A Deterministic Concurrent Product, Production, and Capacity Planning Model for Design, Manufacture, and Support

by

William Kenneth Hoehn

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Industrial and Systems Engineering

APPROVED:

Wolter J. Fabrycky, Chairman

Benjamin S. Blanchard

C. Patrick Koelling

Scott F. Midkiff

William G. Sullivan

February, 1994

Blacksburg, Virginia

A Deterministic Concurrent Product, Production, and Capacity Planning Model for Design, Manufacture, and Support

by

William Kenneth Hoehn

Wolter J. Fabrycky, Chairman

Industrial and Systems Engineering

(ABSTRACT)

This research shows that a model that concurrently determines product, production, and capacity can provide better results than a model that makes these decisions sequentially. Three versions of a life-cycle complete concurrent model and corresponding sequential model that solve product, production, and capacity planning problems are mathematically formulated and compared through an hypothetical example. All versions of the concurrent and sequential models consider three types of work centers and corresponding types of capital budgeting projects.

Each version of the concurrent and sequential models considers a specific type of capital budgeting project. These are: (1) projects that may be partially accepted with no upper limit, (2) projects that may be partially accepted with an upper limit of one, and (3) projects restricted to values of zero and one.

An hypothetical example spanning twelve fiscal periods that included five products, two product sub-groups, one pre-production/post-support project, three production projects, and one support project was developed to enable comparison of the models. Nine comparisons between each of the three versions of the concurrent and sequential models were conducted resulting in a total of 27 comparisons.

Comparisons of the models required the development of five computer programs. Four of these programs provided Mathematical Programming System (MPS) formatted mixed integer linear programs (MIP) that were solved using the LINDO/386 optimization

program. The fifth program read output from LINDO/386 and provided the future worth of the sequential model.

Results obtained from tests of the concurrent and sequential models were compared based upon future worth, capital budgeting projects funded, products funded, production and inventory quantities, and regular and overtime labor levels. For every combination of product sub-group demand and capital budgeting constraint type, the concurrent model always provided the highest future worth. In addition, while both models funded the same set of products, the concurrent model provided more level labor, production and inventory quantities, and support for products. Following these comparisons, it was concluded that, for any capital budgeting constraint type and product sub-group demand, a concurrent model can provide better planning results than a sequential model.

ACKNOWLEDGMENTS

This work is dedicated to my parents, Dorothy Rittel Hoehn and Kenneth William Hoehn, without whose insight and support, the work could not have been completed.

When undertaking a piece of work of this size, one must call on the help and assistance of many individuals. Their contributions range from technical through moral support. I would like to acknowledge several individuals who have helped me develop into a researcher.

First, I would like to thank my advisor, Dr. Wolter J. Fabrycky, for his help and support during my master's and doctoral work. The initial idea for this work was made possible by a project entitled *Systems Engineering for Product Portfolio* sponsored by Bell Northern Research. The project was jointly supervised by Dr. Fabrycky and Dr. Scott F. Midkiff. I would especially thank Dr. Fabrycky and Dr. C. Patrick Koelling for jointly acquiring LINDO/386, a math programming package.

I would especially like to thank my father, Kenneth W. Hoehn, CPA. Mr. Hoehn read several draft copies of the proposal and dissertation document, and made several important contributions.

Dr. Koelling also helped me during my search for a validation strategy. His efforts have shown me that validation is a critical part of the research process. That is, models developed in academia should be validated to demonstrate their worth. This means that some type of comparison is required. Proof of concept (i.e., solving a problem or several problems with the model using a case study, but including no comparison to other models) is not sufficient to demonstrate a model's usefulness.

During this work, Dr. Midkiff helped me in several ways. First, his knowledge of C programming and the Borland C/C++ compiler was of great help. Second, his help in finding ways of model validation during the proposal process is greatly appreciated.

Dr. William G. Sullivan was very helpful during this process. Dr. Sullivan's knowledge of engineering economy and capital budgeting was most helpful. The idea of defining capacity, and, thus, tightening the definition of the models is attributable to Dr. Sullivan.

I was very fortunate to have Dean Benjamin S. Blanchard's support during the process of preliminary exam, proposal, and defense. Dean Blanchard's knowledge of systems engineering and logistics added significantly to this work. Specifically, the idea of including support in the concurrent and sequential meeting is attributable to Dean Blanchard and Dr. Fabrycky.

I was very fortunate to have the help of my good friend Dr. Chunming Duan during this research. Dr. Duan helped me to learn C programming using the Borland

C/C++ compiler. Further, he gave me moral support. Finally, he helped me formulate a validation strategy.

During the course of this work, I had the opportunity to ask help from Dr. Hanif Sherali on several occasions. Specifically, Dr. Sherali helped me with questions concerning disjunctive programming and mathematical programming packages. His help during this time is greatly appreciated.

Throughout this research effort, I had the chance to talk with several individuals knowledgeable in the area of product, production, and capacity planning. Dr. Jay S. Kim, a professor at Boston University, co-authored an article relating to this subject in 1992. A key component of the research presented herein is the validation strategy. This validation strategy is in part due to phone conversations I had with Dr. Kim during the Summer of 1993. It was Dr. Kim's idea to include a second stage in the sequential model. This stage resolves the differences between the first stage and third stages. I truly appreciate Dr. Kim's willingness to help me.

During the research process, it is important to have friends and associates. I am fortunate to have several. First, I would like to thank my good friend Peter Phusavat for his help and support during this process. Peter contributed several key ideas during this process, and for this, I am eternally grateful. Since coming to Blacksburg in August of 1989, I am fortunate to be associated with Dinesh Verma. Dinesh and I had several key conversations over the last year that, I think, have helped form the work contained herein. Finally, I would like to thank Hunter Nichols for his help and knowledge while selecting the correct mathematical programming package.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. BACKGROUND

This chapter presents an overview of the work performed to show that a concurrent product, production, and capacity planning model can provide better results than can be obtained from a similar sequential model. As such, the chapter contains a discussion of the concurrent and sequential models developed through this research, the comparative approach, and the results obtained from these comparisons. The last section of the chapter provides a description of the organization of the remainder of the dissertation.

1.1. Introduction

Organizations engaged in the design, manufacture, and subsequent support of products typically must make three types of planning decisions each fiscal period. These are:

(1) What products should be produced?

(2) What quantities of these products should be produced in each period?

(3) What capacity must be acquired to enable the design, development, production, support, and phase-out of these products?

In most manufacturing organizations, these decisions are made using a sequential planning approach. Decision (1), the product planning decision, is normally determined by top-level management. Decision (2), the production planning problem, is usually solved by operations management using product demand estimates supplied by marketing. Finally, decision (3), the capacity planning problem, is addressed by various departments throughout the organization using budgets and assignments approved by top-level management.

It has been demonstrated that concurrent linkages between product planning and production capacity planning models may result in increased control of the products offered, stabilize process requirements, improve process technology choices, and increase net cash flows over time (Kim et al. 1992). Because of this, Kim et al. (1992) suggested that the typical sequential approach taken towards product planning and production capacity planning should be replaced with a concurrent approach.

A significant shortcoming of the Kim et al. model is that the product life cycle considered consists of only two phases: the production phase, and one general phase to account for design, development, support, and phase-out activities. Specific activities related to the design, development, support, and phase-out of products are not included in the model. Rather, a single expected cost and revenue outcome from these four activities is considered. Because the model does not explicitly consider all six product life-cycle phases, it is not life-cycle complete.

The research presented herein provided three versions of a concurrent product, production, and capacity planning model (i.e., the concurrent model shown in Figure 3.1) that is life-cycle complete. Each version of the concurrent model simultaneously considers all six product life-cycle phases, rather than only the production phase and one general phase, as in the Kim et al. model. Further, each version considers a specific type of capital budgeting project constraint. Version one of the model allows unconstrained partial capital budgeting project selection. Version two allows partial capital budgeting project selection with projects less than or equal to one. Version three restricts capital budgeting projects to values of zero and one.

Several comparisons using a hypothetical example were made between the three versions of the concurrent model and three similar life-cycle complete versions of a sequential model. These comparisons conclusively demonstrated that, for any level of demand and type of capital budgeting project constraint, the concurrent model can provide a future worth of all revenues less costs that is higher than could be obtained from the sequential model.

1.2. Concurrent Versus Sequential Planning Models

In most cases, concurrent linkages between product design, production planning, and manufacturing process technology do not exist in planning models. In practice, a sequence of models is solved starting with product planning and ending with production planning. The sequential solution process is most likely the result of research specific to product planning, production planning, or capacity planning rather than research that considers these three planning problems concurrently. Some reasons for specialization often cited are: (1) the difficulty associated with solving large-scale discrete mathematical programs, (2) the lack of sufficient information, (3) the sequential approach to design, development, and production is taken by the organization (thus, a concurrent model may be inappropriate), and (4) the lack of adequate computer software and hardware.

An outcome of specialization is that models simultaneously considering production planning and capacity planning are rare. Further, prior to this research, no life-cycle complete models existed that simultaneously considered product planning, production planning, and capacity planning.

It is readily apparent that the production plan relies heavily on the availability of manufacturing equipment. Complicating this problem even more is the fact that product planning and manufacturing planning are usually conducted separately. Thus, the best set of products and production quantities may not be offered because of lack of equipment availability and/or manufacturing technology.

This research alleviates the inherent problems associated with sequential planning through the development of a concurrent model that simultaneously determines:

(1) The set of products that **should be funded**[1.1], and the set of products that **should not be funded**[1.2],
(2) The quantity of the funded products that should be produced each period, and
(3) The design, development, and phase-out resources, and production and support equipment that should be procured.

In practice, this concurrent model may be used by an organization to determine the products, production quantities, and capital budgeting projects that will be most beneficial. Benefit is defined as the products, projects, and production quantities in each fiscal period that results in the highest future worth of all revenues less costs at the expected end-of-life for the longest-lived product.

The concurrent model is dynamic in that the model should be run each fiscal period. Thus, products funded in one period may not be funded in future periods. Because of this, the concurrent planning model should be viewed as a strategic level decision support tool. It should be used by management to guide, rather than determine final planning decisions.

1.3. The Concurrent Model

This research was conducted to develop and present a product, production, and capacity planning model that simultaneously determines:

(1) The set of products that should be funded, and the set of products that should not be funded,
(2) Production quantities of each product in each fiscal period, and
(3) The pre-production[1.3], production, support, and post-support[1.4] capacity that must be acquired.

1.1 The set of funded products includes products funded in the previous period that will be continued in the current period of analysis and new products about to enter the first phase of their life cycles. Once funded, a new product may be designed, developed, produced, supported, and phased-out. However, a product funded in the previous fiscal period may be discontinued in the current period of analysis. Because of this, a product selected for funding may not actually complete all of its life-cycle phases.

1.2 The set of products that are not funded are those that should be discontinued and any new products that were considered but should not be funded.

1.3 Pre-production resources are those resources that are consumed during the design and development phases of the life cycle.

1.4 Post-support resource are those resources that are consumed during the phase-out life-cycle phase.

This model is known as the concurrent model and has the following properties: (1) in any specific period, production quantities may be less than the quantity demanded, (2) backlogging is not permitted, (3) the objective function is comprised of a single specific linear continuous function for each cost and revenue component, and (4) consideration of the complete product life cycle is considered.

Three versions of the concurrent model were developed. Each version considered a specific type of capital budgeting project.

1.4. Development of a Comparative Model

As noted above, this research resulted in the development of three versions of a concurrent product, production, and capacity planning model. The main conjecture in this research is that a concurrent model can provide better planning results in terms of the future worth of revenues less costs than can be obtained from a similar sequential planning model.

Because no life-cycle complete sequential model existed, it was necessary to develop a comparative sequential model. As such, three versions of a sequential model were developed and presented herein. These versions correspond to the three versions of the concurrent model. As such, each version could be tested using an hypothetical example. The results were then directly compared to the results obtained from tests of the same version of the concurrent model.

The sequential model relies on an input set that is identical to the input set required by the concurrent model[1.5]. Thus, a fair basis of comparison was established.

1.5. Comparative Approach

Two types of comparisons were used in this research to demonstrate that: (1) the concurrent model can provide superior results to a similar sequential model, and (2) a different version of the concurrent model is applicable to each type of capital budgeting project constraint. In the first comparison, results obtained from three versions of the concurrent and sequential models were compared based upon: (1) the future worth of revenues less costs, (2) capital budgeting projects funded, (3) production quantities and inventory levels, and (4) regular and overtime labor levels. A total of 27 comparisons were made. It was found that, for any demand level and capital budgeting constraint type, the

---

[1.5] An exception to this is that the first stage of the sequential model estimates production capacity based upon the existing cost of production capacity. Because of this, the input that adjusts labor levels in the first stage is also estimated. Thus, the first stage requires an input not used in the concurrent model.

versions of the concurrent model always provided a higher future worth[1.6] than the corresponding version of the sequential model.

Test results obtained from the three versions of the concurrent model were compared to determine differences in future worth and capital budgeting project funding levels for three types of capital budgeting project constraints. It was found that the future worth is dependent upon the level of demand and capital budgeting constraint type considered by the concurrent model. In addition, it was found that each version of the concurrent model is applicable to a specific type of capital budgeting constraint type.

During the comparison of test results, it was found that in situations when unconstrained partial capital budgeting project selection is feasible, the highest future worth was obtained from the version of the concurrent model that allows partial unconstrained project selection. In addition, it was found that when partial project selection is feasible, but projects can be no greater than one, the highest future worth was obtained from the version of the concurrent model that allows partial project selection, and which bounds projects by one. Finally, it was found that when it is not feasible to select parts of a capital budgeting project, the version of the concurrent model that restricts capital budgeting projects to values of zero and one provided the highest future worth.

The set of products funded by all versions of the concurrent and sequential models is identical. However, each version of the concurrent model provided more level labor, and production and inventory quantities than was obtained from the same version of the sequential model.

1.6. Need for this Research

Product planning is defined as "the evaluation of the range, mix, specification and pricing of existing and new products in relation to present and future market requirements and competition; planning of product range, mix, specification and pricing to satisfy company objectives; and specifying the research, design and development support required" (Baker and McTavish, 1976). It is implied in this definition that the organization has "...an overall corporate plan which provides a framework within which these activities may take place" (Baker and McTavish, 1976). In many cases, product planning and capital budgeting activities are conducted separately (Stone, 1976). Because of this, formal product planning and/or capital budgeting may not exist at all (Baker and McTavish, 1976; Gurnani, 1984).

[1.6] When the average cost of capital was 10%.

A review of the literature indicates that the majority of companies do not make use of sophisticated capital budgeting tools (e.g., linear programming) (Petty, 1975; Kim and Farragher, 1981; Gurnani, 1984). In addition, some literature states that many companies conduct little or no product planning activities at all (Baker and McTavish, 1976; Stone, 1976). Instead, they are technically dominated firms, with their focus being on R&D and its requisite acquisition and consumption of a vast array of modern capital equipment, personnel, and materials. Thus, the use of a concurrent planning model that can successfully integrate these management planning functions may help alleviate the aforementioned problems.

1.7. Research Contribution

The research contribution is that a concurrent model that simultaneously considers all product life-cycle phases and also alleviates the problems encountered in sequential product, production, and capacity planning was developed and is presented herein. It is shown that the simultaneous solution of product, production, and capacity planning problems can yield plans that are better (e.g., provide a greater future worth) than plans obtained from a similar sequential model. Since no sequential model existed that could be directly used or expanded to demonstrate this point, a sequential model with the same set of inputs as the concurrent model was developed and presented.

It is conclusively shown that the highest future worth is obtained from the concurrent model. To show this, a total of 27 comparisons were made between three versions of the concurrent and sequential models. In every comparison, the concurrent model provided a greater future worth than the sequential model. Thus, it was shown that, for this hypothetical example, the concurrent model is superior to the sequential planning model. Also, it was shown that each version of the concurrent model is applicable to a specific type of capital budgeting project constraint.

1.8. Dissertation Outline

The remainder of this dissertation is divided into seven chapters. A brief outline of each of these chapters is given below:

Chapter 2: Literature Review - Provides a review of product, production, and capacity planning approaches. Presents in detail the seven concurrent planning models that have been developed in the past.

Chapter 3: Research and Development Methodology - Provides a discussion of the approach used to develop and compare the concurrent and sequential planning models.

Discusses the factors that make this research unique, the limitations and decision environment for the models, the assumptions made, and significant contributions made by the concurrent model.

Chapter 4: The Concurrent Model - Presents a mathematical formulation of the version of the concurrent model that restricts capital budgeting projects to values of zero and one. Constraints to allow partial project selection are incorporated in Chapter 6.

Chapter 5: The Sequential Model - Presents a mathematical formulation of the version of the sequential model that restricts capital budgeting projects to values of zero and one. Constraints to allow partial project selection are incorporated in Chapter 6.

Chapter 6: Hypothetical Example and Programs - Provides a discussion of the input set developed to facilitate comparison of the concurrent and sequential models. Also presents the computer programs developed to make each model partially operational. Finally, shows how the input set can be run through the computer programs and then linked to LINDO/386[1.7] to solve both the concurrent and sequential models.

Chapter 7: Results, Discussion, and Relationships - Provides a detailed analysis of the results obtained from the concurrent and sequential models. Two main comparisons are provided. First, 27 comparisons are made between the concurrent and sequential models. It is shown that, for every type of capital budgeting project constraint, the concurrent model is superior to the sequential model. Second, the three versions of the concurrent model are compared. It is shown that the each version is applicable to situations corresponding to the three types of capital budgeting project constraints. Further, it is shown that use of versions of the concurrent model that allow partial project selection when partial project selection is infeasible may result in: (1) a higher future worth reported than can actually be obtained, and (2) inappropriate selection of capital budgeting projects.

Chapter 8: Summary, Conclusions, and Recommendations - Provides a summary of the research, conclusions gained from the comparisons made, and recommendations for additional research.

Chapter 9: References - Provides a list of references cited, and additional references consulted but not cited.

Appendix A: Input and Output Data Sets - Provides a listing of the 873 inputs required by the Borland C/C++ programs.

[1.7] A DOS-based linear optimization software package.

Appendix B: Program Listing for the Concurrent Model - Provides a complete listing of the Borland C/C++ programs (sin.exe) developed to solve a planning problem using the concurrent model. The output from this program is an Mathematical Programming System (MPS) file[1.8] that is used as input to LINDO/386.

Appendix C: Program Listing for Stage One of the Sequential Model - Provides a complete listing of the Borland C/C++ programs (first.exe) developed to solve the first stage of a planning problem using the sequential model. Input is taken from the standard input set (Appendix A). The output from this program is an MPS file that is used as input for stage one optimization using LINDO/386.

Appendix D: Program Listing for Stage Two of the Sequential Model - Provides a complete listing of the Borland C/C++ programs (middle.exe) developed to solve the second stage of a planning problem using the sequential model. Input is taken from the standard input set (Appendix A), and from a file generated by stage one optimization using LINDO/386. The output from this program is an MPS file that is used as input for second stage optimization using LINDO/386.

Appendix E: Program Listing for Stage Three of the Sequential Model - Provides a complete listing of the Borland C/C++ programs (last.exe) developed to solve the third stage of a planning problem using the sequential model. Input is taken from the standard input set (Appendix A), and from a file generated by stage two optimization using LINDO/386. The output from this program is an MPS file that is used as input for third stage optimization using LINDO/386.

Appendix F: Program Listing for the Future Worth from the Sequential Model - Provides a complete listing of the Borland C/C++ programs (fwss.exe) developed to determine the actual future worth obtained from the sequential model. Input is taken from the standard input set (Appendix A), and from a file generated by stage three optimization using LINDO/386. The output from this program is a report indicating the future worth, products and projects funded, production and inventory quantities, and regular and overtime labor levels.

Appendix G: Comparison of Production and Inventory Quantities - This appendix contains the production and inventory quantities supplied by 9 tests of each version of the

---

[1.8] The MPS format was first developed to solve linear optimization problems on IBM mainframes. LINDO/386 provides a facility to read files that follow the MPS format. Because of this, a linear program can be developed using software (e.g., Borland's C/C++), and then solved using LINDO/386.

concurrent and sequential models. The appendix contains a total of 26 production quantities and 21 inventory quantities obtained from each test of each model.

Appendix H: Comparison of Regular Time and Overtime Labor Levels - This appendix contains the regular time and overtime labor levels supplied by all tests of the three versions of the concurrent and sequential models. The appendix contains a total of 24 regular time and 24 overtime labor levels obtained from each test of each model.

2. LITERATURE REVIEW

This research produced a concurrent model appropriate for product, production, and capacity planning. Thus, a discussion is in order of approaches developed in the past to enable these planning activities. This discussion will present planning models and methods in descending order. That is, the presentation will start with product planning models and end with production planning models. In addition, some models and methods developed in the past to link these planning domains will be discussed. Key conclusions from this literature review are then presented.

2.1. Anthony's Framework for Analysis

Organizations routinely implement four types of decision mechanisms when conducting capacity (or resource allocation) planning. These are: (1) strategic budget determination, (2) product planning, (3) capital budgeting, and (4) production planning. These four categories follow the Framework for Analysis developed by Anthony (1965). The Framework for Analysis divides planning into three distinct, but overlapping categories. These categories are: (1) strategic planning, (2) management control, and (3) operational control. These three levels of management planning correspond to strategic, tactical, and operational decision making (Hax and Candea, 1984; Blank, 1985)[2.1].

Most decision models developed in the past address only one of these decision categories. Thus, organizations typically use several linked models to make product, production, and capacity decisions. Table 2.1 shows factors that differentiate each of these three decision types. Strategic planning "...is the process of deciding on objectives, on changes in these objectives, on the resources used to attain these objectives, and the policies that are to govern the acquisition, use, and disposition of these resources" (Anthony, 1965). Thus, strategic planning deals with the development of long-term plans to lead the organization in a specific direction, and protect it from variations in external entities (i.e., variations in the elements of task environment) (Thompson, 1967). Management control "...is the process by which managers assure that resources are obtained and used effectively and efficiently in the accomplishment of the organization's objectives" (Anthony, 1965). Operational control "...is the process of assuring that specific tasks are carried out effectively and efficiently" (Anthony, 1965).

2.1 It should be noted that Kurstedt (1989) has defined four levels of endeavor - strategic, tactical, operational, and clerical. Addition of a clerical level decision is not required for this discussion on capacity.

Table 2.1. Differentiating factors of three decision types (Hax and Candea, 1984).

FACTOR	STRATEGIC PLANNING	MANAGEMENT CONTROL (TACTICAL PLANNING)	OPERATIONAL CONTROL
Purpose	Management of change, resource acquisition	Resource utilization	Execution, evaluation, and control
Implementation instruments	Policies, objectives, capital investments	Budgets	Procedures, reports
Planning horizon	Long	Medium	Short
Scope	Broad, corporate level	Medium, plant level	Narrow, job shop level
Level of management involvement	Top	Middle	Low
Frequency of replanning	Low	Medium	High
Source of information	Largely external	External and internal	Largely internal
Level of aggregation of information	Highly aggregated	Moderately aggregated	Detailed
Required accuracy	Low	Medium	High
Degree of uncertainty	High	Medium	Low
Degree of risk	High	Medium	Low

Note that this definition of operational control does not clearly separate it from management control. However, a means of distinguishing these two types of planning are by contrasting activities inherent in each. First, "operational control is concerned with tasks and the tasks to which operational control relates are specified, so that little or no judgment is required as to what is to be done (Anthony, 1965). Second, "in operational control, the focus is on execution; in management control it is both on planning and execution" (Anthony, 1965). Finally, management control decisions are typically not structured (or programmed) and the focus is on planning and allocating resources, not on execution.

2.2. Strategic Planning

Organizations are affected by four major environmental factors: (1) novelty, (2) intensity, (3) speed, and (4) complexity (Ansoff, 1979). Novelty refers the concept that "...important events which affect the [organization] are progressively disconnected from past experience." The strategic intensity of a firm is given "...by the strategic budgets[2.2] of the participant [organizations]." Speed refers to the rate of environmental change. Complexity refers to the level of difficulty the organization has trying to understand the environment. Any strategic planning activity must balance these four areas.

2.2 The strategic budget is the total of the marketing, entrepreneurial, and operating budgets (Ansoff, 1979).

In addition to customers, the organization must work with suppliers and distributors, and also compete with other producers supplying similar products. When many firms:

(1) *Sell similar products/services to a common pool of customers/clients;*

(2) *Buy their inputs from a common group of suppliers;*

(3) *Obtain their subsidies from a common pool of donors;*

(4) *Share a common body of knowhow, called technology, which is essential for their commercial activity.*

they are engaged in a common industry (Ansoff, 1979). An industry is composed of firms, "...customers, suppliers, and financing sources" (Ansoff, 1979). An industry will attempt to produce only as much of a particular good as will be consumed. When this condition exists (i.e., the level of supply just meets the level of demand), a Pareto Optimum has been reached and the industry is said to be Pareto Efficient (Stiglitz, 1988).

Under normal competitive conditions, an industry will usually attempt to produce the volume of goods and services that will result in a Pareto Optimal condition[2.3]. However, in reality, the Pareto Optimal condition is probably not obtainable. This is primarily because the market is dynamic, characterized by changing demands, new firms entering the market, existing firms leaving the market, changing prices, and changing quality levels. Industries, however, do attempt to set their production capacity such that a Pareto Optimum can be obtained. Individual firms, conscious of the need to produce only as much as can be consumed, attempt to set their production and distribution capacities to satisfy just the amount of demand for their products[2.4]. This attempt to set production and distribution capacity leads to the need for strategic planning.

One result of strategic planning is the development of a strategic budget (Ansoff, 1979). The strategic budget must be at least equal to the critical mass. The critical mass is defined as "...the budget level just adequate to enable a [firm] to continue to recover its costs from the commercial environment" (Ansoff, 1979). The critical mass is the sum of three components: (1) entrepreneurial critical mass, (2) marketing critical mass, and (3) operating critical mass. The entrepreneurial critical mass "...is the minimum expense needed to keep the [firm's] offerings up to date" (Ansoff, 1979). The marketing critical mass is the

---

[2.3] It is assumed that competition exists since an unregulated monopoly will generate maximum profit when production quantities are lower than demand (Stiglitz, 1988). Hence, an unregulated monopoly will attempt to set supply less than demand. This is because, without subsidies, the monopoly may incur a loss.

[2.4] In some cases, organizations attempt to create an increase in demand. This may be done either through marketing strategy, price reduction, quality improvement, better distribution capabilities, or a combination of all four.

"...minimum budget needed to assure sales of the [firm's] output, sufficient for a break-even operation" (Ansoff, 1979). The operating critical mass is the minimum budget needed to "...keep production costs low enough to make the output saleable at a break-even" (Ansoff, 1979).

Ansoff (1979) has hypothesized that "...at a given point in time there is a region of budget mixes which will produce the optimal economic results." Because operating budgets are determined along with market and entrepreneurial budgets, it follows that the strategic planning activities of an organization will effect the level of funds available for operations. These funds then affect the set of products, projects, and production volumes that the firm can offer.

2.3. Product Planning

Stone (1976) has stated that a commonly applied decision rule in organizations is:

Produce any product which will produce more than a given annual revenue and yield a given profit margin on sales.

This decision rule has the effect of separating the product strategy from the market policy, and, as such, "...such a procedure is unlikely (except by chance) to lead to a firm's aims being realised if that marketing policy is changed" (Stone, 1976). In some cases, organizations do not employ product planning at all. Lack of product planning usually occurs in one of three ways. These are: (1) small firms with few product lines, (2) technical orientation of management, and (3) large firms with a technical orientation (Baker and McTavish, 1976). Reasons for lack of product planning in small firms stems from the fact that there exist a few products in production and in prospect, and management believes they know the product mix well. In these smaller firms, however, management may not understand and convey their objectives, thus bringing into doubt the effectiveness of the "product plan." Some technically oriented firms are run by management with largely technical backgrounds. Because of this, their focus is on the technical management of R&D, and the new products the process may bring. "The result is that the product-planning activity is incomplete" (Baker and McTavish, 1976). In the third case, large technically oriented firms, product planning deficiencies stems from "...the rapid growth that gave them the large size...," and the technical orientation of management (Baker and McTavish, 1976).

A study conducted to examine the product planning practices of seven large firms reported that: (1) two companies were not undertaking product planning activities at all, (2) four companies had not translated the concepts of product planning into action, and (3) only one company employed product planning (Baker and McTavish, 1976). However, even the

company that had employed product planning experienced problems because of the "...informal nature of the product planning activity itself" (Baker and McTavish, 1976). Further "...empirical studies tend to suggest incomplete integration of product policies and objectives, new-product development, and maintenance of established product lines" (Baker and McTavish, 1976). Some reasons that may account for this are:

(1) *The importance of new-product development in terms of time, cost and historical contribution;*

(2) *Technically oriented management; and*

(3) *Reliance on informal methods appropriate to small firms which have not kept pace with the growth of company size.*

The "...failure to integrate all phases of product planning and product development may lead to a number of dangers" (Baker and McTavish, 1976). The following list presents some of these dangers:

(1) *A firm may discover that it is selling a large volume of unprofitable products with no profitable new-product replacements potentially available;*

(2) *Inadequate attention to product objectives and policies may lead the firm in directions in which it had not intended to proceed;*

(3) *Reliance on R&D may result in the initiating of too many development projects with the result that the effort is dissipated over a wide area rather than concentrated on those with good commercial prospects.*

The reasons for this failure to integrate are further frustrated by difficulties encountered when a firm does attempt to adopt a formal product planning system. Many of these difficulties can be traced to problems encountered in determining an optimal portfolio (Baker and McTavish, 1976). Baker and McTavish (1976) have stated that many problems exist in attempting to determine the optimal product portfolio. Some of these are:

(1) *The several marketing-mix elements (product, price, channels and promotion) are mutually interdependent; no single element can be fully effective in the market without support from other elements. Thus planners need to consider the full range of marketing-mix elements in seeking to formulate an integrated marketing strategy.*

(2) *Marketing planners [or product planners] seldom have any precise knowledge about the relationship between any single marketing-mix element and sales in a specific target market [or when many products flow into a single market]. In the case of advertising, for example, estimates of advertising impact on sales are frequently imprecise and must allow for a*

sizable margin of error. When changes in several elements of the marketing-mix are in prospect (the normal situation in strategic planning), the problem of estimating effects on sales becomes even more formidable.

(3) *The marketing-mix is planned for use over a future period which can be forecast in probabilistic terms only.*

(4) *The planned marketing-mix should not exceed the physical, financial and managerial capabilities of the company.*

In addition to these difficulties associated with product planning, Stone (1976) has also pointed out that organizations, in many cases, restrict their product portfolio by adopting decision approaches that limit their ability to produce and compete. There are two types of decision approaches that will place a growth type constraint on the organization. The first type, the "resource-constrained" approach, limits the amount and range of products that can be produced by restricting analysis to include only the "...set of resources [currently] owned or controlled by the firm" (Stone, 1976)[2.5]. The resource-constrained approach is usually adopted because of uncertainty that exists with the success of acquisition of resources required by an expanding product portfolio or increased production capacity. However, this approach does have the long-term effect of limiting the growth of the firm. The "market-constrained" approach is a related approach often adopted by firms, and is used to constrain the firm's business to a specific market or industry (Stone, 1976). Unfortunately, an approach such as this places an "...*a priori* restriction on the firm's product strategy" (Stone, 1976). In some cases, the market-constrained approach may be more limiting than the resource-constrained approach because its adoption may combine the explicit market restriction with an implicit resource limitation (Stone, 1976). Hence, the effects of resource-constrained and market-constrained approaches may be detrimental to the firm.

Stone (1976) has suggested that, to alleviate the problems caused by the two approaches discussed above, the firm should adopt a "rational" approach. In the case of the resource-constrained approach, Stone (1976) suggests that an approach be adopted that considers resources not to be constrained by current resources, but rather, allows for the "...consequences of relaxing [the] constraints." The rational approach suggests that the firm

---

[2.5] It should be noted that adoption of the resource-constrained approach means that capital budgeting activities will precede product planning activities. The capital budgeting activities should still fulfill the strategic goals of the organization (if they exist), but may be subject to similar problems as those discussed for product planning activities (i.e., strategic plans may not exist, thus, these activities may not occur at all, or will not result in efficient or effective selection).

concentrate on the set of possible consequences from the relaxation of various constraints. The amount of information required for an approach of this type would be used as a determinant of cost. Therefore, the firm would have to be careful to limit the number of possible consequences that are investigated. The alternative to the market-approach is an approach that attempts to identify possible markets that may be beneficial to the firm. The rational approach, therefore, to some degree alleviates the detrimental effects of the market-constrained and resource-constrained approaches.

2.4. Life-Cycle Analysis

Fabrycky and Blanchard (1991) have shown that products follow a specific life-cycle made up of six discrete phases. These six phases are: (1) conceptual design, (2) preliminary design, (3) detailed design and development, (4) production, (5) product use and support, and (6) phase-out. Further, products are supported by some type of production and logistics systems (Figure 2.1). Thus, an integrated planning model should ideally consider the three systems life cycles simultaneously.

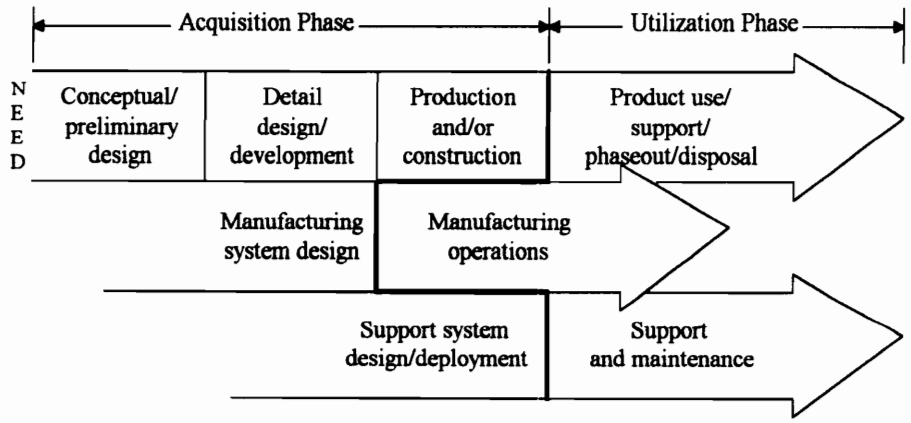

Figure 2.1. The product, production, and logistics system life cycles (Fabrycky and Blanchard, 1991).

2.5. Capital Budgeting

For the most part, capital budgeting decisions deal "...with investments in a company's plant and equipment" (Clark et. al., 1984). However, Nunamaker and Truitt (1987) have shown that capital budgeting - or as they termed it, rationing - "...can be extended to include not only new fixed-asset purchases but also expenditures for any other

discretionary activity such as advertising, research and development, or maintenance." In practice, capital budgeting models are decision mechanisms designed to control capacity, and vary in their level of sophistication. Because of this, the specific capital budgeting mechanism being used will dictate how well life-cycle factors are incorporated into a capital budgeting decision. Therefore, the sophistication of the capital budgeting mechanism plays an important role in determining the degree to which life-cycle factors are considered.

2.5.1. Capital Budgeting as a Means to Control Capacity

In general, an organization's capacity may be divided into two types[2.6]. These are: (1) long-term, and (2) short-term (Groover and Zimmers, 1984). It is generally accepted that of the two, long-term capacity costs more to acquire and cannot be readily acquired (i.e., there is a time-lag between the recognition of need and actual acquisition of the capacity). Short-term capacity, on the other hand, may be acquired or abandoned rapidly, and thus, in most cases, the time-lag between recognition of need and acquisition is relatively short.

Long-term capacity is generally decided through strategic budgets and tactical decision approaches. Groover and Zimmers (1984) state that some factors that affect a firm's long-term capacity are:

(1) Number and production capability of machines
(2) The amount of new plant construction or expansion
(3) Purchase of a plant from another firm

Short-term capacity is affected by long-term capacity decisions, but is ultimately controlled through production planning approaches. Some examples of short-term capacity are (Groover and Zimmers, 1984):

(1) Employment levels
(2) Number of work shifts
(3) Inventory stock levels
(4) Order backlogs
(5) Sub-contracting

Both models developed during this research simultaneously adjust long-term and short-term capacity. Even though the models will find the best solution with respect to future worth at the horizon, it is apparent that the firm may acquire capacity that will not be

---

2.6 For a complete review of capacity management techniques, see Luss (1982).

used[2.7]. This is because, under most conditions, capacity is usually only obtainable in discrete blocks of finite size. Thus, to move from a capacity deficit level to a sufficient level, it may be necessary to acquire excess and/or idle capacity. In addition to this, two other factors influence capacity acquisition. First, the capacity required in one period may be less than that required in future periods. Because of the time required to obtain needed capacity and the increments it is available in, idle capacity may be required in early periods to ensure that demand may be met in the future. Second, the models depend upon specific capital budgeting project proposals. Thus, there is no guarantee that the proposals will exactly fulfill capacity demands in any given fiscal period.

Variable consumer demand may cause the capacity needs of an organization to fluctuate by: (1) trending[2.8], and/or (2) cycling. Thus, a firm may experience cycling along with a trend. Because of fluctuating demand, the organization's capacity at any given time may be: (1) too little, (2) too much, or (3) just right. However, it is doubtful that the third condition could ever be achieved. Because of this, organizations tend to meet variable consumer demand by varying their short-term capacities, and setting their long-term capacities to meet probable future demand.

2.5.2. Measures of Capacity

Capital budgeting mechanisms must determine the best project or projects to fund by comparing the potential for increase in profit (or decrease in cost) resulting from the increases in capacity that could be provided by each project. Thus, the factors taken into account when determining the increase in capacity provided by each project will have a significant effect on the accuracy of the capital budgeting mechanism.

The field of cost accounting recognizes four measures of capacity. These are: (1) theoretical maximum plant capacity, (2) practical plant capacity, (3) normal sales expectancy[2.9], and (4) expected actual volume or estimated production (Crowningshield and Gorman, 1979; Dickey, 1964; Matz et. al., 1984; Bulloch et. al., 1983; and Chatfield and Neilson, 1983). These capacity measures may be applied by cost accountants to determine the portion of product cost attributable to production equipment (i.e., overhead cost

---

[2.7] Capacity that is not used in the present period can be divided into two categories. First, capacity that is unused in one period, but used in any subsequent period is termed idle capacity. Second, capacity that is available but never used is commonly referred to as excess capacity (Crowningshield and Gorman, 1979).

[2.8] Note that the trend can be upwards, downwards, or have a slope equal to zero (i.e., identical demand in each period).

[2.9] Normal sales expectancy is also referred to as normal capacity.

attributable to production capacity costs). These overhead costs may be included in all budgets of the firm, and are allocated to product inventory[2.10].

No references reviewed by the researcher relating to capital budgeting include the aforementioned classification of capacity measures used by cost accountants in developing overhead rates. Thus, it is likely that these measures have not often been applied in the area of capital budgeting. However, a brief review of these measures will be provided because the type of capacity measure employed during the capital budgeting process may have a large impact on the accuracy of some decision mechanisms.

2.5.2.1. Theoretical Capacity

The theoretical capacity of a plant or department is represented by line AB in Figure 2.2. This is the capacity of a plant or department that would be achieved under 100% operating time. "This involves no limitation for waits and delays of any character or for shutdowns on weekends or holidays. It reflects no regard for sales expectancy. The maximum activity level is greater than the practical goal. It cannot be reached in an actual situation and accordingly is seldom, if ever, used" (Dickey, 1964).

2.5.2.2. Practical Capacity

Line AC in Figure 2.2 represents the practical capacity of a plant or department. Practical capacity is usually used "...when a normal activity concept is related primarily to physical production facilities rather than to sales expectancy. This practical plant capacity is theoretical maximum plant capacity less unavoidable operating interruptions" (Dickey, 1964). Practical capacity makes allowances for "...unavoidable interruptions, such as time lost for repairs, inefficiencies, breakdowns, setups, failures, unsatisfactory materials, delay in delivery of raw materials or supplies, labor shortages and absences, Sundays, holidays, vacations, inventory taking, and pattern and model changes. These allowances reduce theoretical capacity to a level often called practical capacity level. It should be noted that this reduction is based on internal influences and does not consider the chief external cause, lack of customers' orders. The reduction from theoretical capacity to practical capacity ranges from 15 to 25 percent, which results in a practical capacity level of 75% to 85% of theoretical capacity" (Dickey, 1964).

2.5.2.3. Normal Sales Expectancy

Line AD in Figure 2.2 represents the normal sales expectancy of an organization. The normal sales expectancy is usually "...40% to 50% of theoretical capacity...", and further

---

[2.10] The Federal Tax Code (Section 1.147-11) allows the use of practical capacity for allocation of overhead costs to products in inventory (Federal Tax Regulations, 1993).

reduces practical capacity by removing the average idle capacity (Dickey, 1964). Normal sales expectancy is set such that the "...peaks and valleys which come with seasonal and cyclical variations..." are leveled out (Dickey, 1964).

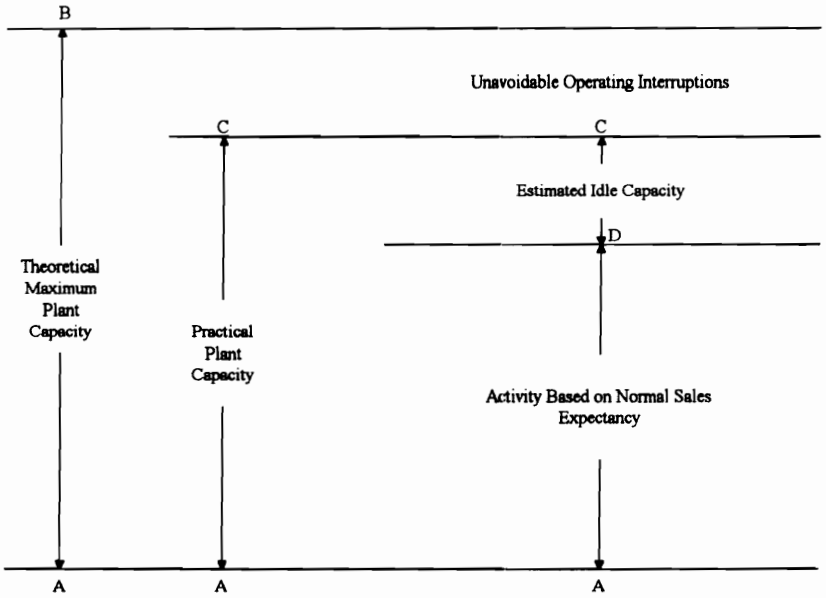

Figure 2.2. Relationships of activity levels (Dickey, 1964).

2.5.2.4. Expected Actual Capacity

The expected actual capacity is similar to normal capacity in that idle capacity caused by the lack of sales is accounted for. However, expected capacity is established on a per period basis, and as such, does not reflect or exceed the long-term capacity of the firm.

2.5.2.5. Advantages of Capacity Measurement Approaches

These four measures of capacity may be further divided into two categories; those that consider sales volume and those that do not. The advantages of plant capacity approaches (i.e., theoretical and practical) are (Dickey, 1964):

(1) Costs of excess and idle facilities are provided to management in the form of unabsorbed charges.

(2) Volume variations are subject to reasonable explanation and interpretation.

(3) Product costs are not inflated by low sales volume so as to overstate inventories and affect income calculations.

(4) Costs are computed upon a relatively fixed and determinable base, with a minimum amount of subjective decision with regard to activity levels.

Presumably rates can be set more accurately than is the case with bases involving sales estimates.

The advantages of capacity measures based upon sales expectancy (i.e., normal and expected actual) are (Dickey, 1964):

(1) They provides unit cost information which may be readily integrated into an over-all budget or plan for business operation.

(2) They result in costs per unit which serve as a guide or basis for pricing decisions.

2.5.2.6. Effect on Capital Budgeting Accuracy

It is evident that, while the measures of capacity that account for sales more accurately represent the capacity utilized, they also tend to overstate the portion of machine cost required to produce each product. This is most likely the reason why the MAPI Accounting Manual[2.11] recommends that capacity be set such that a balance between normal capacity and theoretical capacity exists (Dickey, 1964). Further, during the capital budgeting process, the actual or expected sales volume may not be known. Thus, the use of normal or expected capacity may not be appropriate. Second, capital budgeting mechanisms that rely solely on cost may be biased if normal or expected capacity is supplied[2.12]. Finally, sophisticated models, such as the concurrent and sequential models provided herein, that simultaneously determine production quantity and capacity cannot take as input normal or expected capacity[2.13].

The concurrent and sequential models included herein rely on a form of practical capacity that is measured at each work center rather than measured at the department or plant level. As such, the effects of expected sales are not included in the work center capacity calculations supplied as input to both models. However, expected sales is supplied in the form of product sub-group demand. Because of this, idle capacity contained in the solution emanating from either model can be determined subsequent to optimization. The assumptions in Chapter 3 and the input descriptors in Chapters 4 and 5 provide a more

---

2.11 The MAPI Accounting Manual was published by the Machinery and Allied Products Institute in 1952.

2.12 This is because the portion of production capacity cost attributable to each product may be overstated.

2.13 Sophisticated product, production, and capacity planning models determine both the **production quantities**, and the additional capacity to be acquired. These models **cannot** utilize normal or expected capacity because these capacity measures include factors for the expected amount of products to be sold. Since an objective of the sophisticated model is to determine production quantities (and, thus, also to determine idle and excess capacity), sales based measures cannot be used. Thus, normal and expected actual capacities are used only when the production quantity is known in advance.

complete description of the factors accounted for in the capacity measure utilized by the concurrent and sequential models.

2.5.3. Capital Budgeting Mechanisms

As previously stated, capital budgeting mechanisms are managerial decision mechanisms, and range in sophistication from mechanisms based on economic justification of a single project to probabilistic mechanisms developed to account for risk and uncertainty.

2.5.3.1. Single Project Selection Mechanisms

The simplest form of capital budgeting mechanisms rely on the present worth (PW), future worth (FW), or annual worth (AW) of a single project. The PW, FW, or AW is calculated by use of a minimum attractive rate of return (MARR). The MARR is essentially an expected rate of return and includes components to account for both inflation, the earning power due to interest, and an expected return above and beyond these two (Hodder and Riggs, 1985). An underlying assumption with the use of a MARR is that lending and borrowing opportunities are unlimited at the MARR (Park and Sharp-Bette, 1990). Another underlying assumption is that the firm can reinvest proceeds gained from current investments in the future at the MARR (Thuesen and Fabrycky, 1993).

2.5.3.2. Deterministic Programming Mechanisms

Deterministic capital budgeting (DCB) mechanisms

may have either linear or nonlinear objective functions that maximize either PW or FW (Park and Sharp-Bette, 1990). The advantage of DCBs is that they can be used to simultaneously consider monetary and non-monetary constraints. Models that optimize PW or internal rate of return (IRR) are known as pure capital rationing models. Pure capital rationing models do not consider different reinvestment rates, and thus, most likely will not provide the best results possible (Park and Sharp-Bette, 1990). Finally, the use of an external discount rate along with budget constraints violates the underlying assumption implicit in the use of a discount rate, i.e., borrowing is unlimited (Park and Sharp-Bette, 1990).

The Lorie-Savage formulation of the pure capital rationing model for N planning periods is given by:

$$
\begin{aligned}
Max \quad & \sum_j p_j x_j \\
S.T. \quad & -\sum_j a_{nj} x_j \le M_n \qquad \forall\, n = 0,\ 1,\ldots,\ N \\
& x_j \le 1 \qquad \forall\, j = 1,\ldots,\ J \\
& x_j \ge 0 \qquad \forall\, j = 1,\ldots,\ J
\end{aligned}
$$

where p_j is the present value of project j using some MARR, x_j is the project selection variable, a_{nj} the cash flow at time n for project j, and M_n is the budget limit at time n (Park and Sharp-Bette, 1990).

Another type of DCB are the horizon models. Horizon models are appealing because they avoid many of the conceptual difficulties associated with pure capital rationing models (Park and Sharp-Bette, 1990). These models typically include provision for "...borrowing and lending activities and may have other constraints added" (Park and Sharp-Bette, 1990).

The primal formulation of Weingartner's Horizon Model for N planning periods with equal lending and borrowing rates r is given by:

$$
\begin{aligned}
Max \quad & \sum_j \hat{a}_j x_j + \upsilon_N + \omega_N \\
S.T. \quad & -\sum_j a_{0j} x_j + \upsilon_0 + \omega_0 \le M_0 \\
& -\sum_j a_{nj} x_j - (1+r)\,\upsilon_{n-1} + \upsilon_n + (1+r)\,\omega_{n-1} - \omega_n \le M_n \quad \forall\, n = 1,\ 2,\ldots,\ N \\
& x_j \le 1 \qquad \forall\, j = 1,\ldots,\ J \\
& x_j \ge 0 \qquad \forall\, j = 1,\ldots,\ J \\
& \upsilon_n,\ \omega_n \ge 0 \qquad \forall\, j = 0,\ldots,\ J
\end{aligned}
$$

where $\hat{a}_j$ is the time value of the cash flows beyond the horizon, x_j is the project selection variable, a_{nj} is the cash flow for project j at time n, υ_n is the lending amount between time n

and time (n + 1), ω_n is the borrowing amount between time n and time (n + 1), M_n is the external budget limit at time n (Park and Sharp-Bette, 1990). It should be noted here that both pure capital rationing and horizon models allow for partial project selection. Under some cases, this may be unrealistic. In this case, the selection variable can be constrained to 0,1.

2.5.3.3. Goal Programming Mechanisms

A category of decision mechanisms that satisfice, rather than optimize, are those that rely on goal programming (Canada and Sullivan, 1989). Goal programs are of two types: (1) simultaneous solution with priorities, or (2) solutions in stages. While goal programs are most likely to be deterministic, they allow the decision maker to determine the importance of decision attributes, and thus, solve the decision problem with the inclusion specific user needs.

2.5.3.4. Mechanisms that Incorporate Uncertainty

Another category of capital budgeting models are those that consider uncertainty. Uncertainty can be considered either through utility theory or through the use of probability theory. Decision mechanisms that rely on utility theory are important because they can be used to simultaneously compare dissimilar measures of performance based on a decision maker's own interval scales. However, utility decision mechanisms are subject to several deficiencies: Some of these are: (1) framing effects, (2) preference reversal, (3) preference cycles, (4) nontransative indifference, (5) certainty effects, (6) inability to handle probabilities in the loss region, (7) overvaluation of small probabilities and under-valuation of large probabilities, and (8) common ratio effect (Fishburn, 1988).

As an alternative, probability mechanisms for cash flows may be applied to PW analysis. However, the accurate postulation of distributions may fall into question, along with the difficulty encountered when the assumption of independence does not hold. Additionally, even if accurate distributions are found, they may be dissimilar. In this case, simulation would probably be preferred to a closed form solution. Probability could also be applied in multi-attribute decision making, but after a certain point, handling dependencies and differing distributions may become unwieldy.

2.5.4. A Means to Increase Efficiency and Effectiveness

Sometimes theories developed and incorporated into decision approaches in academia by academicians are not popular in industry. There are many reasons for this, but for the most part, this lack of acceptance can be traced to the complexity, cost, and the possible lack of significant returns that adoption of these approaches would bring. In the area of capital budgeting, most organizations that use advanced approaches are fairly large, and thus, can afford the expense of the requisite specialized personnel and equipment

(Gurnani, 1984). Thus, smaller firms use less sophisticated capital budgeting mechanisms. However, studies have shown that there exists a significant positive correlation between the degree of risk encountered and the sophistication of the capital budgeting mechanisms being used (Gurnani, 1984).

Capital budgeting may directly affect the efficiency of an organization because these decisions "...are responsible for maintaining the competitive capabilities of the firm, determining its rate of growth, and eventually defining its success or failure" (Hax and Candea, 1984). Because of this, the capital budgeting process may also indirectly or directly affect organizational effectiveness. However, the degree to which capital budgeting activities affects these two measures may depend upon the organizational structure. Since capital budgeting relies on internally generated information, it is natural to suppose that some organizations may enjoy more success simply by virtue of their organizational structure.

Since capital budgeting deals with the acquisition of an asset or group of assets and the hiring of personnel, it may be said that the organization attempts to find the most efficient use for its capital. Naturally, the complexity and scope of the capital budgeting mechanism and the prevailing organizational structure will have a significant effect on the degree of efficiency that is obtained. That capital budgeting affects the efficiency of an organization can be seen in the Modified Management Systems Model (Sink and Tuttle, 1989) (Figure 2.3). Here we have limited inputs composed of materials, information, capital, energy, and labor. The organization has capital from retained earnings, loans, and sales. Therefore, we may assume that most capital acquisitions will be made from a mixture of these three sources. The capital budgeting process attempts to determine the optimal mix of all inputs entering the organization[2.14]. The optimal mix of inputs should also represent the most efficient mix of inputs. This is because the organization has obtained the most benefit possible for the least cost.

As was noted, in practice, most organizations utilize the concept of satisficing rather than actually optimizing the mix of inputs (Stone, 1976). This is because it may be impossible to look at all of the different courses of action. In this case, the organization will achieve the most efficient allocation of capital for the alternatives that may be considered. The way capital budgeting affects the effectiveness of organization will depend upon the objective of the capital budgeting mechanism. If the mechanism simply minimizes cost, and

---

[2.14] It should be noted that the organization will most likely satisfice rather than find the optimal mixture of capital allocation.

does not address non-monetary constraints, the efficient use of capital is not obtained. However, because the non-monetary constraints have not been addressed, the effectiveness of the organization may not be optimal or the best that could be obtained.

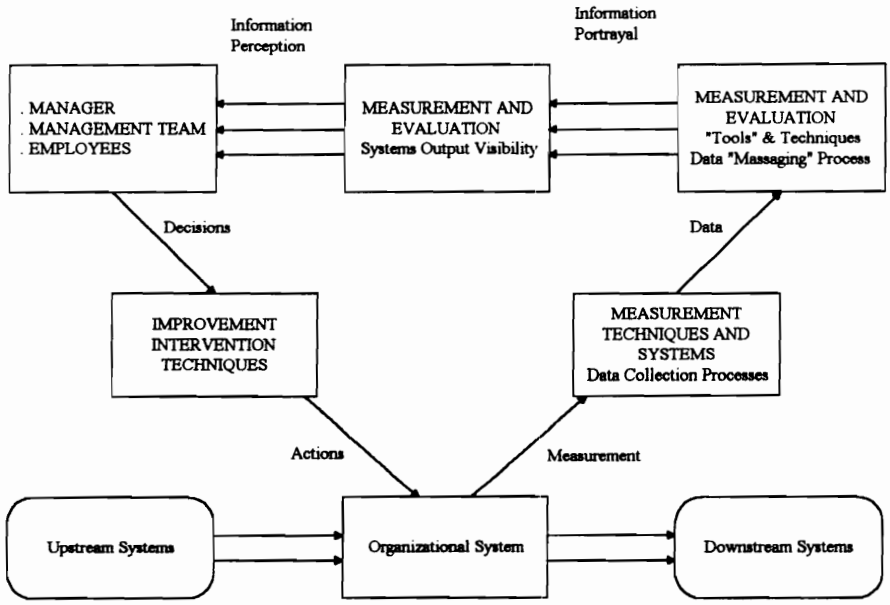

Figure 2.3. The Modified Management Systems Model (Sink and Tuttle, 1989).

For example, if two machines were compared, one costing $2000 with the capability of producing 50 parts per hour and the other costing $1200 with the capability of producing 35 parts per hour, the one costing less would be selected under the aforementioned decision rule. But the fact that the machine can only produce 35 parts per hour may have a significant effect on the profitability of the organization. Thus, the decision has also affected the effectiveness of the organization; possibly in a negative fashion. At the other extreme, if the organization budgets using the maximization of future wealth, with thought given to non-monetary constraints, then both efficiency and effectiveness may be optimized.

Now consider the same example, but with a constraint that the machine chosen must produce at least 40 parts per hour. In this case, the machine with the lesser cost could not be selected, thus helping assure that the organization would maintain a specific level of output. In this case, the mechanism has considered both efficiency and effectiveness simultaneously.

2.5.5. Extensions of Capital Budgeting Models

In recent years, several extensions to capital budgeting models have been proposed. For the most part these extensions concentrate on one of three areas. These are: (1)

incorporation of the dynamic decision environment, (2) widened applicability, and (3) the need to justify advanced manufacturing systems.

While many major contributions to capital budgeting modeling were completed by the early 70's[2.15], recent trends have shown that the philosophies they were based on may not include elements essential to the success of an organization. For example, Blank (1985) has stated that the evaluation of manufacturing systems has taken on a more strategic approach. This approach "...addresses the competitive position of..." products (Blank, 1985).

2.5.5.1. Accounting for the Dynamic Decision Environment

Skipper (1986) and Skipper and Fabrycky (1985) addressed the problem of capital budget allocation in a dynamic, uncertain environment. This model centers around a linear 0,1 integer horizon model featuring continuous cost functions along with constraints to account for contingencies, mutual exclusivities, and must funds with the objective of maximizing future worth at the terminal point of the longest-lived project. It has been shown that maximizing the future worth at the terminal point of the longest lived project will provide good results as long as the MARR is defined to be the average return that the firm receives from typical investments[2.16] (Swalm and Lopez-Leautaud, 1984). Kladivko (1986) extended Skipper's work to include discrete functions and various project funding levels. The models presented by Skipper, Fabrycky, and Kladivko address the dynamic nature of projects by being run on a fiscal or discrete period basis. Thus, projects accepted in a former period may be discarded in a future period. Likewise, projects that were not funded in a former period may be funded in future periods.

2.5.5.2. Extended Applicability

Nunamaker and Truitt (1987) have shown that capital budgeting - or as they termed it, rationing - "...can be extended to include not only new fixed-asset purchases but also expenditures for any other discretionary activity such as advertising, research and development, or maintenance." This extension has the effect of increasing the life cycle applicability, and thus, the number of factors simultaneously considered by the capital budgeting mechanism.

2.15 Note that Bernhard published *Mathematical Programming Models for Capital Budgeting - A Survey, Generalization, and Critique* in 1969.

2.16 The MARR is assumed to be "...the long run average return that the firm receives on a typical investment. The cost of capital is the opportunity cost of an investment, not a required rate of return which is greater than the opportunity cost (Skipper and Fabrycky, 1985)."

2.5.5.3. Inclusion of Intangibles

In many cases, automated manufacturing systems cannot be justified based on economic factors alone but are still important for an organization's survival. Accordingly, some literature has concentrated on approaches that may be used to include non-monetary factors in automated system evaluation. Park and Son (1988) have stated the need for including quality and flexibility measurements and the resulting opportunity costs in the determination of advanced manufacturing system selection. Examples of quality measurements are: (1) prevention costs, (2) appraisal costs, (3) internal failure costs, and (4) external failure costs. Some examples of flexibility measurements are: (1) equipment flexibility (idle cost), (2) product flexibility, (3) process flexibility, and (4) demand flexibility (inventory costs). New approaches that can be used to consider multiple attributes have been proposed. For example, Sullivan (1986) noted the possibility of using an analytic hierarchy process (AHP) to solve capital budgeting decisions. In addition to ensuring adequate evaluation of automated systems, some manufacturing organizations are now forced to consider environmental and social impacts of investment decisions (Evans, 1987). Along with these, some industries are also concerned with the effects that capital budgeting decisions may have on their expansion capability (Smith, 1980; Lawrence et. al., 1980).

2.6. Aggregate Production Planning Models

Aggregate planning affects the utilization of the plant and its equipment through the scheduling of production. Decisions of this type are usually referred to as production planning problems, and affect the arrival of materials, employment profiles, and production and distribution schedules. Production planning systems can be termed budgeting systems since most exist to minimize the costs involved with producing a family, or several families of products. Mathematical models typically used to solve aggregate production planning problems range from definite demand through probabilistic demand, and include forecasting and scheduling algorithms. It may be noted here that forecasting methods commonly employed rely on the assumption that historical events may be used as a good predictor of the future. This, however, may not always be the case (Ansoff, 1979).

Hax and Candea (1984) divide aggregate production planning models into categories by the structure of their cost components. The resulting categories are: (1) linear cost models, (2) quadratic cost models, (3) fixed cost models, and (4) general cost models.

2.6.1. Linear Cost Models

Linear cost models are of two types - those possessing either (1) a single linear goal, or (2) multiple linear goals - and feature linear objective functions and constraints. Models of this type serve to determine the optimal production quantities of specific products in

specific periods over a known length of time. Typically, production capacity is either (1) treated as a constant, or (2) labor levels may be varied in periods.

Formulation of the general variable workforce model is given by:

$$\text{Min } Z = \sum_{i=1}^{N}\sum_{t=1}^{T}(\upsilon_{it}X_{it} + c_{it}I_{it}^{+} + b_{it}I_{it}^{-}) + \sum_{t=1}^{T}(r_tW_t + o_tO_t + h_tH_t + f_tF_t)$$

Subject to:

$$X_{it} + I_{i,t-1}^{+} - I_{i,t-1}^{-} - I_{it}^{+} + I_{it}^{-} = d_{it} \qquad \begin{Bmatrix} i = 1,\ldots,N \\ t = 1,\ldots,T \end{Bmatrix}$$

$$\sum_{i=1}^{N} k_i X_{it} - W_t - O_t \le 0 \qquad t = 1,\ldots,T$$

$$W_t - W_{t-1} - H_t + F_t = 0 \qquad t = 1,\ldots,T$$

$$-pW_t + O_t \le 0 \qquad t = 1,\ldots,T$$

$$X_{it}, I_{it}^{+}, I_{it}^{-} \ge 0 \qquad \begin{Bmatrix} i = 1,\ldots,N \\ t = 1,\ldots,T \end{Bmatrix}$$

$$W_t, O_t, P_t, H_t, F_t \ge 0 \qquad t = 1,\ldots,T$$

where X_{it} is the production quantity of item i in period t, υ_{it} is the per unit production cost for item i in period t, I_{it}^{+} is the number of units of item i in inventory at the end of period t, c_{it} is the carrying cost, I_{it}^{-} is the number of units of i backordered at the end of period t, b_{it} is the backordering cost, W_t is the regular workforce level in period t, r_t is the cost per labor-hour of regular time in period t, O_t is the overtime workforce level in period t, o_t is the cost per overtime hour, H_t is the labor-hours hired in period t, h_t is the cost per labor-hour hired, F_t is the labor-hours fired in period t, f_t is the cost per labor-hour fired, d_{it} is the demand for product i in period t, and p is the overtime as a fraction of regular hours (Hax and Candea, 1984).

Models of the aforementioned type have gained wide acceptance (Hax and Candea, 1984). This is primarily because even large linear programs can be easily solved. However, these linear models do have some disadvantages. The first is their inability to deal with demand uncertainties. However, safety stocks can be introduced to help eliminate this problem. Second, the linear costs implicit in the models may not accurately reflect the true cost behavior. However, this problem can be eliminated by well defined linear cost segments since costs may behave in a linear fashion over a defined range.

2.6.2. Quadratic Cost Models

Quadratic models, also known as linear decision rules, require that all demands and costs for all item types be aggregated into single category (Hax and Candea, 1984). Further, this type of aggregate planning model considers production and employment costs to have a quadratic functional form (Holt et. al., 1960). When these functions are differentiated, they form a set of linear functions.

The first quadratic model was developed by Holt et al. (1960) and has come to be known as the HMMS[2.17] rules. HMMS models determine the optimal aggregate production rate X_t and the work force size W_t in each period t for a total of T periods. Further, the inventory level I_t at the end of each period t are specified by the combinations of X_t and W_t.

Quadratic cost models have three main advantages. These are: (1) the convex cost function more accurately models reality, (2) the derivatives are linear, thus, enabling relatively simple solution methods, and (3) uncertainties are accounted for since the objective is to minimize expected cost. However, these models also have three disadvantages. These are: (1) "the strong need for aggregation, (2) the elaborate estimation procedures that are required to assess the numerical values of the cost coefficients, and (3) the numerical difficulties encountered when the number of decision variables and constraints increase, which limits the model dimensions to a small size" (Hax and Candea, 1984).

2.6.3. Fixed Cost Models

Fixed cost or lot size models add setup cost as a variable to be considered. Thus, these models impose a cost for machine setups. In many cases, fixed cost models are extensions of linear cost models.

An example of a fixed cost model is the capacitated lot size model with variable work force levels. The objective is to minimize the cost of production over several periods by scheduling resources to produce multiple products. Models have been developed that function when demand is not constant and the work force is variable. The formulation of the variable work force, lot size model is given by:

[2.17] The HMMS rules was so named for the developers, Charles C. Holt, Franco Modigliani, John F. Muth, and Herbert A. Simon.

$$Min \quad Z = \sum_{i=1}^{N}\sum_{t=1}^{T}\left[s_{it}\delta(x_{it}) + v_{it}X_{it} + c_{it}I_{it}\right] + \sum_{t=1}^{T}\left[r_tW_t + o_tO_t + h_tH_t + f_tF_t\right]$$

$$S.T. \quad X_{it} + I_{i,t-1} - I_{it} = d_{it} \qquad \forall\, i = 1,\ldots,N;\ t = 1,\ldots,T$$

$$\sum_{i=1}^{N}\left[a_t\delta(x_{it}) + k_tX_{it}\right] - W_{it} - O_{it} \le 0 \qquad \forall\, t = 1,\ldots,T$$

$$W_t - W_{t-1} - H_t + F_t = 0 \qquad \forall\, t = 1,\ldots,T$$

$$-pW_t + O_t \le 0 \qquad \forall\ t = 1,\ldots,T$$

$$X_{it}, I_{it} \ge 0 \qquad \forall\, i = 1,\ldots,N;\ t = 1,\ldots,T$$

$$W_t, O_t, H_t, F_t \ge 0 \qquad \forall\, t = 1,\ldots,T$$

where X_{it} is the quantity of product i produced in period t, I_{it} is the number of units in ending inventory, s_{it} is the setup cost for product i in period t, $\delta(x_{it})$ equals one if product i is produced in period t, zero otherwise, W_t is the labor-hours of regular labor used in period t, H_t is the labor-hours of regular workforce hired in period t, h_t is the cost of hiring per labor-hour, F_t is the labor-hours of regular workforce laid-off in period t, f_t is the per hour cost of laying off, O_t is the labor-hours of overtime labor used during period t, o_t is the cost of overtime in period t, c_{it} is the one period inventory carrying cost, v_{it} is the unit production cost, k_i in the labor-hours required to produce a unit of product i, p is the fraction of regular hours allowed as overtime, a_i is the number of labor-hours consumed for a setup (Hax and Candea, 1984). It should be noted that this model is an extension of the linear cost model presented herein with the exception that backordering is not permitted.

The main advantage of fixed cost models is their inherent ability to incorporate lot sizing problems within the aggregate planning decision. Unfortunately, this addition of information detail requires both money and time to gather. To avoid this, Hax and Candea (1984) propose that a hierarchical production planning approach be employed[2.18].

2.6.4. General Cost Models

Hax and Candea (1984) divide general cost models into four categories. These are as follows:

[2.18] For a detailed discussion of hierarchical planning models see Hax and Meal (1975), and Bitran, Haas, and Hax (1981).

(1) Nonlinear analytic models, which provide a mathematical treatment of general nonlinear cost structures.

(2) Heuristic decision rules, which attempt to bring in the decision maker's intuition about the problem under consideration, by incorporating "rules-of-thumb" that contribute to the solution of the problem.

(3) Search decision rules, which consist of the application of hill-climbing techniques to the response surface defined by the nonlinear cost function and the problem constraints.

(4) Simulation decision rules, which represent the problem under consideration by a set of programmed instructions. The decision maker is able to test various approaches in an iterative fashion, where the outcome of each run suggests what the subsequent run might be. Simulation is particularly suitable to treat the uncertainties present in the decision-making process.

Because of functional form or the process under which they are executed, general cost models have the advantage of added realism. Further, because they more closely parallel the actual decision process, it may be easier to gain management acceptance of general cost models. However, general cost models suffer from three serious disadvantages. These are: (1) "...the models may be expensive to develop and run, and the computational procedures used to solve them seldom guarantee overall optimality," (2) a high level of aggregation may be required (as in the case of convex cost models), and (3) general cost models cannot handle large numbers of constraints (Hax and Candea, 1984).

2.6.5. Some Extensions of Production Planning Models

Damon and Schramm (1972) extended the HMMS rules to include marketing and finance decisions. The model has the objective of maximizing "...the cash equivalent position of the firm at the terminal point of its short-term planning horizon, where cash equivalence is defined as the sum of cash, marketable securities, accounts receivable, and inventories, less short-term debt" (Damon and Schramm, 1972). Decision variables specified by the model are: (1) work force levels, (2) average hours worked, (3) materials purchased, (4) advertising expenditures, (5) product price, (6) investment in marketable securities, and (7) short-term borrowing. From these values, production, materials and finished goods inventory levels, sales quantities and revenues, and cash holdings are determined.

Baker and Damon (1977) developed a model to simultaneously solve for the "....planning of production, inventory, workforce, and working capital levels over several periods." The model takes input from the firm's balance sheet, and solves for the optimal value of decision values through linear programming.

2.7. Existing Models to Simultaneously Solve Capacity and Production Planning Problems

Seven models that simultaneously solve the capacity and production planning problem were identified. These models vary in their implicit assumptions, objective function, number of products and processes they treat. Note that, prior to this research, **no** models existed that could simultaneously solve product, production, and capacity planning problems.

2.7.1. Multiple Period, Single Product Capital and Production Planning

Iglehart (1965) developed a model that enables a firm to make simultaneous periodic production and capital acquisition decisions. This model is applicable to firms producing a single product with identical capital equipment varying by age. Key assumptions in the model are that: (1) the production cost is convex, (2) the firm may hold inventory, (3) capital equipment is of identical type with varying ages, (4) the capital equipment has no fixed lifetime, (5) "...cumulative demand in successive periods...form a sequence of independent, identically distributed random variables, and (6) the fractions of depreciating capital in successive periods...form a sequence of independent, identically distributed random variables" (Iglehart, 1965). This model seeks to find the minimum "...of the expected present value of costs in n periods" (Iglehart, 1965).

By assuming that all capital equipment is identical, it is possible to represent production capacity by a single number in each period. Capital equipment levels are affected by two parameters: (1) how much equipment should be purchased in a period, and (2) depreciation of the capital equipment. Solutions to the single period and the n period problems are derived. Further, it is shown for the single period problem that there exist three unique possible outcomes and that for the n - period problem only one of these outcomes can occur per period. These three unique solutions are shown in Table 2.2 below. Regions indicate these three outcomes as three discrete areas on a two-dimensional graph where each these decisions could occur.

Some limitations of Iglehart's model are that: (1) all units of capital are of the same type, (2) the capital equipment has no fixed life length, (3) a single product is produced, (4) only the production phase of the product's life cycle is considered, and (5) the objective is to minimize undiscounted cost.

Table 2.2. Optimal Production and Capital Accumulation Policies for an *n* - period problem (Iglehart, 1965).

Region	Production Policy	Capital Accumulation Policy
I.	Produce to raise inventory to $\bar{y}_n(x_1)$	Order to raise capital to $\bar{z}_n(x_1)$
II.	Produce to raise inventory to $\bar{y}_n(x_1, x_2)$	Do not order capital
III.	Do not produce	Do not order capital

2.7.2. An Extension of the HMMS Rules

Sypkens (1967) extended the HMMS rules to include multiple period aggregate capacity for a production facility. As was noted earlier, the standard HMMS rules determine aggregate production P_t and workforce levels W_t for each period t of N periods.

Sypken's extension to the traditional HMMS rules was facilitated by adding an additional variable C where C_t is the plant production capacity in period t. Sypkens defined $CC_{t-1,t}$ to be the total cost associated with changes in production, workforce, and capacity levels from period t-1 to t. Thus, $CC_{t-1,t}$ is given by:

$$CC_{t-1,t} = CCW_{t-1,t} + CCP_{t-1,t} + CCC_{t-1,t}$$

where CCW is the "changeover costs connected with a change in W, CCP [is the] changeover costs connected with a change in P, [and] CCC [is the] changeover costs connected with a change in C" (Sypkens, 1967).

The total cost function for the plant becomes:

$$TC = \sum_{t=1}^{N} C_t + CC_{t-1,t} = \sum_{t=1}^{N} FC_t + VC_t + IC_t + CCW_{t-1,t} + CCP_{t-1,t} + CCC_{t-1,t}$$

where FC is the total fixed costs, and VC is the total variable costs.

Sypkens identified three mutually exclusive types of manufacturing, each with the following characteristics: (1) "...capacity can physically be divided in a large number of identical production units each able to perform the essential production operations itself...(e.g., a trucking company)," (2) a plant with "...one or a few large production units...(e.g., a paper mill)," and (3) "...a large number of production units each having a different function in the total production process...(e.g., a job shop)" (Sypkens, 1967). Sypkens noted that the extended model would provide the best results if characteristic one were prevalent. When characteristic three is prevalent, it may be possible to increase the

capacity of the machine with the lowest throughput. Thus, incremental capacity may be possible[2.19]. When characteristic two is prevalent, it is unlikely that the extended model will provide any substantial benefit. This is because any increase in capacity would be very large, and, consequently, very costly.

While Sypken's model has significant advantages over the HMMS rules from which it stems, it does have the same problems inherent to the HMMS rules. These problems are: (1) "the strong need for aggregation, (2) the elaborate estimation procedures required...to assess the numerical value of coefficients, and (3) the numerical difficulties encountered when the number of decision variables and constraints increase..." (Hax and Candea, 1984). In addition, while Sypken's model provides the optimal capacity for the plant, it was also noted that it is unlikely that this precise level of capacity could ever be obtained. This is because capacity is usually acquired in indivisible fixed-size blocks.

2.7.3. A Production Planning and Setup Cost Reduction Model

Billington (1987) developed a model that enables the determination of the economic production quantity (EPQ) for a single product while simultaneously considering investments in capital equipment to reduce setup costs. The EPQ model relies on the assumption that there exists "...a constant, deterministic demand for a single item at a single stage over an infinite horizon" (Billington, 1987). The objective is to minimize the cost of production, setup, annual capital equipment expenditures.

The model developed by Billington balances "...holding, setup, and capital expenses" (Billington, 1987). Two forms of this model were developed: (1) a model which assumes that setup costs vary linearly as a function of capital expense, and (2) a model which assumes that setup costs vary exponentially as a function of capital expense. It is shown that the minimum cost equation for the annual capital expenditure case is given by:

$$\min \quad TC = S\frac{D}{Q} + H\frac{Q}{2}\theta + K$$

where S is the setup cost, D is the demand for the product, Q is the production quantity per run, H is the holding cost, $\theta = \frac{P-D}{P}, 0 < \theta \leq 1$, and K is the additional amount spent each year for capital equipment. Thus, S will never be larger than some upper bound, but will be less as K increases.

[2.19] It should be noted that Sypkens model will not determine the machine with the lowest capacity. Thus, bottlenecks will have to be determined by other means.

Billington treats the relationship between K and S as being either linear or exponential. It is shown that for the exponential relationship between K and S, "once the setup cost reduction is justified,...partial investments up to the optimal K always reduces total cost" (Billington, 1987). In the case of a linear relationship, it is shown that "...when conditions for setup cost reduction are satisfied the optimal solution is to purchase the minimum setup cost technology possible" (Billington, 1987).

Some limitations of this model are that: (1) it treats a single product, (2) it is not life-cycle complete, and (3) the time value of money is not considered. However, treating a single product is conceptually simpler than simultaneously treating all products. Thus, organizations may more readily accept and adopt a model of this type rather than a more complicated multi-product model.

2.7.4. A Model to Balance Utilization and Machine Investment

Vander Veen and Jordan (1989) developed a model that simultaneously considers the "...trade-offs between machine investment and utilization." The model seeks to find the minimum of four costs. These are: (1) investment cost, (2) setup cost, (3) inventory cost, and (4) labor cost. Key assumptions are that: (1) "any part can be built on any machine, (2) each part type is built on one machine only, (3) each machine produces at a fixed rate (its capacity), (4) possible machine speeds vary continuously over a range bounded by the available technology, (5) all part types produced on a single machine are produced in fixed sequence from cycle to cycle, (6) setup times are the same for all part types and are independent of the sequence in which parts are built,...and (7) all part types have the same value and demand and that demand is constant and known with certainty" (Vander Veen and Jordan, 1989).

The model finds the minimum of $C = \sum_{i=1}^{M} P_i + S_i + I_i + W_i$ where P_i is the investment cost, S_i is the setup cost, I_i is the inventory cost, W_i is the labor cost, and M is the total number of machines to be purchased. A solution is found by sequentially solving two nonlinear programs. The first of the programs solves the machine level subproblem (MLSP) for a specific value of n_i (i.e., the number of part types to be built on a machine) by finding the minimum of $P_i + S_i + I_i + W_i$ subject to machine speed measured in parts/unit time and the lot size on machine i. The second program solves the plant level subproblem (PLSP) by minimizing C subject to M and $n_1 ... n_M$ ($n_i \geq 1$). Thus, the MLSP and PLSP are solved sequentially and for all possible values of n_i. The PLSP is solved for each MLSP until "...the optimal number of machines, M*, and the optimal part type allocations, n_i^* for i = 1 to M* [are found]" (Vander Veen and Jordan, 1989).

It should be noted that this planning model is not life-cycle complete. Thus, product planning is not considered. Further, the extension of the model by reducing the number of assumptions made may cause increased complexity in finding solutions. For example, assuming that different part types may be produced on more than one machine could substantially increase the solution possibilities.

2.7.5. A Multiple Objective, Fixed Time Length Model

Trzaskalik (1990) developed a multi-objective, multi-period planning model that enables the determination of the "...optimal schedule for a manufacturing plant in the fixed space of time in the future divided into a finite number of periods." Trzaskalik defines production technology to be "...a logical and usefully well-ordered set of operations, as a result of which we obtain a particular product." It is shown that the model relies on period specific activity condition vectors consisting of the possibilities of:

(1) Using production technologies
(2) Storage of materials and products
(3) Putting new production technologies into service
(4) Buying materials
(5) Utilization of labor resources
(6) Selling manufactured products
(7) Values of technological coefficients
(8) Prices of materials and products
(9) Costs per unit of production

to determine the optimal manufacturing schedule. Objectives that may be taken into consideration are:

(1) Maximization of profits
(2) Maximization of production size
(3) Maximization of production quality
(4) Minimization of utilization of deficit materials
(5) Minimization of costs

Assumptions in the model are that: (1) a definite time horizon exists, and discrete periods can be defined, (2) "it is possible to fix the set of different kinds of products which can be manufactured and the set of production technologies which can be used in the production process, (3) the manufacturing plant receives new production technologies, materials used in the production process and manpower from the environment and delivers manufactured products to the environment, and (4) production possibilities...are determined by an activity condition vector in each period" (Trzaskalik, 1990).

It should be noted that this model does not consider specific work centers. Rather, it considers the possibilities of the acquisition of various production sequences (i.e., production technology). Thus, bottlenecks caused by shared machines cannot be identified. Further, since only the production stage of the product life cycle is considered, this model is not life-cycle complete.

2.7.6. A Capacity Analysis and Aggregate Planning Model

Vercellis (1991) developed a decision support approach to help manage aggregate capacity. This decision support approach is designed "...to aid in planning manufacturing systems and evaluating alternative control policies." The decision support system integrates "...multiple conflicting objectives of the manufacturing strategy, by using different quantitative measures of productivity, service to customers and flexibility" (Vercellis, 1991). Vercellis' main contribution is that "...the model accepts multiple products, and several production stages, with multiple limited resources per each stage; allows to define planned lead times for production stages, and service times for products,...[either in the form] of finished end items or components." It is assumed that "...the system is operating in "stable" conditions, and that the demand requirements are varying in a "normal" range, within the planning horizon."

This decision support approach features multiple objectives. Decision variables are partitioned into one of five groups. These are:

(1) The capacity of multiple stages of the operating system, expressed in terms of standard and extra capacity of the resources, which may be local to a single stage or globally shared among various stages.

(2) The volume of production, either in house or released in subcontracting.

(3) The level of inventories and work in process for the finished products and for components, aggregated into families or similar items.

(4) The planned lead times for the multiple stages of the system.

(5) The service times, expressing the service level to the customer which has to be guaranteed for the different items produced.

Further, the production system consists of a set of aggregate production stages $S = \{s_1, s_2,...,s_n\}$, each of which is applicable to a specific family of items $f_i \ \forall \ i \in 1,2,...,n$. Resources for production are with respect to the four classes of labor that are available each period. These four classes are:

(1) A standard global amount g_{rt}, accessible and sharable in an arbitrary fractioned way among the different stages.

(2) A standard local amount v_{srt} for each aggregate stage s_i, whose consumption is exclusively reserved to stage s_i.

(3) A global extra-amount which can be made available upon planners request, up to a given upper limit g_{rt}, by incurring an extra-cost.

(4) A local extra-amount for each stage s_i, reserved to the stage s_i, which can be increased up to a maximum level u_{srt}, at an extra cost.

Finally, associated with each production stage s_i is a planned lead time l_s. The lead time represents "...a fixed delay imposed to each production order going through stage s_i, starting from the time of its release to the production stage" (Vercellis, 1991).

Using these variables, Vercellis' model schedules resources and production stages for multiple discrete length planning periods. The objective is to minimize total cost while guaranteeing a specified level of service.

Some limitations of Vercellis' model are that: (1) aggregate production capacity of stages is determined by part families, and (2) all families of items must correspond to a specific aggregate production stage. From 2, it is apparent that if a family of items must be produced at more than one stage, the family must be further decomposed. Also, aggregate capacity of the system is determined by labor levels rather than by variable machine capacity. Further, capacity and production is aggregated by the complete stage rather than by work center. Finally, the model is not life-cycle complete.

2.7.7. A Decision Model Linking Product Planning and Process Design

Kim et. al. (1992) developed two models to demonstrate the advantage of providing a simultaneous linkage between product planning and process design. Process design specifies how manufacturing tasks are to be achieved (i.e., the types of manufacturing technology to be employed). Results of the study "...show that a close integration between...[product planning and process design] helps control the product offerings, stabilize process requirements, improve process technology choices, and increase net cash flows over time" (Kim et al., 1992).

Kim's study was conducted by developing two models. Model One includes the complete linkage of product planning and process design. Model Two includes the sequential linkage between product planning and process design. Model One includes "...product substitution, entrance/exit strategies, competitive priorities,...[and the linkages between product plans and] the type, efficiency, and capacity of process technologies" (Kim et. al., 1992). Simulation runs of Model One and Two were compared to determine the effects of product and process integration. It was found that: (1) the linked model had generally higher net cash flows, (2) the unlinked model consistently offered more products,

(3) the unlinked models exhibits a higher acquisition cost and a higher disposal value, and (4) that as complexity increased, the linked model performed better than the unlinked model[2.20].

The advantages of the integrated model are that: (1) sufficient capacity to perform multiple processes must be available, (2) various technology offerings to perform the same process are studied, (3) the objective is to maximize the PW, and (4) interaction of products in the market place are included. However, the integrated model does have some disadvantages. First, the acquisition and abandonment of capacity is specified in machine hours per period. Since capacity is usually acquired in blocks of fixed size, it may not be feasible to acquire or abandon the amount of technology specified by the model. Second, the complete product life cycle is not considered. Thus, design, development, support, and phase-out costs are not considered in either model. Third, it is assumed that demand for products in specific periods, less loss of demand to other products, must be met. Fourth, neither model includes upper spending limits for the acquisition of production equipment, or minimum revenues that must be generated to cover expenses in each period. Finally, the effects that introduction and exit strategies have on product demand must be known with certainty. However, even given these disadvantages, the integrated model points to the need for more research to link product planning, production planning, and technology acquisition/abandonment.

2.8. Conclusions from the Literature

This section contains the conclusions made from a review of the relevant literature. As such, the section addresses constrained planning approaches, the decision domains for the concurrent and sequential models, capital budgeting, production planning, capital budgeting, production planning, and concurrent models.

2.8.1. Constrained Planning Approaches

Stone (1976) suggests that, to alleviate the problems caused by resource-constrained and market-constrained product planning approaches, the firm should adopt a "rational" approach. The rational approach suggests that the firm concentrate on the set of possible consequences from the relaxation of various constraints. In the case of the resource-constrained approach, Stone (1976) suggests that an approach be adopted that considers resources not to be constrained by current resources, but rather, allows for the "...consequences of relaxing [the] constraints." The alternative to the market-approach is an

---

[2.20] Complexity is affected by the number of products to choose from, variance in process requirements, short product life cycles, fluctuating demand for each product, and an increased number of process technologies available.

approach that attempts to identify possible markets that may be beneficial to the firm. Thus, a system of models that simultaneously determines the set of optimal capacity, optimal products, and optimal production quantities follows the "rational approach."

2.8.2. Decision Domain for Planning Models

The planning models developed in this research will take as input some results of strategic planning (e.g., budgets and other constraints, MARR, etc.) and solve the resource allocation and production planning problem at the management control level. However, it should be noted that the models do not clearly fall into the domain of management control. This is because early product life-cycle resource allocation decisions (e.g., investment in R&D) have a great impact on the organization's strategy. Because of this, the models also provide some strategic level decisions (e.g., the products that should be funded). Thus, the models will operate at both the strategic and managerial levels.

2.8.3. Capital Budgeting

The field of capital budgeting is not closed. Over the last ten years many significant contributions have been made. These contributions were either due to a need to incorporate non-economic considerations into capital budgeting decisions, or to accurately model the dynamic environment under which capital budgeting decisions are made. Extension of some of the developments made in the last ten years may allow for the connectivity of strategic, tactical, and operational decisions, thus, possibly providing improved decision making capabilities to the firm.

2.8.4. Production Planning Models

For the most part, production planning models rely on the assumption that long-term capacity is fixed, and thus, adjust regular and overtime labor levels to meet demand (Hax and Candea, 1985). Further, since production planning models seek to find the minimum cost, their must exist a constraint that demand should be met. Demand can be met either through backlogging, producing the exact amount in each period, building inventory, or through any combination of these three.

2.8.5. Concurrent Models

In most cases, the models that have been developed to concurrently make capacity and production planning decisions do not include: (1) specific work center capacity expansion opportunities, (2) specific capacity expansion proposals, (3) the complete product and supporting systems' life cycles, (4) maximization of the future wealth of the firm, (5) a dynamic, fiscal-based execution approach, and (6) partial demand fulfillment. Finally, in many cases capacity is deemed to vary by addition or subtraction of labor rather than by adjustments to long-term capacity and labor levels that the resulting capacity can support.

3. RESEARCH AND DEVELOPMENT METHODOLOGY

This chapter contains the methodology developed to demonstrate that the best product, production, and capacity plan may be obtained from a concurrent model with capital budgeting projects restricted to values of zero and one. The methodology is divided into a research methodology and a development methodology. The research methodology includes: (1) the problem statement, (2) research questions, (3) the uniqueness of the research, and (4) research contributions. The development methodology includes: (1) the comparison approach, (2) the assumptions made when developing the concurrent and sequential models, and (3) the limitations of the models.

3.1. Problem Statement

This research was conducted to demonstrate that the best product, production, and capacity plan may be obtained from a concurrent model. Two types of comparisons were developed to show this. First, it had to be shown that a concurrent model may provide better results than those obtained from a similar sequential model. Second, it had to be shown that each version of the concurrent model is applicable to a specific type of capital budgeting project constraint.

3.2. Research Question

A single research question is required to demonstrate that the best results may be obtained from a concurrent model. This is:

> *Will a life-cycle complete concurrent product, production, and capacity planning model provide solutions that are better than solutions that can be obtained from a similar life-cycle complete sequential model?*

Chapter 7 contains a comprehensive analysis of comparisons of all versions of the concurrent and sequential models developed to answer these two questions. Chapter 8 contains conclusions about the analysis.

3.3. Comparison Approach

Answers to this research question required that: (1) concurrent and sequential product, production, and capacity models be developed, (2) a hypothetical example be constructed, (3) five computer programs to test all versions of the two models be

developed, and (4) a detailed analysis of the results obtained from both models be conducted.

3.3.1. Development of the Concurrent Model

Figure 3.1 depicts the solution approach for the concurrent model. A complete mathematical formulation of the concurrent model that restricts capital budgeting project selection to values of zero and one is presented in Chapter 4. Two versions of the concurrent model that allow partial project selection are presented in Chapter 6.

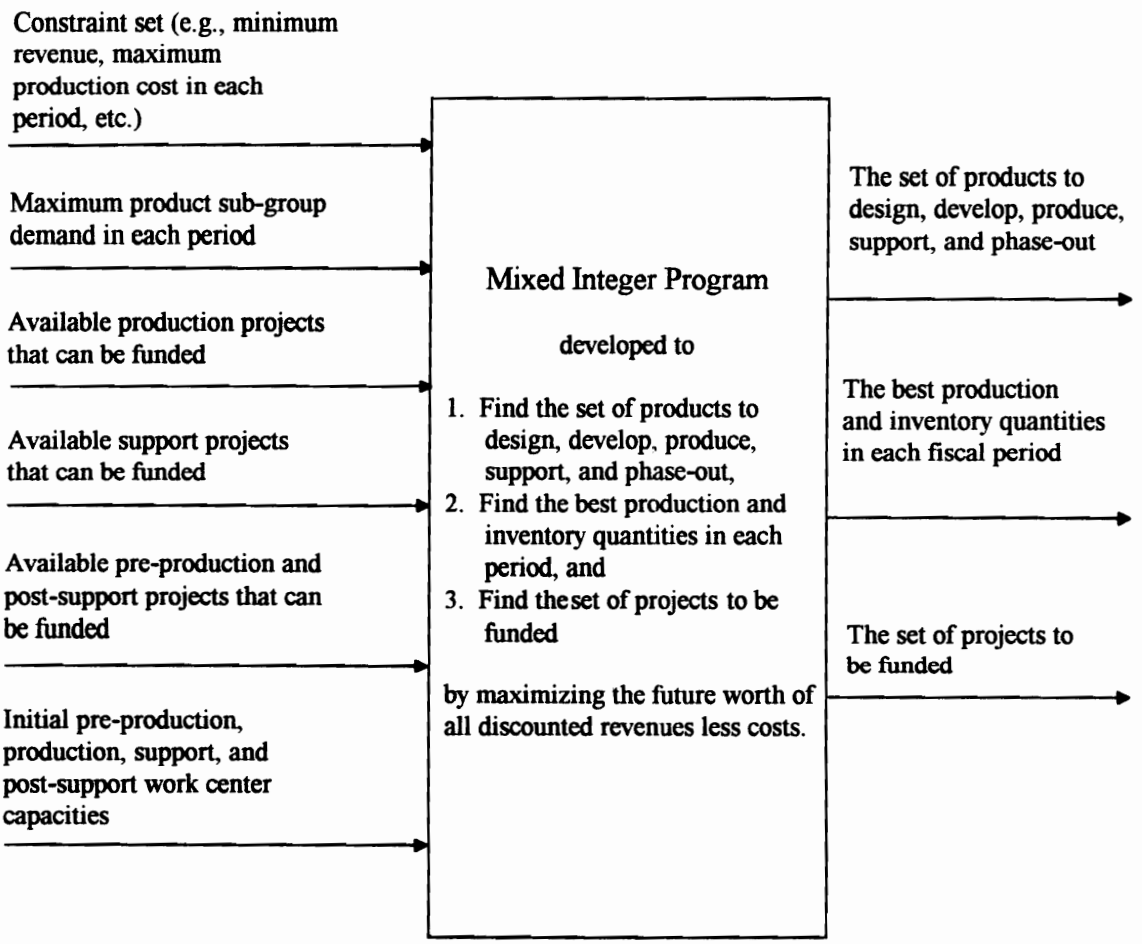

Figure 3.1. Solution Approach for the Concurrent Model.

The concurrent model makes three decisions simultaneously. These are: (1) what products should be produced?, (2) what quantities of these products should be produced

in each period?, and (3) what capacity must be acquired to enable the design, development, production, support, and phase-out of these products?

3.3.2. Development of the Sequential Model

Figure 3.2 depicts the solution approach for the sequential model. A complete mathematical formulation of the sequential model that restricts capital budgeting projects to values of zero and one is presented in Chapter 5. Versions of the sequential model that allow partial capital budgeting project selection are shown in Chapter 6.

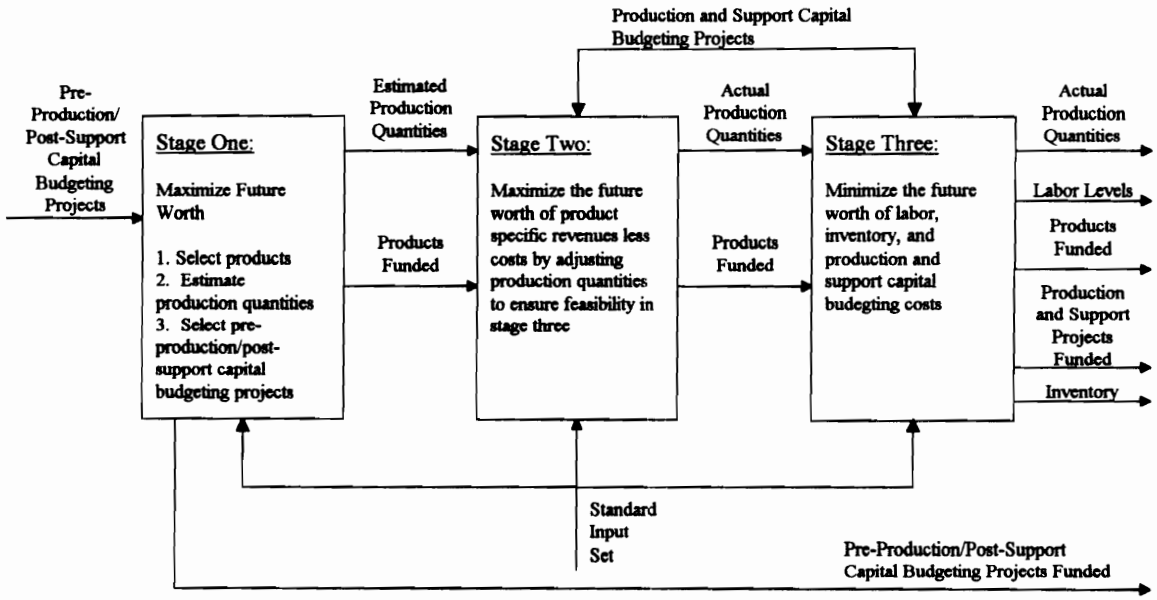

Figure 3.2. Solution Approach for the Sequential Model.

The sequential model solves product, production, and capacity decisions in a sequential fashion. In stage one, the model determines what products should be produced and those not to be produced. In addition, the model determines the best production quantities for each product by estimating production and support capacity cost to be equal to the cost of the existing capacity.

The first stage of the model maximizes the future worth of all revenues less costs, and considers capacity expansion to be unconstrained except for upper limits of project spending in each period. The output from the first stage is the set of products to fund and

those not to fund along with expected production quantities of each of the funded products in each period.

Since the production quantities emanating from stage one may prove to be infeasible when specific production and support capital budgeting projects are considered, stage two is added to ensure feasibility. Stage two maximizes the future worth of product specific revenues and costs and also considers the existing and additional capacity. Stage two may be viewed as a moderating component because it only exists to adjust the production quantities derived in stage one to ensure feasibility in stage three. Since the objective is to find the maximum production levels possible when specific capital budgeting projects are considered, no cost is imposed for selection of a capital budgeting project.

Stage three minimizes the future worth of the costs due to production, inventory, labor, and production and support capital budgeting projects. Production quantities are output from stage two and become input to stage three. Thus, since production quantities are known in advance, stage three functions much like a traditional production planning model.

3.3.3. Development of the Hypothetical Example

The hypothetical example (shown in Chapter 6) spans twelve periods, and is comprised of inputs for five products in two sub-groups, one pre-production/post-support capital budgeting project, three production/support capital budgeting projects and one support capital budgeting project. One product sub-group contains three products while the other sub-group contains the remaining two products. Products are produced for a maximum of eight periods. Two of the products were produced in the previous period. Because of this, the models consider the design, development, production, support, and phase-out life-cycle phases of three of the five products.

The demand for the two product sub-groups is considered to be uniform in each period. Further, the demand for each product sub-group was tested using all combinations of three levels. These levels are known as low, medium, and high. Because each product sub-group demand was varied at three levels, there exist $3^2 = 9$ **combinations** of product sub-group demand. These combinations of sub-group demand were varied in the input set to determine each models' sensitivity to future worth, project funding levels, production and inventory quantities, and labor levels.

3.3.4. Development of the Computer Programs

Comparison of the concurrent and sequential models was enabled through the development of five computer programs and interfaces with LINDO/386 (a DOS-based optimization program). Each model was run with the identical set of inputs. The results from each run of a model were subsequently compared.

Each run of a model was called a **test**. Tests were conducted through each version of the concurrent and sequential models. Since there were three versions of each model, and there were nine **combinations** of demand for each version, there were a total of $3x3^2 = 27$ **tests** run using each model. Thus, a total of $27x2 = 54$ **tests** (i.e., 27 tests on each of 2 models) were made.

A complete description of each of the five computer programs is included in Chapter 6. Listings for the computer programs are included in Appendices B through F.

3.3.5. Analysis Approach

The solutions obtained from the tests of the concurrent and sequential models were compared on the basis of the: (1) future worth of all revenues less costs, (2) capital budgeting project levels, (3) production and inventory quantities in each period, and (4) regular and overtime labor levels in each period. Next, the solutions obtained from the tests of each version of the concurrent model were compared on the basis of the: (1) future worth of all revenues less costs, and (2) capital budgeting project values. A complete analysis is provided in Chapter 7. Further, conclusions from the analysis are presented in Chapter 8.

3.4. Assumptions Made

Twenty-one assumptions were made while developing the concurrent and sequential models. These assumptions are applicable to both models. They are:

(1) The demand can be stated as the amount of products from a group of substitute products (i.e., product sub-group) that consumers are expected to purchase during a specific period[3.1]. That is, all products in a substitute product group serve the same purpose. For example, a product sub-group would be 3 different 1/2" drive impact wrenches produced by the same manufacturer. These 3 impact wrenches form a product sub-group because each impact wrench serves the same general

---

[3.1] A comprehensive definition of the demand constraint and substitute product groups are included under constraint definitions in Chapter 4 and Chapter 5.

purpose, while having slightly different features such as expected life, torque, SCFM requirements, etc. Accordingly, a consumer needing a 1/2" drive impact wrench would only purchase one product from the sub-group.

(2) All work center costs include overhead. This includes cost attributable to base capacity as well as capital outlays for capital budgeting projects. While the specific assignment of overhead is not explicitly considered herein, it is expected that the organization can assign overhead such that all fixed costs incurred (excluding direct labor costs) due to base capacity and increases in capacity are included in each work center's cost. Overhead could be allocated on the basis of direct labor or direct materials. However, since the production quantities are unknown prior to running the model, it is expected that overhead based on the maximum possible labor hours may yield the most accurate allocation of overhead costs.

(3) There exists a facility that engages in the production of finished products or assemblies on a definite number of work centers.

(4) Alternate resource consumption levels for a specific product at pre-production and post-support work centers are not included in either model. This means that products consume a single, specific known amount of resources at work centers that precede production and support activities, and also at work centers that enable product phase-out.

(5) Before a product can be produced or supported, it **must** consume a definite amount of resources at work centers that precede production work centers.

(6) If a product is funded at work centers that precede production and support, then it must also incur the costs due to phase-out activities.

(7) During production, the time spent on each specific product at each specific work center can be quantified.

(8) The sequence of operations has no impact on the accuracy of the models. Thus, sequencing of operations is not modeled.

(9) A primary production routing exists; secondary routings are not considered.

(10) The time spent on a specific operation will be determined by labor standards or similar measures. No consideration is given to the number of shifts and work center availability.

(11) A definite number of capital budgeting projects exist, and each project's ability to increase resources available (i.e., capacity) on a specific work center or a group of

specific work centers is known with certainty[3.2]. Further, the existing capacity at each work center is known with certainty.

(12) The organization will only evaluate capital budgeting projects at the beginning of each fiscal period.

(13) The demand for each product sub-group in a specific fiscal period is known with certainty (i.e., treated as deterministic).

(14) Product interaction (e.g. contingency and mutual exclusivity) is known with certainty (i.e., treated as deterministic).

(15) It is possible to separate production fixed costs (i.e., setup costs) from variable costs. That is, if a product is funded for production, it will incur a certain fixed cost at each production work center regardless of the quantity to be produced, in addition to a linear continuous variable cost.

(16) Revenues from products are considered to be deterministic and linear continuous.

(17) The organization chooses a MARR that is equal to the long run average return that the firm can receive from an average investment.

(18) The organization knows its budget and revenue limitations, and can accurately represent these as constraints in each fiscal period.

(19) All cash flows are discrete and occur at the end of a fiscal period.

(20) Products are differentiable. That is, products within **different** product sub-groups do not meet the same need. Therefore, there may exist several substitute product groups. For example, the same air tool manufacturer may simultaneously produce impact wrenches, air hammers, and air drills. Since these three product sub-groups do not meet the same need, they are differentiable.

(21) All products produced and placed in inventory in a fiscal period (t) must be sold in fiscal period $t + 1$. Thus, all products in beginning inventories (i.e., those products produced in fiscal period $t = -1$, the fiscal period just prior to analysis) will be sold in fiscal period $t = 0$.

[3.2] Note: A practical approach to capacity is adopted at each pre-production/post-support, production, and support work center. Because of this, the effects of work center reliability, maintainability, and planned maintenance are considered in capacity increases. Further, holidays and the feasible hours of operation at each work center are accounted for in the capacity measure.

3.5. Uniqueness of This Research

All versions of the concurrent model are unique in seven ways. Because of this, the concurrent models are original, and as such, make a significant contribution in the areas of product, production, and capacity planning. In addition to this, these models help show the linkages between three planning areas that were, for the most part, treated separately in past research.

The seven differences from models developed in the past are:

(1) The objective of the concurrent model is to maximize the future wealth of the organization at some point in the future. This is done by choosing the longest-lived product, and assuming the reinvestment of the residual worth of items at the end of their life at the MARR.

(2) Capacity of work centers may be increased through the addition of long-term capacity. Long-term capacity is defined to be the acquisition of production and early and late life-cycle resources (e.g., production machines, research and development equipment, salvage equipment). This is in contrast to most models developed to date that treat capacity as fixed, and adjust production regular and overtime labor levels to meet product demand in each fiscal period (Hax and Candea, 1984). Thus, in these existing models, the capital budgeting decision is completed before or after, but not simultaneous with production planning.

(3) Models developed to date do not consider the complete product life cycle. Thus, these models only consider products that are currently being produced. In contrast, this research considers the resources of the facility during the entire product life cycle of all products evaluated. Thus products that are, currently being produced, that are in the planning and design stages, and that may be phased out are considered along with the potential for investment of available capital to increase the facility's capacity in each of the six product life-cycle phases.

(4) Many models assume that demand must be met. This is because the objective of these models is to minimize undiscounted total cost. Hence, not meeting demand would drive the solution to a minimum cost of zero and a production quantity of zero. The need to meet demand is eliminated in the concurrent model because the objective is to maximize future wealth. Thus, production quantities may range from zero to the product sub-group demand level plus product inventory level over several periods.

(5) The models developed through this research treat capital budgeting projects whose costs and potential to increase capacity are known with certainty. Models developed to date either determine: (1) the aggregate level of capacity that should be added to the plant or a work center, or (2) an appropriate level of a specific type of manufacturing technology for the facility. Thus, while previous models only indicate an area where capacity expansion may be beneficial, the concurrent model will select the best set of capital budgeting projects from a known set.

(6) The concurrent model enables the linkage between up-front activities (e.g., R&D) and production. The inherent sequential dependency in the various stages of each product's life-cycle is explicitly modeled as sets of constraints. Because of this, the concurrent model accounts for all resources consumed to design, develop, produce, support, and phase-out a product on all pre-production, production, support, and post-support work centers in all fiscal periods.

(7) The concurrent model enables the concurrency of up-front activities by incorporating the ability to specify that resources from multiple work centers in the same period may be consumed. An example of this is that the concurrent model can include a constraint specifying that known levels of resources from engineering, logistics, testing, and production are simultaneously required to fund a product.

3.6. Research Contributions

This research makes thirteen significant contributions. These are:

(1) The main contribution from the research is that three versions of a concurrent product, production, and capacity planning model are developed that simultaneously solves product, production and capital budgeting problems when capital budgeting projects are known in advance. As was noted previously, the concurrent model developed during this research is different from any existing models both in its scope and underlying assumptions. The two most similar existing models to the concurrent model are the linked models developed by Kim et al. (1992) and Sypkens (1967). The advantages of the concurrent model are that: (1) the Kim et. al. model and the Sypkens' model do not consider specific pre-production and post-support work centers, (2) the Kim et al. model and the Sypkens' model do not include consideration of product support. Further,

key assumptions made by the Kim et al. and Sypkens' models are that: (1) no known capital budgeting projects exist, (2) there are no upper bounds on capital expenditures in periods, and (3) there are no lower bounds on revenues in periods. It is shown herein that models based on the first assumption may yield infeasible solutions and overstated future worths when known capital budgeting opportunities are introduced. Further, models that rely on the second assumption may provide infeasible solutions in the face of capital budget limitations. Models that rely on the third assumption may fail to produce the revenues required by the firm to operate at the specified level. Thus, because of scope and additional information, the concurrent model developed during this research is more comprehensive than other models that already existed.

(2) A second contribution is that the concurrent model includes the simultaneous consideration of all six product life-cycle phases. Thus, the results obtained from this model are likely to be better than if only the production or only the production and support life-cycle phases were considered.

(3) The concurrent model includes upper limits on capital expenditures on capital budgeting projects in each fiscal period. This is important because similar models developed in the past assume that no limits on capital budgeting project expenses exist in any fiscal period.

(4) The concurrent model simultaneously considers the production and the support of products. This is important because the cost of supporting a product may outweigh its net income due to new product sales. In addition, the concurrent model supports the mathematical modeling of revenues and costs generated following the warranty. That is, revenues from repairs after a product's warranty has elapsed and before its life is complete are considered.

(5) The concurrent model includes a provision for the cost of product phase out. This is important because costs due to product phase out may be extremely high. Thus, models that do not include a provision for the cost of phase out may not generate the best solution.

(6) Since the objective is to maximize future worth, it is not necessary to specify that demand for each product sub-group must be met in each

period. This is important because minimum cost models assume that demand must be met. Hence, because the objective of the concurrent model is to maximize future wealth (with adjustments in capacity), this model may find a solution where demand is **not met** that provides higher future worth than if demand were met.

(7) The concurrent model varies production and support work center capacity, and also sets the sum of regular and overtime labor at these work centers. Thus, overall product and support capacity and their corresponding labor levels are treated simultaneously.

(8) The concurrent model considers specific work centers. Thus, traditional fixed cost models are extended in the concurrent models to consider the capacity of each work center within the plant rather than just the aggregate capacity of the plant[3.3]. Because of this, the concurrent model will increase total production capacity by increasing the capacity at the work center with the least slack production time.

(9) By simultaneously considering all six product life-cycle phases and the specific product resource requirements in each fiscal period, the concurrent models enable the mathematical modeling of the **sequential dependency** that exists between up-front (e.g., design, development, and R&D), production, support, and late life-cycle activities (e.g., phase-out). This is important because the concurrent models consider the organization's resources simultaneously resulting in only the set of products that will maximize the organization's future wealth being funded.

(10) Because the concurrent model treats work center capacity rather than aggregate capacity, the **concurrency between activities** is modeled mathematically. Thus, if a product simultaneously requires support from design, logistics, and manufacturing work centers in the same period, then sufficient capacity must exist at all these work centers to support all

---

3.3 Note that Sven Axsäter, *On the Design of the Aggregate Model in a Hierarchical Production Planning System*, **Engineering and Process Economics**, Vol. 4 (1979), pp. 89-97, used a similar approach applied to aggregate production planning models.

required activities. If adequate capacity is not available at each of these work centers, the product will not be funded. Because of this, organizational concurrency is supported.

(11) The concurrent model provides a provision to mathematically describe four types of contingencies that may exist between products in different substitute product groups.

(12) The concurrent model forms a framework that may be used to drive further research. For example, the components that drive the cost and revenue streams may be made explicit, and then included in the concurrent model developed during this research.

(13) The sequential model that was developed and is presented herein requires an additional stage and also incorporates more constraints than the Kim et al. sequential model. These additional constraints fall into two broad categories. First, the sequential model incorporates implied constraints in the form of 0,1 decision variables. Second, the sequential model incorporates explicit constraints in the form of maximum allowable expenditures on capital budgeting projects, and minimum acceptable revenue in each period. Finally, because of the implied and explicit constraints, the sequential model requires three stages rather than two as in the Kim et al. model.

3.7. Limitations of The Models

The concurrent model has four types of limitations. These are: (1) obsolescence, (2) problems associated with pure capital rationing, (3) solution complexity, and (4) data requirements.

3.7.1 Technological Obsolescence

A limitation of the concurrent model is that, as the planning horizon becomes longer, the organization may lack sufficient knowledge to correctly assess the state of technology far in the future. It is obvious that this is a deficiency since the state of technology will affect the competitiveness of both product offerings and capital equipment. However, as the planning horizon becomes shorter, this knowledge deficiency becomes less pronounced and hence, may disappear completely. In any case, production planning and capital budgeting models commonly used today do not address technological obsolescence. Poor results due to technological obsolescence are partially overcome

because the concurrent model: (1) is based on a quasi rolling-horizon approach (i.e., longest lived product), and (2) should be run in each fiscal period.

3.7.2 Problems Associated with Pure Capital Rationing Models

3.7.2.1 Violation of the Assumption of Unlimited Borrowing and Lending

Since this research relies on a horizon model with assumptions similar to pure capital rationing models, the assumption (when using an external discount rate) that both lending and borrowing activities are unlimited is violated (Park and Sharp-Bette, 1990). That is, the model relies on implicitly stated budget constraints while the future worth is determined by an "...externally determined discount rate..." (Park and Sharp-Bette, 1990).

3.7.2.2 No Provision for Outside Investment or Dividend Payments

While there exists no provision for outside investment (i.e., expenditure of unused capital) or for dividend payments, it is assumed that the capital allocation that maximizes the future worth of the organization will also maximize stockholder wealth. In addition, it is assumed that the organization will invest excess capital at the MARR in projects that may be external to the organization, thus insuring that ample funds will be available in subsequent periods. Further, it is assumed that the organization carefully manages capital through planning activities to insure that projected budgets are accurate and available.

3.7.2.3 No Provision for Carryover of Unused Funds

The problem of carryover of excess funds is actually an extension of the previous problem in that all available capital should be either utilized in a fiscal period or made available for consumption in a subsequent fiscal period. However, capital that is not used in a fiscal period should earn a return at the MARR until it is consumed. Again, it is assumed that the organization relies on careful planning, and that unused capital is invested at the MARR either inside or outside the firm.

3.7.3 Solution Complexity

There may exist concern as to whether a real-world problem employing the linear mixed-integer programming (MIP) method used in the concurrent model could be solved. In the case of a small organization producing few products, it is likely that a solution could be readily obtained. In the case of a large organization producing many products, it may be possible to extend the concurrent model to include aggregate production (i.e., use item types and families) and, at the same time, aggregate capital budgeting proposals at the same type/family level. Aggregation may increase the speed with which solutions are found.

3.7.4. Data Requirements

A problem that may be encountered when running the concurrent model is that a large amount of data is required. The hypothetical example presented in Chapter 6 requires 873 inputs and spans a 12 period planning horizon. These inputs are for five capital budgeting projects and five products.

4. THE CONCURRENT MODEL

In this chapter, a mathematical model to enable concurrent product, production, and capacity planning while possibly meeting demand for a group of substitute products is formulated. This model has been developed to: (1) possibly meet consumer demand within a group of substitute products, and (2) simultaneously consider all six life-cycle phases. The six life-cycle phases considered in this concurrent model are: (1) conceptual design, (2) preliminary design, (3) detailed design and development, (4) production, (5) product use and support, and (6) product phase out.

4.1. Subscripts and Objective Function Components

This section contains a description of the subscripts, and objective function components used in the formulation of the concurrent model. The objective function has been divided into several components to make clear the life-cycle phases, and cost and revenue inputs addressed by the model.

4.1.1. Subscripts

k = A group of products that serve the same purpose, i.e., known as substitute products.

K = The number of substitute product groups.

jk = A specific product j within the substitute product group k.

J_k = The number of products j in a specific subgroup k.

l = A specific work center used only in the **production** of products. Work centers with subscripts l are referred to as production work centers.

L = The number of production work centers.

n = A specific work center used only in the non-production and non-utilization phases of the product life cycle, i.e., applicable to conceptual design, preliminary design, detailed design and development, and phase out. Work centers with subscripts n are referred to as pre-production and post-support work centers.

N = The number of non-production work centers.

t = A specific time period.

T = The number of periods considered through the horizon point.

v = Refers to the **one** support work center used to **support** products. The work center with subscript v is referred to herein as the support work center.

m_l = A capital budgeting project that affects a production work center l.

M_L = The number of production capital budgeting projects considered.

m_n = A capital budgeting project that affects non-production work centers n.

M_N = The number of pre-production/post-support capital budgeting projects considered.

m_v = A capital budgeting project that affects the support work center v.

M_V = The number of support capital budgeting projects considered.

T_e = The last period that production occurs.

T_s = The last period that support occurs.

4.1.2. Objective Function Components

$FWNP_T$ = The future worth of revenues from the sales of new products and products held in beginning inventory.

FWI_T = The future worth of inventory holding costs for all products, jk, and all periods, t.

FWL_T = The future worth of labor production costs for all periods, t, and all products, jk.

FWP_T = The future worth of the cost of base production capacity for all periods, t.

$FWCPS_T$ = The future worth of funded pre-production/post-support capital budgeting projects, $e_{m_n} = 1$, for all periods, t.

$FWCP_T$ = The future worth of funded production capital budgeting projects, $e_{m_l} = 1$, for all periods, t.

$FWCS_T$ = The future worth of funded support capital budgeting projects, $e_{m_v} = 1$, for all periods, t.

FWR_T = The future worth of revenues received for labor to repair all products, jk, of age $\tau = PW_{jk}+1, PW_{jk}+2, \ldots, PL_{jk}$ in all periods $t = 0,1,\ldots,T$.

$FWLC_T$ = The future worth of the costs for labor to repair all products, jk, of age $\tau = 1,2,\ldots, PL_{jk}$ in all periods $t = 0,1,\ldots,T$.

$FWPR_T$ = The future worth of the revenues less costs for parts to repair all products of age $\tau = 1,2,\ldots, PL_{jk}$ in all periods $t = 0,1,\ldots,T$.

FWS_T = The future worth of the base capacity of the support work center for all periods $t = 0,1,\ldots,T$.

FWN_T = The future worth of the base capacity of all pre-production and post-support work centers for all periods $t = 0,1,\ldots,T$.

$FWMA_T$ = The future worth of all material cost accrued during production.

4.2. Decision Variables

This concurrent model has nine classes of decision variables. Descriptions of these are as follows:

x_{jkt} = The amount of product j in substitute product group k to be produced in period t. Production may not commence before period $t_{b_{jk}}$ and must cease in period $t_{e_{jk}}$.

i_{jkt} = Ending inventory of product j in substitute product group k at time t. Solved as a real rather than integer variable. Inventory cannot be held prior to period $t_{b_{jk}}$ and must be completely liquidated by the end of period $t_{e_{jk}} - 1$.

$\delta(x_{jkt})$ = Setup decision in period t for product j in substitute product group k. Solved as a 0,1 decision variable. Only one setup can occur per period; hence, production of product jk can only occur once per period. This assumption is consistent with standard production planning modeling practice.

e_{m_l} = A zero-one decision variable that enables selection or rejection of production capital budgeting proposal e_{m_l}.

e_{m_n} = A zero-one decision variable that enables selection or rejection of pre-production/post-support capital budgeting proposal e_{m_n}.

e_{m_v} = A zero-one decision variable that enables selection or rejection of support capital budgeting proposal e_{m_v}.

w_{lt} = Regular labor hours consumed at work center l in period t. Solved as a LP rather than MIP variable. Since w_{lt} is given in hours, it is conceivable that its values do not have to be restricted to integer values.

o_{lt} = Overtime hours consumed at work center l in period t. Since o_{lt} is given in hours, it is conceivable that its values do not have to be restricted to integer values.

z_{jk} = Funding variable of product j in substitute product group k for pre and post production activities (excluding support). Solved as a 0,1 decision variable.

4.3. Model Inputs

The concurrent model requires that the values for 42 types of inputs be known in advance. Descriptions of these inputs are as follows:

I_{jkt} = Inventory holding cost in actual dollars for product j in substitute product group k for period $t = t_{b_{jk}}, \ldots, t_{e_{jk}} - 1$.

d_{kt} = The total demand for all products in subgroup k in period t.

$t_{b_{jk}}$ = The first period that product j in substitute product group k may be produced.

$t_{e_{jk}}$ = The last period that product j in substitute product group k may be produced.

ACC = The long-run average cost of capital with an inflation component.

$\overline{f}$ = Average inflation rate through the horizon.

a_{jkl} = Setup time in hours for product j in substitute product group k at work center l.

h_{jkl} = The time required in hours at work center l to produce one unit of product j in substitute product group k.

$C_{m_l t}$ = The increase in hours in period $t = 1,2,\ldots,T$ at production work centers $l = 1,2,\ldots,L$ if capital budgeting project e_{m_l} is selected.

B_l = The beginning hours available at production work center l in period zero. Note that work centers with subscripts l are production work centers.

B_n = The beginning capacity in some consistent units available at work center n. Note that work centers with subscripts n are pre-production and post-support work centers. Also, note that this MIP has been developed such that each pre-production and post-support work center n consumes only one type of resource. However, the resource types consumed at all pre-production and post-support work centers do not have to be identical. For example, the resources consumed at work center $n = 1$ may be engineering labor hours while the resources consumed at work center $n = 2$ may be computer time.

$C_{m_n t}$ = The increase in some consistent units in period t at pre-production and support work centers $n = 1, 2,\ldots, N$ if capital budgeting project e_{m_n} is selected.

$C_{m_v t}$ = The increase in hours in period t for support if capital budgeting project e_{m_v} is selected.

$c_{m_l t}$ = The cost of production capital budgeting project e_{m_l} in period t. Includes all costs to due to increased capacity of production work centers from capital budgeting project e_{m_l} excluding labor.

$c_{m_n t}$ = The cost of pre-production/post-support capital budgeting project e_{m_n} in period t. Includes all costs to due to increased capacity of pre-production/post-support work centers from capital budgeting project e_{m_n} excluding labor.

$c_{m_v t}$ = The cost of support capital budgeting project e_{m_v} in period t. Includes all costs to due to increased capacity of support work center from capital budgeting project e_{m_v} excluding labor.

E_{t_l} = The maximum allowable that may be spent on production capital budgeting projects in period t.

E_{t_n} = The maximum allowable that may be spent on pre-production/post-support capital budgeting projects in period t.

E_{t_v} = The maximum allowable that may be spent on support capital budgeting projects in period t.

λ_{lt} = Fraction of total hours available at production work center l in period t that may be charged at regular hourly labor rates.

$S_{jk\tau_{jk}}$ = Support hours required for one unit of product j of age $\tau = 1,2,\ldots,PL_{jk}$ in substitute product group k.

S_0 = Support hours available at the beginning of period zero (i.e., base support hours).

PC_t = The maximum allowable cost of production in period $t = 0,1,\ldots,T_e$ including materials and labor.

PL_{jk} = Life length in periods for one unit of product j in substitute product group k.

PW_{jk} = Warranty length in periods for one unit of product j in substitute product group k.

$p_{jk\tau_{jk}}$ = Average cost in constant dollars of parts to repair one unit of product j of age $\tau = 1,2,\ldots,PL_{jk}$ in substitute product group k where $\tau_{jk} \in 1,2,\ldots,PL_{jk}$.

P_t = Maximum repair parts cost in period $t = 0,1,\ldots,T_s$.

$PR_{jk\tau_{jk}}$ = Average revenue in constant dollars from parts to repair one unit of product j in substitute product group k of age τ_{jk} where $\tau_{jk} \in 1,2,\ldots,PL_{jk}$.

r_{jkt} = Revenue in actual dollars from the sale of one unit of product j in substitute product group k in period $t = 0,1,\ldots,T_s$.

R_t = Minimum acceptable total revenue in period $t = 0,1,\ldots,T_s$ - includes sales of new products and repairs.

f_t = Average revenue in actual dollars per hour of repair labor in period t.

W_{lt} = Cost of regular labor in actual dollars per hour in period t at production work center l.

O_{lt} = Cost of overtime labor in actual dollars per hour in period t at production work center l. Note that it is assumed that $O_{lt} > W_{lt}$.

b_l = Base cost to operate work center l (excluding production labor) at beginning capacity B_l. Note that products funded (i.e., Z_{jk} variable equal to one) may consume resources at production work center l. The cost due to this consumption of resources (excluding production labor) appears both in b_l and $c_{m_l t}$ parameters that are funded (i.e., $e_{m_l} = 1$). Because of this, it is not necessary to directly specify product cost at each work center l.

b_n = Base cost to operate pre-production and post-support work center n at beginning capacity B_n. Note that products funded (i.e., Z_{jk} variable equal to one) may consume resources at work center n. The cost due to this consumption of resources appears in both the b_n and $c_{m_n t}$ parameters that are funded (i.e., $e_{m_n} = 1$). Because of this, it is not necessary to directly specify pre-production and post-support product cost at each work center n.

s_0 = Base cost to operate support facilities. Note that products funded (i.e., Z_{jk} variable equal to one) may consume support resources. The cost due to this consumption of resources appears in both the s_0 and $c_{m,t}$ parameters that are funded (i.e., e_m = 1). Because of this, it is not necessary to directly specify product support cost.

MA_{jkt} = The total cost of materials in actual dollars (i.e., raw materials, components, and assemblies) required to produce one unit of product j in substitute product group k in period t.

RC_t = Repair labor cost in actual dollars per hour in period t.

bin_{jk} = The beginning inventory of product j in substitute product group k in period 0.

g_{jknt} = Consistent units (i.e., all units within a work center n must be the same) consumed at pre-production and post-support work center n in period t by product j in substitute product group k.

m_{kt} = Some multiple of product sub-group demand, d_{kt}, used to define the maximum allowable production quantity, x_{jkt}, of product jk in each period t. Enables the capacity to carry inventories of each product, jk.

U = Upper bounds used in various constraints to either: (1) provide linear rather than non-linear solutions methods, or (2) constrain real variables by the value of a zero-one variable.

y = A binary selection variable is used to enable linear rather than nonlinear solutions.

4.4. The Concurrent Model's Life-Cycle Approach

At this point, it may be useful to show, in tabular form, where the decision variables and defined parameters lie with respect to specific products and the facility. This relationship is shown in Table 4.1[4.1]. It is important to recognize that the organization will simultaneously incur costs attributable to products and costs attributable to work centers. Revenues are attributable only to products. Thus, while the concurrent model works only with product specific revenues, costs are attributable to products and work centers. The reason for this is that the work centers used throughout the product life-cycle phases are shared by many products, and, thus, costs at work centers are allocated to many products. However, the model is solved at the work center total cost level, rather than product specific cost level. It should be noted though that after an optimal solution is obtained, it is possible to determine the exact cost per product at each work center. However, the determination of product specific cost at each work center is not necessary to determine the optimal solution.

4.5. Objective Function Formulation

The objective function is presented with thirteen components. The objective is to maximize the future worth of revenues from sales and product support less the future worth of the costs of production, inventory holding (including stocking fee and lost interest on sale of a product produced in a period), support, and pre-production and post-support activities. Table 4.2 shows all revenue and cost components accounted for in the concurrent model.

4.1 The life-cycle phases listed in Table 4.1 follow the six phases discussed previously. These are: (1) conceptual design, (2) preliminary design, (3) detailed design and development, (4) production, (5) product use and support, and (6) product phase out.

Table 4.1. Relationship Between Product Decision Variables, Requirements, and Plant Capacity.

Life-Cycle Phase	1	2	3	4	5	6
Product Specific Decision Variables	z_{jk}	z_{jk}	z_{jk}	x_{jkt}, i_{jkt}, $\delta(x_{jkt})$	x_{jkt}	z_{jk}
Product Requirement Inputs	g_{jknt}	g_{jknt}	g_{jknt}	a_{jkl}, h_{jkl}, bin_{jk}	S_{jkt}	g_{jknt}
Product Life Inputs				$t_{b_{jk}}$, $t_{e_{jk}}$	PL_{jk}, PW_{jk}	
Product Specific Cost Inputs				MA_{jkt}, I_{jkt}	$p_{jk\tau_{jk}}$	
Product Specific Revenue Inputs				r_{jkt}	$PR_{jk\tau_{jk}}$	
Plant Level Decision Variable	e_{m_n}	e_{m_n}	e_{m_n}	e_{m_l}, w_{lt}, o_{lt}	e_{m_v}	e_{m_n}
Plant Level Capacity Constraints	B_n, $C_{m_n t}$	B_n, $C_{m_n t}$	B_n, $C_{m_n t}$	B_l, λ_{lt}, $C_{m_l t}$	S_0, $C_{m_v t}$	B_n $C_{m_n t}$
Plant Level Revenue Inputs				R_t	f_t	
Plant Level Cost Inputs	b_n, E_{t_n}, $c_{m_n t}$	b_n, E_{t_n}, $c_{m_n t}$	b_n, E_{t_n}, $c_{m_n t}$	b_l, W_{lt}, O_{lt}, E_{t_l}, PC_t, $c_{m_l t}$	RC_t, s_0, E_{t_v}, P_t, $c_{m_v t}$	b_n, E_{t_n}, $c_{m_n t}$
External and General Parameters	ACC, $\overline{f}$	ACC, $\overline{f}$	ACC, $\overline{f}$	d_{kt}, ACC, $\overline{f}$	ACC, $\overline{f}$	ACC, $\overline{f}$

Table 4.2. Revenue and Cost Components for the Concurrent Model.

Revenue Components	Cost Components
1. The future worth of revenues from the sales of new products - $FWNP_T$. 2. The future worth of revenues from labor to repair defective, out of warranty products - FWR_T.	1. The future worth of inventory holding cost - FWI_T. 2. The future worth of labor production costs - FWL_T. 3. The future worth of all production work center base capacity costs - FWP_T. 4. The future worth of the total cost to enact all funded pre-production/post-support capital budgeting projects - $FWCPS_T$. 5. The future worth of the total cost to enact all funded production capital budgeting projects - $FWCP_T$. 6. The future worth of the total cost to enact all funded support capital budgeting projects - $FWCS_T$. 7. The future worth of labor repair costs - $FWLC_T$. 8. The future worth of base support costs - FWS_T. 9. The future worth of base pre-production and post-support work center costs - FWN_T. 10. The future worth of all production material costs - $FWMA_T$.
1. The future worth of parts repair revenues less costs[4.2] - $FWPR_T$.	

[4.2] The future worth of parts repair revenues less costs is both a revenue and cost component.

The objective of the concurrent model is to

$$\text{maximize} \begin{Bmatrix} FWNP_T + FWR_T + FWPR_T - FWI_T - FWL_T - FWP_T - FWCPS_T \\ -FWCP_T - FWCS_T - FWLC_T - FWS_T - FWN_T - FWMA_T \end{Bmatrix} \quad (4.1)$$

where $FWNP_T$, FWI_T, FWL_T, FWP_T, $FWCPS_T$, $FWCP_T$, $FWCS_T$, FWR_T, $FWLC_T$, $FWPR_T$, FWS_T, FWN_T, and $FWMA_T$ are given by Equations (4.2), (4.3), (4.4), (4.5), (4.6), (4.7), (4.8), (4.9), (4.10), (4.11), (4.12), (4.13), and (4.14) respectively while subject to the constraints given by Equations (4.15) through (4.56).

4.5.1. Future Worth of Revenues from Sales of New Products

The future worth of revenues $FWNP_T$ from the sales of new products is given by Equation (4.2).

$$FWNP_T = \sum_{k=1}^{K} \sum_{j=1}^{J_k} \left(\left(\sum_{t=t_{b_{jk}}}^{t_{e_{jk}}} r_{jkt} x_{jkt} \right) (1+ACC)^{T-t} + r_{jk0} bin_{jk} (1+ACC)^{T} \right) \quad (4.2)$$

The production quantity decision variables are x_{jkt}. The revenues for a specific product in a specific period are known and represented by r_{jkt}. Further, each product in beginning inventory, bin_{jk}, will contribute period $t = 0$ revenues, r_{jk0}. Thus, all revenues from the products held in beginning inventory, and the products produced and sold during the periods of analysis are considered in this equation. Note that the period t is defined in the equation to start at $t_{b_{jk}}$ and end at $t_{e_{jk}}$. This is because production of a product jk cannot occur before, $t_{b_{jk}}$, the earliest period it may be produced, and must cease at the end of period, $t_{e_{jk}}$, the latest period it may be produced. Thus, the problem is simplified by not considering periods in which a specific product jk cannot be produced. This has the effect of reducing the number of coefficients required to solve this component of the objective function.

4.5.2. Future Worth of Inventory Holding Costs

Inventory holding capability has been included in the concurrent model to make possible to achieve a balance between high demand periods (and the consequent inability to produce all required products for that period in that period) and the cost of producing

in prior periods to meet the demand in future periods. It is assumed that all units of *jk* must be removed from inventory by the end of period $t_{e_{jk}}$, thus ensuring that the production and utilization phases of the product life cycle may be separated.

The inventory holding cost component FWI_T of the objective function (4.3) determines the cost of holding units of products *jk* in inventory to meet demands for these products in the following period. Inventory holding costs include: (1) a stocking fee, $I_{jkt,}$, (2) the future worth of the interest lost from not selling a product in the period it was produced, and (3) the adjustment of support revenues due to support occurring in the periods following the period in which the product was produced. Adjustment of support revenues are given by Equations (4.9), (4.10), and (4.11).

$$FWI_T = \sum_{k=1}^{K}\sum_{j=1}^{J_k}\sum_{t=t_{b_{jk}}}^{t_{e_{jk}}-1}\left[\begin{array}{l}\left(\left(I_{jkt}\right)\left(i_{jkt}\right)+\left(r_{jkt}\right)\left(i_{jkt}\right)\right)(1+ACC)^{T-t} \\ -\left(\left(r_{jkt+1}\right)\left(i_{jkt}\right)\right)(1+ACC)^{T-t-1}\end{array}\right] \tag{4.3}$$

Since FWI_T appears as a negative coefficient in the objective function, adding $\sum_{k=1}^{K}\sum_{j=1}^{J_k}\sum_{t=t_{b_{jk}}}^{t_{e_{jk}}-1}\left(r_{jkt}\, i_{jkt}\right)(1+ACC)^{T-t}$ while subtracting $\sum_{k=1}^{K}\sum_{j=1}^{J_k}\sum_{t=t_{b_{jk}}}^{t_{e_{jk}}-1}\left(r_{jkt+1}\, i_{jkt}\right)(1+ACC)^{T-t+1}$ has two effects. First, the future worth of revenues from product *jk* in period *t* are now accrued in period $T-t+1$. Second, the future worth of the interest earned on the sale of x_{jkt} in period *t* subtracted from the objective function value. Revenues and costs from support are adjusted in the same fashion.

4.5.3. Future Worth of Labor Production Costs

The production cost due to labor FWL_T, given by Equation (4.4), is a function of the number of regular, $w_{lt,}$ and overtime, o_{lt}, hours consumed at production work centers *l* in all periods *t*. Production labor costs in period *t* at work center *l* are given by W_{lt} for regular labor and by O_{lt} for overtime labor. The determination of values for w_{lt} and o_{lt} are given by Equation (4.26).

$$FWL_T = \sum_{t=0}^{T}\sum_{l=1}^{L}\left(W_{lt}w_{lt}+O_{lt}o_{lt}\right)(1+ACC)^{T-t} \tag{4.4}$$

4.5.4. Future Worth of Base Production Costs

It is assumed that the organization may not reduce the amount of specific production work center capacity, B_l, that is possessed at the end of period zero during the optimization. Thus, the organization will incur a cost, b_l, for base production capability at each production work center l for each period of the analysis $t = 0,1,2,...,T$. Since the base production capacity cost, b_l, for production work center l is given in period t = 0 dollars, it is necessary to add t periods of inflation to determine the future worth of all production work center base capacity FWP_T. This determination of future worth is given by Equation (4.5).

$$FWP_T = \sum_{t=0}^{T}\sum_{l=1}^{L} b_l\left(1+\overline{f}\right)^t (1+ACC)^{T-t} \tag{4.5}$$

4.5.5. Future Worth of Pre-Product/Post-Support Capital Budget Project Costs

Each pre-production/post-support capital budgeting project, e_{m_n}, will have associated with it the total cost, $c_{m_n t}$, the organization would incur in each period, t, if the project is selected. Thus, the future worth of the total cost the organization will accrue for all pre-production/post-support capital budgeting projects, $FWCPS_T$, is given by Equation (4.6).

$$FWCPS_T = \sum_{m_n=1}^{M_N}\sum_{t=0}^{T} e_{m_n} c_{m_n t} (1+ACC)^{T-t} \tag{4.6}$$

4.5.6. Future Worth of Product Capital Budget Project Costs

Each production capital budgeting project, e_{m_l}, will have associated with it the total cost, $c_{m_l t}$, the organization would incur in each period, t, if the project is selected. Thus, the future worth of the total cost the organization will accrue for all production capital budgeting projects, $FWCP_T$, is given by Equation (4.7).

$$FWCP_T = \sum_{m_l=1}^{M_L}\sum_{t=0}^{T} e_{m_l} c_{m_l t} (1+ACC)^{T-t} \tag{4.7}$$

4.5.7. Future Worth of Support Capital Budget Project Costs

Each support capital budgeting project, e_{m_v}, will have associated with it the total cost, $c_{m_v,t}$, the organization would incur in each period, t, if the project is selected. Thus, the future worth of the total cost the organization will accrue for all support capital budgeting projects, $FWCS_T$, is given by Equation (4.8).

$$FWCS_T = \sum_{m_v=1}^{M_V} \sum_{t=0}^{T} e_{m_v} c_{m_v,t} (1+ACC)^{T-t} \tag{4.8}$$

4.5.8. Future Worth of Labor Repair Revenues

The organization will receive revenues in each period, t, from repair of defective products that have passed their respective warranty points, PW_{jk}, yet are not older than their effective lives, PL_{jk}. Repair revenues from labor are based upon the number of hours, S_{jkt}, required to repair a product of age $\tau_{jk} = PW_{jk}+1, PW_{jk}+2,\ldots,PL_{jk}$ in period t, multiplied by the number of products of age $\tau_{jk} = PW_{jk}+1, PW_{jk}+2,\ldots,PL_{jk}$, and multiplied by the period t revenue per hour of repair labor, f_t. The future worth of labor repair revenues, FWR_T, is given by Equation (4.9).

$$FWR_T = \sum_{k=1}^{K} \sum_{j=1}^{J_k} \sum_{t=t_{bjk}}^{\left(t_{ejk}+PL_{jk}\right)} f_t \left(\begin{array}{l} S_{jk,PL_{jk}} \begin{pmatrix} x_{jk,t-PL_{jk}+1} \\ -i_{jk,t-PL_{jk}+1} \\ +i_{jk,t-PL_{jk}} \end{pmatrix} \\ +S_{jk,PL_{jk}-1} \begin{pmatrix} x_{jk,t-PL_{jk}+2} \\ -i_{jk,t-PL_{jk}+2} \\ +i_{jk,t-PL_{jk}+1} \end{pmatrix} \\ +\ldots+S_{jk,PW_{jk}+1} \begin{pmatrix} x_{jk,t-PW_{jk}} \\ -i_{jk,t-PW_{jk}} \\ +i_{jk,t-PW_{jk}-1} \end{pmatrix} \end{array} \right) (1+\mathrm{ACC})^{T-t} \tag{4.9}$$

Equation (4.9) enables the repair of products that were produced in several periods. Products that are repaired must have passed their point of warranty, PW_{jk},

expiration yet not have passed their product life, PL_{jk}. Because of this, each period, t, may require that products of age $\tau = PW_{jk}+1,\ldots,PL_{jk}$ be repaired at a cost to the consumer. Note that this equation is based upon the assumption that products produced in a period, t, may also require repairs in period t. Thus, products that have reached their life limit, PL_{jk}, will have been produced in period $t - PL_{jk}+1$.

To enable accurate modeling of product labor repair revenues from all products, jk, held in beginning inventory, bin_{jk}, Equation (4.9) should be modified when any product, jk, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period $t = 0$ through the end of the maximum product life length, $max\, PL_{jk}$ (i.e., $t = 0,\ldots,\max PL_{jk}$).

4.5.9. Future Worth of Labor Repair Costs

The organization will incur labor costs in each period, t, from the repair of all defective products that are not older than their effective lives, PL_{jk}. Repair costs due to labor are based upon the number of hours, S_{jkt}, required to repair a product of age $\tau = 1,\ldots,PL_{jk}$ in period t, multiplied by the number of products, x_{jkt}, of age $\tau = 1,\ldots,PL_{jk}$, and multiplied by the period t cost per hour of repair labor, RC_t. This future worth of labor repair costs, $FWLC_T$, is given by Equation (4.10).

$$FWLC_T = \sum_{k=1}^{K}\sum_{j=1}^{J_k}\sum_{t=t_{bjk}}^{(t_{ebk}+PL_{jk})} RC_t \left(\begin{array}{l} S_{jk,PL_{jk}} \begin{pmatrix} x_{jk,t-PL_{jk}+1} \\ -i_{jk,t-PL_{jk}+1} \\ +i_{jk,t-PL_{jk}} \end{pmatrix} \\ +S_{jk,PL_{jk}-1} \begin{pmatrix} x_{jk,t-PL_{jk}+2} \\ -i_{jk,t-PL_{jk}+2} \\ +i_{jk,t-PL_{jk}+1} \end{pmatrix} \\ +\ldots+S_{jk2}\left(x_{jk,t-1}-i_{jk,t-1}+i_{jk,t-2}\right) \\ +S_{jk1}\left(x_{jkt}+i_{jk,t-1}\right) \end{array} \right)(1+ACC)^{T-t} \qquad (4.10)$$

Equation (4.10) enables the repair of products that were produced in several periods. Products that are repaired must not have passed their effective product life, PL_{jk}.

Because of this, each period, t, may require that products of age $\tau = 1,\ldots,PL_{jk}$ be repaired. Again, please note that this equation is based upon the assumption that products produced in a period, t, may also require repairs in period t. Thus, products that have reached their life limit, PL_{jk}, will have been produced in period $t - PL_{jk} + 1$ while the most recent products requiring repair will have been produced in period t.

To enable accurate modeling of product labor repair costs from all products, jk, held in beginning inventory, bin_{jk}, Equation (4.10) should be modified when any product, jk, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period $t = 0$ through the end of the maximum product life length, $max\ PL_{jk}$ (i.e., $t = 0,\ldots,\max PL_{jk}$).

4.5.10. Future Worth of Parts Repair Revenues Less Costs

The organization will receive parts revenues and incur parts costs in each period, t, from the repair of all defective products that are not older than their effective lives, PL_{jk}. Repair revenues due to parts are based upon the average revenue (in constant dollars) from parts, PR_{jk}, required to repair a product of age $\tau = PW_{jk} + 1,\ldots,PL_{jk}$ in period t multiplied by the number of products, x_{jkt}, of age $\tau = PW_{jk} + 1,\ldots,PL_{jk}$. Repair costs due to parts are based upon the average cost (in constant dollars) of parts, $p_{jk\tau}$, required to repair a product of age $\tau = 1,\ldots,PL_{jk}$ in period t multiplied by the number of products of age $\tau = 1,\ldots,PL_{jk}$. This future worth of parts repair revenues less costs, $FWPR_T$, is given by Equation (4.11).

Again, Equation (4.11) enables the repair of products that were produced in several periods. Products that are repaired must not have passed their effective product life, PL_{jk}. Because of this, each period, t, may require that products of age $\tau = 1,\ldots,PL_{jk}$ be repaired. Note that this equation is based upon the assumption that products produced in a period, t, may also require repairs in period t. Thus, products that have reached their life limit, PL_{jk}, will have been produced in period $t - PL_{jk} + 1$ while the most recent products requiring repair will have been produced in period t.

To enable accurate modeling of product parts repair revenues less costs from all products, jk, held in beginning inventory, bin_{jk}, Equation (4.11) should be modified when any product, jk, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done

by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period $t = 0$ through the end of the maximum product life length, $max\ PL_{jk}$ (i.e., $t = 0,\ldots,\max PL_{jk}$).

$$FWPR_T = \sum_{k=1}^{K}\sum_{j=1}^{J_k}\sum_{t=t_{bjk}}^{\left(t_{ejk}+PL_{jk}\right)} \left(\begin{array}{l} \begin{pmatrix} PR_{jk,PL_{jk}} \\ -p_{jk,PL_{jk}} \end{pmatrix} \begin{pmatrix} x_{jk,t-PL_{jk}+1} \\ -i_{jk,t-PL_{jk}+1} \\ +i_{jk,t-PL_{jk}} \end{pmatrix} \\ +\begin{pmatrix} PR_{jk,PL_{jk}-1} \\ -p_{jk,PL_{jk}-1} \end{pmatrix} \begin{pmatrix} x_{jk,t-PL_{jk}+2} \\ -i_{jk,t-PL_{jk}+2} \\ +i_{jk,t-PL_{jk}+1} \end{pmatrix} \\ +\ldots+\begin{pmatrix} PR_{jk,PW_{jk}+1} \\ -p_{jk,PW_{jk}+1} \end{pmatrix} \begin{pmatrix} x_{jk,t-PW_{jk}} \\ -i_{jk,t-PW_{jk}} \\ +i_{jk,t-PW_{jk}-1} \end{pmatrix} \\ -p_{jk,PW_{jk}} \begin{pmatrix} x_{jk,t-PW_{jk}+1} \\ -i_{jk,t-PW_{jk}+1} \\ +i_{jk,t-PW_{jk}} \end{pmatrix} \\ -\ldots-p_{jk2} \begin{pmatrix} x_{jk,t-1} \\ -i_{jk,t-1} \\ +i_{jk,t-2} \end{pmatrix} \\ -p_{jk1}\left(x_{jkt}+i_{jk,t-1}\right) \end{array} \right) \left(1+\overline{f}\right)^t (1+ACC)^{T-t} \qquad (4.11)$$

4.5.11. Future Worth of Base Support Costs

It is assumed that the organization may not reduce the amount of support capacity, S_0, that is possessed at the end of period zero during the optimization. Thus, the organization will incur a cost, s_0, for base support capacity for each period of the analysis $t = 0,1,2,\ldots,T$. Since the base support capacity cost, s_0, is given in period $t = 0$ dollars, it is necessary to add t periods of inflation to determine the future worth of all production work center base capacity. The future worth of base support costs, FWS_T, is given by Equation (4.12).

$$FWS_T = \sum_{t=0}^{T} s_0\left(1+\overline{f}\right)^t (1+ACC)^{T-t} \qquad (4.12)$$

4.5.12. Future Worth of Base Pre-Production and Post-Support Work Center Costs

It is assumed that the organization may not reduce the amount of specific pre-production and post-support, B_n, capacity that is possessed at the end of period zero during the optimization. Thus, the organization will incur a cost, b_n, for base pre-production and post-support capability for each period of the analysis $t = 0, 1, \ldots, T$. Since the base pre-production and post-support capacity cost, b_n, for work center, n, is given in period $t = 0$ dollars, it is necessary to add t periods of inflation to determine the future worth of all production work center base capacity. The future worth of base pre-production and post-support work center costs, FWN_T, is given by Equation (4.13).

$$FWN_T = \sum_{t=0}^{T} \sum_{n=1}^{N} b_n \left(1 + \bar{f}\right)^t (1 + ACC)^{T-t} \tag{4.13}$$

4.5.13. Future Worth of Production Material Costs

The future worth of production material cost, given by Equation (4.14), accounts for the total cost of all materials required by all products, jk, throughout all periods t. Materials are defined as the components, assemblies, and/or raw materials required by product jk during the production process.

$$FWMA_T = \sum_{k=1}^{K} \sum_{j=1}^{J_k} \sum_{t=t_{bjk}}^{t_{ejk}} MA_{jkt} x_{jkt} (1 + ACC)^{T-t} \tag{4.14}$$

4.6. Constraint Formulation

The concurrent model includes 29 types of constraints. These constraint types fall within product specific, production specific, and capital budgeting categories. The constraint types and their respective categories are shown in Table 4.3.

Table 4.3. Constraints Considered by the Concurrent Model.

Production Specific Constraints -

1. Pre-Production and Post-Support Work Center Capacity
2. Production Set to Zero if a Product is Not Funded
3. Total Allowable Production in Period *t*
4. Production Work Center Constraint
5. Production Setup Constraints
6. Determination of Values for Regular and Overtime Labor
7. Maximum Regular Time Production Labor
8. Maximum Production Overtime Labor
9. Production Labor Balancing Parameter
10. Maximum Production Cost

Product Specific Constraints -

11. Product Exclusivity Constraints
12. Product Contingency Constraints
13. Product Must Fund Constraint
14. Product Support During Production and Utilization
15. Maximum Allowable Support Parts Cost in a Single Period
16. Minimum Acceptable Revenue
17. Maximum Inventory in Each Period

Capital Budgeting Specific Constraints -

18. Maximum Budget for Pre-Production/Post Support Capital Budgeting Constraint
19. Maximum Budget for Production Capital Budgeting Constraint
20. Maximum Budget for Support Capital Budget Constraint
21. Pre-Production/Post-Support Capital Budgeting Mutually Exclusive Project Constraint
22. Production Capital Budgeting Mutually Exclusive Project Constraint
23. Support Capital Budgeting Mutually Exclusive Project Constraint
24. Pre-Production/Post-Support Capital Budgeting Project Contingency Constraint
25. Production Capital Budgeting Project Contingency Constraint
26. Support Capital Budgeting Project Contingency Constraint
27. Pre-Production/Post-Support Capital Budgeting Project Must Fund Constraint
28. Production Capital Budgeting Project Must Fund Constraint
29. Support Capital Budgeting Project Must Fund Constraint

4.6.1. Pre-Production and Post-Support Work Center Capacity

The pre-production and post-support work center constraint, given by Equation (4.15), simultaneously considers the quantity, g_{jknt}, of work center n's resources in period t required by all products jk, the addition of similar resources, $C_{m_n t}$, by the selection of capital budgeting projects, e_{m_n}, and the base capacity, B_n.

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k} g_{jknt} z_{jk} - \sum_{m_n=1}^{M_N} e_{m_n} C_{m_n t} \le B_n \quad \forall\ n \in N;\ t \in T \tag{4.15}$$

Constraint (4.16) allows pre-production and post-support activities, z_{jk}, for product jk to either not occur at all (i.e., $z_{jk} = 0$), or if these activities do occur, they can only occur once (i.e., $z_{jk} = 1$).

$$z_{jk} \in 0,1 \quad \forall \quad j \in J_k;\ k \in K \tag{4.16}$$

4.6.2. Production Set to Zero if a Product is Not Funded

The constraints given by Equations (4.17) and (4.18) have the effect of setting production quantities in period t of product jk equal to zero if pre-production and post-support activities (represented by the variable z_{jk}) are not undertaken. That is, if a product has not been funded through all life-cycle phases prior to production and following support, then the product cannot be produced. This relationship between pre-production and post-support, and production and support can be stated mathematically as

$$\begin{cases} x_{jkt} = 0 \ \textit{if}\ z_{jk} = 0 \\ x_{jkt} \ge 0 \ \textit{if}\ z_{jk} = 1 \end{cases} \quad \forall\ t \in T.$$

$$x_{jkt} - u_{jkt} z_{jk} \le 0 \quad \forall\ j \in J_k;\ k \in K;\ t_{b_{jk}} \le t \le t_{e_{jk}} \tag{4.17}$$

$$x_{jkt} \ge 0 \ \ \forall\ x_{jkt} \in R_n;\ j \in J_k;\ k \in K;\ t \in T \tag{4.18}$$

where u_{jkt} is some multiple of the expected upper limit of production of product jk in period t. Note that if $z_{jk} = 1$, then all pre-production and post-support activities for

product jk have been funded. However, it should be noted here that if $z_{jk} = 1$, x_{jkt} may still be zero[4.3].

4.6.3. Total Allowable Production in Each Period

The total allowable production constraints (4.19), (4.20), and (4.21) control production by allowing the production of products x_{jkt} in substitute product group k plus the inventory on hand, $i_{jk,t-1}$, - i.e., the inventory on hand from the previous period (t-1) - for substitute product group k less inventory to be added, i_{jkt}, in this period, t, to be no greater than the demand, d_{kt}, for all products in substitute product group k in period t.

$$\sum_{j=1}^{J_k} \left(x_{jk0} + bin_{jk} - i_{jk0} \right) \le d_{k0} \quad \forall \; k \in K, \; t = 0 \tag{4.19}$$

$$\sum_{j=1}^{J_k} \left(x_{jkt} + i_{jk,t-1} - i_{jkt} \right) \le d_{kt} \quad \forall \; k \in K; \; 1 \le t \le T_e \tag{4.20}$$

$$i_{jkt} \ge 0 \;\; \forall \;\; i_{jkt} \in R_n; \; j \in J_k; \; k \in K; \; t \in T \tag{4.21}$$

Further, the sum of production and inventories in any period, t, for products in any substitute product group, k, may be less than the substitute product group demand, d_{kt}. If production quantities and inventories are less than the substitute product group demand, then it is more profitable to produce other products and not meet this demand, and/or products in substitute product group k cost more to produce than they return in revenues[4.4].

The constraint given by Equation (4.19) is used only in the current period, i.e., $t = 0$. This constraint sets the quantity of all products, x_{jk0}, in each substitute product group, k, to be produced in period $t = 0$ plus beginning inventory, bin_{jk}, less inventory

---

[4.3] R_n denotes a variable to be assigned a real number, i.e., solved as a linear programming variable rather than as an integer variable.

held, i_{jk0}, at the end of the period $t = 0$ to be less than or equal to the total substitute product group demand, d_{kt}, in period $t = 0$.

The constraint given by Equation (4.20) is used in all periods except the current period. This constraint sets the quantity of all products, x_{jkt}, in each substitute product group, k, to be produced in period $t = 1,2,\ldots,T_e$ plus the ending inventory from the previous period, $i_{jk,t-1}$, less inventory held in the current period, i_{jkt}, to be less than or equal to the total substitute product group demand, d_{kt}, in period t.

4.6.4 Maximum Inventory Level Constraint

The maximum inventory level constraint (4.22) is added to ensure that the inventory variable selected for inventory corresponds to the production variable causing the inventory to occur. This constraint is necessary because the LP would attempt to increase the inventory variable with the least negative coefficient in the objective function. This inventory variable, however, may not correspond to the production variable causing the greater than zero condition in the inventory.

$$i_{jkt} - x_{jkt} \leq 0 \quad \forall \; i_{jkt} \in R_n; \; j \in J_k; \; k \in K; \; t \in T \tag{4.22}$$

4.6.5. Production Work Center Constraint

The production work center constraint, given by Equation (4.23), simultaneously considers: (1) production work center l's base capacity, B_l, in hours, (2) the hours of capacity, C_{mt}, that can be added by the acceptance of production capital budgeting project e_m, and (3) the time in hours required to setup and produce products jk in period t. $\delta(x_{jkt})$ is a zero-one decision variable for the setup of product jk at all work centers required to produce product jk (i.e., if setup occurs at one work center, than it must occur at all work centers), a_{jkl} is the amount of hours required to setup product jk at work center l, and h_{jkl} is the time in hours to perform the process required to produce one unit of product jk at work center l.

4.4 Production planning models that attempt to minimize cost also must meet demand. Thus, use of cost minimization models may result in the production of products that cost more than they return in revenues.

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k}\left(a_{jkl}\delta(x_{jkt})+h_{jkl}x_{jkt}\right)-\sum_{m_l=1}^{M_L}e_{m_l}C_{m_l t}\le B_l \quad \forall\, l\in L; t\in T; e_{m_l}\in 0,1 \tag{4.23}$$

4.6.6. Production Setup Constraints

Before production of product *jk* may commence in period *t*, a setup (i.e., $\delta(x_{jkt})=1$) must occur. Because production of product *jk* must either occur at all work centers in a period or not occur at all (i.e., $x_{jkt}=0$), this relationship between production and setups is not dependent upon production work centers *l*.

The relationship between production and setups can be stated mathematically as:

$$\begin{cases} x_{jkt}=0 \text{ if } \delta(x_{jkt})=0 \\ x_{jkt}\ge 0 \text{ if } \delta(x_{jkt})=1 \end{cases}$$

Constraints given by Equations (4.24), (4.25), and (4.26) rely on disjunctive programming techniques to set $x_{jkt}=0$ if $\delta(x_{jkt})=0$, and $x_{jkt}\ge 0$ if $\delta(x_{jkt})=1$ (Sherali and Shetty, 1980). This is achieved by setting the value of the upper limit on production, u_{jkt}, to some multiple of the demand for products in group *k* during period *t*. The upper limit on production, u_{jkt}, must be larger than the product sub-group demand, d_{kt}, for period *t* because the capability to produce products in excess of demand (and, thus, carry an inventory) and sell them in the subsequent period, $t+1$, is incorporated in the model. Thus, the maximum production, $\max\{x_{jkt}\}$, can be set such that $\max\{x_{jkt}\}=m_{kt}d_{kt}$ where m_{kt} is some multiple of product sub-group demand, d_{kt}.

$$x_{jkt}\le u_{jkt}\delta(x_{jkt}) \quad \forall\ j\in J_k; k\in K; t\in T \tag{4.24}$$

$$\delta(x_{jkt})\in 0,1 \tag{4.25}$$

$$u_{jkt}=\max\{x_{jkt}\} \tag{4.26}$$

4.6.7. Determination of Values for Regular and Overtime Labor

The sum of regular, w_{lt}, and overtime, o_{lt}, labor at a production work center, l, in a period, t, will be equal to the total of setup ($a_{jkl}\delta(x_{jkt})$) and production ($h_{jkl}x_{jkt}$) hours required by all products, jk. Because the objective is to maximize future worth, w_{lt} will also reach its maximum value - given by Equation (4.27) - before $o_{lt} > 0$ [4.5].

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k}\left(a_{jkl}\delta(x_{jkt}) + h_{jkl}x_{jkt}\right) - w_{lt} - o_{lt} = 0 \quad \forall\ l \in L;\ t \in T \tag{4.27}$$

4.6.8. Maximum Regular Time Production Labor

The constraint given by Equation (4.28) limits the value of regular time labor, w_{lt}, by allowing it to be no greater than the number of hours available at production work center l multiplied by λ_{lt}, the maximum fraction of time at work center l that may be charged as regular time. Note it is still possible for w_{lt} to take on values that are less than the maximum work center capacity multiplied by λ_{lt}. This simply means that excess capacity exists at the work center and is not used in that period.

$$w_{lt} - \lambda_{lt}(B_l + \sum_{m_l=1}^{M_L} e_{m_l}C_{m_lt}) \le 0 \quad \forall\ l \in L;\ t \in T \tag{4.28}$$

The constraint given by Equation (4.29) limits values of w_{lt} to zero or greater. It also places w_{lt} in the real number set.

$$w_{lt} \ge 0;\ \ w_{lt} \in R_n \tag{4.29}$$

4.6.9. Maximum Production Overtime Labor

The maximum production overtime labor constraint, given by Equation (4.30), limits the value of overtime labor, o_{lt}, by allowing it to be no greater than the number of hours available at production work center l multiplied by $(1-\lambda_{lt})$, where λ_{lt} is the maximum fraction of time at work center l that may be charged as regular time. Note it is

---

[4.5] This assumes that the cost of regular time labor per hour W_{lt} is less than the cost of overtime labor per hour O_{lt}.

still possible for o_{lt} to take on values that are less than the maximum work center capacity multiplied by $(1-\lambda_{lt})$. This simply means that excess capacity exists at the work center, and is not used in that period.

$$o_{lt} - (1-\lambda_{lt})(B_l + \sum_{m_l=1}^{M_L} e_{m_l} C_{m_l t}) \leq 0 \quad \forall\ l \in L;\ t \in T \tag{4.30}$$

The constraint given by Equation (4.31) limits values of o_{lt} to zero or greater. It also places o_{lt} in the real number set.

$$o_{lt} \geq 0;\ \ o_{lt}\ \in R_n \tag{4.31}$$

4.6.10. Production Labor Balancing Parameter

The labor balancing parameter, given by Equation (4.32), is used to set the maximum available work center capacity that may be used at regular labor rates. λ_{lt} sets the maximum fraction of time at production work center l that may be charged as regular time. Hence, $(1 - \lambda_{lt})$ sets the maximum amount of time at work center l that may be charged as overtime.

$$0 \leq \lambda_{lt} \leq 1 \quad \forall\ l \in L;\ t \in T;\ \ \lambda_{lt}\ \in R_n \tag{4.32}$$

4.6.11. Maximum Production Cost

The maximum production cost constraint, given by Equation (4.33), constrains the feasible set by ensuring that costs from production in any period do not exceed a maximum amount, PC_t. Costs taken into account are regular, w_{lt}, and overtime, o_{lt}, labor, and all materials, MA_{jkt}, consumed to build products.

$$\sum_{l=1}^{L}(W_{lt}w_{lt} + O_{lt}o_{lt}) + \sum_{k=1}^{K}\sum_{j=1}^{J_k} MA_{jkt}x_{jkt} \leq PC_t \quad \forall\ t \in T \tag{4.33}$$

4.6.12. Product Exclusivity Constraints

Exclusivity constraints are of two types. These are: (1) restricting the production in period t of two products, each within different substitute product groups, to produce:

(a) neither, (b) one or the other, and (c) not both; and (2) funding only one of two products.

4.6.12.1. Different Substitute Product Groups

In the first case, an organization may desire to specify that two products, each residing in different substitute product groups, may not both be produced in the same period. This can be stated mathematically as $\begin{cases} x_{j1t} = 0 \text{ if } x_{i2t} > 0 \\ x_{j1t} \geq 0 \text{ if } x_{i2t} = 0 \end{cases}$ where x_{j1t} is the quantity produced in period t of some product j within substitute product group $k = 1$, and x_{i2t} is the quantity produced in period t of some product i within substitute product group $k = 2$.

The simplest way to handle this problem is through the development of the nonlinear constraint given by Equation (4.34a).

$$(x_{j1t})(x_{i2t}) = 0 \quad \forall \ t \in T \tag{4.34a}$$

However, a main objective of this research is that the resulting concurrent model should be based on 0,1 and linear programming. Because of this, a linear formulation of the same constraint given by Equation (4.34a) has been developed. These constraints, based on a disjunctive programming approach, are given by Equations (4.34b), (4.35), and (4.36).

$$0 \leq x_{j1t} \leq u_{j1t} y_{j1t} \quad \forall \ t \in T \tag{4.34b}$$

$$0 \leq x_{i2t} \leq u_{i2t}(1 - y_{j1t}) \quad \forall \ t \in T \tag{4.35}$$

$$y_{j1t} \in 0,1 \tag{4.36}$$

u_{j1t} and u_{i2t} are arbitrary upper bounds, large enough, where u_{j1t} and u_{i2t} may be set at $\max\{x_{j1t}\}$ and $\max\{x_{i2t}\}$ respectively.

4.6.12.3. Funding Only One of Two Products

In the second case, an organization may choose to only fund one of two products. Thus, these products are mutually exclusive. An example of this would be two impact wrenches that are still in the design stage. They are nearly identical, and as such, the organization decides that it will only further fund one of the impact wrench designs. Therefore, the two designs are mutually exclusive.

Mutually exclusive products may either reside in the same substitute product group, or in different substitute product groups. Because of this, two distinct cases of mutual exclusivity have to be accounted for. The first case, known as inter-substitute product group mutual exclusivity, where the two products, j not equal to i, reside in the same substitute product group, k, can be stated mathematically as $\begin{cases} z_{jk} = 0 \text{ if } z_{ik} = 1 \\ z_{jk} \geq 0 \text{ if } z_{ik} = 0 \end{cases}$. Equation (4.37) in conjunction with the constraint given by Equation (4.17) controls this case of product mutual exclusivity.

$$z_{j1k} + z_{ik} \leq 1 \tag{4.37}$$

The second case, known as extra-substitute product group mutual exclusivity, where the two products, denoted by $i1$ and $j2$, reside in different substitute product groups $k = 1$, and $k = 2$ respectively, can be stated mathematically as $\begin{cases} z_{i1} = 0 \text{ if } z_{j2} = 1 \\ z_{i1} \geq 0 \text{ if } z_{j2} = 0 \end{cases}$. Equation (4.38) accounts for this case of mutual exclusivity.

$$z_{i1} + z_{j2} \leq 1 \tag{4.38}$$

Note that subscripts i and j have been adopted here. This is because the products' positions (normally j) within the substitute product group $k = 1$ and $k = 2$ do not have to be the same. However, this does not discount the possibility that $i = j$.

4.6.13. Product Contingency Constraints

Product contingencies may only occur among two products that reside in different substitute product groups. In developing contingency constraints, it will be considered that two products, ik and jk, that have contingency relationships exist. These product contingencies are of four types. These are: (1) less than or equal to, (2) only need to produce, (3) products equal in production volume, and (4) at least as much.

4.6.13.1. Less Than or Equal To

In some cases, an organization may desire to produce a product that will serve no purpose if another product is also not produced. For example, an air tool manufacturer

that produces impact wrenches may also produce sockets of different sizes to be used with the impact wrenches. Assuming that the sockets do not wear out and that all owners may not purchase sockets, the production level of sockets should never be higher than the production level of the impact wrenches.

Stated more precisely, the less than or equal to constraint requires that the contingent product's production volume in every period, t, never exceed the production volume of the product it is contingent upon. This can be stated mathematically as $0 \le x_{j1t} \le x_{i2t} \quad \forall\ t \in T$. The less than or equal to constraint, given by Equation (4.39), enables this relationship.

$$x_{j1t} - x_{i2t} \le 0 \quad \forall\ t \in T \tag{4.39}$$

4.6.13.2. Only Need to Produce

In some cases, an organization may choose to produce a product only if another product will also be produced. A variety of reasons may exist for this restriction. For example, if specific equipment must be procured to produce both products, and the organization anticipates higher production volume for the independent product, acquiring the equipment and then only producing one of the products may be cost prohibitive. Therefore, the organization would choose to enable production only if the independent product were funded.

The only need to produce constraint requires that the independent product's production volume in every period, t, be greater than zero before the contingent product's production can exceed zero. This can be stated mathematically as $\begin{cases} x_{j1t} = 0 \text{ if } x_{i2t} = 0 \\ x_{j1t} \ge 0 \text{ if } x_{i2t} > 0 \end{cases} \quad \forall\ t \in T.$ Note that a minimum production volume greater than zero for the independent product (i.e., $x_{i2t} > \min$) can easily be incorporated into this relationship. The only need to produce constraint is given by Equation (4.40).

$$x_{j1t} - x_{i2t} u_{1t} \le 0 \quad \forall\ t \in T \tag{4.40}$$

This constraint is sufficient for u_{1t} large enough; arbitrary, e.g., $u_{1t} = \max\{x_{j1t}\}$.

4.6.13.3. Products Equal In Production Volume

In some cases products are complimentary or mutually contingent. That is, both products must exist in the same quantity for the system to function. If one or the other does not exist, then the system will not operate. An example of complimentary products would be replacement camshaft and crankshaft gears on an engine. If a timing chain is worn on an engine, then the mechanic must usually replace both gears. While these gears are produced, packaged, and sold separately, the mechanic must purchase both to adequately perform the repair. Thus, these two gears must be produced in the same volume.

The equal to constraint requires that the dependent product's production volume in every period, t, be identical to the independent product's production volume. This can be stated mathematically as $x_{j1t} = x_{i2t} \quad \forall\ t \in T$. The equal to constraint, is given by Equation (4.41).

$$x_{j1t} - x_{i2t} = 0 \quad \forall\ t \in T \tag{4.41}$$

4.6.13.4. At Least as Much

In some cases, an organization must produce more of the dependent product than the independent product. However, the decision to produce the dependent product is still contingent upon producing some quantity of the independent product. A variety of reasons may exist for this restriction. For example, if the manufacturer determines that the demand for the dependent product will increase rapidly at a certain production level of the independent product they would choose to enact this type of constraint.

The at least as much constraint requires that the independent product's production volume in every period, t, be greater than zero before the contingent product's production can exceed zero. This can be stated mathematically as $\begin{cases} x_{j1t} = 0 \ if\ x_{i2t} = 0 \\ x_{j1t} \geq x_{i2t} \ if\ x_{i2t} > 0 \end{cases} \quad \forall\ t \in T.$

Note that a minimum production volume greater than zero for the independent product (i.e., $x_{i2t} > \min$) can easily be incorporated into this relationship. The at least as much constraint, is given by Equations (4.42) and (4.43).

$$x_{i2t} - x_{j1t} \leq 0 \quad \forall\ t \in T \tag{4.42}$$

$$u_{1t}x_{i2t} - x_{j1t} \geq 0 \quad \forall\ t \in T \tag{4.43}$$

This constraint is sufficient for u_{1t} large enough; arbitrary, e.g., $u_{1t} = \max\{x_{j1t}\}$.

4.6.14. Product Must Fund Constraint

Product must fund means that resources at all pre-production and post-support work centers must be allocated to the product. Thus, any solution not containing the product decision variable value $z_{jk} = 1$ will be infeasible. A product may be set as a must fund by setting as given by Equation (4.44).

$$z_{jk} = 1 \tag{4.44}$$

4.6.15. Maximum Budget for Pre-Production/Post-Support Capital Budgeting Constraint

The pre-production/post-support maximum budget constraint, given by Equation (4.45), requires that the sum of the period t costs, $c_{m_n t}$, of all pre-production/post-support capital budgeting projects selected (i.e., $e_{m_n} = 1$) be less than the maximum amount, E_{t_n}, that can be spent on pre-production/post-support capital budgeting projects in period t.

$$\sum_{m_n=1}^{M_N} c_{m_n t} e_{m_n} \leq E_{t_n} \quad \forall\ t \in T; e_{m_n} \in 0,1 \tag{4.45}$$

4.6.16. Maximum Budget for Production Capital Budgeting Constraint

The production maximum budget constraint, given by Equation (4.46), requires that the sum of the period t costs, $c_{m_l t}$, of all production capital budgeting projects selected (i.e., $e_{m_l} = 1$) be less than the maximum amount, E_{t_l}, that can be spent on production capital budgeting projects in period t.

$$\sum_{m_l=1}^{M_L} c_{m_l t} e_{m_l} \leq E_{t_l} \quad \forall\ t \in T; e_{m_l} \in 0,1 \tag{4.46}$$

4.6.17. Maximum Budget for Support Capital Budgeting Constraint

The support maximum budget constraint, given by Equation (4.47), requires that the sum of the period t costs, $c_{m_v t}$, of all support capital budgeting projects selected (i.e., $e_{m_v} = 1$) be less than the maximum amount, E_{t_v}, that can be spent on support capital budgeting projects in period t.

$$c_{m_v t} e_{m_v} \leq E_{t_v} \quad \forall \; t \in T; e_{m_v} \in 0,1 \tag{4.47}$$

4.6.18. Capital Budgeting Pre-Production/Post-Support Mutually Exclusive Project Constraint

When two pre-production/post-support capital budgeting projects, $e1_n$ and $e2_n$, cannot both be funded, they are mutually exclusive. Since $e_{m_n} \in 0,1 \; \forall \; m_n \in M_N$, then the constraint that represents two mutually exclusive pre-production/post-support capital budgeting projects is given by Equation (4.48).

$$e1_n + e2_n \leq 1 \tag{4.48}$$

4.6.19. Capital Budgeting Production Mutually Exclusive Project Constraint

When two production capital budgeting projects, $e1_l$ and $e2_l$, cannot both be funded, they are mutually exclusive. Since $e_{m_l} \in 0,1 \; \forall \; m_l \in M_L$, then the constraint that represents two mutually exclusive production capital budgeting projects is given by Equation (4.49).

$$e1_l + e2_l \leq 1 \tag{4.49}$$

4.6.20. Capital Budgeting Support Mutually Exclusive Project Constraint

When two support capital budgeting projects, $e1_v$ and $e2_v$, cannot both be funded, they are mutually exclusive. Since $e_{m_v} \in 0,1 \; \forall \; m_v \in M_V$, then the constraint that represents two mutually exclusive support capital budgeting projects is given by Equation (4.50).

$$e1_v + e2_v \leq 1 \tag{4.50}$$

4.6.21. Pre-Production/Post Support Capital Budgeting Project Contingency Constraint

When one pre-production/post-support capital budgeting project, $e1_n$, is contingent upon the acceptance of another capital budgeting project, $e2_n$, project $e1_n$ is said to be contingent upon project $e2_n$. Since $e_{m_n} \in 0,1 \ \forall \ m_n \in M_N$, the constraint that represents the contingency relationship between two pre-production/post-support capital budgeting projects is given by Equation (4.51).

$$e1_n - e_{2_n} \leq 0 \tag{4.51}$$

4.6.22. Production Capital Budgeting Project Contingency Constraint

When one production capital budgeting project, $e1_l$, is contingent upon the acceptance of another capital budgeting project, $e2_l$, project $e1_l$ is said to be contingent upon project $e2_l$. Since $e_{m_n} \in 0,1 \ \forall \ m_n \in M_N$, the constraint that represents the contingency relationship between two production capital budgeting projects is given by Equation (4.52).

$$e1_l - e_{2_l} \leq 0 \tag{4.52}$$

4.6.23. Support Capital Budgeting Project Contingency Constraint

When one support capital budgeting project, $e1_v$, is contingent upon the acceptance of another capital budgeting project, $e2_v$, project $e1_v$ is said to be contingent upon project $e2_v$. Since $e_{m_v} \in 0,1 \ \forall \ m_v \in M_V$, the constraint that represents the contingency relationship between two support capital budgeting projects is given by Equation (4.53).

$$e1_v - e_{2_v} \leq 0 \tag{4.53}$$

4.6.24. Pre-Production/Post-Support Capital Budgeting Project Must Fund Constraint

When an organization determines that a pre-production/post-support capital budgeting project (e.g., e_{1_n}) must be accepted, it is said to be a must fund project. An example of a must fund capital budgeting project would be a project that must be accepted before the organization can comply with environmental regulations. Since $e_{m_n} \in 0,1 \ \forall \ m_n \in M_N$, the constraint that accounts for must fund pre-production/post-support capital budgeting projects is given by Equation (4.54).

$$e_{1_n} = 1 \tag{4.54}$$

4.6.25. Production Capital Budgeting Project Must Fund Constraint

When an organization determines that a production capital budgeting project (e.g., e_{1_l}) must be accepted, it is said to be a must fund project. Since $e_{m_l} \in 0,1 \ \forall \ m_l \in M_L$, the constraint that accounts for must fund production capital budgeting projects is given by Equation (4.55).

$$e_{1_l} = 1 \tag{4.55}$$

4.6.26. Support Capital Budgeting Project Must Fund Constraint

When an organization determines that a support capital budgeting project (e.g., e_{1_v}) must be accepted, it is said to be a must fund project. Since $e_{m_v} \in 0,1 \ \forall \ m_v \in M_V$, the constraint that accounts for must fund support capital budgeting projects is given by Equation (4.56).

$$e_{1_v} = 1 \tag{4.56}$$

4.6.27. Product Support During Production and Utilization

The number of hours spent repairing products in a period, t, can never exceed the amount of hours that are available for repair at the support work center. Thus, a constraint, given by Equation (4.57), to account for this must simultaneously consider the number of hours of support, $S_{jk\tau}$, required by all products, x_{jkt}, of ages $\tau = 0,1,...,PL_{jk} - 1$

and all products sold from inventory, i_{jkt}, of ages $\tau = 1,2,...,PL_{jk}$ for each period, t, and the base support, S_0, available along with the increases in available support, C_{m_v}, if support capital budgeting project e_{m_v} were funded.

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k}\left(\begin{array}{l} S_{jk,PL_{jk}}\begin{pmatrix} x_{jk,t-PL_{jk}+1} \\ -i_{jk,t-PL_{jk}+1} \\ +i_{jk,t-PL_{jk}} \end{pmatrix} \\ +S_{jk,PL_{jk}-1}\begin{pmatrix} x_{jk,t-PL_{jk}+2} \\ -i_{jk,t-PL_{jk}+2} \\ +i_{jk,t-PL_{jk}+1} \end{pmatrix} \\ +...+S_{jk2}\begin{pmatrix} x_{jk,t-1} \\ -i_{jk,t-1} \\ +i_{jk,t-2} \end{pmatrix} \\ +S_{jk1}\begin{pmatrix} x_{jkt} \\ +i_{jk,t-1} \end{pmatrix} - \sum_{m_v=1}^{M_v} e_{m_v} C_{m_v t} \end{array}\right) \le S_0 \quad \forall \; t = 0,1,...,T-1; v \in V_m; m_v \in M_N \tag{4.57}$$

To enable accurate modeling of product support hours due to all products, *jk*, held in beginning inventory, bin_{jk}, Equation (4.57) should be modified when any product, *jk*, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period t = 0 through the end of the maximum product life length, $max\, PL_{jk}$ (i.e., $t = 0,\ldots,\max PL_{jk}$).

4.6.28. Maximum Allowable Support Parts Cost in a Single Period

Repair costs due to parts in period t can never exceed the maximum amount, P_t, allocated for this purpose. Repair costs are based upon the average cost (in constant dollars) of parts, $p_{jk\tau}$, adjusted to time t dollars required to repair a product of age $\tau = 1,\ldots,PL_{jk}$ in period t multiplied by the number of products of age $\tau = 1,\ldots,PL_{jk}$. Costs are adjusted by production levels, x_{jkt}, and inventory levels in periods t and t-1. This constraint is given by Equation (4.58).

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k}\begin{pmatrix} p_{jk,PL_{jk}}\left(x_{jk,t-PL_{jk}+1}-i_{jk,t-PL_{jk}+1}+i_{jk,t-PL_{jk}}\right) \\ +p_{jk,PL_{jk}-1}\left(x_{jk,t-PL_{jk}+2}-i_{jk,t-PL_{jk}+2}+i_{jk,t-PL_{jk}+1}\right) \\ +\ldots+p_{jk2}\left(x_{jk,t-1}-i_{jk,t-1}+i_{jk,t-2}\right) \\ +p_{jk1}\left(x_{jkt}+i_{jk,t-1}\right) \end{pmatrix}\left(1+\overline{f}\right)^t \le P_t \quad (4.58)$$

$\forall\ t = 0,1,\ldots,T-1$

To enable accurate modeling of product support parts costs due to all products, *jk*, held in beginning inventory, bin_{jk}, Equation (4.58) should be modified when any product, *jk*, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period t = 0 through the end of the maximum product life length, *max* PL_{jk} (i.e., $t = 0,\ldots,\max PL_{jk}$).

4.6.29. Minimum Acceptable Revenue

The organization receives revenues from three sources. These are: (1) sales of new products (r_{jkt}), (2) labor (f_t) to repair products that are out of warranty, and (3) parts (PR_t) to repair products that are out of warranty. The revenues are adjusted by inventory levels in each period. The organization may choose to impose a constraint that the sum of the revenues from these three sources in period t must be at least as much as the minimum acceptable revenue, R_t, for period t. This constraint is given by Equation (4.59).

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k}\left(\begin{array}{l} r_{jkt}\left(x_{jkt}-i_{jkt}+i_{jk,t-1}\right) \\ +\begin{pmatrix} PR_{jk,PL_{jk}}\left(1+\overline{f}\right)^t \\ +f_t S_{jk,PL_{jk}} \end{pmatrix}\begin{pmatrix} x_{jk,t-PL_{jk}+1} \\ -i_{jk,t-PL_{jk}+1} \\ +i_{jk,t-PL_{jk}} \end{pmatrix} \\ +\begin{pmatrix} PR_{jk,PL_{jk}-1}\left(1+\overline{f}\right)^t \\ +f_t S_{jk,PL_{jk}-1} \end{pmatrix}\begin{pmatrix} x_{jk,t-PL_{jk}+2} \\ -i_{jk,t-PL_{jk}+2} \\ +i_{jk,t-PL_{jk}+1} \end{pmatrix} \\ +\ldots+\begin{pmatrix} PR_{jk,PW_{jk}+1}\left(1+\overline{f}\right)^t \\ +f_t S_{jk,PW_{jk}+1} \end{pmatrix}\begin{pmatrix} x_{jk,t-PW_{jk}} \\ -i_{jk,t-PW_{jk}} \\ +i_{jk,t-PW_{jk}-1} \end{pmatrix} \end{array}\right) \geq R_t \qquad \forall\ t=0,1,\ldots,T-1 \qquad (4.59)$$

The sales and support revenues obtained from each product, *jk*, held in beginning inventory, bin_{jk}, and sold in period $t = 0$ must be included in each period specific minimum acceptable revenue constraint until the product, *jk*, has reached the end of its life, PL_{jk}. Thus, the revenues given by Equation (4.59a) must be added to the left hand side of the $t = 0$ minimum revenue constraint.

$$\sum_{k=0}^{K}\sum_{j=0}^{J_k} r_{jk0} bin_{jk} \qquad (4.59a)$$

Further, to enable accurate modeling of product support and parts revenues obtained from all products, *jk*, held in beginning inventory, bin_{jk}, Equation (4.59) should be modified when any product, *jk*, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period $t = 0$ through the end of the maximum product life length, *max* PL_{jk} (i.e., $t = 0,\ldots,\max PL_{jk}$).

5. THE SEQUENTIAL MODEL

In this chapter, a mathematical model to enable the sequential solution of product, production, and capacity planning while possibly meeting demand for a group of substitute products is formulated. This model has been developed to: (1) possibly meet consumer demand within a group of substitute products, and (2) simultaneously consider all six life-cycle phases. The six life-cycle phases considered in this sequential model are: (1) conceptual design, (2) preliminary design, (3) detailed design and development, (4) production, (5) product use and support, and (6) product phase out.

5.1. Description of the Sequential Model

The sequential model has been developed specifically to demonstrate that the concurrent model provides better results than a related sequential product, production, and capital budgeting model. Development of the sequential model was necessary because sequential models do not exist that could be directly compared to the concurrent model.

The sequential model relies on the same set of inputs as the concurrent model. In addition, for the most part, subscripts and terminology are shared. Further, **all** constraints considered by the concurrent model are considered in one or more stages of the sequential model.

The sequential model, shown schematically in Figure 5.1, relies on three stages[5.1] to determine the best product, production, and capital budgeting plan. Stage one determines the set of products to be produced (and those not to be produced) along with

---

[5.1] This approach is consistent with the Kim et al. approach as abstracted in Section 2.7.7. Note, however, that two stages (as opposed to three) are required by the Kim et al. model. In the Kim et al. model, the production quantities are determined in the first stage. The second stage must then find the capacity increases that provide the minimum discounted cost and also enable production in the periods and quantities supplied by the first stage. Because increases in capacity are unbounded in the second stage, a feasible solution in the second stage is guaranteed for any solution emanating from the first stage. This means that any production schedule determined in the first stage is feasible. The sequential model developed and presented herein requires three stages because the production capacity in the third stage is bounded by: (1) upper limits on project expenses, and (2) the quantity of production projects that can be acquired. A moderating stage (stage two) is added to ensure feasibility in the third stage. This moderating stage reduces the production quantities specified in stage one so that a feasible solution is ensured in stage three. These differences in modeling approach were made apparent in two phone conversations (May 20, 1993 and June 3, 1993) between Dr. Jay S. Kim and William K. Hoehn.

the best production quantities in each period[5.2]. Stage two adjusts the production quantities emanating from stage one so that feasibility is ensured in stage three. Stage three finds the minimum future worth production cost including the cost of production and support capital budgeting projects[5.3].

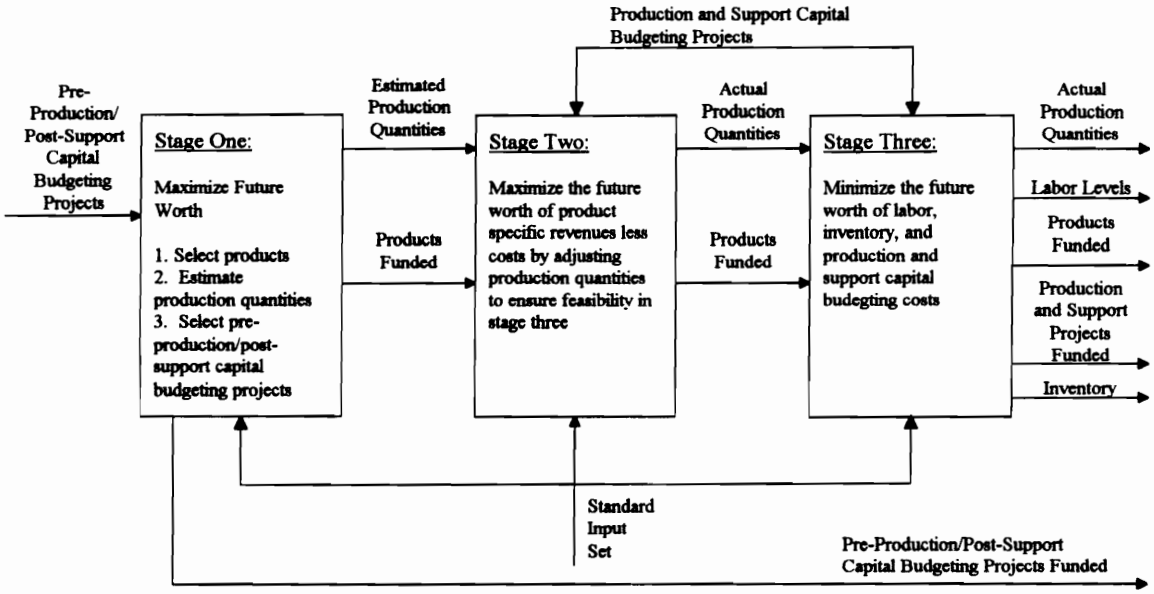

Figure 5.1. Stages and Interaction in the Sequential Model.

Stage one does not consider the addition of production or support capacity through known capital budgeting projects. Rather, production and support work center costs are estimated based upon the cost per hour of the current capacity of production and support. Stage one may increase the capacity of any production or support work center, but the cost to increase capacity is estimated based upon the current cost.

Since the production quantities emanating from stage one may prove to be infeasible when specific production and support capital budgeting projects are considered, stage two is added to ensure feasibility. Stage two maximizes the future worth of product specific revenues and costs and also considers the existing and additional capacity. Stage

---

5.2 Stage one considers four phases of the product life cycle. These are: (1) conceptual design, (2) preliminary design, (3) detailed design, and (4) phase-out.

5.3 Stage three considers two stages of the product life cycle. These are: (1) manufacturing and utilization, and (2) support.

two may be viewed as a moderating component because it only exists to adjust the production quantities derived in stage one to ensure feasibility in stage three. Since the objective is to find the maximum production levels possible when specific capital budgeting projects are considered, no cost is imposed for selection of a capital budgeting project.

Stage three minimizes the future worth of the costs of production, inventory, labor, and production and support capital budgeting projects. Production quantities are output from stage two and become input to stage three. Thus, since production quantities are known in advance, stage three functions much like a traditional production planning model.

5.2. Subscripts and Objective Function Components

This section contains a description of the subscripts, and objective function components used in the formulation of the sequential model. The objective function has been divided into several components to make clear the life-cycle phases, and cost and revenue inputs addressed by the model.

5.2.1. Subscripts

k = A group of products that serve the same purpose, i.e., known as substitute products.

K = The number of substitute product groups.

jk = A specific product j within the substitute product group k.

J_k = The number of products j in a specific subgroup k.

l = A specific work center used only in the **production** of products. Work centers with subscripts l are referred to herein as production work centers.

L = The number of production work centers.

n = A specific work center used only in the non-production and non-utilization phases of the product life cycle, i.e., applicable to conceptual design, preliminary design, detailed design and development, and phase-out. Work centers with subscripts n are referred to herein as pre-production and post-support work centers.

N = The number of non-production work centers.

t = A specific time period.

T = The number of periods considered through the horizon point.

v = Refers to the **one** support work center used to **support** products. The work center with subscript v is referred to herein as the support work center.

m_l = A capital budgeting project that affects a production work center l.

M_L = The number of production capital budgeting projects considered.

m_n = A capital budgeting project that affects non-production work centers n.

M_N = The number of pre-production/post-support capital budgeting projects considered.

m_v = A capital budgeting project that affects the support work center v.

M_V = The number of support capital budgeting projects considered.

T_e = The last period that production occurs.

T_s = The last period that support occurs.

5.2.2. Objective Function Variables

The objective function components have been divided into three classes. Class one components combine to form the stage one objective function. Class two components combine to form the stage two objective function. Class three components combine to form the stage three objective function.

5.2.2.1. Stage One Objective Function Components

$FWNP_T$ = The future worth of revenues from the sales of new products.

$FWSU_T$ = The total estimated future worth of production capacity incurred during production work center setups.

$FWPC_T$ = The total estimated future worth of production capacity due to production (i.e., excluding cost of setups).

$FWUP_T$ = The total estimated future worth of under-utilized production capacity.

$FWSC_T$ = The total estimated future worth of utilized support capacity.

$FWUS_T$ = The total estimated future worth of under-utilized support capacity. Accounts for the cost of support work center time not used in each period t.

FWI_T = The future worth of inventory holding costs for all products, jk, and all periods, t.

$FWIS_T$ = The total adjusted estimated future worth of support work center time in each period when inventories are carried.

FWL_T = The future worth of labor production costs for all periods, t, and all products, jk.

FWP_T = The future worth of the cost of base production capacity for all periods, t.

$FWCPS_T$ = The future worth of funded pre-production/post-support capital budgeting projects, $e_{m_n} = 1$, for all periods, t.

FWR_T = The future worth of revenues received for labor to repair all products, jk, of age $\tau = PW_{jk}+1, PW_{jk}+2, \ldots, PL_{jk}$ in all periods $t = 0,1,\ldots,T$.

$FWLC_T$ = The future worth of the costs for labor to repair all products, jk, of age $\tau = 1,2,\ldots, PL_{jk}$ in all periods $t = 0,1,\ldots,T$.

$FWPR_T$ = The future worth of the revenues less costs for parts to repair all products of age $\tau = 1,2,\ldots,PL_{jk}$ in all periods $t = 0,1,\ldots,T$.

FWS_T = The future worth of the base capacity of the support work center for all periods $t = 0,1,\ldots,T$.

FWN_T = The future worth of the base capacity of all pre-production and post-support work centers for all periods $t = 1, 2,\ldots,T$.

$FWMA_T$ = The future worth of all material cost accrued during production.

5.2.2.2. Stage Two Objective Function Components

$FWNP_T$ = The future worth of revenues from the sales of new products.

FWR_T = The future worth of revenues received for labor to repair all products, *jk*, of age $\tau = PW_{jk}+1, PW_{jk}+2,\ldots,PL_{jk}$ in all periods $t = 0,1,\ldots,T$.

$FWLC_T$ = The future worth of the costs for labor to repair all products, *jk*, of age $\tau = 1,2,\ldots,PL_{jk}$ in all periods $t = 0,1,\ldots,T$.

$FWPR_T$ = The future worth of the revenues less costs for parts to repair all products of age $\tau = 1,2,\ldots,PL_{jk}$ in all periods $t = 0,1,\ldots,T$.

$FWMA_T$ = The future worth of all material cost accrued during production.

5.2.2.3. Stage Three Objective Function Components

FWI_T = The future worth of inventory holding costs for all products, *jk*, and all periods, *t*.

FWL_T = The future worth of labor production costs for all periods, *t*, and all products, *jk*.

$FWLC_T$ = The future worth of the costs for labor to repair all products, *jk*, of age $\tau = 1,2,\ldots,PL_{jk}$ in all periods $t = 0,1,\ldots,T$.

$FWMA_T$ = The future worth of all material cost accrued during production.

$FWPRC_T$ = The future worth of the costs for parts to repair all products of age $\tau = 1,2,\ldots,PL_{jk}$ in all periods $t = 0,1,\ldots,T$.

$FWCP_T$ = The future worth of funded production capital budgeting projects, $e_{m_l} = 1$, for all periods *t*.

$FWCS_T$ = The future worth of funded support capital budgeting projects, $e_{m_v} = 1$, for all periods *t*.

5.3. Decision Variables

The sequential model considers thirteen classes of decision variables. However, these are divided among stages as follows:

5.3.1. Variables Considered by Stages One, Two, and Three

x_{jkt} = The amount of product j in substitute product group k to be produced in period t. Production may not commence before period $t_{b_{jk}}$ and must cease in period $t_{e_{jk}}$.

i_{jkt} = Ending inventory of product j in substitute product group k at time t. Solved as a real rather than integer variable. Inventory cannot be held prior to period $t_{b_{jk}}$ and must be completely liquidated by the end of period $t_{e_{jk}} - 1$.

$\delta(x_{jkt})$ = Setup decision in period t for product j in substitute product group k. Solved as a 0,1 decision variable. Only one setup can occur per period; hence, production of product jk can only occur once per period. This assumption is consistent with standard production planning modeling practice.

w_{lt} = Regular labor hours consumed at work center l in period t. Solved as a LP rather than MIP variable. Since w_{lt} is given in hours, it is conceivable that its values do not have to be restricted to integer values.

o_{lt} = Overtime hours consumed at work center l in period t. Since o_{lt} is given in hours, it is conceivable that its values do not have to be restricted to integer values.

z_{jk} = Funding variable of product j in substitute product group k for pre and post production activities (excluding support). Solved as a 0,1 decision variable.

5.3.2. Decision Variables Specific to Stage One

e_{m_n} = A zero-one decision variable that enables selection or rejection of pre-production/post-support capital budgeting proposal e_{m_n}.

up_{lt} = Variable to account for under-utilization of production work center l in period t. Forces production work center cost to be no less than the cost of base production capacity. Ensures consistency between the concurrent and sequential model.

op_{lt} = Variable to account for production work center resources used beyond base capacity levels. Allows for an increase in production capacity, and ensures consistency between the concurrent and sequential model.

us_t = Variable to account for under-utilization of the support work center in period t. Forces support work center cost to be no less than the cost of base support capacity. Ensures consistency between the concurrent and sequential model.

os_t = Variable to account for support work center resources used beyond base capacity levels. Allows for an increase in support capacity, and ensures consistency between the concurrent and sequential model.

5.3.3. Variables Specific to Stage Two and Three

e_{m_l} = A zero-one decision variable that enables selection or rejection of production capital budgeting proposal e_{m_l}.

e_{m_v} = A zero-one decision variable that enables selection or rejection of support capital budgeting proposal e_{m_v}.

5.4. Model Inputs

The sequential model requires that the values for 42 types of inputs be known in advance. Descriptions of these inputs are as follows:

I_{jkt} = Inventory holding cost in actual dollars for product j in substitute product group k for period $t = t_{b_{jk}}, \ldots, t_{e_{jk}} - 1$.

d_{kt} = The total demand for all products in subgroup k in period t.

$t_{b_{jk}}$ = The first period that product j in substitute product group k may be produced.

$t_{e_{jk}}$ = The last period that product j in substitute product group k may be produced.

ACC = The long-run average cost of capital with an inflation component.

$\overline{f}$ = Average inflation rate through the horizon.

a_{jkl} = Setup time in hours for product j in substitute product group k at work center l.

h_{jkl} = The time required in hours at work center l to produce one unit of product j in substitute product group k.

$C_{m_l t}$ = The increase in hours in period $t = 1,2,\ldots,T$ at production work centers $l = 1,2,\ldots,L$ if capital budgeting project e_{m_l} is selected.

B_l = The beginning hours available at work center l in period zero. Note that work centers with subscripts l are production work centers.

B_n = The beginning capacity in some consistent units available at work center n. Note that work centers with subscripts n are pre-production and post-support work centers. Also, note that this LP has been developed such that each pre-production and post-support work center n consumes only one type of resource. However,

the resource types consumed at all pre-production and post-support work centers do not have to be identical. For example, the resources consumed at work center $n = 1$ may be engineering man hours while the resources consumed at work center $n = 2$ may be computer time.

$C_{m_n t}$ = The increase in some consistent units in period t at pre-production and support work centers $n = 1, 2, \ldots, N$ if capital budgeting project e_{m_n} is selected.

$C_{m_v t}$ = The increase in hours in period t for support if capital budgeting project e_{m_v} is selected.

$c_{m_l t}$ = The cost of production capital budgeting project e_{m_l} in period t. Includes all costs to due to increased capacity of production work centers from capital budgeting project e_{m_l} excluding labor.

$c_{m_n t}$ = The cost of pre-production/post-support capital budgeting project e_{m_n} in period t. Includes all costs to due to increased capacity of pre-production/post-support work centers from capital budgeting project e_{m_n} excluding labor.

$c_{m_v t}$ = The cost of support capital budgeting project e_{m_v} in period t. Includes all costs to due to increased capacity of support work center from capital budgeting project e_{m_v} excluding labor.

E_{t_l} = The maximum allowable that may be spent on production capital budgeting projects in period t.

E_{t_n} = The maximum allowable that may be spent on pre-production/post-support capital budgeting projects in period t.

E_{t_v} = The maximum allowable that may be spent on support capital budgeting projects in period t.

$S_{jk\tau_{jk}}$ = Support hours required for one unit of product j of ages $\tau = 1, 2, \ldots, PL_{jk}$ in substitute product group k.

S_0 = Support hours available at the beginning of period zero (i.e., base support hours).

PC_t = The maximum allowable cost of production in period $t = 0, 1, \ldots, T_e$ including materials and labor.

PL_{jk} = Life length in periods for one unit of product j in substitute product group k.

PW_{jk} = Warranty length in periods for one unit of product j in substitute product group k.

$p_{jk\tau_{jk}}$ = Average cost in constant dollars of parts to repair one unit of product j of age $\tau = 1,2,\ldots,PL_{jk}$ in substitute product group k where $\tau_{jk} \in 1,2,\ldots,PL_{jk}$.

P_t = Maximum repair parts cost in period $t = 0,1,\ldots,T_s$.

$PR_{jk\tau_{jk}}$ = Average revenue in constant dollars from parts to repair one unit of product j in substitute product group k of age τ_{jk} where $\tau_{jk} \in 1,2,\ldots,PL_{jk}$.

r_{jkt} = Revenue in actual dollars from the sale of one unit of product j in substitute product group k in period $t = 0,1,\ldots,T_s$.

R_t = Minimum acceptable total revenue in period $t = 0,1,\ldots,T_s$ - includes sales of new products and repairs. Defaults to zero if no value is specified.

f_t = Average revenue in actual dollars per hour of repair labor in period t.

W_{lt} = Cost of regular labor in actual dollars per hour in period t at production work center l.

O_{lt} = Cost of overtime labor in actual dollars per hour in period t at production work center l. Note that it is assumed that $O_{lt} > W_{lt}$.

b_l = Base cost to operate work center l (excluding production labor) at beginning capacity B_l. Note that products funded (i.e., Z_{jk} variable equal to one) may consume resources at production work center l. The cost due to this consumption of resources (excluding production labor) appears in both the b_l and $c_{m_l t}$ parameters that are funded (i.e., $e_{m_l} = 1$). Because of this, it is not necessary to directly specify product cost at each work center l.

b_n = Base cost to operate pre-production and post-support work center n at beginning capacity B_n. Note that products funded (i.e., Z_{jk} variable equal to one) may consume resources at work center n. The cost due to this consumption of resources appears in both the b_n and $c_{m_n t}$ parameters that are funded (i.e., $e_{m_n} = 1$). Because of this, it is not necessary to directly specify pre-production and post-support product cost at each work center n.

s_0 = Base cost to operate support facilities. Note that products funded (i.e., Z_{jk} variable equal to one) may consume support resources. The cost due to this consumption of resources appears in both the s_0 and $c_{m_v t}$ parameters that are funded (i.e., $e_m = 1$). Because of this, it is not necessary to directly specify product support cost.

MA_{jkt} = The total cost of materials in actual dollars (i.e., raw materials, components, and assemblies) required to produce one unit of product j in substitute product group k in period t.

RC_t = Repair labor cost in actual dollars per hour in period t.

bin_{jk} = The beginning inventory of product j in substitute product group k in period 0.

g_{jknt} = Consistent units (i.e., all units within a work center n must be the same) consumed at pre-production and post-support work center n in period t by product j in substitute product group k.

U = Upper bounds used in various constraints to either: (1) provide linear rather than non-linear solution methods, or (2) constrain real variables by the value of a zero-one variable.

y = A binary selection variable is used to enable linear rather than nonlinear solutions.

reg_l = The maximum amount of labor in each period t that may be charged as regular time at each production work center l. This value is an estimate and is based upon previous experience.

5.5 Values Passed from One Stage to the Next Stage

$X1_{jkt}$ = Maximum production quantities determined by stage one and used in stage two of the sequential model for each product j in group k in each period $t = t_{b_{jk}}, \ldots, t_{e_{jk}}$.

$X2_{jkt}$ = Exact production quantities determined in stage two and used in stage three of the sequential model for each product j in group k in each period $t = t_{b_{jk}}, \ldots, t_{e_{jk}}$.

Z_{jkt} = Products funded: determined in stage one and used in stages two and three.

5.6. Stage One Objective Function Formulation

The objective function has been divided into seventeen components. The objective is to maximize the future worth of revenues from sales and support less the future worth of the costs of production, inventory holding (including stocking fee and lost interest on sale of a product produced in a period), support, and pre-production and post-support activities. It is important to note that the cost of additional production and support capacity is estimated based on the cost of current base production and support capacity.

Table 5.1 shows all revenue and cost components accounted for in stage one of the sequential model.

Table 5.1. Revenue and Cost Components for Stage One.

Revenue Components	Cost Components
1. The future worth of revenues from the sales of new products - $FWNP_T$. 2. The future worth of revenues from labor to repair defective, out of warranty products - FWR_T.	1. The total estimated future worth of production capacity incurred during production work center setups - $FWSU_T$. 2. The total estimated future worth of production capacity due to production (i.e., excluding cost of setups) - $FWPC_T$. 3. The total estimated future worth of under-utilized production capacity - $FWUP_T$. 4. The total estimated future worth of utilized support capacity - $FWSC_T$. 5. The total estimated future worth of under-utilized support capacity. Accounts for the cost of support work center time not used in each period t - $FWUS_T$. 6. The future worth of inventory holding cost - FWI_T. 7. The total adjusted estimated future worth of support work center time in each period when inventories are carried - $FWIS_T$. 8. The future worth of labor production costs - FWL_T. 9. The future worth of all production work center base capacity costs - FWP_T.

Table 5.1. Continued.

Revenue Components	Cost Components
	10. The future worth of the total cost to enact all funded pre-production/post-support capital budgeting projects - $FWCPS_T$. 11. The future worth of labor repair costs - $FWLC_T$. 12. The future worth of base support costs - FWS_T. 13. The future worth of base pre-production and post-support work center costs - FWN_T. 14. The future worth of all production material costs - $FWMA_T$.
1. The future worth of parts repair revenues less costs[5.4] - $FWPR_T$.	

The objective of stage one is to

$$\text{maximize} \left\{ \begin{matrix} FWNP_T + FWR_T + FWPR_T - FWSU_T - FWPC_T - FWUP_T \\ -FWSC_T - FWUS_T - FWI_T - FWIS_T - FWL_T - FWP_T - FWCPS_T \\ -FWLC_T - FWS_T - FWN_T - FWMA_T \end{matrix} \right\} \quad (5.1)$$

where $FWNP_T$, $FWSU_T$, $FWPC_T$, $FWUP_T$, $FWSC_T$, $FWUS_T$, FWI_T, $FWIS_T$, FWL_T, FWP_T, $FWCPS_T$, FWR_T, $FWLC_T$, $FWPR_T$, FWS_T, FWN_T, and $FWMA_T$ are given by Equations (5.2), (5.3), (5.4), (5.5), (5.6), (5.7), (5.8), (5.9), (5.10), (5.11), (5.12), (5.13), (5.14), (5.15), (5.16), (5.17), and (5.18) respectively while meeting the constraints given by Equations (5.19) through (5.51).

[5.4] The future worth of parts repair revenues less costs includes both revenue and cost components.

5.6.1. Future Worth of Revenues from Sales of New Products

The future worth of revenues $FWNP_T$ from the sales of new products is given by Equation (5.2).

$$FWNP_T = \sum_{k=1}^{K}\sum_{j=1}^{J_k}\left(\left(\sum_{t=t_{b_{jk}}}^{t_{e_{jk}}} r_{jkt}x_{jkt}\right)(1+ACC)^{T-t} + r_{jk0}bin_{jk}(1+ACC)^{T}\right) \tag{5.2}$$

The production quantity decision variables are x_{jkt}. The revenues for a specific product in a specific period are known and are represented by r_{jkt}. Further, each product in beginning inventory, bin_{jk}, will contribute period $t = 0$ revenues, r_{jk0}. Thus, all revenues from the products held in beginning inventory, and the products produced and sold during the periods of analysis are considered in this equation. Note that the period t is defined in the equation to start at $t_{b_{jk}}$ and end at $t_{e_{jk}}$. This is because production of a product jk cannot occur before, $t_{b_{jk}}$, the earliest period it may be produced, and must cease at the end of period, $t_{e_{jk}}$, the latest period it may be produced. Thus, the problem is simplified by not considering periods in which a specific product jk cannot be produced. This has the effect of reducing the number of coefficients required to solve this component of the objective function.

5.6.2. Estimated Future Worth of Production Capacity Due to Setup

The estimated future worth of production capacity due to setup, $FWSU_T$, given by Equation (5.3), determines the total cost of production capacity at all production work centers l consumed during the setup of work centers prior to the production of all products, jk, in all periods $t = t_{b_{jk}},\ldots,t_{e_{jk}}$. This objective function component is required in stage one of the sequential model because potential increases in production capacity through know capital budgeting projects are not known.

$$FWSU_T = \sum_{k=1}^{K}\sum_{j=1}^{J_k}\sum_{t=t_{b_{jk}}}^{t_{e_{jk}}}\left(\frac{a_{jkl}\delta(x_{jkt})b_l}{B_l}\right)(1+\bar{f})^{t}(1+ACC)^{T-t} \tag{5.3}$$

5.6.3. Estimated Future Worth of Production Capacity Due to Production

The estimated future worth of production capacity due to production, $FWPC_T$, given by Equation (5.4), determines the total cost of production capacity at all production work centers, l, consumed during the production of all products, jk, in all periods $t = t_{b_{jk}}, \ldots, t_{e_{jk}}$. This objective function component is required in stage one of the sequential model because potential increases in production capacity through known production capital budgeting projects are not known.

$$FWPC_T = \sum_{k=1}^{K} \sum_{j=1}^{J_K} \sum_{t=t_{b_{jk}}}^{t_{e_{jk}}} \left(\frac{x_{jkt} h_{jkl} b_l}{B_l} \right) \left(1 + \overline{f}\right)^t (1 + ACC)^{T-t} \tag{5.4}$$

5.6.4. Estimated Future Worth of Under-Utilized Production Capacity

The estimated future worth of under-utilized production capacity, $FWUP_T$, given by Equation (5.5), determines the total cost of unused production capacity at all production work centers, l, in all periods $t = 0, 1, \ldots, T$. This objective function component forces the cost of production capacity to be at least equal to the base cost of production capacity. Thus, costs are consistent between the concurrent and sequential models.

$$FWUP_T = \sum_{t=0}^{T} \left(\frac{up_t b_l}{B_l} \right) \left(1 + \overline{f}\right)^t (1 + ACC)^{T-t} \tag{5.5}$$

5.6.5. Estimated Future Worth of Utilized Support Capacity

The estimated future worth of utilized support capacity, $FWSC_T$, given by Equation (5.6), determines the total cost of support capacity consumed during the support of all products, jk, in all periods $t = t_{b_{jk}}, \ldots, t_{e_{jk}}$. This objective function component is required in stage one of the sequential model because potential increases in support capacity through known support capital budgeting projects are not known.

$$FWSC_T = \sum_{k=1}^{K}\sum_{j=1}^{J_K}\sum_{t=t_{b_{jk}}}^{t_{e_{jk}}} \left(\frac{s_0}{S_0}\right)\begin{pmatrix} S_{jk,PL_{jk}}x_{jk,t-PL_{jk}+1} \\ +S_{jk,PL_{jk}-1}x_{jk,t-PL_{jk}+2} \\ +\ldots+S_{jk1}x_{jkt} \end{pmatrix}\left(1+\overline{f}\right)^{t}(1+ACC)^{T-t} \qquad (5.6)$$

To enable accurate modeling of the future worth of support hours consumed by all products, *jk*, held in beginning inventory, bin_{jk}, Equation (5.6) should be modified when any product, *jk*, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period $t = 0$ through the end of the maximum product life length, $max\ PL_{jk}$ (i.e., $t = 0,\ldots,\max PL_{jk}$).

5.6.6. Estimated Future Worth of Under-Utilized Support Capacity

The estimated future worth of under-utilized support capacity, $FWUS_T$, given by Equation (5.7), determines the total cost of unused support capacity in all periods $t = 0,1,\ldots,T$. This objective function component forces the cost of support capacity to be at least equal to the base cost of production capacity. Thus, costs are consistent between the concurrent and sequential models.

$$FWUS_T = \sum_{t=0}^{T}\left(\frac{us_t s_0}{S_0}\right)\left(1+\overline{f}\right)^{t}(1+ACC)^{T-1} \qquad (5.7)$$

5.6.7. Future Worth of Inventory Holding Costs

As in the concurrent model, inventory capability has been included in the sequential model so that it is possible to achieve a balance between high demand periods (and the consequent inability to produce all required products for that period in that period) and the cost of producing in prior periods to meet the demand in future periods. It is assumed that all units of *jk* must be removed from inventory by the end of period $t_{e_{jk}}$, thus ensuring that the production and utilization phases of the product life cycle may be separated.

The inventory holding cost component, FWI_T, of the objective function, given by Equation (5.8), determines the cost of holding units of products *jk* in inventory to meet demands for these products in the following period. Inventory holding costs include: (1) a stocking fee, I_{jkt}, (2) the future worth of the interest lost from not selling a product in the

period it was produced, and (3) the adjustment of support revenues due to support occurring in the periods following the period in which the product was produced.

$$FWI_T = \sum_{k=1}^{K}\sum_{j=1}^{J_k}\sum_{t=t_{b_{jk}}}^{t_{e_{jk}}-1}\left[\begin{array}{l}\left((I_{jkt})(i_{jkt})+(r_{jkt})(i_{jkt})\right)(1+ACC)^{T-t}\\ -\left((r_{jkt+1})(i_{jkt})\right)(1+ACC)^{T-t-1}\end{array}\right] \tag{5.8}$$

Since FWI_T is subtracted from the objective function value, adding $\sum_{k=1}^{K}\sum_{j=1}^{J_k}\sum_{t=t_{b_{jk}}}^{t_{e_{jk}}-1}(r_{jkt}\, i_{jkt})(1+ACC)^{T-t}$ while subtracting $\sum_{k=1}^{K}\sum_{j=1}^{J_k}\sum_{t=t_{b_{jk}}}^{t_{e_{jk}}-1}(r_{jkt+1}\, i_{jkt})(1+ACC)^{T-t+1}$ has two effects. First, the future worth of revenues from each unit of product *jk* held in inventory is now accrued in period $T-t+1$. Second, the future worth of the interest earned on the sale of product *jk* in period *t* is subtracted from the objective function value. Revenues and costs from support are adjusted in the same fashion.

5.6.8. Estimated Future Worth of Support Work Center Adjusted for Inventory Levels

The estimated future worth of support work center adjusted for inventory levels $FWIC_T$, given by Equation (5.9), adjusts the costs of support capacity consumed during the support of all products, *jk*, in all periods $t = t_{b_{jk}},\ldots,t_{e_{jk}}$ when inventories are carried. Period *t* costs are moved forward to period $t + 1$. This objective function component is required in stage one of the sequential model because potential increases in support capacity through known support capital budgeting projects are not known.

$$FWIS_T = \sum_{k=1}^{K}\sum_{j=1}^{J_k}\sum_{t=t_{b_{jk}}}^{t_{e_{jk}}-1}\left[\left(\frac{i_{jkt}s_0}{S_0}\right)\left(\begin{array}{l}(1+\overline{f})^{t}(1+ACC)^{T-1}\\ -(1+\overline{f})^{t+1}(1+ACC)^{T-t-1}\end{array}\right)\right] \tag{5.9}$$

5.6.9. Future Worth of Labor Production Costs

The production cost due to labor, FWL_T, given by Equation (5.10), is a function of the number of regular, w_{lt}, and overtime, o_{lt}, labor hours consumed at each production

work center, l, in all periods, t. Production labor costs in period t at work center l are given by W_{lt} for regular labor and by O_{lt} for overtime labor. The determination of values for w_{lt} and o_{lt} are given by Equations (5.31) and (5.32).

$$FWL_T = \sum_{t=0}^{T}\sum_{l=1}^{L}\left(W_{lt}w_{lt} + O_{lt}o_{lt}\right)\left(1+ACC\right)^{T-t} \tag{5.10}$$

5.6.10. Future Worth of Base Production Costs

It is assumed that the organization may not reduce the amount of specific production work center capacity, B_l, that is possessed at the end of period zero during the optimization. Thus, the organization will incur a cost for base production capability, b_l, at each production work center, l, for each period $t = 0,1,\ldots,T$. The future worth of the base cost of each production work center for periods $t = 0,1,\ldots,T_e$ is given by Equations (5.3), (5.4) and (5.5). Thus, an additional cost component must be added to account for base production work center costs incurred in periods after production has ceased (i.e., periods $t = T_e + 1,\ldots,T$).

The future worth of base production work center costs is given by Equation (5.11). These costs are incurred in periods $t = T_e + 1,\ldots,T$. Since the base production capacity cost b_l for production work center l is given in period $t = 0$ dollars, it is necessary to use constant dollar analysis (i.e., the addition of an inflation component) to determine the future worth of all production work center base capacity FWP_T. This determination of future worth is given by Equation (5.11).

$$FWP_T = \sum_{t=T_e+1}^{T}\sum_{l=1}^{L} b_l\left(1+\overline{f}\right)^t\left(1+ACC\right)^{T-t} \tag{5.11}$$

5.6.11. Future Worth of Pre-Product/Post-Support Capital Budget Project Costs

Each pre-production/post-support capital budgeting project, e_{m_n}, will have associated with it the total cost, $c_{m_n t}$, the organization would incur in each period, t, if the project is selected. Thus, the future worth of the total cost the organization will accrue for all pre-production/post-support capital budgeting projects, $FWCPS_T$, is given by Equation (5.12).

$$FWCPS_T = \sum_{m_n=1}^{M_N} \sum_{t=0}^{T} e_{m_n} c_{m_n t} (1 + ACC)^{T-t} \tag{5.12}$$

5.6.12. Future Worth of Labor Repair Revenues

The organization will receive revenues in each period, t, from repair of defective products that have passed their warranty point, PW_{jk}, yet are not older than their effective lives, PL_{jk}. Repair revenues from labor are based upon the number of hours, S_{jkt}, required to repair a product of age $\tau_{jk} = PW_{jk}+1, PW_{jk}+2,\ldots,PL_{jk}$ in period t, multiplied by the number of products of age $\tau_{jk} = PW_{jk}+1, PW_{jk}+2,\ldots,PL_{jk}$, and multiplied by the period t revenue per hour of repair labor, f_t. The future worth of labor repair revenues, FWR_T, is given by Equation (5.13).

$$FWR_T = \sum_{k=1}^{K} \sum_{j=1}^{J_k} \sum_{t=t_{b_{jk}}}^{\left(t_{e_{jk}} + PL_{jk}\right)} f_t \left(\begin{array}{l} S_{jk,PL_{jk}} \begin{pmatrix} x_{jk,t-PL_{jk}+1} \\ -i_{jk,t-PL_{jk}+1} \\ +i_{jk,t-PL_{jk}} \end{pmatrix} \\ +S_{jk,PL_{jk}-1} \begin{pmatrix} x_{jk,t-PL_{jk}+2} \\ -i_{jk,t-PL_{jk}+2} \\ +i_{jk,t-PL_{jk}+1} \end{pmatrix} \\ +\ldots+S_{jk,PW_{jk}+1} \begin{pmatrix} x_{jk,t-PW_{jk}} \\ -i_{jk,t-PW_{jk}} \\ +i_{jk,t-PW_{jk}-1} \end{pmatrix} \end{array} \right) (1+\mathrm{ACC})^{T-t} \tag{5.13}$$

Equation (5.13) enables the repair of products that were produced in several periods. Products that are repaired must have passed their point of warranty expiration, PW_{jk}, yet not have passed their product life, PL_{jk}. Because of this, each period, t, may require that products of age $\tau = PW_{jk}+1,\ldots,PL_{jk}$ be repaired at a cost to the consumer. Note that this equation is based upon the assumption that products produced in a period, t, may also require repairs in period t. Thus, products that have reached their life limit, PL_{jk}, will have been produced in period $t - PL_{jk} + 1$.

To enable accurate modeling of the future worth of revenues due to labor obtained all products, *jk*, held in beginning inventory, bin_{jk}, Equation (5.13) should be modified when any product, *jk*, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period $t = 0$ through the end of the maximum product life length, $max\ PL_{jk}$ (i.e., $t = 0,\ldots,\max PL_{jk}$).

5.6.13. Future Worth of Labor Repair Costs

The organization will incur labor costs in each period, t, from the repair of all defective products that are not older than their effective lives, PL_{jk}. Repair costs due to labor are based upon the number of hours, S_{jkt}, required to repair a product of age $\tau = 1,\ldots,PL_{jk}$ in period t multiplied by the number of products x_{jkt} of age $\tau = 1,\ldots,PL_{jk}$ multiplied by the period t cost per hour of repair labor, RC_t. This future worth of labor repair costs, $FWLC_T$, is given by Equation (5.14).

$$FWLC_T = \sum_{k=1}^{K}\sum_{j=1}^{J_k}\sum_{t=t_{bjk}}^{\left(t_{ebk}+PL_{jk}\right)} RC_t \left(\begin{array}{l} S_{jk,PL_{jk}} \begin{pmatrix} x_{jk,t-PL_{jk}+1} \\ -i_{jk,t-PL_{jk}+1} \\ +i_{jk,t-PL_{jk}} \end{pmatrix} \\ +S_{jk,PL_{jk}-1} \begin{pmatrix} x_{jk,t-PL_{jk}+2} \\ -i_{jk,t-PL_{jk}+2} \\ +i_{jk,t-PL_{jk}+1} \end{pmatrix} \\ +\ldots+S_{jk2}\left(x_{jk,t-1}-i_{jk,t-1}+i_{jk,t-2}\right) \\ +S_{jk1}\left(x_{jkt}+i_{jk,t-1}\right) \end{array} \right) (1+ACC)^{T-t} \qquad (5.14)$$

Equation (5.14) enables the repair of products that were produced in several periods. Products that are repaired must not have passed their effective product life, PL_{jk}. Because of this, each period, t, may require that products of age $\tau = 1,\ldots,PL_{jk}$ be repaired. Again, please note that this equation is based upon the assumption that products produced in a period, t, may also require repairs in period t. Thus, products that have reached their life limit, PL_{jk}, will have been produced in period $t - PL_{jk} + 1$ while the youngest products requiring repair will have been produced in period t.

To enable accurate modeling of the future worth of costs due to labor from all products, *jk*, held in beginning inventory, bin_{jk}, Equation (5.14) should be modified when any product, *jk*, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period $t = 0$ through the end of the maximum product life length, $max\ PL_{jk}$ (i.e., $t = 0,\ldots,\max PL_{jk}$).

5.6.14. Future Worth of Parts Repair Revenues Less Costs

The organization will receive parts revenues and incur parts costs in each period, *t*, from the repair of all defective products that are not older than their effective lives, PL_{jk}. Repair revenues due to parts are based upon the average revenue (in constant dollars) from parts, PR_{jk}, required to repair a product of age $\tau = PW_{jk}+1,\ldots,PL_{jk}$ in period *t* multiplied by the number of products, x_{jkt}, of age $\tau = PW_{jk}+1,\ldots,PL_{jk}$. Repair costs due to parts are based upon the average cost (in constant dollars) of parts, $p_{jk\tau}$, required to repair a product of age $\tau = 1,\ldots,PL_{jk}$ in period *t* multiplied by the number of products of age $\tau = 1,\ldots,PL_{jk}$. This future worth of parts repair revenues less costs, $FWPR_T$, is given by Equation (5.15).

Equation (5.15) enables the repair of products that were produced in several periods. Products that are repaired must not have passed their effective product life, PL_{jk}. Because of this, each period, *t*, may require that products of age $\tau = 1,\ldots,PL_{jk}$ be repaired. Note that this equation is based upon the assumption that products produced in a period, *t*, may also require repairs in period *t*. Thus, products that have reached their life limit, PL_{jk}, will have been produced in period $t - PL_{jk} + 1$ while the youngest products requiring repair will have been produced in period *t*.

To enable accurate modeling of the future worth of parts repair revenues less costs from all products, *jk*, held in beginning inventory, bin_{jk}, Equation (5.15) should be modified when any product, *jk*, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period $t = 0$ through the end of the maximum product life length, $max\ PL_{jk}$ (i.e., $t = 0,\ldots,\max PL_{jk}$).

$$FWPR_T = \sum_{k=1}^{K}\sum_{j=1}^{J_k}\sum_{t=t_{bjk}}^{\left(t_{ejk}+PL_{jk}\right)} \left(\begin{array}{l} \begin{pmatrix} PR_{jk,PL_{jk}} \\ -p_{jk,PL_{jk}} \end{pmatrix} \begin{pmatrix} x_{jk,t-PL_{jk}+1} \\ -i_{jk,t-PL_{jk}+1} \\ +i_{jk,t-PL_{jk}} \end{pmatrix} \\ +\begin{pmatrix} PR_{jk,PL_{jk}-1} \\ -p_{jk,PL_{jk}-1} \end{pmatrix} \begin{pmatrix} x_{jk,t-PL_{jk}+2} \\ -i_{jk,t-PL_{jk}+2} \\ +i_{jk,t-PL_{jk}+1} \end{pmatrix} \\ +\ldots+\begin{pmatrix} PR_{jk,PW_{jk}+1} \\ -p_{jk,PW_{jk}+1} \end{pmatrix} \begin{pmatrix} x_{jk,t-PW_{jk}} \\ -i_{jk,t-PW_{jk}} \\ +i_{jk,t-PW_{jk}-1} \end{pmatrix} \\ -p_{jk,PW_{jk}} \begin{pmatrix} x_{jk,t-PW_{jk}+1} \\ -i_{jk,t-PW_{jk}+1} \\ +i_{jk,t-PW_{jk}} \end{pmatrix} \\ -\ldots-p_{jk2} \begin{pmatrix} x_{jk,t-1} \\ -i_{jk,t-1} \\ +i_{jk,t-2} \end{pmatrix} \\ -p_{jk1}\left(x_{jkt}+i_{jk,t-1}\right) \end{array} \right) \left(1+\overline{f}\right)^t (1+ACC)^{T-t} \qquad (5.15)$$

5.6.15. Future Worth of Base Support Costs

It is assumed that the organization may not reduce the amount of support capacity, s_0, that is possessed at the end of period zero during the optimization. Equations (5.6), (5.7), and (5.9) account for the future worth of base support capacity through the last period support is provided to products. However, since the support work center will incur costs after the last period of support, an equation must be added to account for this cost component.

The organization will incur a cost, s_0, for base support capacity for each period of the analysis $t = T_s + 1, \ldots, T$ following the last period in which support is provided to products (i.e., $t = T_s$). Since the base support capacity cost, s_0, is given in period $t = 0$ dollars, t periods of inflation are added to determine the future worth of all production work center base capacity. The future worth of base support costs, FWS_T, is given by Equation (5.16).

$$FWS_T = \sum_{t=T_s+1}^{T} s_0(1+\overline{f})^t (1+ACC)^{T-t} \tag{5.16}$$

5.6.16. Future Worth of Base Pre-Production and Post-Support Work Center Costs

It is assumed that the organization may not reduce the amount of specific pre-production and post-support, B_n, capacity that is possessed at the end of period zero during the optimization. Thus, the organization will incur a cost, b_n, for base pre-production and post-support capability for each period of the analysis $t = 0, 1, \ldots, T$. Since the base pre-production and post-support capacity cost, b_n, for each pre-production/post-support work center, n, is given in period $t = 0$ dollars, it is necessary to use constant dollar analysis (i.e., no inflation component) to determine the future worth of all production work center base capacity. The future worth of base pre-production and post-support work center costs, FWN_T, is given by Equation (5.17).

$$FWN_T = \sum_{t=0}^{T} \sum_{n=1}^{N} b_n(1+\overline{f})^t (1+ACC)^{T-t} \tag{5.17}$$

5.6.17. Future Worth of Production Material Costs

The future worth of production material cost, given by Equation (5.18), accounts for the total cost of all materials required by all products, *jk*, throughout all periods, *t*. Materials are defined as the components, assemblies, and/or raw materials required by product, *jk*, during the production process.

$$FWMA_T = \sum_{k=1}^{K} \sum_{j=1}^{J_k} \sum_{t=t_{bjk}}^{t_{ejk}} MA_{jkt} x_{jkt} (1+ACC)^{T-t} \tag{5.18}$$

5.7. Stage One Constraint Formulation

Stage one of the sequential model includes 19 types of constraints. These constraint types fall within product specific, production specific, and capital budgeting categories. The constraint types and their respective categories are shown in Table 5.2.

Table 5.2. Constraints Considered by Stage One of the Sequential Model.

Production Specific Constraints - 1. Pre-Production and Post-Support Work Center Capacity 2. Production Set to Zero if a Product is Not Funded 3. Total Allowable Production in Period *t* 4. Production Work Center Constraint 5. Production Setup Constraints 6. Determination of Values for Regular and Overtime Labor 7. Balance between Regular Time and Overtime Labor Levels 8. Maximum Production Cost
Product Specific Constraints - 9. Product Exclusivity Constraints 10. Product Contingency Constraints 11. Product Must Fund Constraint 12. Product Support During Production and Utilization 13. Maximum Allowable Support Parts Cost in a Single Period 14. Minimum Acceptable Revenue 15. Maximum Inventory in Each Period
Capital Budgeting Specific Constraints - 16. Maximum Budget for Pre-Production/Post Support Capital Budgeting Constraint 17. Pre-Production/Post-Support Capital Budgeting Mutually Exclusive Project Constraint 18. Pre-Production/Post-Support Capital Budgeting Project Contingency Constraint 19. Pre-Production/Post-Support Capital Budgeting Project Must Fund Constraint

5.7.1. Pre-Production and Post-Support Work Center Capacity

The pre-production and post-support work center constraint, given by Equation (5.19), simultaneously considers the quantity, g_{jknt}, of work center n's resources in period t required by all products, jk, the addition of similar resources, $C_{m_n t}$, by the selection of capital budgeting projects, e_{m_n}, and the base capacity, B_n.

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k} g_{jknt} z_{jk} - \sum_{m_n=1}^{M_N} e_{m_n} C_{m_n t} \le B_n \qquad \forall\ n \in N;\ t \in T \tag{5.19}$$

Constraint (5.20) allows pre-production and post-support activities, z_{jk}, for product *jk* to either not occur at all (i.e., $z_{jk} = 0$), or if these activities do occur, they can only occur once (i.e., $z_{jk} = 1$).

$$z_{jk} \in 0,1 \quad \forall \quad j \in J_k;\ k \in K \tag{5.20}$$

5.7.2. Production Set to Zero if a Product is Not Funded

The constraints given by Equations (5.21) and (5.22) have the effect of setting production quantities in period *t* of product *jk* equal to zero if pre-production and post-support activities (represented by the variable z_{jk}) are not undertaken. That is, if a product has not been funded through all life-cycle phases prior to production and following support, then the product cannot be produced. This relationship between pre-production and post-support, and production and support can be stated mathematically as

$$\begin{cases} x_{jkt} = 0 \ \text{if} \ z_{jk} = 0 \\ x_{jkt} \ge 0 \ \text{if} \ z_{jk} = 1 \end{cases} \quad \forall\ t \in T.$$

$$x_{jkt} - u_{jkt} z_{jk} \le 0 \quad \forall\ j \in J_k;\ k \in K;\ t_{b_{jk}} \le t \le t_{e_{jk}} \tag{5.21}$$

$$x_{jkt} \ge 0 \quad \forall\ x_{jkt} \in R_n;\ j \in J_k;\ k \in K;\ t \in T \tag{5.22}$$

where u_{jkt} is some multiple of the expected upper limit of production of product *jk* in period *t*. Note that if $z_{jk} = 1$, then all pre-production and post-support activities for product *jk* have been funded. However, it should be noted here that if $z_{jk} = 1$, x_{jkt} may still be zero.

5.7.3. Total Allowable Production in Each Period

The total allowable production constraints (5.23), (5.24), and (5.25) control production by allowing the production of products in substitute product group *k* plus the

inventory on hand, $i_{jk,t-1}$ - i.e., the inventory from the previous period (t-1) - for substitute product group k less inventory to be added in this period, i_{jkt}, to be no greater than the demand, d_{kt}, for products in substitute product group k in period t.

$$\sum_{j=1}^{J_k}\left(x_{jk0}+bin_{jk}-i_{jk0}\right)\le d_{k0} \quad \forall\ k\in K,\ t=0 \tag{5.23}$$

$$\sum_{j=1}^{J_k}\left(x_{jkt}+i_{jk,t-1}-i_{jkt}\right)\le d_{kt} \quad \forall\ k\in K, t>0 \tag{5.24}$$

$$i_{jkt}\ge 0\ \ \forall\ i_{jkt}\ \in R_n;\ j\in J_k;\ k\in K;\ t\in T \tag{5.25}$$

Further, the sum of production and inventories for products in substitute product groups, k, may be less than the demand. If demand is not met, then it is more profitable to produce other products and not meet this demand, and/or products in substitute product group k cost more to produce than they return in revenues[5.5].

The constraint given by Equation (5.23) is used only in the current period, i.e., $t=0$. This constraint sets the quantity of all products, x_{jk0}, in each substitute product group, k, to be produced in period $t=0$ plus beginning inventory, bin_{jk}, less inventory held, i_{jk0}, at the end of the period $t=0$ to be less than or equal to the total substitute product group demand, d_{kt}, in period $t=0$.

The constraint given by Equation (5.24) is used in all periods except the current period. This constraint sets the quantity of all products, x_{jkt}, in each substitute product group, k, to be produced in period $t=1,2,\ldots,T_e$ plus the ending inventory from the previous period, $i_{jk,t-1}$, less inventory held in the current period, i_{jkt}, to be less than or equal to the total substitute product group demand, d_{kt}, in period t.

5.5 Note that production planning models that attempt to minimize cost also **must meet demand**. Thus, use of cost minimization models may result in the production of products that cost more than they return in revenues.

5.7.4. Maximum Inventory Level Constraint

The maximum inventory level constraint (5.26) is added to ensure that the inventory variable selected for inventory corresponds to the production variable causing the inventory to occur. This constraint is necessary because the MIP would attempt to increase the inventory variable with the least negative coefficient in the objective function. This inventory variable, however, may not correspond to the production variable causing the greater than zero condition in the inventory.

$$i_{jkt} - x_{jkt} \leq 0 \quad \forall \; i_{jkt} \in R_n; \; j \in J_k; \; k \in K; \; t \in T \tag{5.26}$$

5.7.5. Production Work Center Constraint

The production work center constraint, given by Equation (5.27), simultaneously considers: (1) production work center *l*'s base capacity, B_l, in hours, (2) the under-utilization of base capacity, up_{lt}, (3) the over-utilization (addition of production capacity), op_{lt}, and (4) the time in hours required to set up and produce products, *jk*, in period *t*. $\delta(x_{jkt})$ is a zero-one decision variable for the setup of product *jk* at all work centers required to produce product *jk* (i.e., if setup occurs at one work center, than it must occur at all work centers), a_{jkl} is the amount of hours required to setup product *jk* at work center *l*, and h_{jkl} is the time in hours to perform the process required to produce one unit of product *jk* at work center l.

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k}\left(a_{jkl}\delta(x_{jkt}) + h_{jkl}x_{jkt}\right) + up_{lt} - op_{lt} = B_l \qquad \forall \; l \in L; t \in T \tag{5.27}$$

5.7.6. Production Setup Constraints

Before production of product *jk* may commence in period *t*, a setup (i.e., $\delta(x_{jkt}) = 1$) must occur. Because production of product *jk* must either occur at all work centers in a period or not occur at all (i.e., $x_{jkt} = 0$), this relationship between production and setups is not dependent upon production work centers *l*.

The relationship between production and setups can be stated mathematically as:

$$\begin{cases} x_{jkt} = 0 \text{ if } \delta(x_{jkt}) = 0 \\ x_{jkt} \geq 0 \text{ if } \delta(x_{jkt}) = 1 \end{cases}$$

Constraints given by Equations (5.28), (5.29), and (5.30) rely on disjunctive programming techniques to set $x_{jkt} = 0$ if $\delta(x_{jkt}) = 0$, and $x_{jkt} \geq 0$ if $\delta(x_{jkt}) = 1$. This is achieved by setting the value of u_{jkt} to some multiple of the expected upper limit of demand for products in group k during period t.

$$x_{jkt} \leq u_{jkt}\delta(x_{jkt}) \quad \forall\ j \in J_k; k \in K; t \in T \tag{5.28}$$

$$\delta(x_{jkt}) \in 0,1 \tag{2.29}$$

$$u_{jkt} = \max\{x_{jkt}\} \tag{5.30}$$

5.7.7. Determination of Values for Regular and Overtime Labor

The sum of regular, w_{lt}, and overtime, o_{lt}, labor at a production work center, l, in a period, t, will be equal to the total of setup ($a_{jkl}\delta(x_{jkt})$) and production ($h_{jkl}x_{jkt}$) hours required by all products, jk. Because the objective is to maximize future worth, w_{lt} will also reach its maximum value - given by Equation (5.31) - before $o_{lt} > 0$ [5.6].

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k}\left(a_{jkl}\delta(x_{jkt}) + h_{jkl}x_{jkt}\right) - w_{lt} - o_{lt} = 0 \quad \forall\ l \in L;\ t \in T \tag{5.31}$$

5.7.8. Balance Between Regular and Overtime Labor

When production capital budgeting projects are considered, it is possible to determine the exact quantity of labor that will be charged at regular and overtime rates. Since stage one of the sequential model does not consider known production capital budgeting projects, the exact number of hours that may be charged as regular time and overtime are not known. Because of this, a variable, reg_l, must be included to set the

---

[5.6] This assumes that the cost of regular time labor per hour W_{lt} is less than the cost of overtime labor per hour O_{lt}.

quantity of total labor at each production work center, l, that will be charged at regular rates in each period, t.

The constraint given by Equation (5.32) forces overtime labor, o_{lt}, to be a percentage of regular time labor, w_{lt}. For example, if $reg_l = 0.75$, then 25% of the total labor consumed at each production work center, l, in each period, t, must be charged at overtime rates.

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k}(1-reg_l)w_{lt} - o_{lt} \le 0 \quad \forall\ l \in L;\ t \in T \tag{5.32}$$

5.7.9. Maximum Production Cost

The maximum production cost constraint, given by Equation (5.33), constrains the feasible set by ensuring that costs from production in any period do not exceed a maximum amount, PC_t. Costs taken into account are regular and overtime labor, and all materials, MA_{jkt}, consumed to build products.

$$\sum_{l=1}^{L}(W_{lt}w_{lt} + O_{lt}o_{lt}) + \sum_{k=1}^{K}\sum_{j=1}^{J_k} MA_{jkt}x_{jkt} \le PC_t \quad \forall\ t \in T \tag{5.33}$$

5.7.10. Product Exclusivity Constraints

Exclusivity constraints are of two types. These are: (1) restricting the production in period t of two products, each within different substitute product groups, to produce: (a) neither, (b) one or the other, and (c) not both; and (2) funding only one of two products.

5.7.10.1. Different Substitute Product Groups

In the first case, an organization may desire to specify that two products, each residing in different substitute product groups, may not both be produced in the same period. This can be stated mathematically as $\begin{cases} x_{j1t} = 0 \ if\ x_{i2t} > 0 \\ x_{j1t} \ge 0 \ if\ x_{i2t} = 0 \end{cases}$ where x_{j1t} is the quantity produced in period, t, of some product, j, within substitute product group $k = 1$, and x_{i2t} is

the quantity produced in period, t, of some product, i, within substitute product group k = 2.

The constraints limiting production of both products are given by Equations (5.34), (5.35), and (5.36).

$$0 \le x_{j1t} \le u_{j1t} y_{j1t} \quad \forall \ t \in T \tag{5.34}$$

$$0 \le x_{i2t} \le u_{i2t}(1 - y_{j1t}) \quad \forall \ t \in T \tag{5.35}$$

$$y_{j1t} \in 0,1 \tag{5.36}$$

u_{j1t} and u_{i2t} are arbitrary upper bounds, large enough, where u_{j1t} and u_{i2t} may be set at $\max\{x_{j1t}\}$ and $\max\{x_{i2t}\}$ respectively.

5.7.10.2. Funding Only One of Two Products

In the second case, an organization may choose to only fund one of two products. Thus, these products are mutually exclusive. An example of this would be two impact wrenches that are still in the design stage. They are nearly identical, and as such, the organization decides that it will only further fund one of the impact wrench designs. Therefore, the two designs are mutually exclusive.

Mutually exclusive products may either reside in the same substitute product group, or in different substitute product groups. Because of this, two distinct cases of mutual exclusivity have to be accounted for. The first case, known as inter-substitute product group mutual exclusivity, where the two products, j not equal to i, reside in the same substitute product group k, can be stated mathematically as $\begin{cases} z_{jk} = 0 \ if \ z_{ik} = 1 \\ z_{jk} \ge 0 \ if \ z_{ik} = 0 \end{cases}$. Equation (5.37) in conjunction with the constraint given by Equation (5.21) controls this case of product mutual exclusivity.

$$z_{j1k} + z_{ik} \le 1 \tag{5.37}$$

The second case, known as extra-substitute product group mutual exclusivity, where the two products, denoted by $i1$ and $j2$, reside in different substitute product groups

$k = 1$, and $k = 2$ respectively, can be stated mathematically as $\begin{cases} z_{i1} = 0 \text{ if } z_{j2} = 1 \\ z_{i1} \geq 0 \text{ if } z_{j2} = 0 \end{cases}$. Equation (5.38) accounts for this case of mutual exclusivity.

$$z_{i1} + z_{j2} \leq 1 \tag{5.38}$$

Note that subscripts i and j have been adopted here. This is because the products' positions (normally j) within the substitute product group $k = 1$ and $k = 2$ do not have to be the same. However, this does not discount the possibility that $i = j$.

5.7.11. Product Contingency Constraints

Product contingencies may only occur among two products that reside in different substitute product groups. In developing contingency constraints, it will be considered that two products, ik and jk, that have contingency relationships, exist. These product contingencies are of four types. These are: (1) less than or equal to, (2) only need to produce, (3) products equal in production volume, and (4) at least as much.

5.7.11.1. Less Than or Equal To

In some cases, an organization may desire to produce a product that will serve no purpose if another product is also not produced. For example, an air tool manufacturer that produces impact wrenches may also produce sockets of different sizes to be used in the impact wrenches. Assuming that the sockets do not wear out and that all owners may not purchase sockets, the production level of sockets should never be higher than the production level of the impact wrenches.

Stated more precisely, the less than or equal to constraint requires that the contingent product's production volume in every period, t, never exceed the production volume of the product it is contingent upon. This can be stated mathematically as $0 \leq x_{j1t} \leq x_{i2t} \quad \forall \ t \in T$. The less than or equal to constraint, given by Equation (5.39), enables this relationship.

$$x_{j1t} - x_{i2t} \leq 0 \quad \forall \ t \in T \tag{5.39}$$

5.7.11.2. Only Need to Produce

In some cases, an organization may choose to produce a product only if another product will also be produced. A variety of reasons may exist for this restriction. For example, if specific equipment must be procured to produce both products, and the organization anticipates higher production volume for the independent product, acquiring the equipment and then only producing one of the products may be cost prohibitive. Therefore, the organization would choose to enable production only if the independent product were funded.

The only need to produce constraint requires that the independent product's production volume in every period, t, be greater than zero before the contingent product's production can exceed zero. This can be stated mathematically as

$$\begin{cases} x_{j1t} = 0 \text{ if } x_{i2t} = 0 \\ x_{j1t} \geq 0 \text{ if } x_{i2t} > 0 \end{cases} \quad \forall\ t \in T.$$

Note that a minimum production volume greater than zero for the independent product (i.e., $x_{i2t} > \min$) can easily be incorporated into this relationship. The only need to produce constraint is given by Equation (5.40).

$$x_{j1t} - x_{i2t} u_{1t} \leq 0 \quad \forall\ t \in T \tag{5.40}$$

The constraint is sufficient for u_{1t} large enough; arbitrary, e.g., $u_{1t} = \max\{x_{j1t}\}$.

5.7.11.3. Products Equal In Production Volume

In some cases products are complimentary or mutually contingent. That is, both products must exist in the same quantity for the system to function. If one or the other does not exist, then the system will not operate. An example of complimentary products would be replacement camshaft and crankshaft gears on an engine. If a timing chain is worn on an engine, then the mechanic must usually replace both gears. While these gears are produced, packaged, and sold separately, the mechanic must purchase both to adequately perform the repair. Thus, these two gears must be produced in the same volume.

The equal to constraint requires that the dependent product's production volume in every period, t, be identical to the independent product's production volume. This can be stated mathematically as $x_{j1t} = x_{i2t} \quad \forall\ t \in T$. The equal to constraint is given by Equation (5.41).

$$x_{j1t} - x_{i2t} = 0 \quad \forall \ t \in T \tag{5.41}$$

5.7.11.4. At Least as Much

In some cases, an organization must produce more of the dependent product than the independent product. However, the decision to produce the dependent product is still contingent upon producing some quantity of the independent product. A variety of reasons may exist for this restriction. For example, if the manufacturer determines that the demand for the dependent product will increase rapidly at a certain production level of the independent product they would choose to enact this type of constraint.

The at least as much constraint requires that the independent product's production volume in every period, t, be greater than zero before the contingent product's production can exceed zero. This can be stated mathematically as $\begin{cases} x_{j1t} = 0 \ if \ x_{i2t} = 0 \\ x_{j1t} \geq x_{i2t} \ if \ x_{i2t} > 0 \end{cases} \quad \forall \ t \in T.$ Note that a minimum production volume greater than zero for the independent product (i.e., $x_{i2t} > \min$) can easily be incorporated into this relationship. The at least as much constraint, is given by Equations (5.42) and (5.43).

$$x_{i2t} - x_{j1t} \leq 0 \quad \forall \ t \in T \tag{5.42}$$

$$u_{1t} x_{i2t} - x_{j1t} \geq 0 \quad \forall \ t \in T \tag{5.43}$$

The constraint is sufficient for u_{1t} large enough; arbitrary, e.g., $u_{1t} = \max\{x_{j1t}\}$.

5.7.12. Product Must Fund Constraint

Product must fund means that resources at all pre-production and post-support work centers must be allocated to the product. Thus, any solution not containing the product decision variable $z_{jk} = 1$, will be infeasible. A product may be set as a must fund as given by Equation (5.44).

$$z_{jk} = 1 \tag{5.44}$$

5.7.13. Maximum Budget for Pre-Production/Post-Support Capital Budgeting Constraint

The pre-production/post-support maximum budget constraint, given by Equation (5.45), requires that the sum of the period t costs, $c_{m_n t}$, of all pre-production/post-support capital budgeting projects selected (i.e., $e_{m_n} = 1$) be less than the maximum amount, E_{t_n}, that can be spent on pre-production/post-support capital budgeting projects in period t.

$$\sum_{m_n=1}^{M_N} c_{m_n t} e_{m_n} \leq E_{t_n} \quad \forall \; t \in T; e_{m_n} \in 0,1 \tag{5.45}$$

5.7.14. Capital Budgeting Pre-Production/Post-Support Mutually Exclusive Project Constraint

When two pre-production/post-support capital budgeting projects, e_{1_n} and e_{2_n}, cannot both be funded, they are mutually exclusive. Since $e_{m_n} \in 0,1 \;\; \forall \;\; m_n \in M_N$, the constraint that represents two mutually exclusive pre-production/post-support capital budgeting projects, Equation (5.46), is used.

$$e_{1_n} + e_{2_n} \leq 1 \tag{5.46}$$

5.7.15. Pre-Production/Post-Support Capital Budgeting Project Contingency Constraint

When one pre-production/post-support capital budgeting project, e_{1_n}, is contingent upon the acceptance of another capital budgeting project, e_{2_n}, project e_{1_n} is said to be contingent upon project e_{2_n}. Since $e_{m_n} \in 0,1 \;\; \forall \;\; m_n \in M_N$, the constraint that represents the contingency relationship between two pre-production/post-support capital budgeting projects, Equation (5.47), is used.

$$e_{1_n} - e_{2_n} \leq 0 \tag{5.47}$$

5.7.16. Pre-Production/Post-Support Capital Budgeting Project Must Fund Constraint

When an organization determines that a pre-production/post-support capital budgeting project (e.g., e_{1_n}) must be accepted, it is said to be a must fund project. An example of a must fund capital budgeting project would be a project that must be accepted before the organization can comply with environmental regulations. Since $e_{m_n} \in 0,1 \ \forall \ m_n \in M_N$, the constraint that accounts for must fund pre-production/post-support capital budgeting projects, Equation (5.48), is used.

$$e_{1_n} = 1 \tag{5.48}$$

5.7.17. Product Support During Production and Utilization

The number of hours spent repairing products in a period t can never exceed the amount of hours that are available for repair at the support work center. Thus, a constraint, given by Equation (5.49), to account for this must simultaneously consider the total number of hours of support, $S_{jk\tau}$, required by an individual product of age t = 1,2,..., τ for each period, t. The total support hours, $S_{jk\tau}$, required by all products, jk, in a period t can exceed the base support hours available, S_0. However, if this occurs, then the support work center over-utilization variable, os_t, will be greater than zero. If the total number of support hours required in a period are less than the base support hours, S_0, then the support work center under-utilization variable, us_t, will be greater than zero.

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k}\left(\begin{array}{l} S_{jk,PL_{jk}}\begin{pmatrix} x_{jk,t-PL_{jk}+1} \\ -i_{jk,t-PL_{jk}+1} \\ +i_{jk,t-PL_{jk}} \end{pmatrix} + S_{jk,PL_{jk}-1}\begin{pmatrix} x_{jk,t-PL_{jk}+2} \\ -i_{jk,t-PL_{jk}+2} \\ +i_{jk,t-PL_{jk}+1} \end{pmatrix} \\ +\ldots+S_{jk2}\begin{pmatrix} x_{jk,t-1} \\ -i_{jk,t-1} \\ +i_{jk,t-2} \end{pmatrix} + S_{jk1}\begin{pmatrix} x_{jkt} \\ +i_{jk,t-1} \end{pmatrix} + us_t - os_t \end{array}\right) = S_0 \tag{5.49}$$

$$\forall \ \ t = 0,1,\ldots,T-1; v \in V_m; m_v \in M_N$$

To enable accurate modeling of the hours of support consumed by all products, *jk*, held in beginning inventory, bin_{jk}, Equation (5.49) should be modified when any product, *jk*, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period $t = 0$ through the end of the maximum product life length, *max* PL_{jk} (i.e., $t = 0,\ldots,\max PL_{jk}$).

5.7.18. Maximum Allowable Support Parts Cost in a Single Period

Repair costs due to parts in period t can never exceed the maximum amount, P_t, allocated for this purpose. Repair costs are based upon the average cost (in constant dollars) of parts, $p_{jk\tau}$, adjusted to time t dollars required to repair a product of age $\tau = 1,\ldots,PL_{jk}$ in period t multiplied by the number of products of age $\tau = 1,\ldots,PL_{jk}$. Costs are adjusted by production levels, x_{jkt}, and inventory levels in periods t and t-1. This constraint is given by Equation (5.50).

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k}\begin{pmatrix} p_{jk,PL_{jk}}\left(x_{jk,t-PL_{jk}+1} - i_{jk,t-PL_{jk}+1} + i_{jk,t-PL_{jk}}\right) \\ +p_{jk,PL_{jk}-1}\left(x_{jk,t-PL_{jk}+2} - i_{jk,t-PL_{jk}+2} + i_{jk,t-PL_{jk}+1}\right) \\ +\ldots+p_{jk2}\left(x_{jk,t-1} - i_{jk,t-1} + i_{jk,t-2}\right) \\ +p_{jk1}\left(x_{jkt} + i_{jk,t-1}\right) \end{pmatrix}\left(1+\overline{f}\right)^t \le P_t \quad (5.50)$$

$$\forall\ t = 0,1,\ldots,T-1$$

To enable accurate modeling of the parts repair costs due to all products, *jk*, held in beginning inventory, bin_{jk}, Equation (5.50) should be modified when any product, *jk*, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period $t = 0$ through the end of the maximum product life length, *max* PL_{jk} (i.e., $t = 0,\ldots,\max PL_{jk}$).

5.7.19. Minimum Acceptable Revenue

The organization receives revenues from three sources. These are: (1) sales of new products, r_{jkt}, (2) labor, f_t, to repair products that are out of warranty, and (3) parts, PR_t, to repair products that are out of warranty. The revenues are adjusted by inventory

levels in each period. The organization may choose to impose a constraint that the sum of the revenues from these three sources in period t must be at least as much as the minimum acceptable revenue, R_t, for period t. This constraint is given by Equation (5.51).

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k}\left(\begin{array}{l} r_{jkt}\left(x_{jkt}-i_{jkt}+i_{jk,t-1}\right) \\ +\left(\begin{array}{l} PR_{jk,PL_{jk}}\left(1+\overline{f}\right)^t \\ +f_t S_{jk,PL_{jk}} \end{array}\right)\left(\begin{array}{l} x_{jk,t-PL_{jk}+1} \\ -i_{jk,t-PL_{jk}+1} \\ +i_{jk,t-PL_{jk}} \end{array}\right) \\ +\left(\begin{array}{l} PR_{jk,PL_{jk}-1}\left(1+\overline{f}\right)^t \\ +f_t S_{jk,PL_{jk}-1} \end{array}\right)\left(\begin{array}{l} x_{jk,t-PL_{jk}+2} \\ -i_{jk,t-PL_{jk}+2} \\ +i_{jk,t-PL_{jk}+1} \end{array}\right) \\ +\ldots+\left(\begin{array}{l} PR_{jk,PW_{jk}+1}\left(1+\overline{f}\right)^t \\ +f_t S_{jk,PW_{jk}+1} \end{array}\right)\left(\begin{array}{l} x_{jk,t-PW_{jk}} \\ -i_{jk,t-PW_{jk}} \\ +i_{jk,t-PW_{jk}-1} \end{array}\right) \end{array}\right) \geq R_t \qquad \forall\ t=0,1,\ldots,T-1 \qquad (5.51)$$

The sales and support revenues obtained from each product, *jk*, held in beginning inventory, bin_{jk}, and sold in period $t = 0$ must be included in each period specific minimum acceptable revenue constraint until the product, *jk*, has reached the end of its life, PL_{jk}. Thus, the revenues given by Equation (5.51a) must be added to the left hand side of the $t = 0$ minimum revenue constraint.

$$\sum_{k=0}^{K}\sum_{j=0}^{J_k} r_{jk0} bin_{jk} \qquad (5.51a)$$

Further, to enable accurate modeling of the labor and parts repair revenues obtained from all products, *jk*, held in beginning inventory, bin_{jk}, Equation (5.51) should be modified when any product, *jk*, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period $t = 0$ through the end of the maximum product life length, *max* PL_{jk} (i.e., $t = 0,\ldots,\max PL_{jk}$).

5.8. Stage Two Objective Function Formulation

Stage two finds the largest production quantities possible given production quantities determined in stage one and the constraints imposed by known production and support capital budgeting projects. The objective function has been divided into five components. The objective is to maximize the future worth of revenues from sales and support less the future worth of the direct costs of production. Table 5.3 shows all revenue and cost components accounted for in stage two of the sequential model.

Table 5.3. Revenue and Cost Components for Stage Two.

Revenue Components	Cost Components
1. The future worth of revenues from the sales of new products - $FWNP_T$. 2. The future worth of revenues from labor to repair defective, out of warranty products - FWR_T.	1. The future worth of labor repair costs - $FWLC_T$. 2. The future worth of all production material costs - $FWMA_T$.
1. The future worth of parts repair revenues less costs - $FWPR_T$.	

The objective of stage two is to

$$\text{maximize } \{FWNP_T + FWR_T + FWPR_T - FWLC_T - FWMA_T\} \qquad (5.52)$$

where $FWNP_T$, FWR_T, $FWLC_T$, $FWPR_T$, and $FWMA_T$ are given by Equations (5.2), (5.13), (5.14), (5.15), and (5.18) respectively while meeting the constraints given by Equations (5.19), (5.20), (5.21), (5.22), (5.24). (5.25), (5.26), (5.28), (5.29), (5.30), (5.31), (5.33), (5.34), (5.35), (5.36), (5.37), (5.38), (5.39), (5.40), (5.41), (5.42), (5.43), (5.44), (5.51), (5.53), (5.54), (5.55), (5.56), (5.57), (5.58), (5.59), (5.60), (5.61), (5.62), (5.63), (5.64), (5.65), (5.66), (5.67), (5.68), and (5.69).

5.9. Stage Two Constraint Formulation

Stage two of the sequential model includes 25 types of constraints. These constraint types fall within product specific, production specific, and capital budgeting categories. The constraint types and their respective categories are shown in Table 5.4.

Table 5.4. Constraints Considered by Stage Two of the Sequential Model.

Production Specific Constraints - 1. Production Set to Zero if a Product is Not Funded 2. Total Allowable Production in Period *t* 3. Production Work Center Constraint 4. Production Setup Constraints 5. Determination of Values for Regular and Overtime Labor 6. Maximum Regular Time Production Labor 7. Maximum Production Overtime Labor 8. Production Labor Balancing Parameter 9. Maximum Production Cost
Product Specific Constraints - 10. Product Quantity Constraint 11. Product Exclusivity Constraints 12. Product Contingency Constraints 13. Product Must Fund Constraint 14. Product Support During Production and Utilization 15. Maximum Allowable Support Parts Cost in a Single Period 16. Minimum Acceptable Revenue 17. Maximum Inventory in Each Period

Table 5.4. Continued.

Capital Budgeting Specific Constraints -
18. Maximum Budget for Production Capital Budgeting Constraint
19. Maximum Budget for Support Capital Budget Constraint
20. Production Capital Budgeting Mutually Exclusive Project Constraint
21. Support Capital Budgeting Mutually Exclusive Project Constraint
22. Production Capital Budgeting Project Contingency Constraint
23. Support Capital Budgeting Project Contingency Constraint
24. Production Capital Budgeting Project Must Fund Constraint
25. Support Capital Budgeting Project Must Fund Constraint

5.9.1. Production Work Center Constraint

The production work center constraint, given by Equation (5.53), simultaneously considers: (1) production work center *l*'s base capacity, B_l, in hours, (2) the hours of capacity, $C_{m_l t}$, that can be added by the acceptance of production capital budgeting project e_{m_l}, and (3) the time in hours required to setup and produce products *jk* in period *t*. $\delta(x_{jkt})$ is a zero-one decision variable for the setup of product *jk* at all work centers required to produce product *jk* (i.e., if setup occurs at one work center, than it must occur at all work centers), a_{jkl} is the amount of hours required to setup product *jk* at work center *l*, and h_{jkl} is the time in hours to perform the process required to produce one unit of product *jk* at work center l.

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k}\left(a_{jkl}\delta(x_{jkt})+h_{jkl}x_{jkt}\right)-\sum_{m_l=1}^{M_L}e_{m_l}C_{m_l t}\le B_l \quad \forall\, l\in L; t\in T; e_{m_l}\in 0,1 \qquad (5.53)$$

5.9.2. Maximum Regular Time Production Labor

The maximum regular time production labor constraint, given by Equation (5.54), limits the value of regular time labor by allowing w_{lt} to be no greater than the number of hours available at work center *l* multiplied by λ_{lt}, the maximum fraction of time at work center *l* that may be charged as regular time. Note that it is still possible for w_{lt} to take on

values that are less than the maximum work center capacity multiplied by λ_{lt}. This simply means that excess capacity exists at the work center, and is not used in that period.

$$w_{lt} - \lambda_{lt}(B_l + \sum_{m_l=1}^{M_L} e_{m_l} C_{m_l t}) \leq 0 \quad \forall\ l \in L;\ t \in T \tag{5.54}$$

The constraint given by Equation (5.55) limits values of w_{lt} to zero or greater, and also places w_{lt} in the real number set.

$$w_{lt} \geq 0;\ \ w_{lt} \in R_n \tag{5.55}$$

5.9.3. Maximum Production Overtime Labor

The maximum production overtime constraint, given by Equation (5.56), limits the value of overtime time labor o_{lt} by allowing it to be no greater than the number of hours available at work center l multiplied by $(1-\lambda_{lt})$, where λ_{lt} is the maximum fraction of time at work center l that may be charged as regular time. Note it is still possible for o_{lt} to take on values that are less than the maximum work center capacity multiplied by $(1-\lambda_{lt})$. This simply means that excess capacity exists at the work center, and is not used in that period.

$$o_{lt} - (1-\lambda_{lt})(B_l + \sum_{m_l=1}^{M_L} e_{m_l} C_{m_l t}) \leq 0 \quad \forall\ l \in L;\ t \in T \tag{5.56}$$

The constraint given by Equation (5.57) limits values of o_{lt} to zero or greater, and also places o_{lt} in the real number set.

$$o_{lt} \geq 0;\ \ o_{lt} \in R_n \tag{5.57}$$

5.9.4. Production Labor Balancing Parameter

The labor balancing parameter, given by Equation (5.58), is used to set the maximum available work center capacity that may be used at regular labor rates. λ_{lt} sets the maximum fraction of time at work center l that may be charged as regular time.

Hence, (1 - λ_{lt}) set the maximum amount of time at work center l that may be charged as overtime.

$$0 \le \lambda_{lt} \le 1 \quad \forall \; l \in L;\; t \in T;\; \lambda_{lt} \in R_n \tag{5.58}$$

5.9.5. Maximum Budget for Production Capital Budgeting Constraint

The production maximum budget constraint, given by Equation (5.59), requires that the sum of the period t costs, $c_{m_l t}$, of all production capital budgeting projects selected (i.e., $e_{m_l} = 1$) be less than the maximum amount, E_{t_l}, that can be spent on production capital budgeting projects in period t.

$$\sum_{m_l=1}^{M_L} c_{m_l t} e_{m_l} \le E_{t_l} \quad \forall \; t \in T; e_{m_l} \in 0,1 \tag{5.59}$$

5.9.6. Maximum Budget for Support Capital Budgeting Constraint

The support maximum budget constraint, given by Equation (5.60), requires that the sum of the period t costs, $c_{m_v t}$, of all support capital budgeting projects selected (i.e., $e_{m_v} = 1$) be less than the maximum amount, E_{t_v}, that can be spent on support capital budgeting projects in period t.

$$c_{m_v t} e_{m_v} \le E_{t_v} \quad \forall \; t \in T; e_{m_v} \in 0,1 \tag{5.60}$$

5.9.7. Production Capital Budgeting Project Must Fund Constraint

When an organization determines that a production capital budgeting project (e.g., e_{1_l}) must be accepted, it is said to be a must fund project. Since $e_{m_l} \in 0,1 \;\; \forall \;\; m_l \in M_L$, the constraint that accounts for must fund production capital budgeting projects is given by Equation (5.61).

$$e_{1_l} = 1 \tag{5.61}$$

5.9.8. Support Capital Budgeting Project Must Fund Constraint

When an organization determines that a support capital budgeting project (e.g., $e1_v$) must be accepted, it is said to be a must fund project. Since $e_{m_v} \in 0,1 \ \forall \ m_v \in M_V$, the constraint that accounts for must fund support capital budgeting projects is given by Equation (5.62).

$$e1_v = 1 \tag{5.62}$$

5.9.9. Capital Budgeting Production Mutually Exclusive Project Constraint

When two production capital budgeting projects, $e1_l$ and $e2_l$, cannot both be funded, they are mutually exclusive. Since $e_{m_l} \in 0,1 \ \forall \ m_l \in M_L$, then the constraint that represents two mutually exclusive production capital budgeting projects, Equation (5.63), is used.

$$e1_l + e2_l \leq 1 \tag{5.63}$$

5.9.10. Capital Budgeting Support Mutually Exclusive Project Constraint

When two support capital budgeting projects, $e1_v$ and $e2_v$, cannot both be funded, they are mutually exclusive. Since $e_{m_v} \in 0,1 \ \forall \ m_v \in M_V$, then the constraint that represents two mutually exclusive support capital budgeting projects, Equation (5.64), is used.

$$e1_v + e2_v \leq 1 \tag{5.64}$$

5.9.11. Production Capital Budgeting Project Contingency Constraint

When one production capital budgeting project, $e1_l$, is contingent upon the acceptance of another capital budgeting project, $e2_l$, project $e1_l$ is said to be contingent upon project $e2_l$. Since $e_{m_n} \in 0,1 \ \forall \ m_n \in M_N$, the constraint that represents the contingency relationship between two production capital budgeting projects, Equation (5.65), is used.

$$e_{1_l} - e_{2_l} \leq 0 \tag{5.65}$$

5.9.12. Support Capital Budgeting Project Contingency Constraint

When one support capital budgeting project, e_{1_v}, is contingent upon the acceptance of another capital budgeting project, e_{2_v}, project e_{1_v} is said to be contingent upon project e_{2_v}. Since $e_{m_v} \in 0,1 \ \forall \ m_v \in M_V$, the constraint that represents the contingency relationship between two support capital budgeting projects, Equation (5.66), is used.

$$e_{1_v} - e_{2_v} \leq 0 \tag{5.66}$$

5.9.13. Product Support During Production and Utilization

The number of hours spent repairing products in a period, t, can never exceed the amount of hours that are available for repair at the support work center. Thus, a constraint, given by Equation (5.67), to account for this must simultaneously consider the number of hours of support, $S_{jk\tau}$, required by an individual product of age t = 1,2,..., τ for each period, t. The total number of support hours, $S_{jk\tau}$, required for all products, jk, cannot exceed the base support, S_0, available along with the increases in available support, C_{m_v}, if support capital budgeting project e_{m_v} were funded.

$$\sum_{k=1}^{K}\sum_{j=1}^{J_k}\left(\begin{array}{l} S_{jk,PL_{jk}}\begin{pmatrix} x_{jk,t-PL_{jk}+1} \\ -i_{jk,t-PL_{jk}+1} \\ +i_{jk,t-PL_{jk}} \end{pmatrix} \\ +S_{jk,PL_{jk}-1}\begin{pmatrix} x_{jk,t-PL_{jk}+2} \\ -i_{jk,t-PL_{jk}+2} \\ +i_{jk,t-PL_{jk}+1} \end{pmatrix} \\ +\ldots+S_{jk2}\begin{pmatrix} x_{jk,t-1} \\ -i_{jk,t-1} \\ +i_{jk,t-2} \end{pmatrix} \\ +S_{jk1}\begin{pmatrix} x_{jkt} \\ +i_{jk,t-1} \end{pmatrix} - \sum_{m_v=1}^{M_v} e_{m_v} C_{m_v,t} \end{array}\right) \leq S_0 \qquad \forall \ t = 0,1,\ldots,T-1; v \in V_m; m_v \in M_N \tag{5.67}$$

To enable accurate modeling of the hours spent repairing all products, *jk*, held in beginning inventory, bin_{jk}, Equation (5.67) should be modified when any product, *jk*, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period $t = 0$ through the end of the maximum product life length, $max\ PL_{jk}$ (i.e., $t = 0,\ldots,\max PL_{jk}$).

5.9.14. Stage Two Product Quantity Constraint

The stage two product quantity constraint, given by Equation (5.68), limits the production quantity of each product, *jk*, in each period, *t*. This is accomplished by forcing x_{jkt} to be less than or equal to the estimated production quantities, $X1_{jkt}$, supplied by stage one. The product quantity constraint makes it possible for stage two to find the largest possible production quantities with respect to the anticipated returns from products funded in stage one.

$$x_{jkt} \le X1_{jkt} \qquad \forall \ \ t = t_{b_{jk}},\ldots,t_{e_{jk}}; j \in J_K; k \in K \tag{5.68}$$

5.9.15. Products Funded Constraint

The products funded constraint, given by Equation (5.69), specifies the products that will be funded (i.e., $z_{jk} = 1$) and those that will not be funded (i.e., $z_{jk} = 0$). Values for Z_{jk} are determined in stage one and used to set the values of product funding variables z_{jkt} in stages two and three.

$$z_{jk} = Z_{jk} \qquad \forall\ j \in J_K; k \in K \tag{5.69}$$

5.10. Stage Three Objective Function Formulation

Stage three finds the minimum cost of production and support given production quantities determined in stage two and the constraints imposed by known production and support capital budgeting projects. The objective function has been divided into five components. The objective is to minimize the future worth of the costs of production and support. Table 5.5 shows all cost components accounted for in stage three of the sequential model.

Table 5.5. Cost Components for Stage Three.

1. The future worth of inventory holding costs - FWI_T
2. The future worth of labor production costs - FWL_T
3. The future worth of the labor costs to repair products - $FWLC_T$
4. The future worth of all production material costs - $FWMA_T$
5. The future worth of the cost of base production capacity - FWP_T
6. The future worth of the base support capacity - FWS_T
7. The future worth of the costs of parts to repair products - $FWPRC_T$
8. The future worth of production capital budgeting projects - $FWCP_T$
9. The future worth of support capital budgeting projects - $FWCS_T$

The objective of stage three, given by Equation (5.70), is to

$$\text{minimize} \begin{Bmatrix} FWI_T + FWL_T + FWLC_T + FWMA_T + FWP_T \\ + FWS_T + FWPRC_T + FWCP_T + FWCS_T \end{Bmatrix} \tag{5.70}$$

where FWI_T, FWL_T, $FWLC_T$, $FWMA_T$, FWP_T, FWS_T, $FWPRC_T$, $FWCP_T$, and $FWCS_T$ are given by Equations (5.8), (5.10), (5.14), (5.18), (5.71), (5.72), (5.73), (5.74), and (5.75) respectively while subject to the constraints given by Equations (5.19), (5.20), (5.21), (5.22), (5.24). (5.25), (5.26), (5.28), (5.29), (5.30), (5.31), (5.33), (5.34), (5.35), (5.36), (5.37), (5.38), (5.39), (5.40), (5.41), (5.42), (5.43), (5.44), (5.51), (5.53), (5.54), (5.55), (5.56), (5.57), (5.58), (5.59), (5.60), (5.61), (5.62), (5.63), (5.64), (5.65), (5.66), (5.67), (5.69), and (5.76).

5.10.1. Future Worth of Base Production Costs

It is assumed that the organization may not reduce the amount of specific production work center capacity, B_l, that is possessed at the end of period zero during the optimization. Thus, the organization will incur a cost, b_l, for base production capability at each production work center, l, for each period of the analysis $t = 0,1,2,...,T$. Since the base production capacity cost, b_l, for production work center l is given in period t = 0 dollars, it is necessary to use constant dollar analysis (i.e., addition of an inflation component) to determine the future worth of all production work center base capacity, FWP_T. This determination of future worth is given by Equation (5.71).

$$FWP_T = \sum_{t=0}^{T} \sum_{l=1}^{L} b_l \left(1 + \overline{f}\right)^t (1 + ACC)^{T-t} \tag{5.71}$$

5.10.2. Future Worth of Base Support Costs

It is assumed that the organization may not reduce the amount of support capacity, S_0, that is possessed at the end of period zero during the optimization. Thus, the organization will incur a cost, s_0, for base support capacity for each period of the analysis $t = 0,1,2,...,T$. Since the base support capacity cost, s_0, is given in period $t = 0$ dollars, it is necessary to use constant dollar analysis (i.e., no inflation component) to determine the future worth of all production work center base capacity. The future worth of base support costs, FWS_T, is given by Equation (5.72).

$$FWS_T = \sum_{t=0}^{T} s_0 \left(1+\overline{f}\right)^t (1+ACC)^{T-t} \tag{5.72}$$

5.10.3. Future Worth of Parts Repair Costs

Repair costs due to parts are based upon the average cost (in constant dollars) of parts, $p_{jk\tau}$, required to repair a product of age $\tau = 1,\ldots,PL_{jk}$ in period t multiplied by the number of products of age $\tau = 1,\ldots,PL_{jk}$. This future worth of parts repair costs, $FWPRC_T$, is given by Equation (5.73).

$$FWPRC_T = \sum_{k=1}^{K}\sum_{j=1}^{J_k}\sum_{t=t_{bjk}}^{\left(t_{ejk}+PL_{jk}\right)} \left(\begin{array}{l} p_{jk,PL_{jk}} \begin{pmatrix} x_{jk,t-PL_{jk}+1} \\ -i_{jk,t-PL_{jk}+1} \\ +i_{jk,t-PL_{jk}} \end{pmatrix} \\ +p_{jk,PL_{jk}-1} \begin{pmatrix} x_{jk,t-PL_{jk}+2} \\ -i_{jk,t-PL_{jk}+2} \\ +i_{jk,t-PL_{jk}+1} \end{pmatrix} \\ +\ldots+p_{jk2} \begin{pmatrix} x_{jk,t-1} \\ -i_{jk,t-1} \\ +i_{jk,t-2} \end{pmatrix} \\ +p_{jk1}\left(x_{jkt}+i_{jk,t-1}\right) \end{array} \right) \left(1+\overline{f}\right)^t (1+ACC)^{T-t} \tag{5.73}$$

To enable accurate modeling of the parts repair cost from all products, *jk*, held in beginning inventory, bin_{jk}, Equation (5.73) should be modified when any product, *jk*, appears in beginning inventory (i.e., $bin_{jk} > 0$). This modification is done by setting all x_{jk0} equal to $x_{jk0} + bin_{jk}$, and is required in period $t = 0$ through the end of the maximum product life length, $max\ PL_{jk}$ (i.e., $t = 0,\ldots,\max PL_{jk}$).

5.10.4. Future Worth of Product Capital Budget Project Costs

Each production capital budgeting project, e_{m_l}, will have associated with it the total cost, $c_{m_l t}$, the organization would incur in each period, *t*, if the project is funded. The future worth of the total cost the organization will incur for all production capital budgeting projects, $FWCP_T$, is given by Equation (5.74).

$$FWCP_T = \sum_{m_l=1}^{M_L} \sum_{t=0}^{T} e_{m_l} c_{m_l t} (1 + ACC)^{T-t} \tag{5.74}$$

5.10.5. Future Worth of Support Capital Budget Project Costs

Each support capital budgeting project, e_{m_v}, will have associated with it the total cost, $c_{m_v t}$, the organization would incur in each period, *t*, if the project is selected. The future worth of the total cost the organization will incur for all support capital budgeting projects, $FWCS_T$, is given by Equation (5.75).

$$FWCS_T = \sum_{m_v=1}^{M_V} \sum_{t=0}^{T} e_{m_v} c_{m_v t} (1 + ACC)^{T-t} \tag{5.75}$$

5.11. Stage Three Constraint Formulation

Stage three of the sequential model includes 25 types of constraints. These constraint types fall within product specific, production specific, and capital budgeting categories. The constraint types and their respective categories are shown in Table 5.6.

Table 5.6. Constraints Considered by Stage Three of the Sequential Model.

Production Specific Constraints -

1. Production Set to Zero if a Product is Not Funded
2. Total Allowable Production in Period *t*
3. Production Work Center Constraint
4. Production Setup Constraints
5. Determination of Values for Regular and Overtime Labor
6. Maximum Regular Time Production Labor
7. Maximum Production Overtime Labor
8. Production Labor Balancing Parameter
9. Maximum Production Cost

Product Specific Constraints -

10. Product Quantity Constraint
11. Product Exclusivity Constraints
12. Product Contingency Constraints
13. Product Must Fund Constraint
14. Product Support During Production and Utilization
15. Maximum Allowable Support Parts Cost in a Single Period
16. Minimum Acceptable Revenue
17. Maximum Inventory in Each Period

Capital Budgeting Specific Constraints -

18. Maximum Budget for Production Capital Budgeting Constraint
19. Maximum Budget for Support Capital Budget Constraint
20. Production Capital Budgeting Mutually Exclusive Project Constraint
21. Support Capital Budgeting Mutually Exclusive Project Constraint
22. Production Capital Budgeting Project Contingency Constraint
23. Support Capital Budgeting Project Contingency Constraint
24. Production Capital Budgeting Project Must Fund Constraint
25. Support Capital Budgeting Project Must Fund Constraint

5.11.1. Stage Three Product Quantity Constraint

The stage three product quantity constraint, given by Equation (5.76), limits the production quantity, x_{jkt}, of each product, jk, in each period, t. This is accomplished by forcing x_{jkt} to be equal to the production quantities, $X2_{jkt}$, supplied by stage two. The constraint makes it possible for stage three to find the minimum cost of production and support.

$$x_{jkt} = X2_{jkt} \qquad \forall \quad t = t_{b_{jk}}, \ldots, t_{e_{jk}}; j \in J_K; k \in K \tag{5.76}$$

Note that if production quantities of x_{jkt} were not predetermined in stage two, the minimum cost solution would be found where $x_{jkt} = 0 \ \forall \quad t = t_{b_{jk}}, \ldots, t_{e_{jk}}; j \in J_K; k \in K$.

6. HYPOTHETICAL EXAMPLE AND PROGRAMS

The procedure used to compare the concurrent and sequential planning models embraces three components. The first component includes a realistic hypothetical example composed of a set of hypothetical inputs. These inputs allow the comparison of the results obtained from all three versions of the concurrent and sequential product, production, and capacity planning models. The example includes 873 inputs that cover five products, one pre-production/post-support capital budgeting project, three production capital budgeting projects, and one support capital budgeting project. The input set spans a twelve year planning horizon. The second component includes five computer programs developed in C using Borland's C/C++ compiler. Four of the programs supply inputs to LINDO/386 (a linear mixed-integer programming software package) while one program determines the future worth obtained from the sequential model. The third component consists of an analysis of the results obtained while testing the three versions of the concurrent and sequential models with various values of product sub-group demand. Results obtained from both models are critically compared to demonstrate that the concurrent model provides the best results.

This chapter is divided into two major sections. The first section describes the hypothetical planning example. The second section discusses the software and requisite interfaces. The comparison of results obtained from tests of the three versions of the concurrent and sequential models are given in Chapter 7.

6.1. Hypothetical Example

The hypothetical example spans a twelve year planning horizon, and contains inputs for five products, one pre-production/post-support capital budgeting project, three production capital budgeting projects, and one support capital budgeting project. Construction of the hypothetical example required that 873 realistic inputs be developed and placed in an IBM PC DOS-based computer file that could be read by the Borland C/C++ programs that were developed to compare the concurrent and sequential models. Appendix A contains a complete listing of this set of inputs.

6.1.1. Product Life Cycles

The example presented in this chapter consists of the inputs required to solve a hypothetical twelve year product, production, and capacity planning problem for five products in two product sub-groups. The example also includes all inputs for one pre-

production/post-support capital budgeting project, three production capital budgeting projects, and one support capital budgeting project.

The life-cycle phases for each product are shown in Figure 6.1. Products z_{00}, z_{01}, and z_{02}, reside in product sub-group $k = 0$ while the remaining two products, z_{10} and z_{11}, reside in product sub-group $k = 1$. Further, products z_{01} and z_{02} are mutually exclusive. That is, if $z_{01} = 1$ then $z_{02} = 0$ and if $z_{02} = 1$ then $z_{01} = 0$. Products z_{00} and z_{01} were produced in the previous fiscal period. Because of this, they both must at least receive funding for their respective phase-out life-cycle phases. Thus, $z_{00} = 1$ and $z_{10} = 1$.

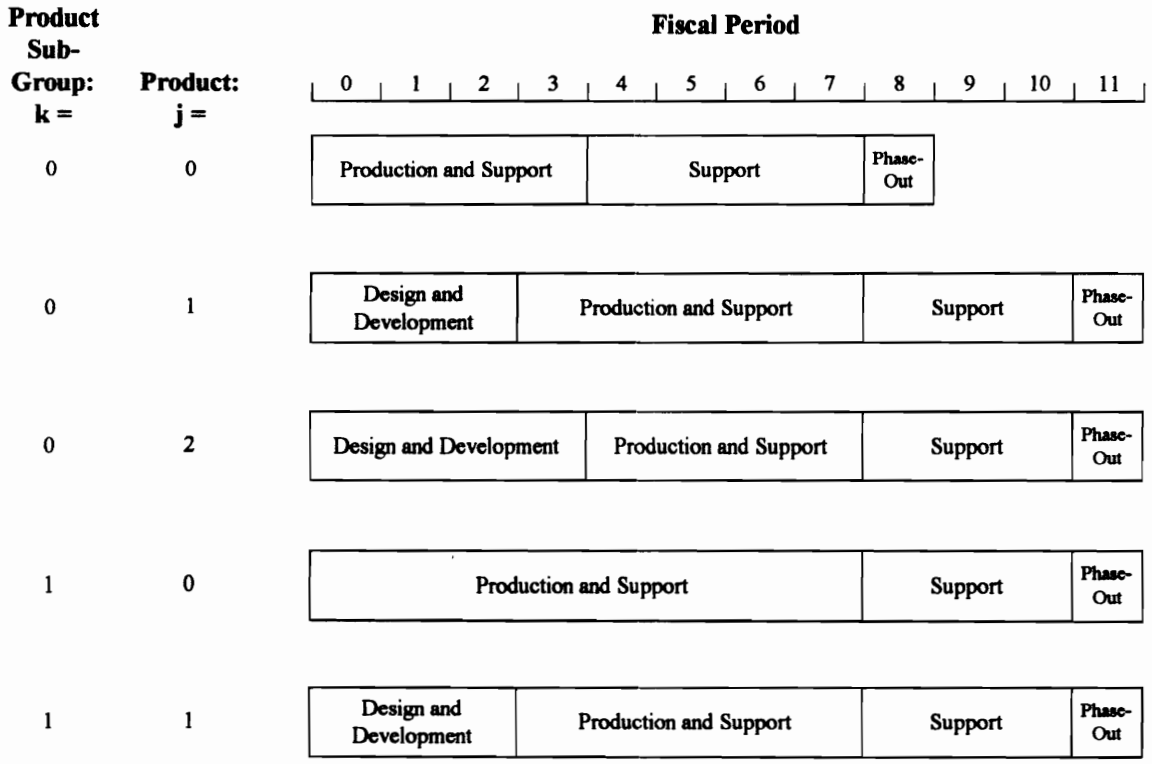

Figure 6.1. Life Cycles of Five Products.

This hypothetical example allows both the concurrent and sequential models to determine the set of products that should be funded, the set of products not funded, and the production quantities in each period. Because production is determined in each period, it is possible to cease production of a product prior to phase-out. Thus, it is possible to abandon one product when another replacement product becomes available. As an example, it is possible to produce product z_{10} through fiscal period $t = 2$. At that

point, the production of z_{10} may cease, and production of product z_{11} may begin. Thus, product z_{11} has replaced product z_{10}. Note, though, that it is also possible to produce z_{10} and z_{11} simultaneously.

6.1.2. Demand for Product Subgroups

The total demand, d_{kt}, for all products in subgroup k in a fiscal period, t, is considered to be uniform and deterministic throughout the planning horizon. Since demand is uniform in each fiscal period, t, for each product sub-group, k, a variable, D_k, to represent demand for each product subgroup, k, was developed such that:

$$D_k = d_{k0} = d_{k1} = \ldots = d_{kT_e} \quad \forall \quad k \in K \tag{6.1}$$

All versions of the concurrent and sequential models were tested by permitting each D_k to take on three levels of demand. These levels were low ($D_k = 50$), medium ($D_k = 100$), and high ($D_k = 150$). Because there were three levels of demand and two product sub-groups, there existed nine **combinations** of demand across the sub-groups ($3^2 = 9$). These combinations are shown in Table 6.1.

6.1.4. Average Cost of Capital

The long-run average cost of capital ACC is set such that $ACC = 0.10$.

6.1.5. Average Rate of Inflation

The average rate of inflation $\overline{f}$ is set such that $\overline{f} = 0.050$.

6.1.6. Must Fund

Since products z_{00} and z_{10} were produced in previous periods, both models must at least fund the phase-out periods for both of these products. Because of this, the input set includes a component such that $z_{00} = 1$ and $z_{10} = 1$.

6.1.3. Life Lengths of Each Product

The life length in periods for one unit of product j in substitute product group k is shown in Table 6.2 below.

6.1.7. Mutually Exclusive Products

The hypothetical example contains a set of inputs that control product mutual exclusivity. There is one mutually exclusive input for each product. The inputs are base 10 conversions of the binary relationships that exist between a product and all other products.

Table 6.1. Combinations of Product Sub-Group Demand.

Demand Combination	Product Sub-Group	Demand for Each Sub-Group in Each Period							
		d_{k0}	d_{k1}	d_{k2}	d_{k3}	d_{k4}	d_{k5}	d_{k6}	d_{k7}
1	0	50	50	50	50	50	50	50	50
	1	50	50	50	50	50	50	50	50
2	0	50	50	50	50	50	50	50	50
	1	100	100	100	100	100	100	100	100
3	0	50	50	50	50	50	50	50	50
	1	150	150	150	150	150	150	150	150
4	0	100	100	100	100	100	100	100	100
	1	50	50	50	50	50	50	50	50
5	0	100	100	100	100	100	100	100	100
	1	100	100	100	100	100	100	100	100
6	0	100	100	100	100	100	100	100	100
	1	150	150	150	150	150	150	150	150
7	0	150	150	150	150	150	150	150	150
	1	50	50	50	50	50	50	50	50
8	0	150	150	150	150	150	150	150	150
	1	100	100	100	100	100	100	100	100
9	0	150	150	150	150	150	150	150	150
	1	150	150	150	150	150	150	150	150

Table 6.2. Product Life Lengths in Years.

Product Sub-Group	Product	Product Life Length
0	0	4
0	1	4
0	2	4
1	0	4
1	1	4

Table 6.3 shows the binary relationships between a product and all other products, and the base 10 conversion of the resulting binary number. If two products are mutually exclusive, then a 1 will be entered for their relationship. If no mutual exclusivity exists,

then a 0 will be entered for the product relationship. After the binary relationship has been determined for each product, the binary number is converted to a base 10 number. The base 10 conversion forms the input to the Borland C/C++ programs[6.1].

Table 6.3. Product Mutual Exclusivity.

	Product Sub-Group	1	1	0	0	0	
Product Sub-Group	Product	1	0	2	1	0	Base 10 Conversion
0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	4
0	2	0	0	0	1	0	2
1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0

6.1.8. Mutually Exclusive Production Capital Budgeting Projects

The hypothetical example contains a set of inputs that control production project mutual exclusivity. There is one mutually exclusive input for each production project. The inputs are base 10 conversions of the binary relationships that exist between a production project and all other production projects.

Table 6.4 shows the binary relationships between a production project and all other production projects. The table also shows the base 10 conversion of the binary number formed by each relationship. If two production projects are mutually exclusive, then a 1 will be entered for their relationship. If no mutual exclusivity exists, then a 0 will be entered for the production project relationship. After the binary relationship has been

---

6.1 The Borland C/C++ programs developed during this research read the product mutually exclusive information contained in the input file. However, since the programs were developed to solve problems using only this hypothetical example, it was later determined that the product mutually exclusive constraint could be more easily modeled internally (i.e., rather than requiring external inputs) within the programs. Thus, the product mutually exclusive input set is not required to represent the product mutually exclusive constraint. However, a decision was made to keep the product mutually exclusive inputs in the input set so that the relationships between products are made explicit.

determined for each production project, the binary number is converted to a base 10 number. The base 10 conversion forms the input to the Borland C/C++ programs[6.2].

Table 6.4. Production Project Mutual Exclusivity.

Production Project	2	1	0	Base 10 Conversion
0	1	0	0	4
1	0	0	0	0
2	0	0	1	1

6.1.9. Product Warranty Length

The warranty length, PW_{jk}, in periods for one unit of product j in substitute product group k is shown in Table 6.5.

Table 6.5. Product Warranty Lengths in Years.

Product Sub-Group	Product	Product Warranty Length
0	0	2
0	1	2
0	2	2
1	0	2
1	1	2

6.1.10. Product Contingency

The hypothetical example contains a set of inputs that control product contingencies. There is one contingency input for each product. The inputs are base 10 conversions of the binary relationships that exist between a product, itself, and all other products.

Table 6.6 shows the binary relationships between a product and all other products, and the base 10 conversion of the resulting binary number. If one product is contingent on another product, then a 1 will be entered for their relationship. If no contingency exists,

---

6.2 The Borland C/C++ programs developed during this research read the mutually exclusive production project information contained in the input file. This information is read but not used in the programs for the same reasons given in footnote 6.1.

then a 0 will be entered for the product relationship. Finally, each product is contingent upon itself. Thus, a one is always entered for the relationship between a product and itself.

Table 6.6. Product Contingencies.

	Product Sub-Group	1	1	0	0	0	
Product Sub-Group	Product	1	0	2	1	0	Base 10 Conversion
0	0	0	0	0	0	1	1
0	1	0	0	0	1	1	3
0	2	0	0	1	0	1	5
1	0	0	1	0	0	0	8
1	1	1	1	0	0	0	24

After the binary relationship has been determined for each product, the binary number is converted to a base 10 number. The base 10 conversion forms the input to the Borland C/C++ programs[6.3].

6.1.11. Base Hours at Production Work Centers

Table 6.7 shows the beginning hours, B_l, available at each production work center, l, in period $t = 0$.

Table 6.7. Base Hours at Production Work Centers.

Work Center	0	1	2
Base Hours	8320	8320	8320

6.1.12. Earliest Period Production of a Product May Commence

Table 6.8 shows, $t_{b_{jk}}$, the first period that product j in substitute product group k may be produced.

6.3 The Borland C/C++ programs developed during this research read the product contingency inputs contained in the input file. This information is read but not used in the programs for the same reasons given in footnote 6.1.

Table 6.8. Earliest Period Production of a Product May Commence.

Product Sub-Group	Product	Earliest Period Production May Commence
0	0	0
0	1	3
0	2	4
1	0	0
1	1	3

6.1.13. Maximum Number of Production Periods for Each Product

Table 6.9 shows the last period that product j in substitute product group k may be produced. The Borland C/C++ programs add these numbers to $t_{b_{jk}}$ to determine, $t_{e_{jk}}$, the latest period that a product may be produced.

Table 6.9. Maximum Number of Production Periods.

Product Sub-Group	Product	Maximum Number of Production Periods
0	0	4
0	1	5
0	2	4
1	0	8
1	1	5

6.1.14. Revenue from the Sale of Products in Each Period

The revenue in actual dollars, r_{jkt}, from the sale of one unit of product j in substitute product group k in period t is shown in Table 6.10. A product can only have revenues associated with it in periods that it can be produced. In periods when a product cannot be produced, product revenues are not applicable (i.e., N/A) as noted in Table 6.9[6.4].

[6.4] Because rectangular arrays were used in the Borland C/C++ programs, a 0 was placed in the input file for periods in which a product could not be produced (i.e., N/A). This procedure was consistent throughout the development of the programs. Hence, any time an N/A appears in a table, it was replaced by a zero in the input file.

Table 6.10. Revenue from the Sale of Each Product.

	Period	0	1	2	3	4	5	6	7
Product Sub-Group	**Product**	**Revenue from Each Product Sold**							
0	0	60,000	61,000	62,000	63,000	N/A	N/A	N/A	N/A
0	1	N/A	N/A	N/A	61,500	62,500	63,500	64,500	65,500
0	2	N/A	N/A	N/A	N/A	62,250	63,250	64,250	65,250
1	0	59,500	59,500	59,500	59,500	59,500	59,500	59,500	59,500
1	1	N/A	N/A	N/A	62,250	63,250	64,250	65,250	66,250

6.1.15. Setup Hours for Each Product

Since there are five products and three production work centers, there are $(3x5) = 15$ inputs to account for the time in hours, a_{jkl}, required to setup each production work center, l, for a production run of each product, j, in substitute product sub-group k. These times are shown in Table 6.11.

Table 6.11. Setup Hours at Each Work Center.

Product Sub-Group	0	0	0	1	1
Product	0	1	2	0	1
Work Center	**Setup Hours Required**				
0	35	40	30	60	45
1	100	78	65	95	78
2	45	40	30	50	25

6.1.16. Production Hours for Each Product

Since there are five products and three production work centers, there are $(3x5) = 15$ inputs to account for the time in hours, h_{jkl}, required to produce each unit of product j in substitute product sub-group k at each production work center, l. These times are shown in Table 6.12.

Table 6.12. Production Hours at Each Work Center.

Product Sub-Group	0	0	0	1	1
Product	0	1	2	0	1
Work Center	Production Hours Required				
0	70	62	50	85	65
1	95	72	60	69	55
2	50	45	30	60	40

6.1.17. Regular Rate Labor Costs in Each Production Period

The cost per hour of regular time labor, W_{lt}, at each production work center, l, in each production period $t = 0,1,\ldots,T_e$ is shown in Table 6.13.

Table 6.13. Regular Time Labor Cost in Each Period.

Period	0	1	2	3	4	5	6	7
Work Center	Regular Time Labor Costs in Each Period							
0	30.00	30.00	30.00	30.00	30.00	30.00	30.00	32.00
1	35.00	35.00	35.00	35.00	35.00	35.00	35.00	38.00
2	25.00	25.00	25.00	25.00	25.00	25.00	25.00	27.00

6.1.18. Overtime Labor Costs in Each Production Period

The cost per hour of overtime labor, O_{lt}, at each production work center, l, in each production period $t = 0,1,\ldots,T_e$ is shown in Table 6.14.

Table 6.14. Overtime Labor Cost in Each Period.

Period	0	1	2	3	4	5	6	7
Work Center	Overtime Labor Costs in Each Period							
0	45.00	45.00	45.00	45.00	45.00	45.00	45.00	48.00
1	52.50	52.50	52.50	52.50	52.50	52.50	52.50	57.00
2	37.50	37.50	37.50	37.50	37.50	37.50	37.50	40.50

6.1.19. Increases in Production Capacity from Project 0

If funded, production capital budgeting project zero, $e0_l$, will increase the capacity of each production work center, l, by a known amount, $C0_l t$, in each production period $t = 0,1,\ldots,T_e$. In this hypothetical example, $T_e = 7$. Consequently, a set of inputs is required to account for these potential increases. This set of inputs is shown in Table 6.15.

Table 6.15. Increases in Capacity if Project $e0_l$ is Funded.

Period	0	1	2	3	4	5	6	7
Work Center	Increases in Hours at Production Work Centers							
0	5000	5000	5000	5000	5000	5000	5000	5000
1	0	0	0	0	0	0	0	0
2	3000	3000	3000	3000	3000	3000	3000	3000

6.1.20. Increases in Production Capacity from Project 1

If funded, production capital budgeting project one, $e1_l$, will increase the capacity of each production work center, l, by a known amount, $C1_l t$, in each production period $t = 0,1,\ldots,T_e$ where $T_e = 7$. Consequently, a set of inputs is required to account for these potential increases. This set of inputs is shown in Table 6.16.

Table 6.16. Increases in Capacity if Project $e1_l$ is Funded.

Period	0	1	2	3	4	5	6	7
Work Center	Increases in Hours at Production Work Centers							
0	0	0	0	0	0	0	0	0
1	6240	6240	6240	6240	6240	6240	6240	6240
2	0	0	0	0	0	0	0	0

6.1.21. Increases in Production Capacity from Project 2

If funded, production capital budgeting project one, $e2_l$, will increase the capacity of each production work center, l, by a known amount, $C2_l t$, in each production period

$t = 0,1,\ldots,T_e$ where $T_e = 7$. Consequently, a set of inputs is required to account for these potential increases. This set of inputs is shown in Table 6.17.

Table 6.17. Increases in Capacity if Project e_{2_l} is Funded.

Period	0	1	2	3	4	5	6	7
Work Center	Increase in Hours at Production Work Centers							
0	9000	9000	9000	9000	9000	9000	9000	9000
1	0	0	0	0	0	0	0	0
2	4000	4000	4000	4000	4000	4000	4000	4000

6.1.22. Increases in Capacity from Pre-Production/Post Support Capital Budgeting Project

If funded, the pre-production/post-support capital budgeting project, e_{0_n}, will increase the capacity of each pre-production/post-support work center n = 1, 2,..., N by a known amount, $C_{0_n t}$, in each period $t = 0,1,\ldots,T$. In this hypothetical example, $N = 4$ and $T = 11$. The set of inputs required to account for these potential increases is shown in Table 6.18.

Table 6.18. Increases in Capacity if Project e_{0_n} is Funded.

Period	0	1	2	3	4	5	6	7	8	9	10	11
Work Center	Increase in Units at Pre-Production/Post-Support Work Centers											
0	50	50	50	50	50	50	50	50	50	50	50	50
1	75	75	75	75	75	75	75	75	75	75	75	75
2	50	50	50	50	50	50	50	50	50	50	50	50
3	30	30	30	30	30	30	30	30	30	30	30	30

6.1.23. Increase in Hours from Each Support Capital Budgeting Project

If funded, the support capital budgeting project, e_{0_v}, will increase the capacity of the support work center by a known amount, $C_{0_v t}$, in each period $t = 0,1,\ldots,T_s$. In this hypothetical example $T_s = 10$. The set of inputs required to account for these potential increases is shown in Table 6.19.

Table 6.19. Increase in Hours from Support Capital Budgeting Project.

Period	0	1	2	3	4	5	6	7	8	9	10
Hours	6000	6000	6000	6000	6000	6000	6000	6000	6000	6000	6000

6.1.24. Base Units of Capacity at Pre-Production/Post-Support Work Centers

Table 6.20 shows the beginning capacity, B_n, in some consistent units available at each pre-production and post-support work center, n.

Table 6.20. Base Units of Capacity at Pre-Production/Post-Support Work Centers.

Work Center	0	1	2	3
Base Hours	20.00	30.00	15.00	50.00

6.1.25. Cost of Production Capital Budgeting Projects in Each Period

Table 6.21 shows, $c_{m_l t}$, the cost of each production capital budgeting project, e_{m_l}, in each period $t = 0,1,\ldots,T$.

6.1.26. Cost of Pre-Production/Post-Support Project in Each Period

Table 6.22 shows, $c_{0_n t}$, the cost of the pre-production/post-support capital budgeting project, e_{m_n}, in each period $t = 0,1,\ldots,T$. This cost is uniform in each period.

6.1.27. Cost of Support Capital Budgeting Project in Each Period

Table 6.23 shows, $c_{0_v t}$, the cost of support capital budgeting project, e_{0_v}, in each period $t = 0,1,\ldots,T$. This cost is uniform in each period.

6.1.28. Maximum Allowable Production Project Expense in Each Period

Table 6.24 shows, E_{t_l}, the maximum amount that may be spent on all production capital budgeting projects, e_{m_l}, in each period $t = 0,1,\ldots,T$. This maximum allowable expense is uniform in each period.

Table 6.21. Cost of Production Capital Budgeting Projects in Each Period.

Period	0	1	2	3	4	5	6	7	8	9	10	11
Project	Cost of Production Capital Budgeting Projects in Each Period											
0	180,289	180,289	180,289	180,289	180,289	180,289	180,289	180,289	180,289	180,289	180,289	180,289
1	187,500	187,500	187,500	187,500	187,500	187,500	187,500	187,500	187,500	187,500	187,500	187,500
2	250,000	250,000	250,000	250,000	250,000	250,000	250,000	250,000	250,000	250,000	250,000	250,000

Table 6.22. Cost of Pre-Production/Post-Support Capital Budgeting Project in Each Period.

Period	0	1	2	3	4	5	6	7	8	9	10	11
Project Cost	150,000	150,000	150,000	150,000	150,000	150,000	150,000	150,000	150,000	150,000	150,000	150,000

Table 6.23. Cost of Support Capital Budgeting Project in Each Period.

Period	0	1	2	3	4	5	6	7	8	9	10	11
Project Cost	80,000	80,000	80,000	80,000	80,000	80,000	80,000	80,000	80,000	80,000	80,000	80,000

Table 6.24. Maximum Allowable Production Capital Budgeting Expense in Each Period.

Period	0	1	2	3	4	5	6	7	8	9	10	11
Maximum Expense	750,000	750,000	750,000	750,000	750,000	750,000	750,000	750,000	750,000	750,000	750,000	750,000

6.1.29. Maximum Allowable Pre-Production/Post-Support Project Expense in Each Period

Table 6.25 shows, E_{t_n}, the maximum amount that may be spent on the pre-production/post-support capital budgeting project, $e0_n$, in each period $t = 0,1,\ldots,T$.

6.1.30. Maximum Allowable Support Project Expense in Each Period

Table 6.26 shows, E_{t_v}, the maximum amount that may be spent on the support capital budgeting project, $e0_v$, in period $t = 0,1,\ldots,T$.

6.1.31. Maximum Repair Parts Cost in Each Period

Table 6.27 shows, P_t, the maximum total cost of parts consumed due to repair in each period $t = 0,1,\ldots,T_s$.

6.1.32. Minimum Acceptable Revenue

Table 6.28 shows, R_t, the minimum acceptable total revenue in each period $t = 0,1,\ldots,T_s$.

6.1.33. Material Cost for the Production of Each Product

Table 6.29 shows, MA_{jkt}, the total cost of materials in actual dollars required to produce one unit of product j in substitute product group k in each period $t = 0,1,\ldots,T_e$. A product can only have material costs associated with it in periods that it can be produced. In periods when a product cannot be produced, material costs are not applicable (i.e., N/A) as noted in Table 6.29[6.5].

6.1.34. Units Required at Pre-Production/Post-Support Work Center Zero

Table 6.30 shows, g_{jk0t}, the units of resources consumed at pre-production and post-support work center $n = 0$ in each period $t = 0,1,\ldots,T$ by each product j in substitute product group k.

6.1.36. Units Required at Pre-Production/Post-Support Work Center One

Table 6.31 shows, g_{jk1t}, the units of resources consumed at pre-production and post-support work center $n = 1$ in each period $t = 0,1,\ldots,T$ by each product j in substitute product group k.

[6.5] See footnote 6.4.

Table 6.25. Maximum Allowable Pre-Production/Post-Support Capital Budgeting Expense in Each Period.

Period	0	1	2	3	4	5	6	7	8	9	10	11
Maximum Expense	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000

Table 6.26. Maximum Allowable Support Capital Budgeting Expense in Each Period.

Period	0	1	2	3	4	5	6	7	8	9	10	11
Maximum Expense	100,000	100,000	100,000	100,000	100,000	100,000	100,000	100,000	100,000	100,000	100,000	100,000

Table 6.27. Maximum Allowable Repair Parts Cost in Each Support Period.

Period	0	1	2	3	4	5	6	7	8	9	10
Maximum Cost	225,000	225,000	225,000	225,000	225,000	225,000	225,000	225,000	225,000	225,000	225,000

Table 6.28. Minimum Acceptable Revenue in Each Support Period.

Period	0	1	2	3	4	5	6	7	8	9	10
Minimum Acceptable Revenue	100	100	100	100	100	100	100	100	100	100	100

Table 6.29. Material Cost to Produce Each Product.

	Period	0	1	2	3	4	5	6	7
Product Sub-Group	**Product**	**Material Cost to Produce Product**							
0	0	8500.00	8500.00	8500.00	8500.00	N/A	N/A	N/A	N/A
0	1	N/A	N/A	N/A	7500.00	7600.00	7700.00	7700.00	7800.00
0	2	N/A	N/A	N/A	N/A	7100.00	7100.00	7100.00	7100.00
1	0	8500.00	8500.00	8500.00	8500.00	8500.00	8600.00	9000.00	9000.00
1	1	N/A	N/A	N/A	7750.00	7750.00	7750.00	7800.00	7800.00

Table 6.30. Units Required by Each Product at Pre-Production/Post-Support Work Center Zero.

	Period	0	1	2	3	4	5	6	7	8	9	10	11
Product Sub-Group	**Product**	**Units Required by Each Product at Pre-Production/Post-Support Work Center**											
0	0	2.00	2.00	2.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	1	12.25	6.15	3.00	3.00	3.00	3.00	3.00	3.00	0.00	0.00	0.00	0.00
0	2	0.00	20.50	10.25	5.00	3.00	2.00	2.00	2.00	0.00	0.00	0.00	0.00
1	0	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	0.00	0.00	0.00	0.00
1	1	12.20	6.15	3.00	2.00	2.00	2.00	2.00	2.00	0.00	0.00	0.00	0.00

Table 6.31. Units Required by Each Product at Pre-Production/Post-Support Work Center One.

	Period	0	1	2	3	4	5	6	7	8	9	10	11
Product Sub-Group	**Product**	**Units Required by Each Product at Pre-Production/Post-Support Work Center**											
0	0	3.00	3.00	3.00	3.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	1	0.00	12.00	6.00	3.00	3.00	3.00	3.00	3.00	0.00	0.00	0.00	0.00
0	2	0.00	0.00	12.00	6.00	3.00	3.00	3.00	3.00	0.00	0.00	0.00	0.00
1	0	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	0.00	0.00	0.00	0.00
1	1	0.00	20.00	10.00	5.00	3.00	3.00	3.00	3.00	0.00	0.00	0.00	0.00

6.1.35. Maximum Fraction of Regular Time Labor at Each Production Work Center

The maximum fraction of regular time labor, λ_{lt}, at each production work center, l, in each production period $t = 0,1,\ldots,T_e$ is shown in Table 6.32.

Table 6.32. Maximum Fraction of Regular Time Labor.

Period	0	1	2	3	4	5	6	7
Work Center	Maximum Fraction of Regular Time Labor in Each Production Period							
0	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
1	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
2	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

6.1.41. Maximum Allowable Production Cost in Each Period

Table 6.33 shows the maximum allowable cost of production, PC_t, in each period $t = 0,1,\ldots,T_e$. This constraint includes all materials and labor costs.

Table 6.33. Maximum Allowable Production Cost in Each Period.

Period	0	1	2	3	4	5	6	7
Maximum Cost	300,000	300,000	300,000	300,000	300,000	300,000	300,000	300,000

6.1.37. Units Required at Pre-Production/Post-Support Work Center Two

Table 6.34 shows, g_{jk2t}, the units of resources consumed at pre-production and post-support work center $n = 2$ in each period $t = 0,1,\ldots,T$ by each product j in substitute product group k.

6.1.38. Units Required at Pre-Production/Post-Support Work Center Three

Table 6.35 shows, g_{jk3t}, the units of resources consumed at pre-production and post-support work center $n = 3$ in each period $t = 0,1,\ldots,T$ by each product j in substitute product group k.

Table 6.34. Units Required by Each Product at Pre-Production/Post-Support Work Center Two.

	Period	0	1	2	3	4	5	6	7	8	9	10	11
Product Sub-Group	**Product**	**Units Required by Each Product at Pre-Production/Post-Support Work Center**											
0	0	5.00	5.00	5.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	1	0.00	0.00	10.00	8.00	6.00	5.00	5.00	5.00	0.00	0.00	0.00	0.00
0	2	0.00	0.00	0.00	10.00	8.00	6.00	5.00	5.00	0.00	0.00	0.00	0.00
1	0	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	0.00	0.00	0.00	0.00
1	1	0.00	0.00	9.00	7.00	6.00	5.00	5.00	5.00	0.00	0.00	0.00	0.00

Table 6.35. Units Required by Each Product at Pre-Production/Post-Support Work Center Three.

	Period	0	1	2	3	4	5	6	7	8	9	10	11
Product Sub-Group	**Product**	**Units Required by Each Product at Pre-Production/Post-Support Work Center**											
0	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	20.00	0.00	0.00	0.00
0	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	20.00
0	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	20.00
1	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	20.00
1	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	20.00

6.1.39. Hours of Support Required by Products of Age τ

Table 6.36 shows the support hours, $S_{jk\tau_{jk}}$, required for each unit of product j of age $\tau = 1,2,\ldots,PL_{jk}$ in substitute product group k. In this hypothetical example, $PL_{jk} = 3 \quad \forall\ j \in J_k,\ k \in K$.

Table 6.36. Hours of Support Required for Products of Age τ.

	Age	0	1	2	3
Product Sub-Group	**Product**	**Support Hours Required by Each Product**			
0	0	25.30	26.10	27.25	28.40
0	1	24.00	24.50	25.10	26.20
0	2	18.30	20.10	21.25	22.50
1	0	27.20	29.30	29.50	30.00
1	1	20.10	21.30	22.50	23.00

6.1.40. Base Support Hours

The base quantity of support hours available in each period, S_0, is equal to 12,000.0 hours.

6.1.42. Average Parts Cost to Repair Each Product of Age τ

Table 6.37 shows, $p_{jk\tau_{jk}}$, the average cost in constant dollars of parts to repair one unit of product j of age τ_{jk} in substitute product group k where $\tau_{jk} = 0,1,\ldots,PL_{jk}$.

Table 6.37. Average Parts Cost to Repair Product of Age τ.

	Age	0	1	2	3
Product Sub-Group	**Product**	**Average Parts Cost to Repair Product**			
0	0	200.00	250.00	300.00	350.00
0	1	150.00	200.00	250.00	300.00
0	2	75.00	100.00	200.00	250.00
1	0	225.00	275.00	300.00	350.00
1	1	125.00	175.00	250.00	290.00

6.1.43. Repair Labor Revenue Per Hour in Each Period

Table 6.38 shows, f_t, the average revenue in actual dollars per hour of repair labor in each support period $t = 0,1,\ldots,T_s$.

Table 6.38. Hourly Labor Revenue from the Repair of Products in Each Period.

Period	0	1	2	3	4	5	6	7	8	9	10
Labor Revenue	30.00	31.00	32.00	33.00	34.00	35.00	36.00	37.00	38.00	39.00	40.00

6.1.44. Repair Labor Cost

Table 6.39 shows, RC_t, the repair labor cost in actual dollars per hour in each support period $t = 0,1,\ldots,T_s$.

Table 6.39. Hourly Labor Costs to Repair Products in Each Period.

Period	0	1	2	3	4	5	6	7	8	9	10
Labor Cost	25.00	25.50	25.75	25.90	26.00	26.25	26.50	26.75	26.95	27.00	27.30

6.1.45. Base cost to Operate Each Pre-Production/Post-Support Work Center

Table 6.40 shows, b_n, the base cost to operate pre-production and post-support work centers $n = 0,1,..,N$ in period $t = 0$ at the beginning capacity, B_n.

Table 6.40. Base Cost to Operate Each Pre-Production/Post-Support Work Center.

Work Center	0	1	2	3
Base Cost	200,000	300,000	125,000	50,000

6.1.46. Revenues from the Sale of Parts to Repair a Product of Age τ

Table 6.41 shows, $PR_{jk\tau_{jk}}$, the average revenue in constant dollars from parts to repair one unit of product j in substitute product group k of age τ_{jk} where $\tau_{jk} = 0,1,\ldots,PL_{jk}$. A product can only generate parts revenues after its warranty has

expired. In periods when a product repair costs are covered by the warranty, revenues from the sale of parts are not applicable (i.e., N/A) as noted in Table 6.41[6.6].

Table 6.41. Revenues from the Sale of Parts to Repair Product of Age τ.

	Age	0	1	2	3
Product Sub-Group	**Product**	**Parts Revenues to Repair Product**			
0	0	N/A	N/A	500.00	650.00
0	1	N/A	N/A	450.00	500.00
0	2	N/A	N/A	400.00	450.00
1	0	N/A	N/A	525.00	670.00
1	1	N/A	N/A	500.00	575.00

6.1.47. Base Cost to Operate Each Production Work Center

Table 6.42 shows, b_l, the base cost to operate each production work center, l, at the beginning capacity, B_l. This cost is incurred in each period $t = 0,1,\ldots,T$, and does not include production labor costs.

Table 6.42. Base Cost to Operate Each Production Work Center.

Work Center	0	1	2
Base Cost	300,000	250,000	350,000

6.1.48. Base Cost to Operate the Support Work Center

The base cost to operate support facilities, s_0, at the base capacity level in hours, S_0, is \$125,000 in period $t = 0$ dollars.

6.1.49. Inventory Holding Costs for Each Product in Each Period

Table 6.43 shows the inventory holding cost, I_{jkt}, in actual dollars for each product, j, in substitute product group k for each period $t = t_{b_{jk}},\ldots,t_{e_{jk}} - 1$. A product can only have inventory holding costs associated with it in periods that it can be produced.

[6.6] See footnote 6.4

In periods when a product cannot be produced, inventory holding costs are not applicable (i.e., N/A) as noted in Table 6.43[6.7].

Table 6.43. Inventory Holding Cost for Each Product in Each Period.

	Period	0	1	2	3	4	5	6
Product Sub-Group	**Product**	**Units Required by Each Product**						
0	0	100.00	100.00	100.00	N/A	N/A	N/A	N/A
0	1	N/A	N/A	N/A	100.00	100.00	100.00	100.00
0	2	N/A	N/A	N/A	N/A	100.00	100.00	100.00
1	0	100.00	100.00	100.00	100.00	100.00	100.00	100.00
1	1	N/A	N/A	N/A	100.00	100.00	100.00	100.00

6.1.50. Beginning Inventory Level for Each Product

Table 6.44 shows the beginning inventory levels, bin_{jk0}, of each product, j, in substitute product group k in period 0. Only products that were produced in the period prior to the current beginning period $t = 0$, may have inventories that are greater than zero. Thus, products that have not been produced yet cannot have any beginning inventory levels associated with them. In these cases, beginning inventory levels are not applicable (i.e., N/A) as noted in Table 6.44[6.8].

Table 6.44. Beginning Inventory Levels for Each Product.

Product Sub-Group	Product	Inventory Level
0	0	1
0	1	N/A
0	2	N/A
1	0	1
1	1	N/A

[6.7] See footnote 6.4.

[6.8] See footnote 6.4.

6.2. Software Modules

This section contains a discussion of the software modules developed to enable comparison of the results obtained from all three versions of the concurrent and sequential models. Software modules comprise the linkages between LINDO/386 and five computer programs developed using Borland's C/C++.

6.2.1. LINDO/386

LINDO/386 is a commercially available DOS (disk operating system) based IBM/IBM compatible linear programming (LP) software package[6.9]. The LINDO/386 version used in this research runs only on 80386 and 80486 processors with a floating point coprocessor and requires DOS 3.0 or higher.

The LINDO/386 software package will handle real as well as zero-one decision variables. LINDO/386 can read data from MPS formatted files and also has an internal editor to enable MIP creation and modification. Thus, MIP's can either be read from files, created on-line using the internal editor, or read from an MPS file and then edited using the internal editor. During tests of both models, MPS files were read into LINDO/386, and then modified several times using LINDO/386's internal editor.

In addition to reading files in MPS format, LINDO/386 can also save the solutions it generates after an MIP has been solved. Solutions are stored in a standard DOS formatted file. Because of this, it was possible to use Microsoft Corporation's DOS based editor, Edit, to add a file name and quantities of each product sub-group demanded, and remove constraint slack and right hand side information in each file created by LINDO/386. Since constraint slack and right hand side information were not required during the comparison of the concurrent and sequential models, this data was removed to decrease the size of the solution files generated by LINDO/386. These files were then saved using a standardized naming convention.

6.2.2. Borland C/C++

When solving the hypothetical example, the concurrent model simultaneously finds the values of 132 decision variables and considers 300 constraints. While the concurrent model considers all decision variables and constraints at once, the sequential model considers a sub-set of the decision variables and constraints in each stage. However, some

---

[6.9] LINDO/386 can also perform 0,1 linear programming.

constraints are considered in more than one stage. Further, stage one has a set of decision variables to account for capacity utilization that are not required in the concurrent model.

In most cases, the coefficients contained in the objective function and constraints of each linear program were values that had to be determined using more than one input. There were nine comparisons of tests made of each of the three versions of the concurrent and sequential models (a total of 27 comparisons). This means that a total of approximately 3,564 objective function coefficients and 8,100 constraint equations[6.10] were developed while testing the three versions of the concurrent model. Further, a total of approximately 10,692 objective function coefficients and 24,300 constraint equations[6.11] were developed while testing the three versions of the sequential model.

Because of the large size of each MIP, four computer programs that provide MIP's in MPS format were developed using Borland's C/C++. These programs create MPS formatted files that are input to LINDO/386.

6.2.3. Conversion to Consecutive Notation

To ease programming difficulty, the product sub-group notation k was not included in the Borland C/C++ programs. Product sub-groups do not have to be explicitly considered because the Borland C/C++ programs were developed to solve a specific problem. Hence, the Borland C/C++ programs only consider the hypothetical problem where three products reside in product sub-group zero, and the remaining two products reside in product sub-group one.

Since product sub-groups are not considered, it is possible to convert product notation to a single subscript, consecutive format. This conversion to consecutive notation is shown in Table 6.45. Note that the comparison of results obtained from tests of the concurrent and sequential models (shown in Chapter 7) refers to products by the consecutive notation.

6.10 The total number of coefficients in the concurrent model constraint set has not been determined. However, if each constraint averaged five decision variables, then a total of 300 x 5 = 1,500 coefficients would exist in the constraint set. Thus, since a total of 27 tests were made using the three versions of the concurrent model, a total of 27 x 1,500 = 40,500 coefficients would be determined by the Borland C/C++ program.

6.11 The total number of coefficients in the three stages of the sequential constraint set has not been determined. However, if each constraint averaged five decision variables, then a total of 300 x 3 x 5 = 4,500 coefficients would exist in the constraint set. Thus, since a total of 27 tests were made using the three versions of the sequential models, a total of 27 x 4,500 = 121,500 coefficients would be determined by the Borland C/C++ programs..

Table 6.45. Conversion from Product Sub-Group Notation to Consecutive Notation.

Old Notation		Equivalent New Notation
Product Sub-Group	Product	Product Number
0	0	0
0	1	1
0	2	2
1	0	3
1	1	4

As an example of the conversion to the new notation, the production variable x_{113} (i.e., product $j = 1$, product sub-group $k = 1$, and fiscal period $t = 3$) is equivalently known as x_{43} (i.e., product number $j = 1$, and fiscal period $t = 3$). As can be seen, this change in notation significantly reduces the number of subscripts involved with each variable. This reduction in the number of subscripts involved with each variable enabled the consideration of each set of product specific return and resource consumption in the Borland C/C++ programs as a two-dimensional array rather than as a single three dimensional or five two-dimensional arrays.

6.2.4. File Naming Convention

A file naming convention was developed for this research. All files created by Borland C/C++ programs to become input to LINDO/386 carry a DOS .dat extension. For example, the file mps1.dat is a MPS formatted MIP file developed for input into LINDO/386. All solution files emanating from LINDO/386 carry a DOS .rep extension. For example, the file mps1.rep is a solution file saved after LINDO/386 has found the optimal solution.

6.2.5. Additional Constraints Required by the Concurrent and Sequential Models

Constraint types (i.e., unconstrained and partial project selection allowed, less than or equal to one and partial project selection allowed, and restricted to values of zero and one) and the number of 0,1 integer variables were adjusted using LINDO/386's on-line editing function.

An additional 0,1 decision variable, g, was required during optimization runs when partial capital budgeting project selection was allowed. The 0,1 integer variable enabled retention of the project mutually exclusive constraint that existed between production capital budgeting projects e_{0_l} and e_{2_l}.

The MIP formulation (i.e., the formulation given in Chapter 4) of the mutually exclusive constraint is given in equation (6.2).

$$e_{1_l} + e_{2_l} \leq 1 \tag{6.2}$$

However, this constraint requires that both e_{0_l} and e_{2_l} be constrained to 0,1 integer values.

During tests when no upper limits were placed on capital budgeting project values and partial project selection was allowed, equation (6.1) was deleted and a set of substitute constraints was added to account for the increased range and partial project selection. In these cases, equations (6.3), (6.4), and (6.5) were added to the MIP and equation (6.2) was deleted.

$$-u_1 g + e_{2_l} \leq 0 \tag{6.3}$$

$$e_{1_l} + u_1 g \leq u_1 \tag{6.4}$$

$$g \in 0,1 \tag{6.5}$$

When production capital budgeting projects e_{0_l} and e_{2_l} were allowed to take values that were less than or equal to one and partial project selection was allowed, equation (6.3) and (6.4) were deleted, and equations (6.6) and (6.7) were added.

$$-g + e_{2_l} \leq 0 \tag{6.6}$$

$$e_{1_l} + g \leq 1 \tag{6.7}$$

During tests when capital budgeting projects were less than or equal to zero, and partial project selection was allowed, constraints had to be added to bound projects. During these tests, three constraints to account for upper bounds of one on the pre-production/post-support project e_{0_n}, the support project e_{0_v}, and the production project e_{1_l} were added. These constraints are shown in equations (6.8), (6.9), and (6.10) respectively.

$$e_{0_n} \leq 1 \tag{6.8}$$

$$e_{0_v} \leq 1 \qquad (6.9)$$

$$e_{1_t} \leq 1 \qquad (6.10)$$

6.2.6. Software Modules Required by the Concurrent Model

Each test of the concurrent model was initiated by running the input set contained in the DOS file input.dat (listed in Appendix A) through sin.exe, a computer program developed in Borland's C/C++. Sin.exe is comprised of four Borland C/C++ files that have been compiled and linked. These four files, sinmain.c, sinmain.h, constrnt.c and constrnt.h, are listed in Appendix B.

Sin.exe creates an MPS formatted file named mps.dat. LINDO/386 (lindo.exe) reads as input mps.dat and finds the MIP optimal solution. This solution is then stored in a file called mps.rep.

The linkages between sin.exe and lindo.exe are shown in Figure 6.2. The software modules representing the concurrent model were run a total of 27 times. Each run was called a test. Tests included all combinations of nine different product sub-group demand levels and three capital budgeting project constraint types; i.e., $(9)(3) = 27$. Constraint types considered were: (1) unconstrained and partial project selection allowed, (2) less than or equal to one and partial project selection allowed, and (3) restricted to values of zero and one.

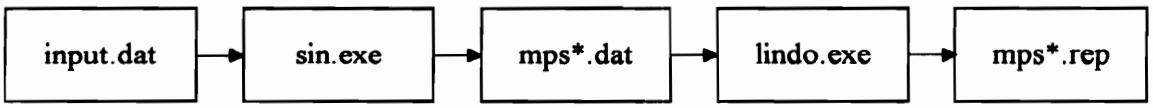

Figure 6.2. Integration of Computer Modules Required by the Concurrent Model.

The demand for product sub-groups was varied by manually editing the input file (input.dat), and setting the demand, d_{kt}, for each product sub-group, k, in each period $t = 0,1,\ldots,7$ to either 50, 100, or 150[6.12]. The concurrent model was run using all possible combinations of these three demand levels for each of the two product sub-

---

6.12 To preserve generality of the concurrent model, the Borland C/C++ program sin.exe requires that the product sub-group demand d_{kt} be known for each fiscal period $t = 0,1,\ldots,7$. However, during all tests of the concurrent model, product sub-group demand was considered to be uniform (i.e., $d_{k0}=d_{k1}=\ldots=d_{k7}$).

groups. Thus, concurrent model MPS input files are named mps1.dat through mps9.dat (Table 6.46).

After editing the input file, input.dat, to set the correct product sub-group demand levels, the Borland C/C++ file sin.exe was run to create the DOS file mps.dat. The DOS function "rename oldfile_name newfile_name" was then used to assign the correct name to the MPS file.

Table 6.46. Combination of Demand in each mps.dat File.

MPS File Name: mps*.dat	Product Sub-Group One Demand	Product Sub-Group Two Demand
mps1.dat	50	50
mps2.dat	50	100
mps3.dat	50	150
mps4.dat	100	50
mps5.dat	100	100
mps6.dat	100	150
mps7.dat	150	50
mps8.dat	150	100
mps9.dat	150	150

6.2.7. Process to Test the Concurrent Model

Figure 6.3 shows the process required to run the concurrent model. The first step in the process was to run the Borland C/C++ program sin.exe. This program reads the input file input.dat. An MPS formatted file named mps.dat was created each time sin.exe was run. The MPS file was saved as mps*.dat where $* = 1,2,\ldots,9$ depending upon the value of D_0 and D_1 in the input file.

LINDO/386 was executed and the MPS formatted file mps*.dat was read as input. LINDO/386's edit function was initiated, and a 0,1 integer variable g with a coefficient of negative one was added to the objective function. Next, the production capital budgeting mutually exclusive constraint shown in equation (6.2) was replaced by the constraints shown in equations (6.3) and (6.4). The number of 0,1 integer variables was then set to 32. Zero-one integer variables included five product funding variables (z_{jk}), 26 product setup variables ($\delta(x_{jkt})$), and the project mutually exclusive variable, g. The edit function was then terminated and optimization was initiated. When an optimal solution was found, the solution was saved as mps*.rep where $* = 1,2,\ldots,9$ and corresponded to the nine

combinations of demand for the model that allowed partial project selection with no upper bound.

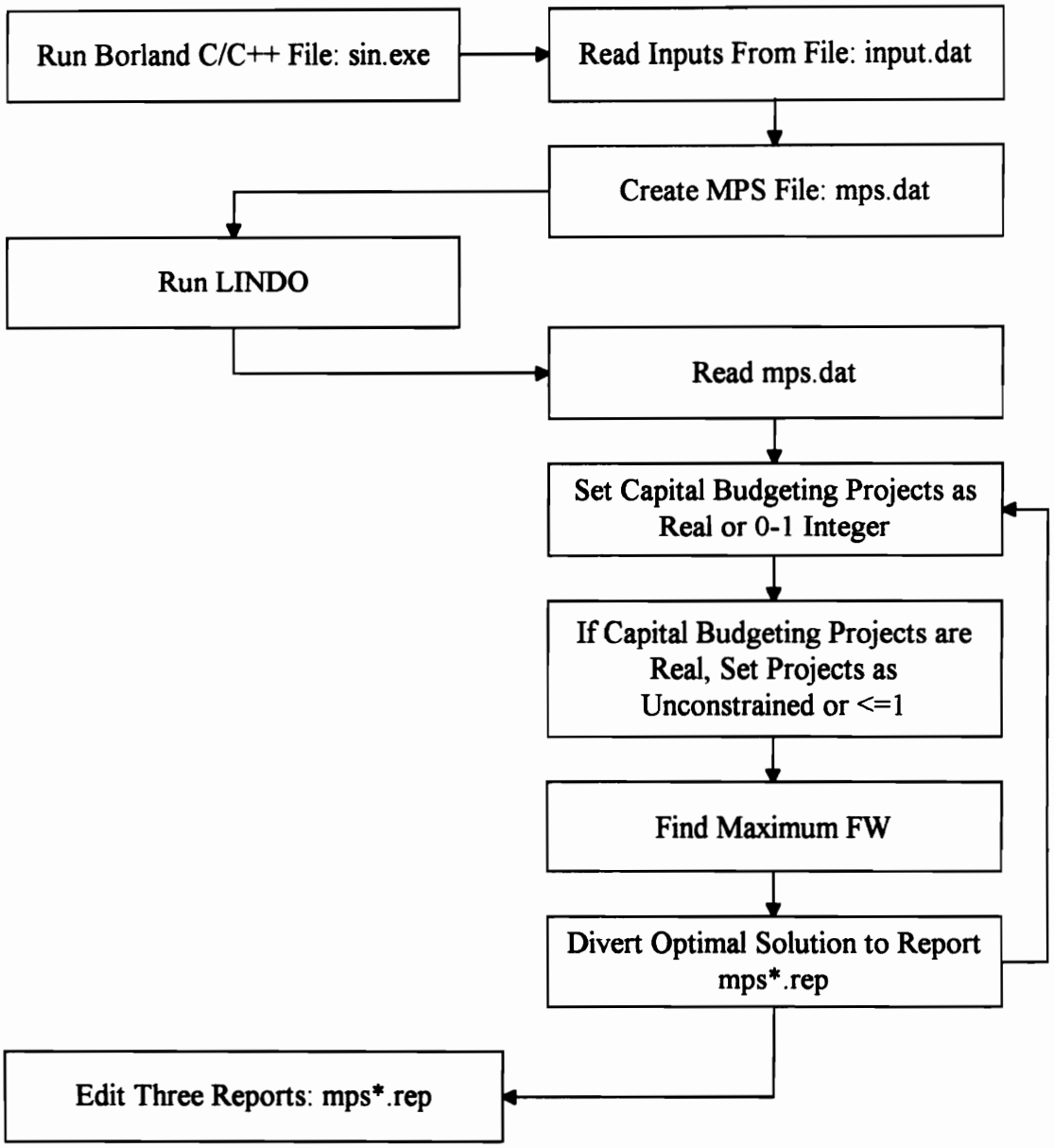

Figure 6.3. Process Chart for Concurrent Model Execution.

At this point, the LINDO/386 edit function was once again executed. The constraints shown in equations (6.3) and (6.4) were replaced with the constraints shown in

equations (6.6), (6.7), (6.8). (6.9), and (6.10). The edit function was then terminated, and optimization was initiated. When an optimal solution was found, the solution was saved to a file named mps*.rep where $* = 10,11,\ldots,18$ and corresponded to the nine combinations of demand for the model that allowed partial project selection with an upper bound of one on all projects.

The LINDO/386 edit function was re-called once again, and the variable g was removed from the objective function. The constraints shown in equations (6.5), (6.6), (6.7), (6.8), (6.9), and (6.10) were replaced with the constraint shown in equation (6.2). The number of 0,1 integer variables was then set to 36. Zero-one integer variables included five product funding variables (z_{jk}), 26 product setup variables ($\delta(x_{jkt})$), one pre-production/post-support capital budgeting variable (e_{m_n}), three production capital budgeting variables (e_{m_l}), and one support capital budgeting variable (e_{m_v}). At this point, the edit function was terminated, and optimization was initiated. When an optimal solution was found, the solution was saved as mps*.rep where $* = 19,20,\ldots,27$ and corresponded to the nine combinations of demand for the model that restricted projects to values of zero and one.

As a final step, each mps*.rep file created during the optimization process was edited to remove slack and right-hand side information. In addition, the file name, constraint type, and values of D_0 and D_1 were added. After the completion of all tests, there existed 27 MPS formatted files named mps1.rep, mps2.rep,...,mps27.rep.

6.2.8. Software Modules Required by the Sequential Model

The three versions of the sequential model were tested by running the input set contained in the DOS file input.dat (listed in Appendix A) through four Borland C/C++ files. The four Borland files are: first.exe, middle.exe, last.exe, and fwss.exe. Each Borland C/C++ file is comprised of four Borland C/C++ files that have been compiled and linked. These relationships are shown in Table 6.47.

Table 6.47. Sequential Model C/C++ Executable Files.

EXE File	Borland C/C++ Files				Listed in
first.exe	frstmain.c	frstmain.h	first.c	first.h	Appendix C
middle.exe	midlmain.c	midlmain.h	middle.c	middle.h	Appendix D
last.exe	lastmain.c	lastmain.h	last.c	last.h	Appendix E
fwss.exe	fwssmain.c	fwssmain.h	fwss.c	fwss.h	Appendix E

Figure 6.4 shows the linkages between the four Borland C/C++ programs, and LINDO/386. The Borland C/C++ program first.exe creates an MPS formatted file named mpfs.dat. LINDO/386 (lindo.exe) reads as input mpfs*.dat and finds the MIP solution for stage one of the sequential model. This solution is then stored in a file called mpfs*.rep. Middle.exe creates an MPS formatted file named mpms.dat.

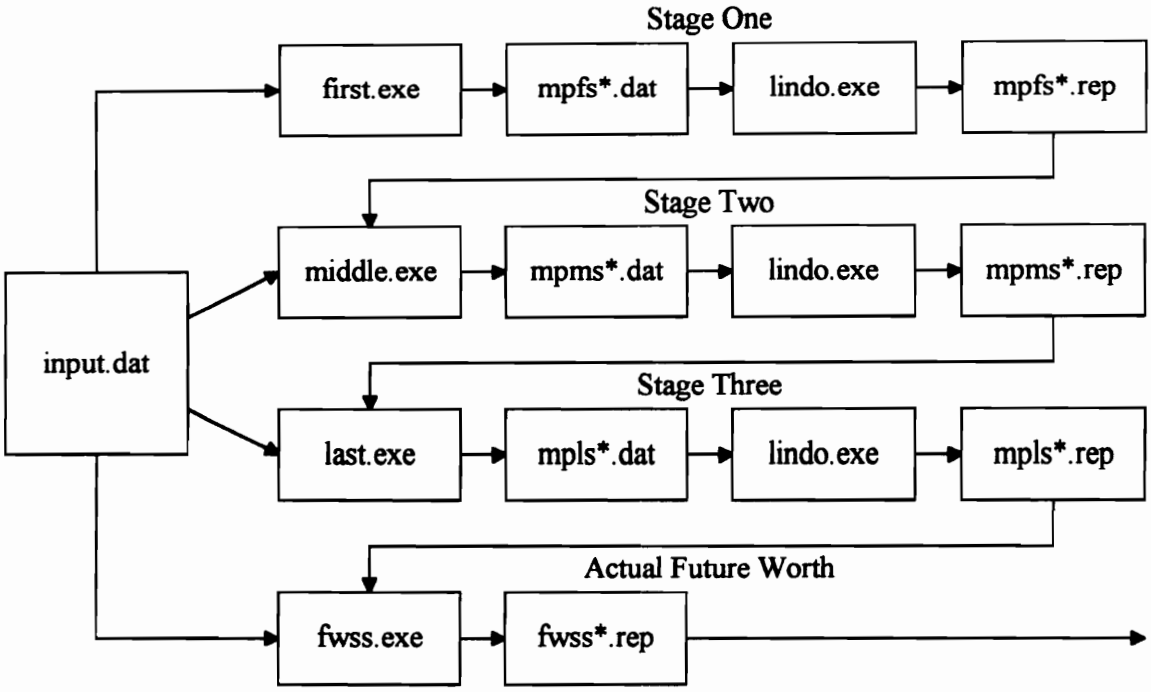

Figure 6.4. Integration of Computer Modules Required by the Sequential Model.

The Borland C/C++ program middle.exe reads inputs from the input file, input.dat, and also from the optimal first stage solution stored in mpfs*.rep. LINDO/386 reads as input mpms.dat and finds the MIP solution for stage two of the sequential model. This solution is then saved in a file named mpms*.rep.

The Borland C/C++ program last.exe creates and MPS formatted file named mpls.dat. Last.exe reads inputs from the input file, input.dat, and also from the optimal second stage solution stored in mpms*.rep. LINDO/386 reads as input mpls.dat and find the MIP solution for stage three of the sequential model. The optimal stage three solution is stored in a file named mpls*.rep. The Borland C/C++ program fwss.exe reads as input the input file, input.dat, and the optimal stage three solution stored in mpls*.rep. Fwss.exe

finds the actual future worth of the sequential model given the values of all decision variables located in mpls*.rep.

Software modules representing the sequential model were run a total of 27 times. Runs included all combinations of nine different product sub-group demand levels and three versions of the sequential model (i.e., $9x3 = 27$).

The demand for product sub-groups was varied by manually editing the input file (input.dat), and setting the uniform demand D_k for each product sub-group k in each period to either: $D_k = 50$, $D_k = 100$, or $D_k = 150$. The sequential model was run using all possible combinations of these three demand levels for each of the two product sub-groups. Thus, stage one sequential model MPS input files are named mpfs1.dat through mpfs9.dat (Table 6.48).

Table 6.48. Values of Demand for each mpfs.dat File.

MPS File Name: mpfs*.dat	Product Sub-Group Zero Demand: D_0	Product Sub-Group One Demand: D_1
mpfs1.dat	50	50
mpfs2.dat	50	100
mpfs3.dat	50	150
mpfs4.dat	100	50
mpfs5.dat	100	100
mpfs6.dat	100	150
mpfs7.dat	150	50
mpfs8.dat	150	100
mpfs9.dat	150	150

6.2.9. Process to Test the Sequential Model

Four Borland C/C++ programs and three optimization runs using LINDO/386 must be completed for each test of the sequential model. This section presents the solution process required in each of the three stages of the sequential model, and also discusses the process to determine future worth.

6.2.9.1. Process to Test Stage One of the Sequential Model

Figure 6.5 shows the process required to run stage one of the sequential model. The first step was to run the Borland C/C++ program first.exe. This program reads the input file input.dat. An MPS formatted file named mpfs.dat was created each time

first.exe was run. The MPS file was saved as mpfs*.dat where * = 1,2,...,9 depending upon the value of D_0 and D_1 in the input file.

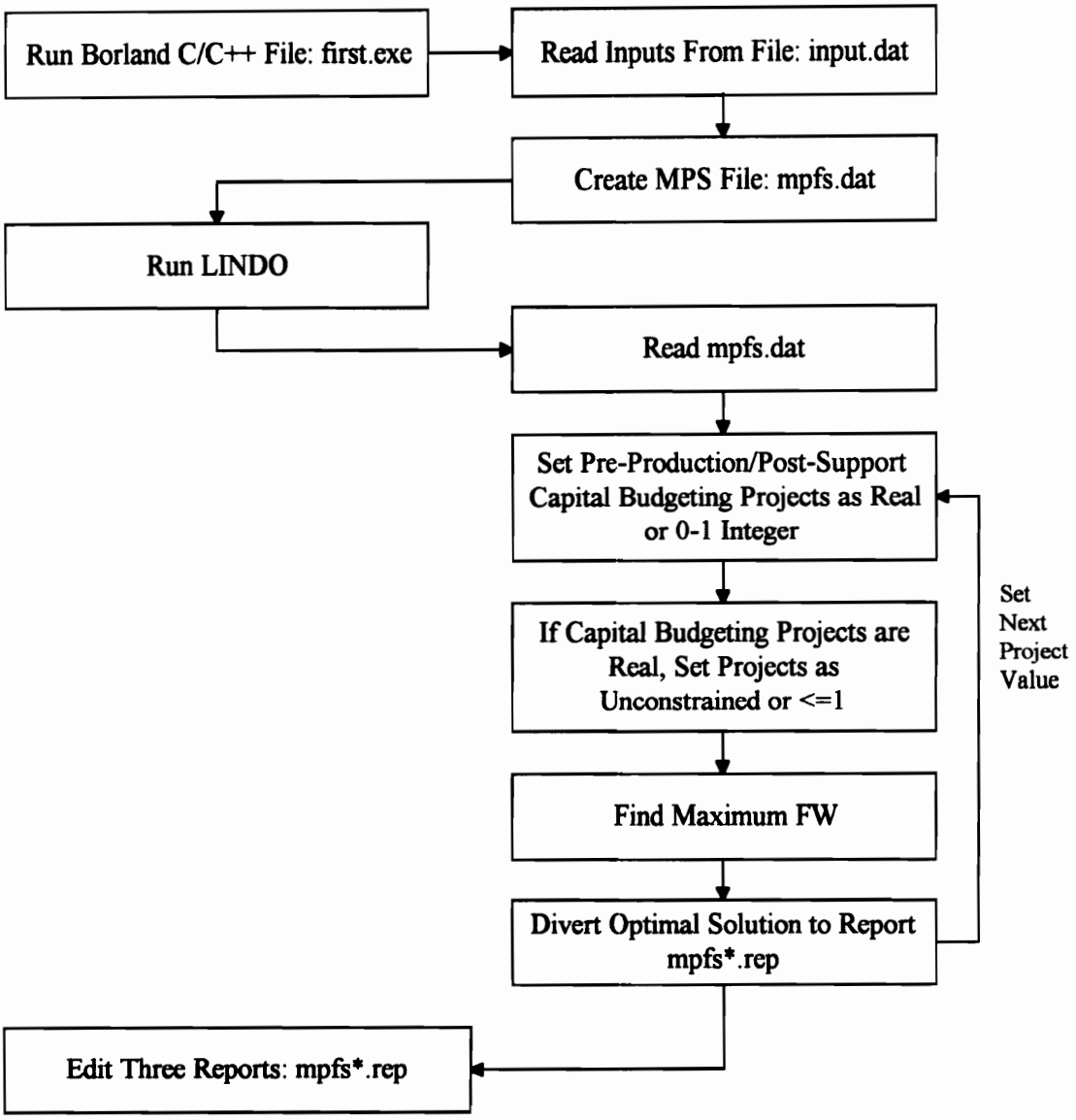

Figure 6.5. Process Chart for Stage One of the Sequential Model.

LINDO/386 was executed and the MPS formatted file mpfs*.dat was read as input. LINDO/386's edit function was initiated, and the number of 0,1 integer variables was then set to 31. Zero-one integer variables included five product funding variables

(z_{jk}), and 26 product setup variables ($\delta(x_{jkt})$). The edit function was then terminated and optimization initiated. When an optimal solution was found, the solution was saved as mpfs*.rep where * = 1,2,...,9 and corresponded to the nine combinations of demand for the model that allowed partial project selection with no upper bound.

At this point, the LINDO/386 edit function was once again executed. An additional constraint shown in equation (6.8) was added. The edit function was then terminated, and optimization was initiated. When an optimal solution was found, the solution was saved to a file named mpfs*.rep where * = 10,11,...,18 and corresponded to the nine combinations of demand for the model that allowed partial project selection and included an upper bound of one on all projects.

The LINDO/386 edit function was re-called once again and the constraint shown in equation (6.8) was removed. The number of 0,1 integer variables was increased to 32. Zero-one integer variables included five product funding variables (z_{jk}), 26 product setup variables ($\delta(x_{jkt})$), and one pre-production/post-support capital budgeting variable (e_{m_n}). At this point, the edit function was terminated, and optimization was initiated. When an optimal solution was found, the solution was saved as mpfs*.rep where * = 19,20,...,27 and corresponded to the nine combinations of demand for the model that restricted projects to values of zero and one.

As a final step, each mpfs*.rep file created during the optimization process was edited to remove slack and right-hand side information. In addition, the file name, constraint type, and values of D_0 and D_1 were added. After the completion of all tests, there existed 27 MPS formatted files named mpfs1.rep, mpfs2.rep,...,mpfs27.rep.

6.2.9.2. Process to Test Stage Two of the Sequential Model

Figure 6.6 shows the process required to run stage two of the sequential model. The first step in the process was to run the Borland C/C++ program middle.exe. This program reads the input file input.dat and the stage one solution file mpfs*.rep. An MPS formatted file named mpms.dat was created each time middle.exe was run. The MPS file was saved as mpms*.dat where * = 1,2,...,9 when projects were unconstrained and partial project selection was allowed, * = 10,11,...,18 when projects had an upper bound of one and partial project selection was allowed, and * = 19,20,...,27 when projects were restricted to values of zero and one.

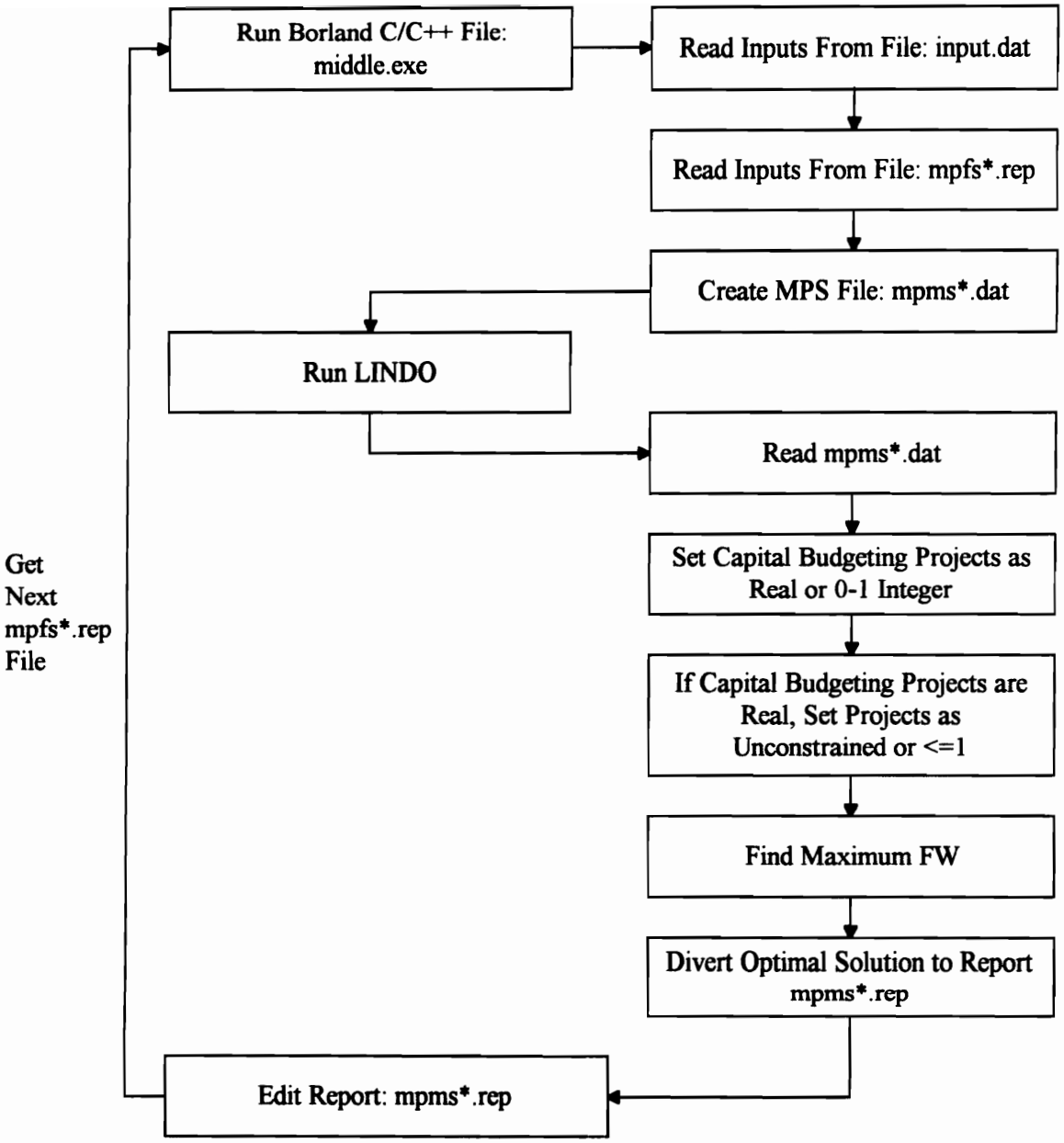

Figure 6.6. Process Chart for Stage Two of the Sequential Model.

There were three specific processes that occurred depending upon the type of capital budgeting constraint considered. When projects were unconstrained and partial project selection was allowed, LINDO/386 was executed and the MPS formatted file mpms*.dat was read as input. LINDO/386's edit function was initiated, and a 0,1 integer variable g with a coefficient of negative one was added to the objective function. Next,

the production capital budgeting mutually exclusive constraint shown in equation (6.2) was replaced by the constraints shown in equations (6.3) and (6.4). The number of 0,1 integer variables was then set to 32. Zero-one integer variables included five product funding variables (z_{jk}), 26 product setup variables ($\delta(x_{jkt})$), and the project mutually exclusive variable g. The edit function was then terminated and optimization was initiated. When an optimal solution was found, the solution was saved as mpms*.rep where $* = 1,2,\ldots,9$ and corresponded to the nine combinations of demand for the model that allowed partial project selection with no upper bound.

When projects were less than or equal to one and partial project selection were allowed, LINDO/386 was executed and the MPS formatted file mpms*.dat was read as input. LINDO/386's edit function was initiated, and a 0,1 integer variable g with a coefficient of negative one was added to the objective function. Next, the production capital budgeting mutually exclusive constraint shown in equation (6.2) was replaced by the constraints shown in equations (6.6) and (6.7). The number of 0,1 integer variables was then set to 32, and the additional constraints shown in equations (6.9) and (6.10) were added. Zero-one integer variables included five product funding variables (z_{jk}), 26 product setup variables ($\delta(x_{jkt})$), and the production project mutually exclusive variable g. The edit function was then terminated, and optimization was initiated. When an optimal solution was found, the solution was saved to a file named mpms*.rep where $* = 10,11,\ldots,18$ and corresponded to the nine combinations of demand for the model that allowed partial project selection and included an upper bound of one on all projects.

When projects were restricted to values of zero and one, LINDO/386 was executed and the MPS formatted file mpms*.dat was read as input. The LINDO/386 edit function was initiated and the number of 0,1 integer variables was set to 35. Zero-one integer variables included five product funding variables (z_{jk}), 26 product setup variables ($\delta(x_{jkt})$), one pre-production/post-support capital budgeting variable (e_{m_n}), three production capital budgeting variables (e_{m_l}), and one support capital budgeting variable (e_{m_v}). At this point, the edit function was terminated, and optimization was initiated. When an optimal solution was found, the solution was saved as mpms*.rep where $* = 19,20,\ldots,27$ and corresponded to the nine combinations of demand for the model that restricted projects to values of zero and one.

As a final step, each mpms*.rep file created during the optimization process was edited to remove slack and right-hand side information. In addition, the file name, constraint type, and values of D_0 and D_1 were added. After the completion of all tests, there existed 27 MPS formatted files named mpms1.rep, mpms2.rep,...,mpms27.rep.

6.2.9.3. Process to Test Stage Three of the Sequential Model

Figure 6.7 shows the process required to run stage three of the sequential model. The first step was to run the Borland C/C++ program last.exe. This program reads the input file input.dat and the stage two solution file mpms*.rep. An MPS formatted file named mpls.dat was created each time last.exe was run. The MPS file was saved as mpls*.dat where $* = 1,2,\ldots,9$ when projects were unconstrained and partial project selection was allowed, $* = 10,11,\ldots,18$ when projects had an upper bound of one and partial project selection was allowed, and $* = 19,20,\ldots,27$ when projects were restricted to values of zero and one.

There were three specific processes that occurred depending upon the type of capital budgeting constraint considered. When projects were unconstrained and partial project selection was allowed, LINDO/386 was executed and the MPS formatted file mpls*.dat was read as input. LINDO/386's edit function was initiated, and a 0,1 integer variable g with a coefficient of negative one was added to the objective function. Next, the production capital budgeting mutually exclusive constraint shown in equation (6.2) was replaced by the constraints shown in equations (6.3) and (6.4). The number of 0,1 integer variables was then set to 32. Zero-one integer variables included five product funding variables (z_{jk}), 26 product setup variables ($\delta(x_{jkt})$), and the project mutually exclusive variable g. The edit function was then terminated and optimization was initiated. When an optimal solution was found, the solution was saved as mpls*.rep where $* = 1,2,\ldots,9$ and corresponded to the nine combinations of demand for the model that allowed partial project selection with no upper bound.

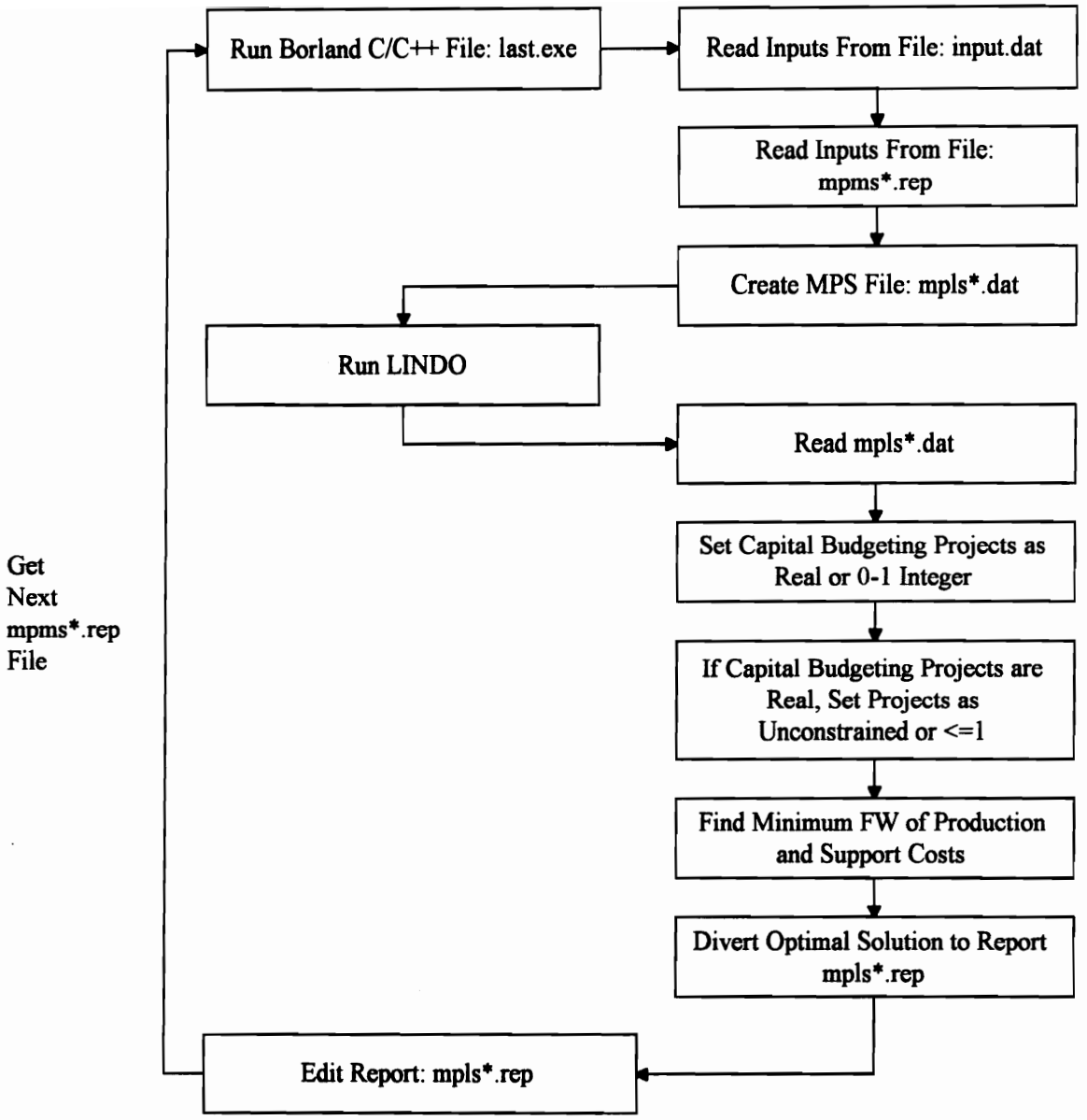

Figure 6.7. Process Chart for Stage Three of the Sequential Model.

When projects could take on values that were less than or equal to one and partial project selection were allowed, LINDO/386 was executed and the MPS formatted file mpls*.dat was read as input. LINDO/386's edit function was initiated, and a 0,1 integer variable g with a coefficient of negative one was added to the objective function. Next, the production capital budgeting mutually exclusive constraint shown in equation (6.2) was replaced by the constraints shown in equations (6.6) and (6.7). The number of 0,1

integer variables was then set to 32, and the additional constraints shown in equations (6.9) and (6.10) were added. Zero-one integer variables included five product funding variables (z_{jk}), 26 product setup variables ($\delta(x_{jkt})$), and the project mutually exclusive variable g. The edit function was then terminated, and optimization was initiated. When an optimal solution was found, the solution was saved to a file named mpls*.rep where * = 10,11,...,18 and corresponded to the nine combinations of demand for the model that allowed partial project selection and included an upper bound of one on all projects.

When projects were restricted to values of zero and one, LINDO/386 was executed and the MPS formatted file mpls*.dat was read as input. The LINDO/386 edit function was initiated and the number of 0,1 integer variables was set to 35. Zero-one integer variables included five product funding variables (z_{jk}), 26 product setup variables ($\delta(x_{jkt})$), one pre-production/post-support capital budgeting variable (e_{m_n}), three production capital budgeting variables (e_{m_l}), and one support capital budgeting variable (e_{m_v}). At this point, the edit function was terminated, and optimization was initiated. When an optimal solution was found, the solution was saved as mpls*.rep where * = 19,20,...,27 and corresponded to the nine combinations of demand for the model that restricted projects to values of zero and one.

As a final step, each mpls*.rep file created during the optimization process was edited to remove slack and right-hand side information. In addition, the file name, constraint type, and values of D_0 and D_1 were added. After the completion of all tests, there existed 27 MPS formatted files named mpls1.rep, mpls2.rep,...,mpls27.rep.

6.2.9.4. Process to Test the Future Worth of the Sequential Model

Figure 6.8 shows the process required to run the future worth component of the sequential model. This component determines the future worth provided by the stage one and stage three solutions to the sequential model. During tests, the set of products funded and the funding levels of the pre-production/post-support capital budgeting projects were determined in stage one, while the funding levels of the product and support capital budgeting projects, and the production, inventory, and labor levels in each fiscal period were determined in stage three. Thus, solutions from stage one and three are combined to determine the future worth. The future worth value yielded by fwss.exe was **directly comparable** to the future worth value obtained from the concurrent model.

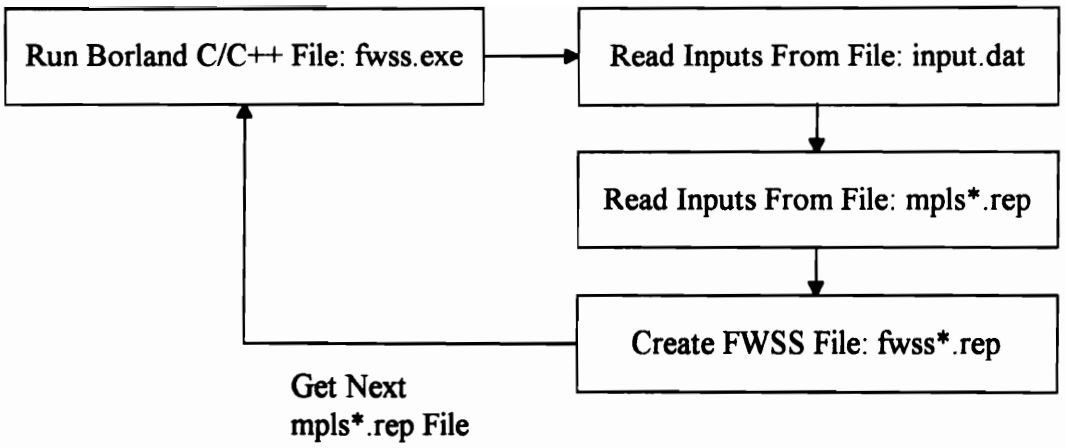

Figure 6.8. Process Chart for Future Worth Module of the Sequential Model.

The first step in the process was to run the Borland C/C++ program fwss.exe. This program reads the input file input.dat and the stage three solution file mpls*.rep. A file named fwssout.dat was created each time last.exe was run. This file contained a listing of decision variable values, and the actual future worth of the solution provided by the sequential model. During development, an identical problem was run through LINDO/386 and fwss.exe to ensure accuracy of the answers provided by fwss.exe. Fwssout.dat was renamed fwss*.dat where $* = 1,2,\ldots,9$ when projects were unconstrained and partial project selection was allowed, $* = 10,11,\ldots,18$ when projects had an upper bound of one and partial project selection was allowed, and $* = 19,20,\ldots,27$ when projects were restricted to values of zero and one.

7. RESULTS, DISCUSSION, AND RELATIONSHIPS

This chapter contains a detailed analysis of the results obtained when all versions of the concurrent and sequential models were tested using the hypothetical example presented in Chapter 6. The chapter is divided into two main sections. The first section contains comparisons of future worth, capital budgeting project funding values, products selected for funding, production and inventory levels, and regular time and overtime labor levels. The second section contains a detailed discussion of the results.

7.1. Results

This section contains a detailed analysis of the 27 tests carried out on three versions (i.e., nine tests of each version) of the concurrent model, and the 27 tests carried out on the three versions of the sequential model. Tests numbered one through nine were conducted on the version of the concurrent and sequential models that considered all five capital budgeting projects ($e_{0_n}, e_{0_{l,}}, e_{1_l}, e_{2_l}$, and e_{0_v}) to be unconstrained, and also allowed partial project selection. Tests conducted with this type of capital budgeting project constraints are denoted by $0 \leq P$ in the following tables. Tests numbered ten through eighteen were conducted on the version of the concurrent and sequential models that considered all five capital budgeting projects ($e_{0_n}, e_{0_{l,}}, e_{1_l}, e_{2_l}$, and e_{0_v}) to be bounded at one, and also allowed partial project selection. Tests with this type of capital budgeting project constraints are denoted by $0 \leq P \leq 1$ in the following tables. Finally, tests numbered nineteen through twenty seven were conducted on the version of the concurrent and sequential models that restricts all five capital budgeting projects ($e_{0_n}, e_{0_{l,}}, e_{1_l}, e_{2_l}$, and e_{0_v}) to values of zero and one. Tests with this type of capital budgeting project constraints are denoted by $P \in 0,1$ in the following tables.

Each test resulted in the determination of the optimal future worth that could be obtained from the inputs in the hypothetical example, and the combination of product sub-group demand. Thus, a total of 54 future worth calculations were made. In addition to future worth, each test of the version of the concurrent and sequential models that restricts capital budgeting project values to zero and one resulted in the determination of the values

of 131 decision variables[7.1]. Each group of 131 decision variables was composed of: five product funding variables (z_{jk}), one pre-production/post-support capital budgeting project variable (e_{m_n}), three production capital budgeting project variables (e_{m_l}), one support capital budgeting project variable (e_{m_v}), 26 production setup variables ($\delta(x_{jkt})$), 26 production quantity variables (x_{jkt}), 21 inventory level variables (i_{jkt}), 24 regular time labor level variables (w_{lt}), and 24 overtime labor level variables (o_{lt}).

A total of $(26x27x2) = 1,404$ setup variable values, $\delta(x_{jkt})$, were determined. These values are not considered in the following analysis because they are either zero when the corresponding production variable, x_{jkt}, equals zero, and one when the corresponding production variable, x_{jkt}, is greater than zero. Since the value of $\delta(x_{jkt})$ is contingent upon the value of x_{jkt}, and all values of x_{jkt} are known, values of $\delta(x_{jkt})$ are not considered in the analysis.

The values of the utilization variables (up_{lt}, op_{lt}, us_t, and os_t) required during optimization of stage one of all versions of the sequential model are not considered in the analysis[7.2]. Each test of the sequential model resulted in a total of 24 values of up_{lt}, 24 values of op_{lt}, 11 values of us_t, and 11 values of os_t. Thus, a total of $(24+24+11+11)(27) = 1,890$ values were determined.

The values of g, the 0,1 variable used to facilitate mutually exclusive projects in versions of the concurrent and sequential models that allowed partial capital budgeting project selection, are not considered in the analysis. A total of eighteen (i.e., $(9x2) = 18$) values of g were determined in two versions of the concurrent models. Further, a total of thirty six (i.e., $(9x2x2) = 36$) values of g were determined in two stages of two versions of the sequential models. Thus, a total of $(18+36) = 54$ values of g were determined.

7.1 As noted in Chapter 6, tests of the two versions of the concurrent and sequential models that allow partial project selection required an additional 0,1 decision variable. In these tests, the value of 132 decision variable values were found.

7.2 The additional variables required during stage one optimization are: up_{lt}, the variable to account for under-utilization of production work center l in period t; op_{lt}, the variable to account for production work center resources used beyond base capacity levels; us_t, the variable to account for under-utilization of the support work center in period t; and os_t, the variable to account for support work center resources used beyond base capacity levels.

The results obtained from all versions of the models were compared based upon $(1x27) = 27$ future worth values, $(5x27) = 135$ product funding variables, $(1x27) = 27$ pre-production/post-support capital budgeting project variables, $(3x27) = 81$ product capital budgeting project variables, $(1x27) = 27$ support capital budgeting project variables, $(26x27) = 702$ production quantity variables, $(21x27) = 567$ inventory level variables, $(24x27) = 648$ regular time labor variables, and $(24x27) = 648$ overtime labor variables. Thus, a total of $(27+135+27+81+27+702+567+648+648) = 2,862$ comparisons between $(2,862x2) = 5,724$ values are made in the following analysis. Further, while a total of $(1,890+5,724+1,404+54) = 9,072$ values were determined during the 54 tests, 5,724 of these are used in the following analysis.

7.1.1. Comparison of Future Worth

The future worth obtained from tests of the concurrent and sequential models were compared in three ways. First, as shown in Table 7.1 and Figure 7.1, the future worth obtained during each test of the concurrent model was compared to the future worth obtained during the corresponding test of the sequential model (a total of 27 comparisons). Next, as shown in Table 7.2 and Figure 7.2, the future worth obtained from each version of the concurrent model run with the nine combinations of product sub-group demand were compared. A total of 27 tests were compared three at a time. Thus, nine comparisons were made. Finally, as shown in Table 7.3 and Figure 7.3, the future worth obtained from each version of the sequential model run with the nine combinations of product sub-group demand were compared. A total of 27 tests were compared three at a time. Thus, nine comparisons were made.

7.1.1.1. Comparison of Future Worth Obtained from All Versions of the Concurrent and Sequential Models

Table 7.1 shows the future worth obtained from each test of the three versions of the concurrent and sequential models. Further, the table shows the difference in future worth obtained from each model.

As shown in Table 7.1, every test of the concurrent model resulted in a higher future worth than was obtained from the same test of the sequential model. Further, the tests showed that, in many cases, as the demand increased the difference between the future worth provided by the concurrent and the sequential models also increased.

Figure 7.1 graphically depicts the data presented in Table 7.1. As such, Figure 7.1 depicts the future worth obtained from the concurrent and sequential models when each

model considered the same product sub-group demand, $D_k \quad \forall \; k \in K$ and capital budgeting constraint type. Further, the figure also depicts the differences in future worth obtained from the concurrent and sequential models.

Table 7.1. Paired Comparisons of Future Worth Obtained from Tests of the Concurrent and Sequential Models.

				Future Worth		
Test No.	**Demand for Products in Sub-Group One**	**Demand for Products in Sub-Group Two**	**Project (P) Constraint Type**	**Concurrent Model**	**Sequential Model**	**Difference**
1	50	50	$0 \le P$	$31,191,558	$29,799,880	$1,391,678
10	50	50	$0 \le P \le 1$	31,191,558	29,799,880	1,391,678
19	50	50	$P \in 0,1$	29,001,844	24,836,808	4,165,036
2	50	100	$0 \le P$	64,097,504	63,126,312	971,192
11	50	100	$0 \le P \le 1$	64,097,504	63,126,312	971,192
20	50	100	$P \in 0,1$	58,730,492	58,473,264	257,228
3	50	150	$0 \le P$	91,142,592	78,492,048	12,650,544
12	50	150	$0 \le P \le 1$	87,303,648	83,134,976	4,186,672
21	50	150	$P \in 0,1$	83,394,808	80,972,016	2,422,792
4	100	50	$0 \le P$	64,127,916	63,568,064	559,852
13	100	50	$0 \le P \le 1$	64,127,916	63,568,064	559,852
22	100	50	$P \in 0,1$	59,019,124	58,864,408	154,716
5	100	100	$0 \le P$	91,309,376	76,216,000	15,093,376
14	100	100	$0 \le P \le 1$	87,394,808	78,453,328	8,941,480
23	100	100	$P \in 0,1$	84,173,360	76,185,312	7,988,048
6	100	150	$0 \le P$	99,539,656	82,053,184	17,486,472
15	100	150	$0 \le P \le 1$	92,794,176	81,567,776	11,226,400
24	100	150	$P \in 0,1$	89,723,728	79,429,264	10,294,464
7	150	50	$0 \le P$	90,632,416	73,517,408	17,115,008
16	150	50	$0 \le P \le 1$	82,986,992	75,420,816	7,566,176
25	150	50	$P \in 0,1$	82,986,992	72,134,320	10,852,672
8	150	100	$0 \le P$	101,956,712	85,887,616	16,069,096
17	150	100	$0 \le P \le 1$	93,415,144	81,966,608	11,448,536
26	150	100	$P \in 0,1$	90,425,200	79,828,064	10,597,136
9	150	150	$0 \le P$	103,195,472	87,219,792	15,975,680
18	150	150	$0 \le P \le 1$	94,582,040	82,243,952	12,338,088
27	150	150	$P \in 0,1$	91,633,024	80,105,440	11,527,584

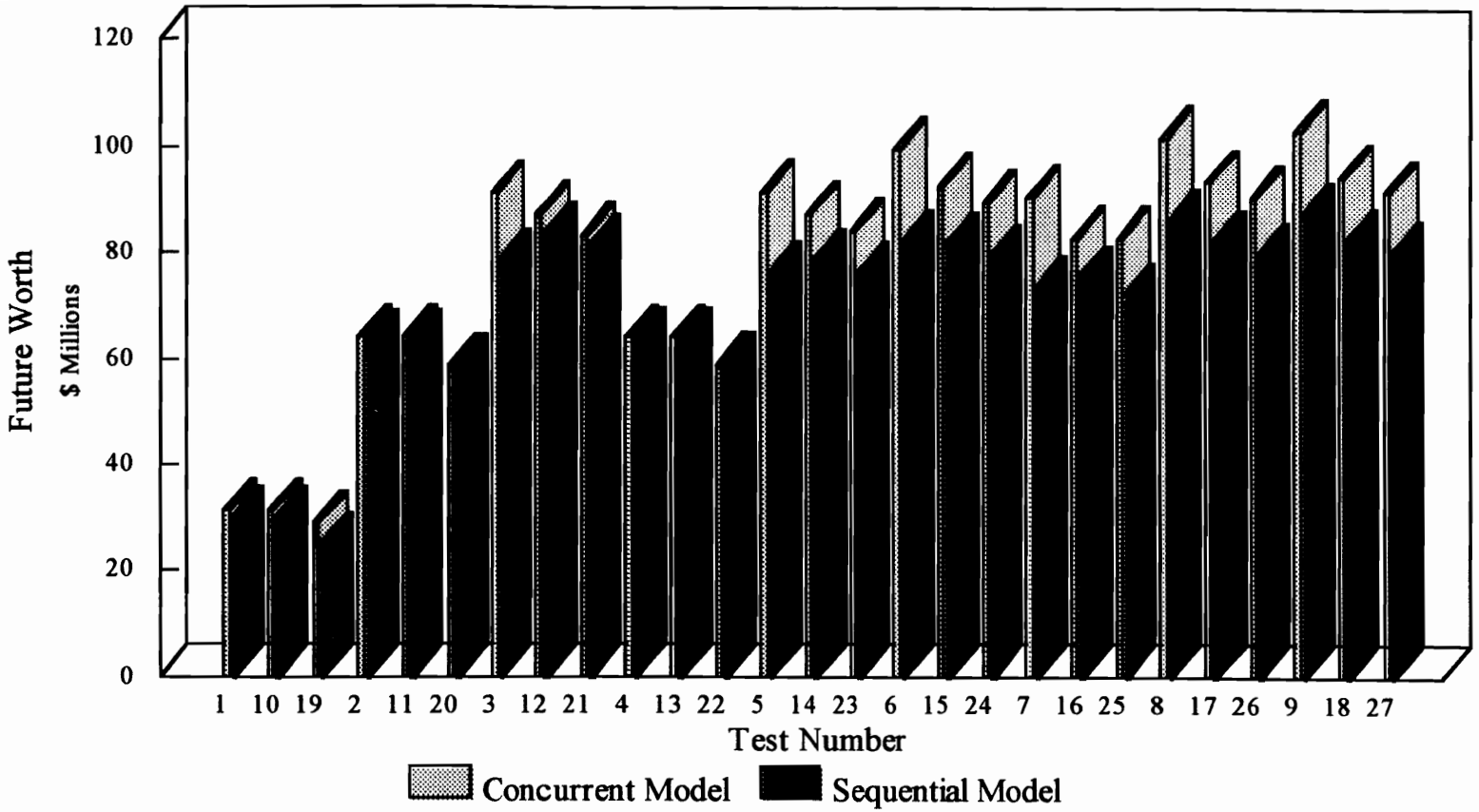

Figure 7.1. Chart Depicting the Future Worth Obtained from Tests of the Concurrent and Sequential Models.

As can be readily seen, the future worth provided by each test of the concurrent model was always higher than the future worth provided by the same test of the sequential model. Further, as demand increased, in most comparisons, the differences between the future worth provided by the two models also increased.

7.1.1.2. Comparison of Future Worth Obtained from the Three Versions of the Concurrent Model

Table 7.2 shows the future worth obtained from each test of the three versions of the concurrent model plus the difference in future worth obtained from each model. As shown in Table 7.2, the version of the concurrent model that restricts capital budgeting projects to values of zero and one provided the lowest future worth in every test[7.3]. The version of the concurrent model that allows partial capital budgeting project selection, and places no upper bound on projects provided the highest future worth in six out of the nine tests. In tests where the future worth was not higher, the future worth was identical to the future worth provided by the version of the concurrent model that allows partial capital budgeting project selection, and places an upper bound of one on projects. Further, the future worth provided by all three versions of the concurrent model increased when demand increased.

Figure 7.2 graphically depicts the data presented in Table 7.2. As such, Figure 7.2 shows the future worth obtained from the three versions of the concurrent model when each version considered the same product sub-group demand, $D_k \quad \forall \; k \in K$. Further, the figure also depicts the differences in future worth obtained from the three versions of the concurrent model when the same product sub-group demand is supplied to each version the concurrent model.

As can be readily seen, the future worth provided by the version of the concurrent model that restricted capital budgeting projects to values of zero and one was always less than the future worth provided by the other two versions of the concurrent model. The

---

7.3 The solution to test 16 - the test where demand for product sub-group zero was 150, and the demand for product sub-group one was 50 - provided in this discussion is actually the solution for test 25. This is because LINDO/386 did not find a solution after 1,559,221 iterations. It was concluded that LINDO/386 was cycling, and as such, probably would never find a feasible solution. Since an optimal solution could not be obtained for test 16, test 16 was assigned the zero-one solution of test 25. As such, the future worth for test 16 reported herein may be understated. It is expected that the cycling experienced in this run of LINDO/386 stems from two primary causes. First, the problem as entered to LINDO/386 may have been poorly scaled. Second, after discussions with Dr. Hanif Sherali, a noted expert in the field of math programming, it is thought that LINDO/386 may be prone to cycling in certain situations.

effect that demand has on each version of the concurrent model is also readily apparent. As demand increased, so too did the future worth provided by each version of the concurrent model.

Table 7.2. Comparison of Future Worth Generated by Three Versions of the Concurrent Model.

			Future Worth				
Test No.	**Sub-Group One Demand**	**Sub-Group Two Demand**	**Version 1** $0 \leq P$	**Version 2** $0 \leq P \leq 1$	**Version 3** $P \in 0,1$	**Maximum Difference**	**Max. Future Worth**
1,10,19	50	50	$31,191,558	$31,191,558	$29,001,844	2,189,714	1 and 2
2,11,20	50	100	64,097,504	64,097,504	58,730,492	5,367,012	1 and 2
3,12,21	50	150	91,142,592	87,303,648	83,394,808	7,747,784	1
4,13,22	100	50	64,127,916	64,127,916	59,019,124	5,108,792	1 and 2
5,14,23	100	100	91,309,376	87,394,808	84,173,360	7,136,016	1
6,15,24	100	150	99,539,656	92,794,176	89,723,728	9,815,928	1
7,16,25	150	50	90,632,416	82,986,992	82,986,992	7,645,424	1
8,17,26	150	100	101,956,712	93,415,144	90,425,200	11,531,512	1
9,18,27	150	150	103,195,472	94,582,040	91,633,024	11,562,448	1

7.1.1.3. Comparison of Future Worth Obtained from the Sequential Models

Table 7.3 shows the future worth obtained from each test of the three versions of the sequential model. Further, the table shows the difference in future worth obtained from each model.

As shown in Table 7.3, the version of the sequential model that allows partial capital budgeting project selection, and also places an upper bound of one on the projects provided the highest future worth in three of the nine tests. Further, in three out of nine tests, the version if the sequential model that allows unconstrained partial project selection provided the highest future worth. In the remaining three tests, the two versions of the

sequential model that allow partial projects selection provided the same future worth. It also places no upper bound on project values. In the three tests where versions of the sequential model that allow partial capital budgeting selection provided the same future worth, the future worth was higher than the future worth provided by the version of the sequential model that restricts capital budgeting projects to values of zero. In one test, the version of the sequential model that restricts capital budgeting projects to values of zero and one provided a higher future worth than the version of the sequential model that allows unconstrained partial project selection. However, the version of the sequential model that restricts projects to values of zero and one provided the lowest future worth in the remaining eight tests.

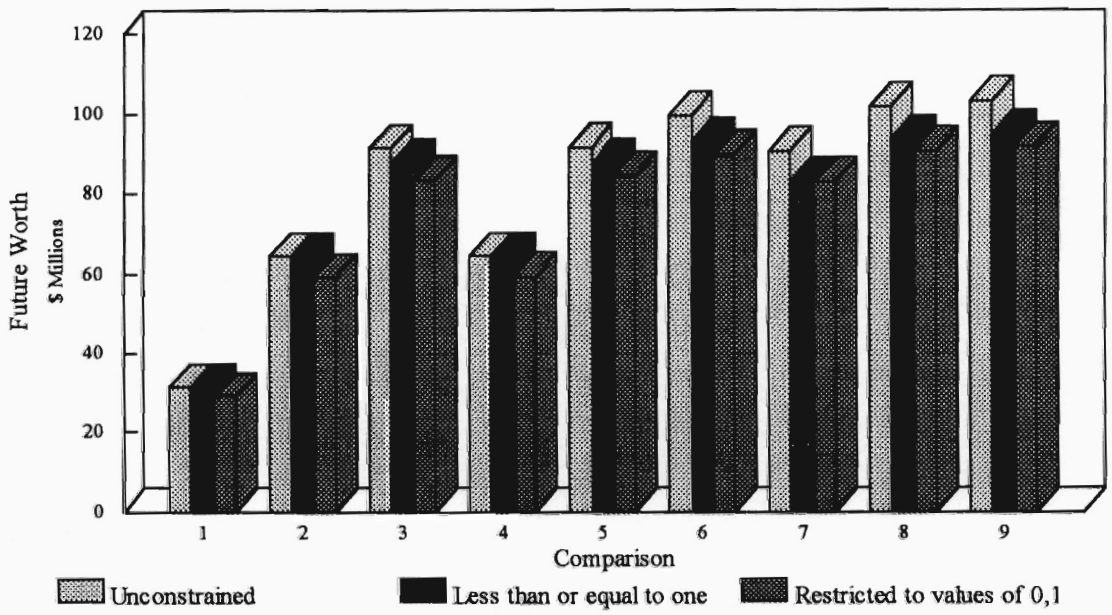

Figure 7.2. Chart Depicting the Future Worth Obtained from Tests of Three Versions of the Concurrent Model.

Figure 7.3 graphically depicts the data presented in Table 7.3. As such, Figure 7.3 shows the future worth obtained from each of the three versions of the sequential model when each version considered the same product sub-group demand, $D_k \quad \forall \; k \in K$. Further, the figure also shows the differences in future worth obtained from the three

versions of the sequential model when the same product sub-group demand is supplied to each version the sequential model.

Table 7.3. Comparison of Future Worth From Sequential Model.

Test No.	Sub-Group One Demand	Sub-Group Two Demand	Future Worth				
			Version 1 $0 \le P$	Version 2 $0 \le P \le 1$	Version 3 $P \in 0,1$	Maximum Difference	Maximum Future Worth
1,10,19	50	50	$29,799,880	29,799,880	24,836,808	$4,963,072	1 and 2
2,11,20	50	100	63,126,312	63,126,312	58,473,264	4,653,048	1 and 2
3,12,21	50	150	78,492,048	83,134,976	80,972,016	4,642,928	2
4,13,22	100	50	63,568,064	63,568,064	58,864,408	4,703,656	1 and 2
5,14,23	100	100	76,216,000	78,453,328	76,185,312	2,268,016	2
6,15,24	100	150	83,053,184	81,567,776	79,429,264	3,623,920	1
7,16,25	150	50	73,517,408	75,420,816	72,134,320	3,256,496	2
8,17,26	150	100	85,887,616	81,966,608	79,828,064	6,059,552	1
9,18,27	150	150	87,219,792	82,243,952	80,105,440	7,114,352	1

As can be readily seen, the future worth provided by the version of the sequential model that restricts capital budgeting projects to values of zero and one was never the best future worth provided in any test. In addition, the tests reveal that the best future worth is always obtained from the two versions of the sequential model that allow partial capital budgeting project selection.

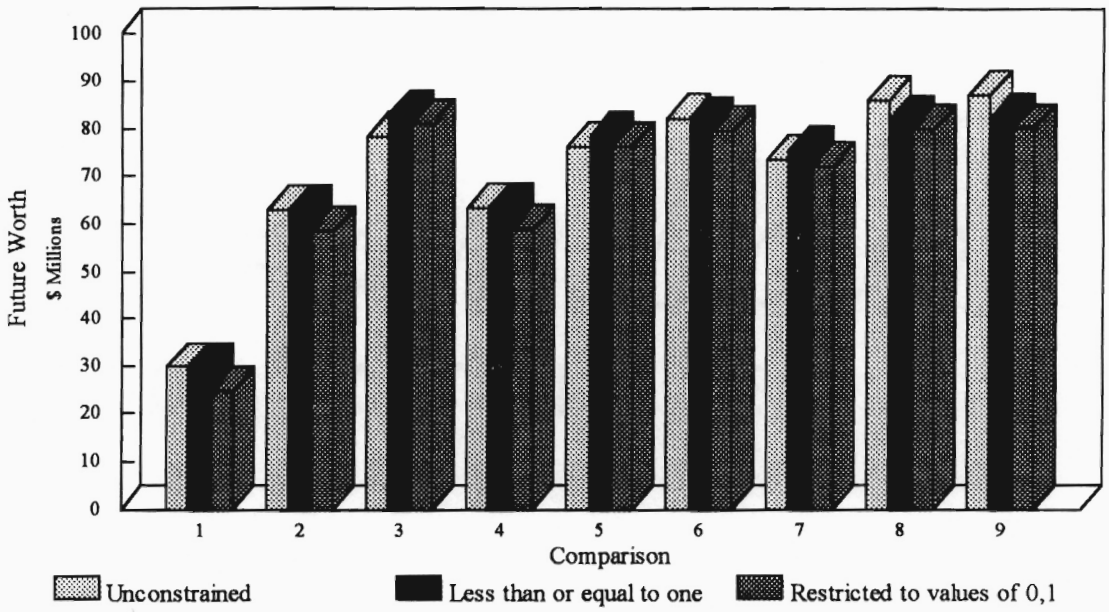

Figure 7.3. Chart Depicting the Future Worth Obtained from Tests of the Three Versions of the Sequential Model.

7.1.2. Comparison of Pre-Production/Post-Support Project Funding Values

The value of the pre-production/post-support capital budgeting project obtained from tests of the concurrent and sequential models were compared in three ways. First, as shown in Figure 7.4, the value of the pre-production/post-support capital budgeting project obtained during each test of the concurrent model was compared to the value of the pre-production/post-support capital budgeting project obtained during the corresponding test of the sequential model (a total of 27 comparisons). Next, as shown in Figure 7.5, the value of the pre-production/post-support capital budgeting project obtained from each version of the concurrent model run with the nine combinations of demand were compared. A total of nine comparisons of three corresponding test results was made. Finally, the value of the pre-production/post-support capital budgeting project obtained from each version of the sequential model run with the nine combinations of demand was compared. A total of nine comparisons of three corresponding test results was made.

7.1.2.1. Comparison of Pre-Production/Post-Support Project Funding Values Obtained from All Tests of The Concurrent and Sequential Models

Table 7.4 shows the value of the pre-production/post-support capital budgeting project obtained from each test of the three versions of the concurrent and sequential models. The data reveal that the corresponding tests of the concurrent and sequential models always resulted in the selection of the same value for the pre-production/post-support capital budgeting project. That is, corresponding tests of the concurrent and sequential models provided the identical results. Thus, for this example problem, there are no differences in the pre-production/post-support capital budgeting funding values provided by the concurrent and sequential models.

Figure 7.4 graphically depicts the data presented in Table 7.4. As such, Figure 7.4 shows the value of the pre-production/post-support capital budgeting project obtained from each of the three versions of the concurrent and sequential models when each model considered the same product sub-group demand, $D_k \quad \forall \; k \in K$ and capital budgeting constraint type. Further, the figure reveals that corresponding versions of both models always provided the same pre-production/post-support project funding value.

7.1.2.2. Comparison of Pre-Production/Post-Support Project Funding Values Obtained from All Versions of The Concurrent Model

Figure 7.5 provides a graphical comparison of the pre-production/post-support project value provided by all three versions of the concurrent model. The data for this figure are contained in Table 7.4.

The figure shows the values of the three production capital budgeting projects obtained from each of the three versions of the concurrent model when each version considered the same product sub-group demand, $D_k \quad \forall \; k \in K$. In every test, both versions of the concurrent model that allow partial capital budgeting projects provided the identical project value. Further, in every test, the version of the concurrent model that restricted capital budgeting projects to values of zero and one always provided the same project value. Finally, in every comparison, the version of the concurrent model that restricts capital budgeting project values to zero and one always provided a higher project value than the project values provided by the other two versions.

Table 7.4. Comparison of Pre-Production/Post-Support Project Values from Tests of All Versions of the Concurrent and Sequential Models.

Test No.	Product Sub-Group 1	Product Sub-Group 2	Project Constraint Type	Value of Capital Budgeting Project e_{0_n}: Concurrent Model	Value of Capital Budgeting Project e_{0_n}: Sequential Model
1	50	50	$0 \leq P$	0.333333	0.333333
10	50	50	$0 \leq P \leq 1$	0.333333	0.333333
19	50	50	$P \in 0,1$	1	1
2	50	100	$0 \leq P$	0.333333	0.333333
11	50	100	$0 \leq P \leq 1$	0.333333	0.333333
20	50	100	$P \in 0,1$	1	1
3	50	150	$0 \leq P$	0.333333	0.333333
12	50	150	$0 \leq P \leq 1$	0.333333	0.333333
21	50	150	$P \in 0,1$	1	1
4	100	50	$0 \leq P$	0.333333	0.333333
13	100	50	$0 \leq P \leq 1$	0.333333	0.333333
22	100	50	$P \in 0,1$	1	1
5	100	100	$0 \leq P$	0.333333	0.333333
14	100	100	$0 \leq P \leq 1$	0.333333	0.333333
23	100	100	$P \in 0,1$	1	1
6	100	150	$0 \leq P$	0.333333	0.333333
15	100	150	$0 \leq P \leq 1$	0.333333	0.333333
24	100	150	$P \in 0,1$	1	1
7	150	50	$0 \leq P$	0.333333	0.333333
16	150	50	$0 \leq P \leq 1$	1	1
25	150	50	$P \in 0,1$	1	1
8	150	100	$0 \leq P$	0.333333	0.333333
17	150	100	$0 \leq P \leq 1$	0.333333	0.333333
26	150	100	$P \in 0,1$	1	1
9	150	150	$0 \leq P$	0.333333	0.333333
18	150	150	$0 \leq P \leq 1$	0.333333	0.333333
27	150	150	$P \in 0,1$	1	1

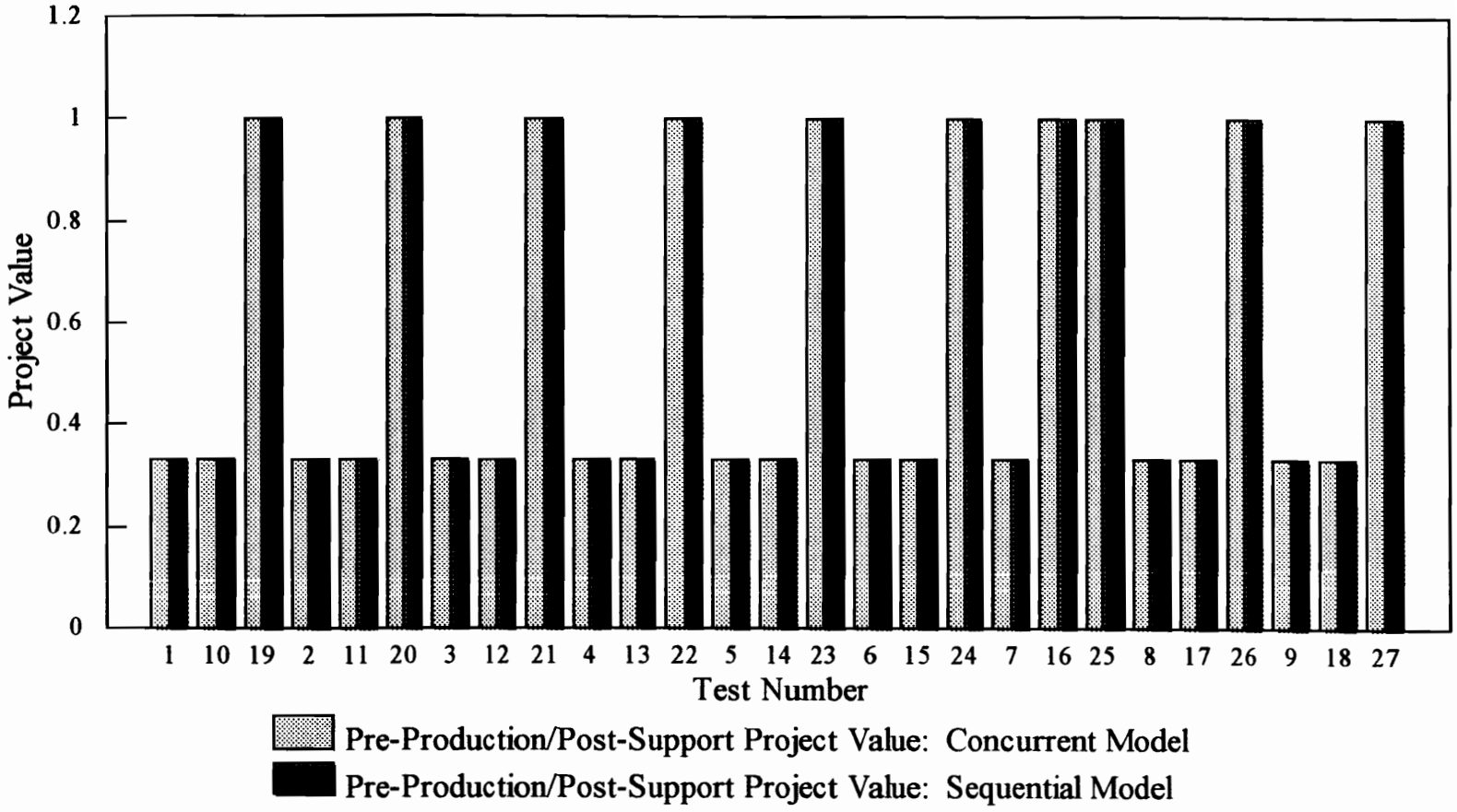

Figure 7.4. Comparison of Funding Values of Pre-Production/Post-Support Project Obtained from Tests of the Concurrent and Sequential Models.

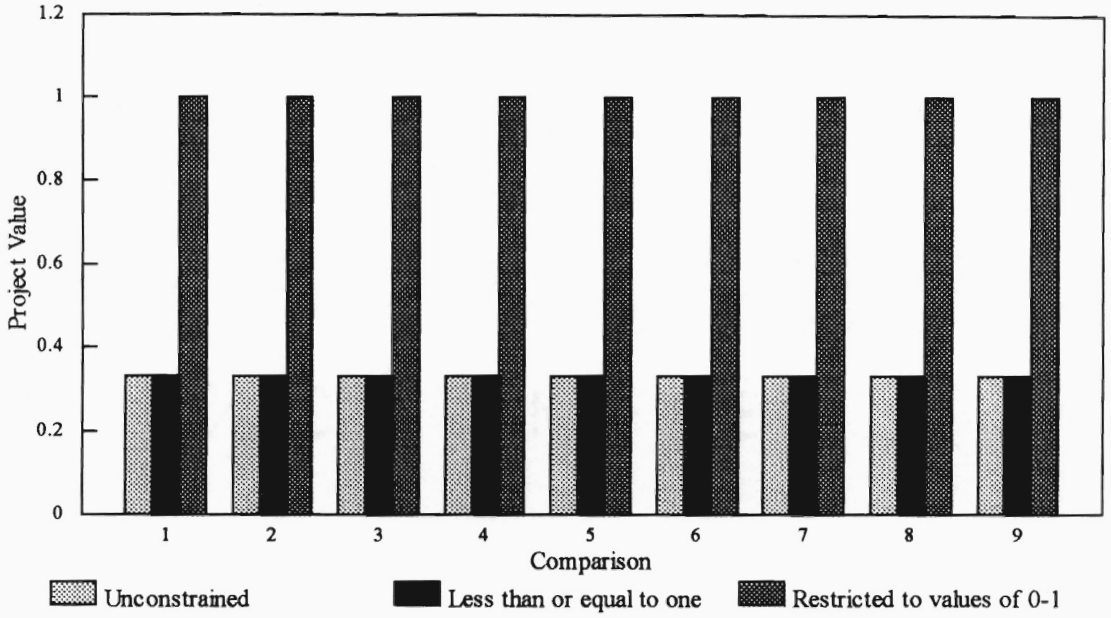

Figure 7.5. Comparison of Pre-Production/Post-Support Project Funding Values Obtained from Tests of the Concurrent Model.

7.1.2.3. Comparison of Pre-Production/Post-Support Project Funding Values Obtained from All Versions of The Sequential Model

Figure 7.6 provides a graphical comparison of the pre-production/post-support project value provided by all three versions of the sequential model. The data for this figure are contained in Table 7.4.

The figure shows the values of the three production capital budgeting projects obtained from each of the three versions of the sequential model when each version considered the same product sub-group demand, $D_k \quad \forall \; k \in K$. In every test, both versions of the sequential model that allow partial capital budgeting project selection provided the identical project value. Further, in every test, the version of the sequential model that restricted capital budgeting projects to values of zero and one always provided the same project value. Finally, in every comparison, the version of the sequential model that restricts capital budgeting project values to zero and one always provided a higher project value than the project values provided by the two other versions.

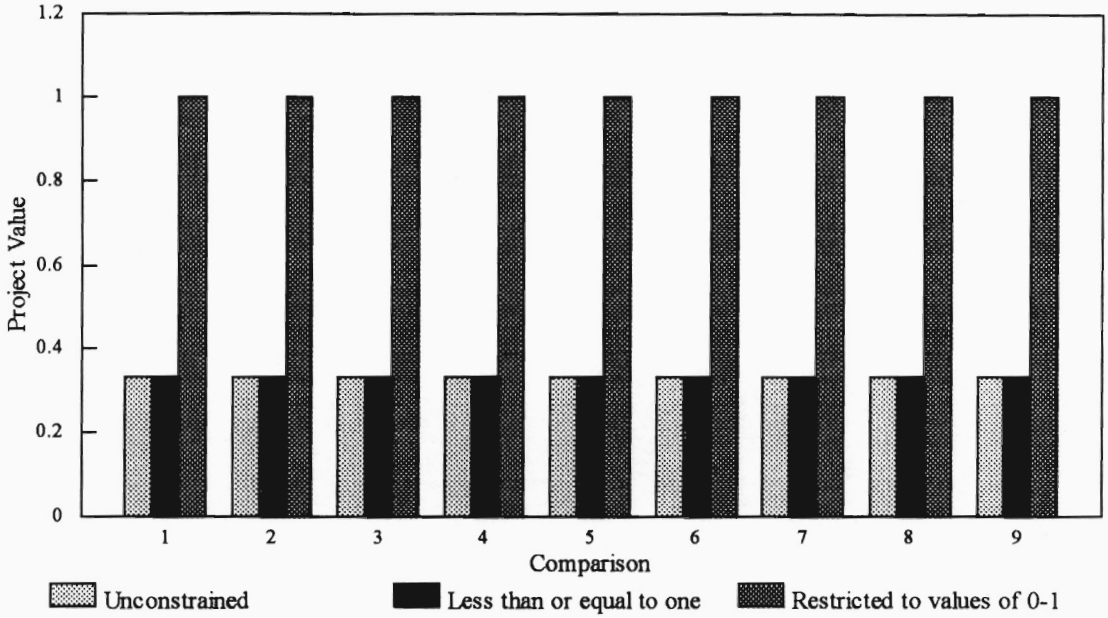

Figure 7.6. Comparison of Pre-Production/Post-Support Project Funding Values Obtained from Tests of the Sequential Model.

7.1.3. Comparison of Production Project Values

The values of the production capital budgeting projects obtained from tests of the concurrent and sequential models were compared in three ways. First, as shown in Table 7.5 and Figures 7.7, 7.8, and 7.9, the values of the production capital budgeting projects obtained during each test of the concurrent model were compared to the values of the production capital budgeting projects obtained during the corresponding test of the sequential model. Thus, a total of 27 comparisons were made between 54 corresponding test results. Next, the values of the production capital budgeting projects obtained from each version of the concurrent model run with the nine combinations of demand were compared. A total of nine comparisons were made between three corresponding tests. Finally, the values of the production capital budgeting projects obtained from each version of the sequential model run with the nine combinations of demand were compared. A total of nine comparisons were made between three corresponding test results.

7.1.3.1. Comparison of Production Project Funding Values Obtained from All Versions of the Concurrent and Sequential Models

Table 7.5 shows the values of the three production capital budgeting projects obtained from each test of the three versions of the concurrent and sequential models.

The data reveal that the version of the concurrent and sequential model that allows unconstrained partial capital budgeting selection always funded production projects one and two. Further, this version of the sequential model always provided higher values of project one and two.

Table 7.5. Comparison of Production Projects Values Provided by Both Models.

Test No.	Sub-Group One	Sub-Group Two	Constraint	Value of Capital Budgeting Projects: Concurrent Model			Value of Capital Budgeting Projects: Sequential Model		
				e_{0_1}	e_{1_1}	e_{2_1}	e_{0_1}	e_{1_1}	e_{2_1}
1	50	50	$0 \leq P$	0	0.003259	0	0	0.122751	0
10	50	50	$0 \leq P \leq 1$	0	0.003259	0	0	0.122751	0
19	50	50	$P \in 0,1$	0	0	0	0	1	0
2	50	100	$0 \leq P$	0	0.556143	0.413704	0	0.706085	0.487774
11	50	100	$0 \leq P \leq 1$	0	0.556143	0.413704	0	0.706085	0.487774
20	50	100	$P \in 0,1$	1	1	0	1	1	0
3	50	150	$0 \leq P$	0	0.846553	0.659405	0	0.887355	2.334455
12	50	150	$0 \leq P \leq 1$	0	0.688705	0.52184	0	0.999945	1.0
21	50	150	$P \in 0,1$	1	1	0	0	1	1
4	100	50	$0 \leq P$	0	0.764476	0.330730	0	0.848953	0.371108
13	100	50	$0 \leq P \leq 1$	0	0.764476	0.330370	0	0.848953	0.371108
22	100	50	$P \in 0,1$	1	1	0	1	1	0
5	100	100	$0 \leq P$	0	1.084948	0.602345	0	1.261068	2.054181
14	100	100	$0 \leq P \leq 1$	0	0.941208	0.484802	0	0.999998	0.999983
23	100	100	$P \in 0,1$	1	1	0	0	1	1
6	100	150	$0 \leq P$	0	1.091546	0.607981	0	1.506692	1.869962
15	100	150	$0 \leq P \leq 1$	0	0.953797	0.495554	0	1.0	0.999982
24	100	150	$P \in 0,1$	1	1	0	0	1	1
7	150	50	$0 \leq P$	0	1.293131	0.517263	0	1.505731	1.869939
16	150	50	$0 \leq P \leq 1$	1	1	0	0	1.0	0.601729
25	150	50	$P \in 0,1$	1	1	0	1	1	0
8	150	100	$0 \leq P$	0	1.298409	0.629979	0	1.506731	1.869939
17	150	100	$0 \leq P \leq 1$	0	1.0	0.533848	0	1.0	0.999977
26	150	100	$P \in 0,1$	1	1	0	0	1	1
9	150	150	$0 \leq P$	0	1.299854	0.654854	0	1.506723	1.869946
18	150	150	$0 \leq P \leq 1$	0	1.0	0.591317	0	1.0	0.999985
27	150	150	$P \in 0,1$	1	1	0	0	1	1

The data reveal that, except for test 16[7.4], the versions of the concurrent and sequential models that allow partial project selection and place an upper bound of one on projects always funded production projects one and two. Further, with the exception of tests 1, 10, 16, 17, and 18, the sequential model always provided higher values of project one and two[7.5].

In three out of nine comparisons[7.6], tests of the versions of the concurrent and sequential models that restricted capital budgeting project to values of zero and one funded the same projects at the same value. Also, in five out of the nine comparisons[7.7], the concurrent model funded production projects zero and one, while the sequential model funded productions projects one and two. Finally, in one comparison[7.8], the concurrent model funded no projects, while the sequential model funded only project one.

Figures 7.7, 7.8, and 7.9 graphically depict the data presented in Table 7.5. As such, the figures show the value of the three production capital budgeting projects obtained from each of the three versions of the concurrent and sequential models when each model considered the same product sub-group demand ($D_k \quad \forall \; k \in K$) and capital budgeting constraint type.

7.1.3.2. Comparison of Production Project Funding Values Obtained from All Versions of the Concurrent Model

Figure 7.10 provides a graphical comparison of the production project values provided by all three versions of the concurrent model. The data for this figure are contained in Table 7.5.

The figure shows the values of the three production capital budgeting projects obtained from each of the three versions of the concurrent model when each version considered the same product sub-group demand ($D_k \quad \forall \; k \in K$).

7.4 As noted previously, test 16 results are actually test 25 results.

7.5 In tests 17 and 18 the concurrent and sequential models both set project one equal to one. In these tests, project one was set at its maximum value. In tests 1 and 10, the concurrent and sequential models both set project two equal to zero.

7.6 These comparisons are tests 20, 22, and 25.

7.7 These comparisons are tests 21, 23, 24, 26, and 27.

7.8 This comparison is test 19.

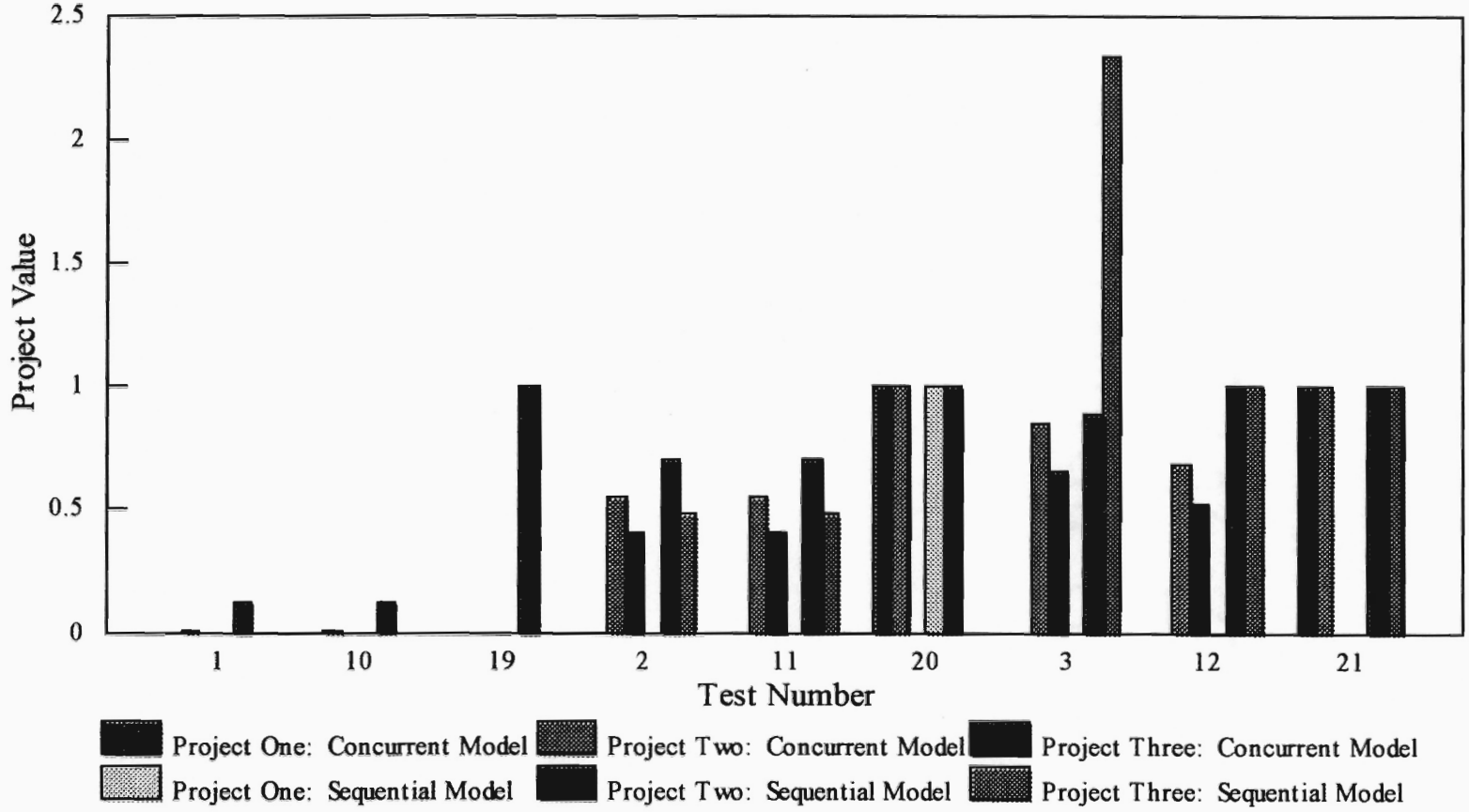

Figure 7.7. Comparison of Production Project Funding Values Obtained from Tests of the Concurrent and Sequential Models - Part One.

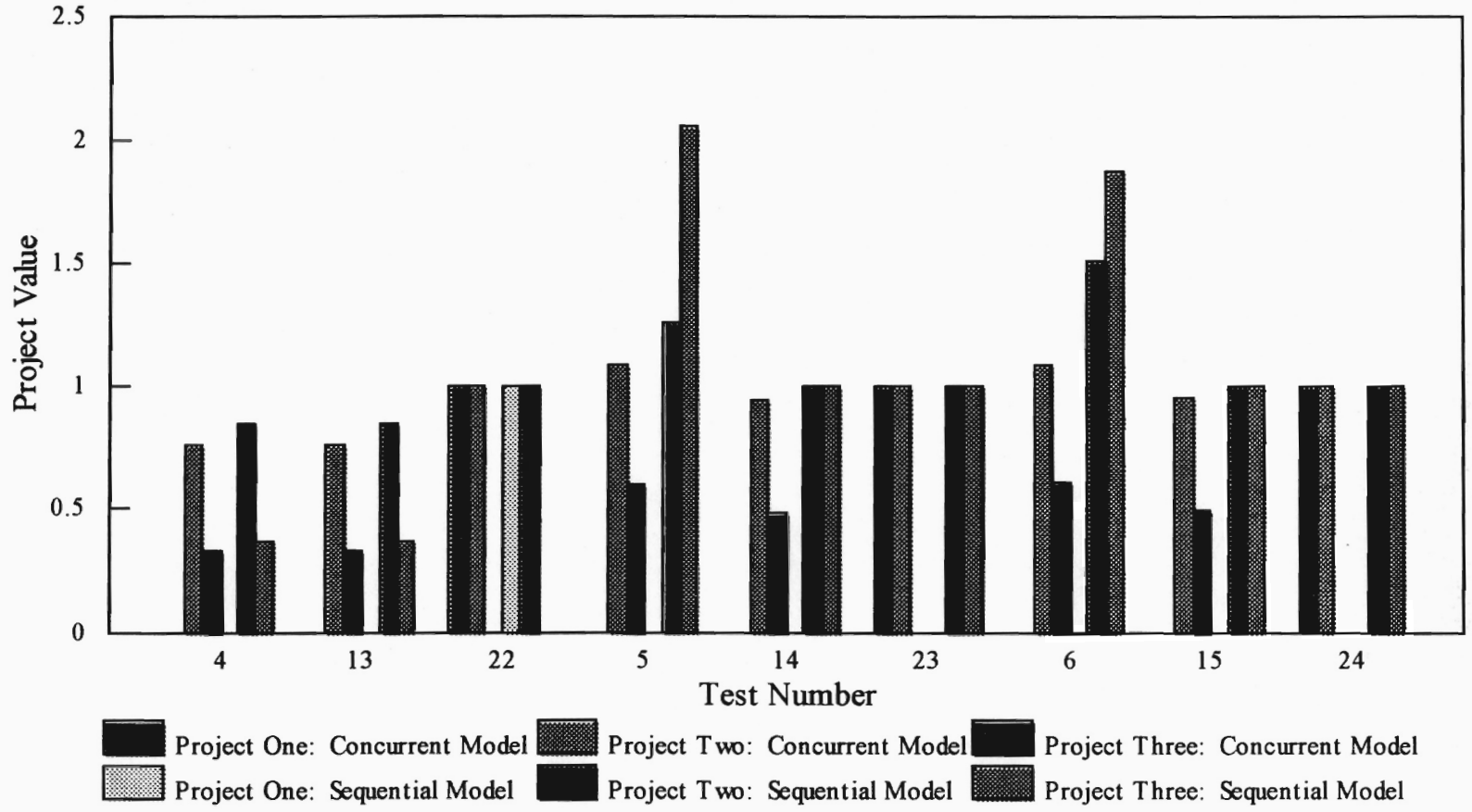

Figure 7.8. Comparison of Production Project Funding Values Obtained from Tests of the Concurrent and Sequential Models - Part Two.

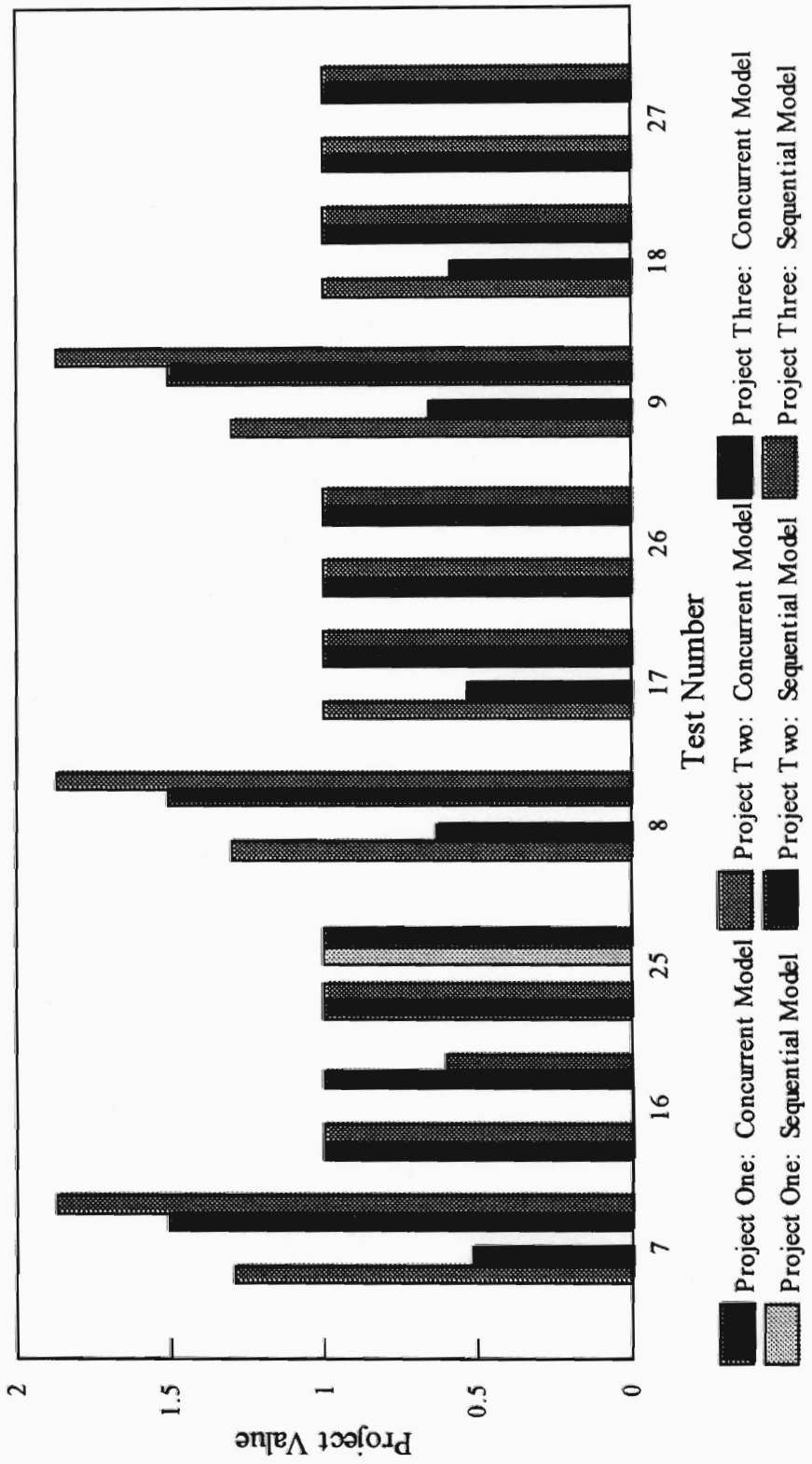

Figure 7.9. Comparison of Production Project Funding Values Obtained from Tests of the Concurrent and Sequential Models - Part Three.

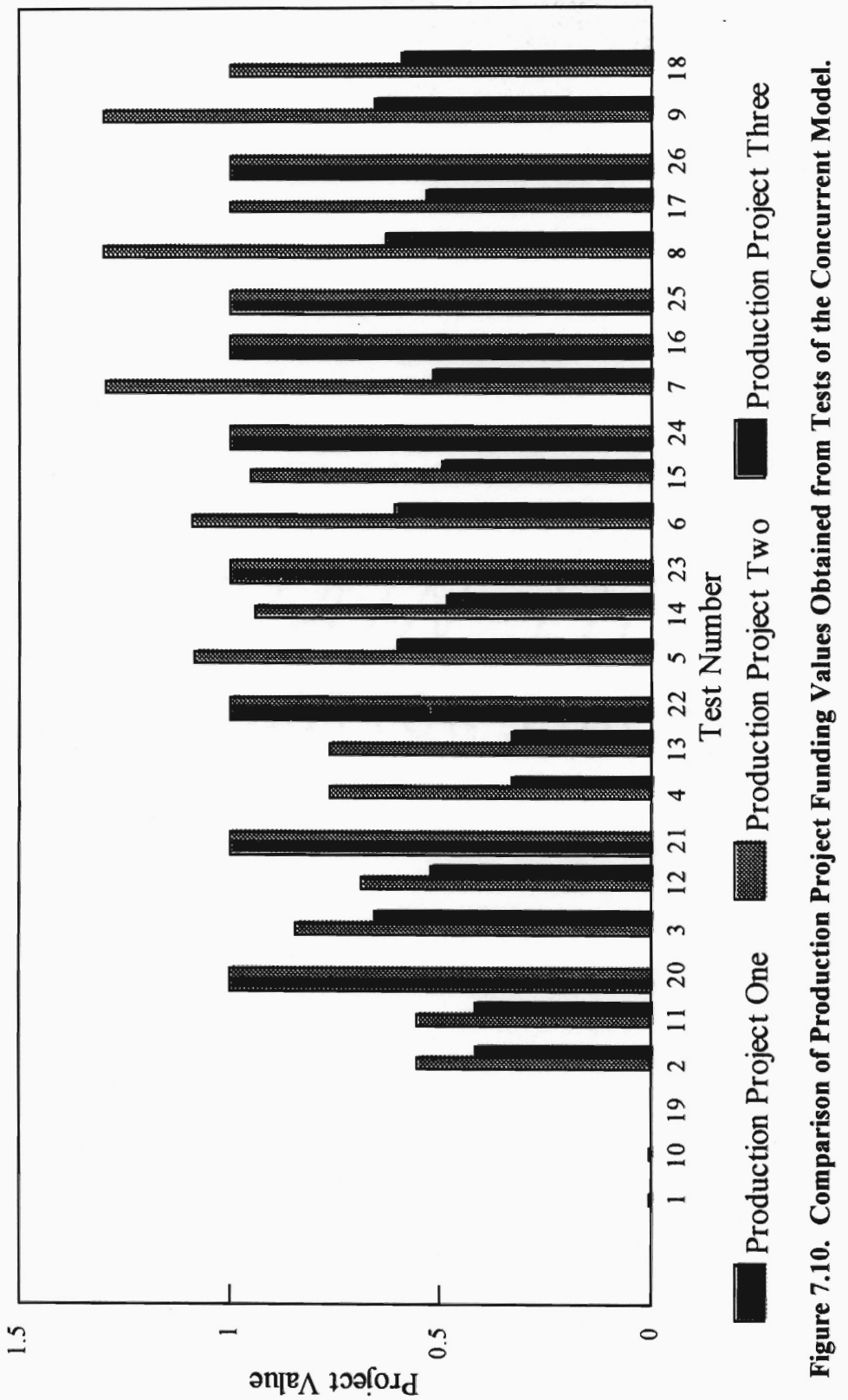

Figure 7.10. Comparison of Production Project Funding Values Obtained from Tests of the Concurrent Model.

The horizontal axis of Figure 7.10 contains the number of each test performed on the concurrent model. Each group of three tests (e.g., 1, 10, 19; 2, 11, 20; 3, 12, 21; etc.) were conducted using the same levels of product sub-group demand. Test one was conducted on the version of the concurrent model that allowed partial project selection with no upper bound, test 10 was conducted on the version of the concurrent model that allowed partial project selection with an upper bound of one on projects, and test 19 was conducted on the version of the concurrent model that restricted capital budgeting projects to values of zero and one. Thus, for each group of three tests, the product sub-group demand was identical while the set of capital budgeting project constraints was not.

Analysis of the data reveal that the version of the concurrent model that restricted capital budgeting project values to zero and one always funded production projects zero and one, and never funded production project two. In all cases, except tests 1 and 16, both versions of the concurrent model that allowed partial project selection funded production project two at values that were less than one and greater than zero. In addition, these models never funded project zero. Thus, the models that allow partial project selection always selected one project that was the same as that selected by the version that restricted values to zero and one, and one project that was different.

7.1.3.3. Comparison of Production Project Funding Values Obtained from All Versions of the Sequential Model

Figure 7.11 provides a graphical comparison of the production project values provided by all three versions of the sequential model. The data for this figure are contained in Table 7.5.

The figure shows the values of the three production capital budgeting projects obtained from each of the three versions of the sequential model when each version considered the same product sub-group demand ($D_k \quad \forall \; k \in K$).

The horizontal axis of Figure 7.11 contains the number of each test performed on the sequential model. Each group of three tests (e.g., 1, 10, 19; 2, 11, 20; 3, 12, 21; etc.) were conducted using the same levels of product sub-group demand. Thus, test one was conducted on the version of the sequential model that allowed partial project selection with no upper bound, test 10 was conducted on the version of the sequential model that allowed partial project selection with an upper bound of one on projects, and test 19 was conducted on the version of the sequential model that restricted capital budgeting projects to values of zero and one. Thus, for each group of three tests, the product sub-group demand was identical while the set of capital budgeting project constraints was not.

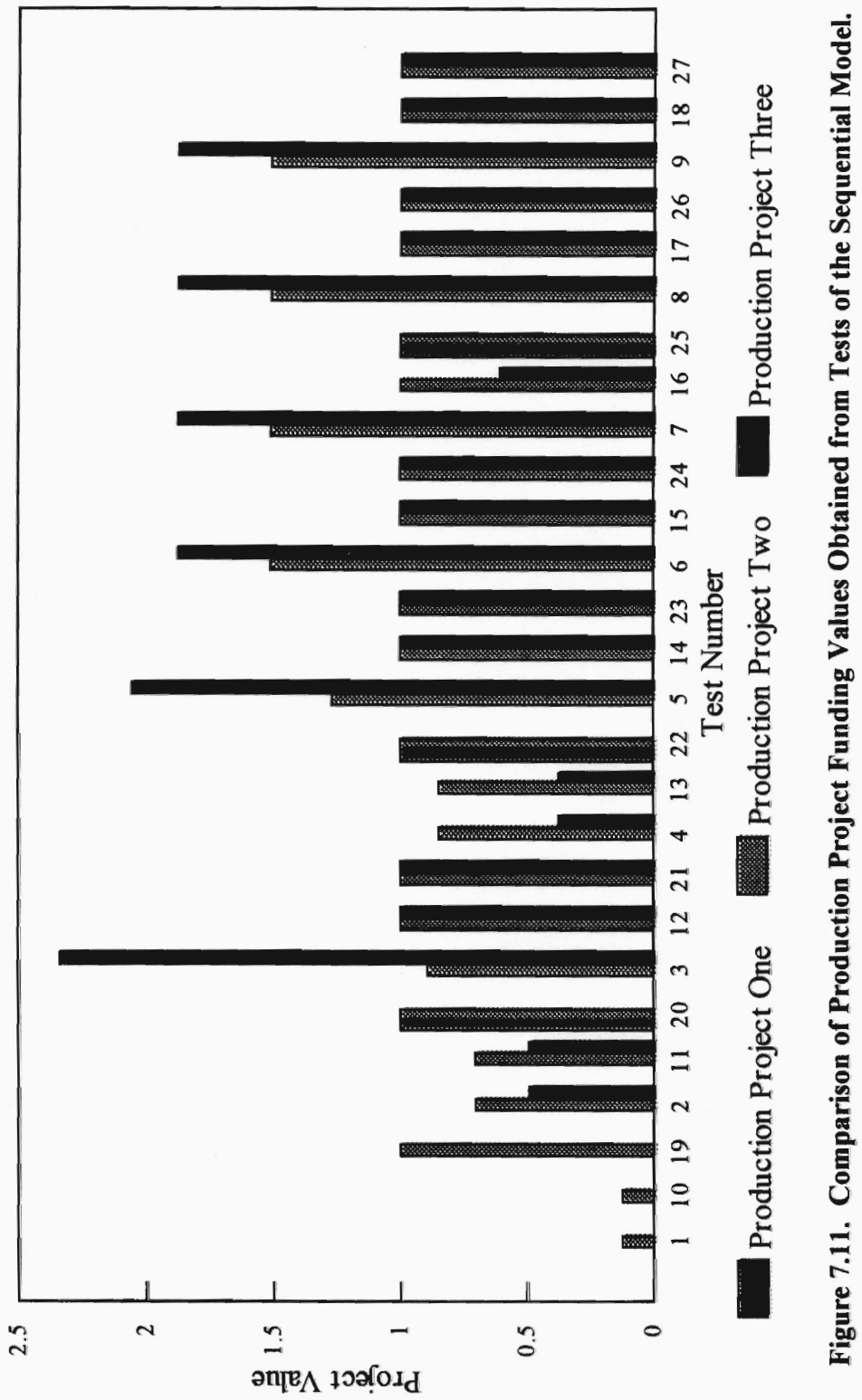

Figure 7.11. Comparison of Production Project Funding Values Obtained from Tests of the Sequential Model.

Analysis of the data reveal that the version of the sequential model that restricted capital budgeting project values to zero and one funded production projects zero and one in three out nine tests[7.9], funded production projects one and two in five out of the nine tests[7.10], and funded only production project one in one out of the nine tests[7.11]. The versions of the sequential model that allow partial project selection funded production projects one and two in eight out of nine tests[7.12], while both versions funded only project one in the remaining test[7.13].

In five out of nine comparisons[7.14], the version of the sequential model that allows unconstrained partial project selection provided higher values of production capital budgeting projects one and two than were provided by the version of the sequential model that allows partial project selection and bounds project values by one. In three out of nine comparisons[7.15], the versions that allowed partial project selection provided production project values that were identical. In the remaining comparison, each version of the sequential model that allows partial project selection provided one project value that is higher and one that is lower[7.16].

7.1.4. Comparison of Support Project Funding Values

The values of the support capital budgeting projects obtained from tests of the concurrent and sequential models were compared in three ways. First, as shown in Table 7.6 and Figure 7.12, the values of the support capital budgeting projects obtained during each test of the concurrent model were compared to the values of the support capital

---

7.9 Production projects zero and one were funded by the version of the sequential model that restricts project values to zero and one in tests 20, 22, and 25.

7.10 Production projects one and two were funded by the version of the sequential model that restricts project values to zero and one in tests 21, 23, 24, 26, and 27.

7.11 Production project one was funded by the version of the sequential model that restricts project values to zero and one in test 19.

7.12 Production projects one and two were funded by the versions of the sequential model that allow partial project selection in tests 2 through 9, and 11 through 18.

7.13 Production project one was funded by the versions of the sequential model that allow partial project selection in tests 1 and 10.

7.14 Higher values of production projects were provided in comparisons of tests 5 and 14 through tests 9 and 27.

7.15 Identical values of production projects were provided in comparisons of tests 1 and 10, 2 and 11, and 4 and 13.

7.16 These values were provided in tests 3 and 12.

budgeting projects obtained during the corresponding test of the sequential model (a total of 27 comparisons). Also, as shown in Figure 7.13, the values of the support capital budgeting project obtained from each version of the concurrent model run with the nine combinations of demand were compared. A total of nine comparisons were made. Finally, as shown in Figure 7.14, the values of the support capital budgeting project obtained from each version of the sequential model run with the nine combinations of demand were compared.

7.1.4.1. Comparison of Support Project Funding Values Obtained from All Versions of the Concurrent and Sequential Models

Table 7.6 shows the value of the support capital budgeting project obtained from each test of the three versions of the concurrent and sequential models. The data reveal that in 24 out of 27 tests, all versions of the concurrent model funded the support project. Further, it is revealed that in the same 24 tests of the sequential model, the support project was also funded.

Figure 7.12 provides a graphical comparison of the support project value provided by all tests of the concurrent and sequential models. The data for this figure are contained in Table 7.6.

Figure 7.12 clearly shows the value of the support capital budgeting project obtained from the tests of the concurrent and sequential models when each version considered the same product sub-group demand ($D_k \quad \forall \; k \in K$), and capital budgeting project type. In seven out of nine tests of the versions of the concurrent and sequential models that allow unconstrained partial project selection, the sequential model provided a lower value for the support project than was obtained from the same version of the concurrent model[7.17]. In one out of the nine tests of this version, the sequential model provided a higher value for the support project than was obtained from the same version of the concurrent model[7.18]. In the remaining test of this version, both the concurrent and sequential models did not fund the support project.

[7.17] The version of the sequential model that allows unconstrained partial project selection provided lower support project values in tests 2, 3, 4, 5, 6, 7, and 8.

[7.18] The version of the sequential model that allows unconstrained partial project selection provided the highest support project value in test 9.

Table 7.6. Comparison of Support Projects Funded by All Versions of the Concurrent and Sequential Models.

Test No.	Demand for Sub-Group One	Demand for Sub-Group Two	Project Constraint	Value of Support Capital Budgeting Project $e0_v$: Concurrent Model	Value of Support Capital Budgeting Project $e0_v$: Sequential Model
1	50	50	$0 \leq P$	0	0
10	50	50	$0 \leq P \leq 1$	0	0
19	50	50	$P \in 0,1$	0	0
2	50	100	$0 \leq P$	0.697350	0.697344
11	50	100	$0 \leq P \leq 1$	0.697350	0.697344
20	50	100	$P \in 0,1$	1	1
3	50	150	$0 \leq P$	1.25	1.015766
12	50	150	$0 \leq P \leq 1$	1	0.985777
21	50	150	$P \in 0,1$	1	1
4	100	50	$0 \leq P$	0.681933	0.681927
13	100	50	$0 \leq P \leq 1$	0.681933	0.681927
22	100	50	$P \in 0,1$	1	1
5	100	100	$0 \leq P$	1.25	0.975281
14	100	100	$0 \leq P \leq 1$	1.0	0.924301
23	100	100	$P \in 0,1$	1	1
6	100	150	$0 \leq P$	1.25	1.131778
15	100	150	$0 \leq P \leq 1$	1.0	1.0
24	100	150	$P \in 0,1$	1	1
7	150	50	$0 \leq P$	1.203193	0.8232
16	150	50	$0 \leq P \leq 1$	1	0.703186
25	150	50	$P \in 0,1$	1	1
8	150	100	$0 \leq P$	1.232139	1.181151
17	150	100	$0 \leq P \leq 1$	1.0	1.0
26	150	100	$P \in 0,1$	1	1
9	150	150	$0 \leq P$	1.235136	1.25
18	150	150	$0 \leq P \leq 1$	1.0	1.0
27	150	150	$P \in 0,1$	1	1

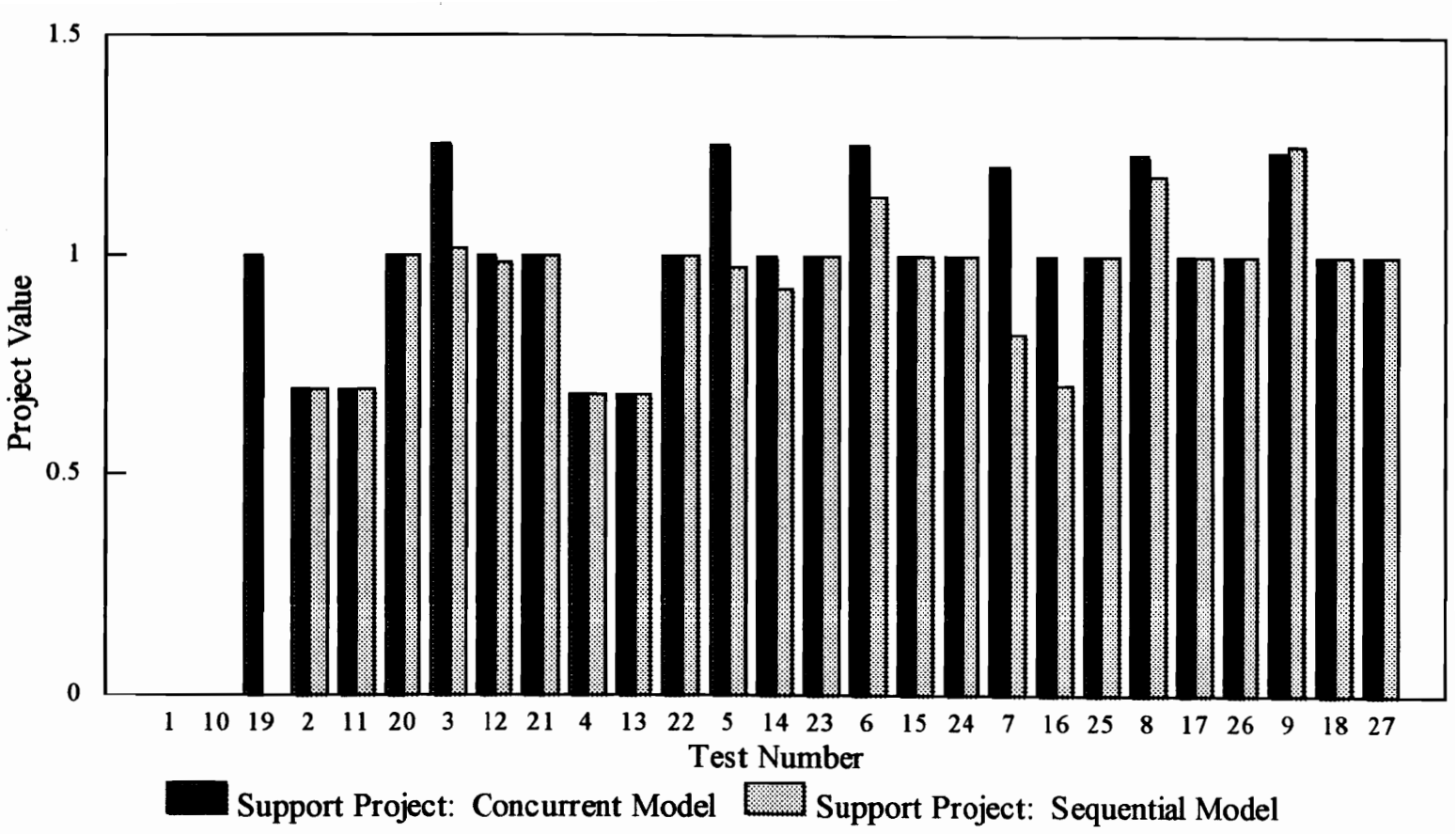

Figure 7.12. Comparison of Support Project Funding Values Obtained from Tests of All Versions of the Concurrent and Sequential Models.

In four out of nine tests of the versions of the concurrent and sequential models that allow partial project selection and bound projects at one, the sequential model provided a lower value for the support project than was obtained from the same version of the concurrent model[7.19]. In the remaining five tests of this version, both models provided the same support project value[7.20].

7.1.4.2. Comparison of Support Project Funding Values Obtained from All Versions of the Concurrent Model

Figure 7.13 provides a graphical comparison of the support project level provided by all three versions of the concurrent model. The data for this table are contained in Table 7.6.

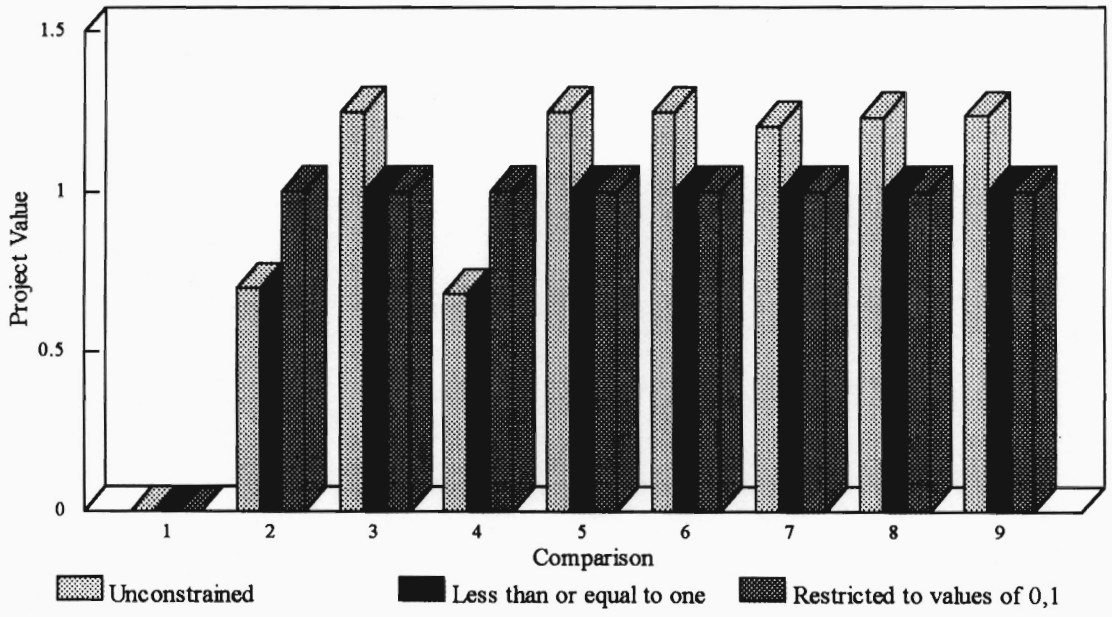

Figure 7.13. Comparison of Support Project Funding Values Obtained from Tests of All Versions of the Concurrent Model.

7.19 The version of the sequential model that allows partial project selection and bounds projects at one provided lower support project values in tests 12, 13, 14, and 15.

7.20 The version of the concurrent and sequential models that allow partial project selection and bound projects at one both provided a support project value of one in tests 15, 17, and 18. Further, this version of these models provided a support project value of zero in test 10. Finally, both models provided identical values that are less than one and greater than zero in test 11.

The figure clearly shows the value of the support capital budgeting project obtained from each of the three versions of the concurrent model when each version considered the same product sub-group demand ($D_k \quad \forall \; k \in K$).

As shown in Figure 7.13, in six out of the nine comparisons, the version of the concurrent model that allows unconstrained partial capital budgeting project selection selected the highest value of the support project. In two of the remaining three comparisons, the version of the concurrent model that restricts capital budgeting projects to values of zero and one selected the highest value of the support project. Finally, all versions of the concurrent model provided a value of zero for the support project in the remaining comparison.

7.1.4.3. Comparison of Support Project Funding Values Obtained from All Versions of the Sequential Model

Figure 7.14 provides a graphical comparison of the support project level provided by all three versions of the sequential model. The data for this table are contained in Table 7.6.

The figure clearly shows the value of the support capital budgeting project obtained from each of the three versions of the sequential model when each version considered the same product sub-group demand ($D_k \quad \forall \; k \in K$).

As shown in Figure 7.14, in four out of the nine comparisons, the version of the sequential model that allows unconstrained partial capital budgeting project selection selected the highest value of the support project[7.21]. In four of the remaining five comparisons, the version of the sequential model that restricts capital budgeting projects to values of zero and one selected the highest value of the support project[7.22]. Finally, all versions of the sequential model provided a value of zero for the support project in the remaining comparison.

[7.21] The version of the sequential model that allows unconstrained partial project selection provided the highest values of the support project in comparisons 3, 6, 8 and 9.

[7.22] The version of the sequential model that restricts projects values to zero and one provided the highest values of the support project in comparisons 2, 4, 5, and 7.

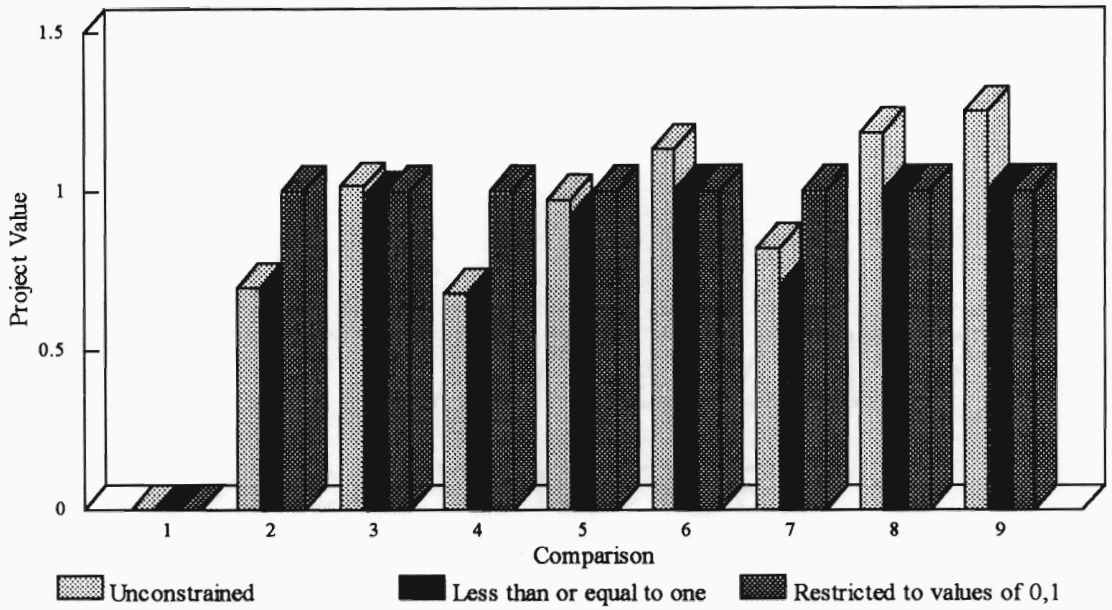

Figure 7.14. Comparison of Support Project Values Obtained from Tests of All Versions of the Sequential Model.

7.1.5. Comparison of Products Funded by All Versions of the Concurrent and Sequential Models

Table 7.7 shows the values of the product funding variables (z_{jk}) obtained from each test of the three versions of the concurrent and sequential models. A product with a one in its column was funded, while a zero means that product was not funded.

Note that product funding variables are not dependent upon the fiscal period, t. That is, when a funding variable, z_{jk}, is set equal to one, products may be produced in each production period $t = t_{b_{jk}},\ldots,t_{e_{jk}}$ (i.e., $x_{jkt} \geq 0$) and also may be stored in inventory in each period $t = t_{b_{jk}},\ldots,t_{e_{jk}} - 1$ (i.e., $i_{jkt} \geq 0$). However, when a product funding variable, z_{jk}, is set equal to zero, no production in each production period $t = t_{b_{jk}},\ldots,t_{e_{jk}}$ can occur (i.e., $x_{jkt} = 0$). Further, since no production can occur, no inventories can be carried (i.e., $i_{jkt} = 0 \quad \forall\ t = t_{b_{jk}},\ldots,t_{e_{jk}} - 1$).

Table 7.7. Comparison of Products Funded by All Versions of the Concurrent and Sequential Models.

Test No.	Group 1 Demand	Group 2 Demand	Constraint Type	Products Funded: Concurrent Model					Products Funded: Sequential Model				
				z_{00}	z_{00}	z_{10}	z_{20}	z_{01}	z_{11}	z_{10}	z_{20}	z_{01}	z_{11}
1	50	50	$0 \le P$	1	1	0	1	1	1	0	0	1	1
10	50	50	$0 \le P \le 1$	1	1	0	1	1	1	0	1	1	1
19	50	50	$P \in 0,1$	1	1	0	1	1	1	0	1	1	1
2	50	100	$0 \le P$	1	1	0	1	1	1	0	1	1	1
11	50	100	$0 \le P \le 1$	1	1	0	1	1	1	0	1	1	1
20	50	100	$P \in 0,1$	1	1	0	1	1	1	0	1	1	1
3	50	150	$0 \le P$	1	1	0	1	1	1	0	1	1	1
12	50	150	$0 \le P \le 1$	1	1	0	1	1	1	0	1	1	1
21	50	150	$P \in 0,1$	1	1	0	1	1	1	0	1	1	1
4	100	50	$0 \le P$	1	1	0	1	1	1	0	1	1	1
13	100	50	$0 \le P \le 1$	1	1	0	1	1	1	0	1	1	1
22	100	50	$P \in 0,1$	1	1	0	1	1	1	0	1	1	1
5	100	100	$0 \le P$	1	1	0	1	1	1	0	1	1	1
14	100	100	$0 \le P \le 1$	1	1	0	1	1	1	0	1	1	1
23	100	100	$P \in 0,1$	1	1	0	1	1	1	0	1	1	1
6	100	150	$0 \le P$	1	1	0	1	1	1	0	1	1	1
15	100	150	$0 \le P \le 1$	1	1	0	1	1	1	0	1	1	1
24	100	150	$P \in 0,1$	1	1	0	1	1	1	0	1	1	1
7	150	50	$0 \le P$	1	1	0	1	1	1	0	1	1	1
16	150	50	$0 \le P \le 1$	1	1	0	1	1	1	0	1	1	1
25	150	50	$P \in 0,1$	1	1	0	1	1	1	0	1	1	1
8	150	100	$0 \le P$	1	1	0	1	1	1	0	1	1	1
17	150	100	$0 \le P \le 1$	1	1	0	1	1	1	0	1	1	1
26	150	100	$P \in 0,1$	1	1	0	1	1	1	0	1	1	1
9	150	150	$0 \le P$	1	1	0	1	1	1	0	1	1	1
18	150	150	$0 \le P \le 1$	1	1	0	1	1	1	0	1	1	1
27	150	150	$P \in 0,1$	1	1	0	1	1	1	0	1	1	1

The data contained in Table 7.7 reveal that the corresponding tests of the concurrent and sequential models always resulted in the selection of the same value for each product funding variable. That is, corresponding tests of the concurrent and sequential models provided the identical results. Thus, for this hypothetical example, there

are no differences in the values of product funding variables provided by the concurrent and sequential models.

7.1.6. Comparison of Production and Inventory Values

A total of 27 comparisons of production and inventory values determined during the tests of all versions of the concurrent and sequential models are contained in Appendix G. In general, these comparisons reveal that the sequential model usually carried higher inventory quantities than the concurrent model. Because of this, all versions of the concurrent model usually had more periods of production, with lower production quantities in each period. Thus, production was more level from period to period in all versions of the concurrent model.

In every test, all versions of the concurrent model specified that product zero should be produced in all fiscal periods zero through three[7.23]. Further, in every test of the version of the concurrent model that allows unconstrained partial project selection, product three was produced in all fiscal periods zero through two[7.24]. In seven out of nine tests of the version of the concurrent model that allows partial project selection with project values bounded at one, product three was produced in all fiscal periods zero through two[7.25]. In the remaining two tests of this version, product three was produced only in fiscal periods zero and one[7.26]. In seven out of nine tests of the version of the concurrent model that restricts projects to values of zero and one, product three was produced in all fiscal periods zero through two[7.27]. In the remaining two tests of this version, product three was produced only in fiscal periods zero and one[7.28].

[7.23] These production quantities of product zero were obtained in tests 1 through 9, and are shown in Tables G.1 through G.9 of Appendix G.

[7.24] These production quantities were obtained in tests 1 through 9, and are shown in Tables G.1 through G.9 of Appendix G.

[7.25] These production quantities of product three were obtained in tests 10 through 16, and are shown in Tables G.10 through G.16 of Appendix G.

[7.26] These production quantities of product three were obtained in tests 17 and 18, and are shown in Tables G.17 and G.18 of Appendix G.

[7.27] These production quantities of product three were obtained tests 19 through 25, and are shown in Tables G.19 through G.25 of Appendix G.

[7.28] These production quantities of product three were obtained in tests 26 and 27, and are shown in Tables G.26 and G.27 of Appendix G.

In all tests of the sequential model, units of product zero[7.29] were always produced in at least two fiscal periods. Further, in all tests of the sequential model, product three was produced in at least one period.

7.1.7. Comparison of Regular and Overtime Values

A total of 27 comparisons between regular time and overtime labor levels determined during all tests of all versions of the concurrent and sequential models are contained in Appendix H. In general, these comparisons reveal that, in periods when all versions of the sequential model specified production quantities greater than zero, all versions of the sequential model usually specified more overtime hours than the corresponding version of the concurrent model. Thus, the sequential model specified more total labor hours in these periods.

7.2. Discussion

After analysis of the data, it is clear that, for this hypothetical example, all versions of the concurrent model always provided the best results with respect to the future worth when compared to the corresponding version of the sequential model. Thus, the best product, production, and capacity plan may be obtained from the concurrent model.

The discussions in this section address the future worth, capital budgeting project values, and products funded. The major differences between performance of the concurrent and sequential models are shown.

7.2.1. Future Worth

In every test, the future worth provided by the concurrent model was greater than the future worth provided by the sequential model. Because the future worth found by each model was an optimum value[7.30], neither model could provide a higher future worth. Thus, in this hypothetical example, the optimal solution provided by the concurrent model is always superior to the optimal solution provided by the sequential model.

The future worth provided by all versions of the concurrent and sequential models is dependent upon the product sub-group demand. This relationship is evident in Figure 7.1. For the most part, as the product sub-group demand increases, the future worth obtained from both models increases.

[7.29] These production and inventory quantities are shown in Tables G.1 through G.27 of Appendix G.

[7.30] Except in the case of test 16 for the concurrent model.

The capital budgeting constraint considered by the concurrent model has a direct effect on the future worth obtained from tests. This relationship is readily apparent in Figure 7.2. In general, partial project selection always resulted in a higher future worth than when projects are restricted to values of zero and one. Further, in three out of nine comparisons[7.31], the future worth obtained from the version of the concurrent model that allows unconstrained partial project selection was identical to the future worth obtained from the version that bounds projects at one.

The capital budgeting constraint considered by the sequential model has a direct effect on the future worth obtained from tests. This relationship is readily apparent in Figure 7.3. In general, partial project selection always resulted in a higher future worth than when projects are restricted to values of zero and one. Further, in three out of nine comparisons[7.32], the future worth obtained from the version of the sequential model that bounds projects at one was higher than was obtained from the unconstrained version. In addition, in three out of nine comparisons, the future worth obtained from the version of the sequential model that allows partial project selection was higher than was obtained from the version that allows partial project selection and also bounds projects at one[7.33]. In the remaining three tests, both versions of the sequential model that allow partial project selection provided that same future worth.

In comparisons 3, 5, 6, 7, 8, and 9, the version of the concurrent model that allows unconstrained partial project selection reported the highest values of future worth. These comparisons are shown in Figure 7.2. For the most part, when product sub-group demand was high, the best results from the concurrent model were obtained from the version that allows unconstrained partial project selection. In the same comparisons of the sequential model, the versions that allow partial project selection provided the highest future worth. These comparisons are shown in Figure 7.3. In every comparison, the concurrent model reported a higher future worth than the sequential model. This relationship is shown in Figure 7.1.

7.31 These comparisons (3, 5, 6, 7, 8, and 9) are shown in Figure 7.2.

7.32 These comparisons (3, 5, and 7) are shown in Figure 7.3.

7.33 These comparisons (6, 8, and 9) are shown in Figure 7.3.

7.2.2. Support Project Funding Values

In this example, the future worth obtained in tests of the concurrent model are dependent on the constraint limiting support projects values. This is evident, because in six out of nine comparisons[7.34] between the versions of the concurrent model that allow partial project selection, the unconstrained version selected support project values that were greater than one while the version that limited project values to one or less selected support project equal to one. In addition, the future worth provided by the unconstrained version was greater in these six comparisons than the future worth provided by the version that limited support project values to one or less.

In all versions of the concurrent model, the value of the support project is dependent upon product sub-group demand. When demand was low[7.35], the versions of the concurrent and sequential models that allow partial project selection yielded values of the support project that were less than one.

7.2.3. Production Project Values

The production projects selected for funding differ between the version of the concurrent model that restricts project values to zero and one, and the versions that allow partial project selection. However, versions that allow partial project selection always selected the same two production projects.

Values of the production projects selected for funding by all versions of the concurrent model are dependent upon product sub-group demand. In the version of the concurrent model that restricts projects to values of zero and one, production projects zero and one were funded in eight out of nine tests[7.36]. In most cases, as product sub-group demand increased, both versions of the concurrent model selected higher values of production projects. This was true for these versions of the sequential model also.

7.34 These comparisons (3, 5, 6, 7, 8, and 9) are shown in Figure 7.13.

7.35 Both versions of the concurrent model that allow partial project selection provided a value of the support project that was less than one in comparisons 1, 2, and 4. Product sub-group demand in these comparisons was (50, 50), (50, 100), and (100, 50) respectively. Both versions of the sequential model that allow partial project selection provided a value of the support project that was less than one in comparisons 1, 2, 4, 5, and 7. Product sub-group demand in these comparisons was (50, 50), (50, 100), (100, 50), (100, 100), and (150, 5) respectively. In comparison 3 (demand = 50, 150), the version of the sequential model that allows unconstrained partial project selection provided a value of the support project that was greater than one. In the same comparison, the version of the sequential model that allows partial project selection with an upper limit of one provided a value of the support project that was less than one.

7.36 These were tests 20 through 27.

7.2.4. Pre-Production/Post-Support Project Values

The pre-production/post-support project value supplied by all versions of the concurrent and sequential models may not be dependent upon the level of product sub-group demand. This is evident because tests of all versions of the concurrent and sequential models that allow partial project selection resulted in a value of 1/3 for the pre-production/post-support project. This relationship is shown in Figure 7.4. Further, tests of the versions of the concurrent and sequential models that restrict project values to zero and one resulted in a value of one for the pre-production/post-support project. This relationship is shown in Figure 7.4.

7.2.5. Products Funded

Every test of the concurrent and sequential model resulted in products zero, two, three, and four being funded. In all tests of the concurrent and sequential models, product zero was abandoned at the inception of fiscal period four[7.37], and production of product two began at that time. Activities to design, develop, and manufacture product two were funded in fiscal periods zero through three, and continued in diminished amounts through the end of product life.

In all tests of the concurrent model, product three was abandoned at the inception of fiscal period three, and production of product four began at that time. In all tests of the sequential model, product three was abandoned before or at the inception of fiscal period three, and production of product four began at that time. Activities to design, develop, and manufacture product four were funded by both models in fiscal periods zero through two, and continued in diminished amounts through the end of product life.

7.3. Relationships

This section contains a discussion of the cause and effect relationships that exist between concurrent and sequential model inputs, and their outputs (i.e., the objective function and decision variable values). Some reasons for the differences in the results obtained from the concurrent and sequential models are identified. Further, some reasons for the differences in the results obtained from the three versions of the concurrent model are identified.

[7.37] Although in tests 6, 9, 15, 18, and 24 the sequential model retained some level of inventory at the end of period 2, and did not produce product zero in period 3.

7.3.1. Comparison of Future Worth Obtained from Concurrent and Sequential Models

As shown in Table 7.1, for every combination of product sub-group demand and capital budgeting constraint type, the concurrent model always provided a higher future worth than the sequential model. It is likely that the concurrent model provided the highest future worth for two main reasons. First, the sequential model estimates the cost of additional capacity (i.e., the marginal cost of capital budgeting projects) based upon the cost of the present capacity. Thus, the production quantities selected by the first stage of the sequential model may not be optimal when specific capital budgeting projects are considered. Second, the objective of the third stage in the sequential model is to minimize the future worth of production costs, and production and support capital budgeting project costs, rather than to maximize the future worth of **all** revenues less costs. Thus, it is likely that the minimum cost solution, even though it is based upon the future worth criterion, will not yield the maximum revenue less cost solution.

7.3.2. Comparison of Future Worth Obtained from Three Versions of the Concurrent Model

Table 7.2 and Figure 7.2 showed the difference in the future worth obtained from the three versions of the concurrent model when nine combinations of demand were considered. In every test, the two versions of the concurrent model that allow partial project selection provided a higher future worth than the version that restricts projects to values of zero and one. For low demand levels (e.g., combination one where product sub-group demand equals (50, 50)), the two versions of the concurrent model that allow partial project selection usually provided the same future worth. This future worth was higher than the future worth provided by the version of the concurrent model that restricts projects to values of zero and one. Further, for high demand levels (e.g., combination nine where product sub-group demand equal (150, 150)), the version of the concurrent model that allows partial unbounded project selection usually provided the highest future worth. In comparison nine (i.e., product sub-group demand combination nine), the version of the concurrent model that allows partial project selection with a limit of one provided a future worth that was greater than the future worth provided by the 0,1 version of the concurrent model.

From these comparisons of the results obtained from the three versions of the concurrent model, it is clear that the type of project constraint (i.e., partial unbounded, partial bounded at one, and 0,1) may have a great impact on the future worth provided. In

the case of comparison one, both versions that allow partial project selection funded the capital budgeting projects at identical levels[7.38]. In all cases, project values were less than one. In the same comparison, the version of the concurrent model that restricts projects to values of zero and one funded some projects (i.e., assigned the project a value of one) and did not fund others (i.e., assigned the project a value of zero). In this comparison, the future worth obtained from the versions that allow partial project selection was higher than was obtained from the 0,1 version because the versions that allow partial project selection selected a smaller percentage of the capital budgeting projects. Thus, the versions of the concurrent model that allow partial project selection have a lower future worth of costs due to capital budgeting projects.

In comparison nine, the version of the concurrent model that allows unconstrained partial project selection provided the highest future worth. In this comparison, this version outperformed the version that allows partial project selection with an upper limit of one because projects could be assigned values that were greater than one. In fact, the version that allows unconstrained partial project selection assigned the support project a value of 1.235136 and production project one a value of 1.299854. In this same comparison, the version that allows partial project selection with an upper limit of one assigned these two projects values of one. Thus, the upper limit of one had the effect of reducing the production quantity and, thus, also reducing the future worth obtained from the version of the concurrent model that allows partial project selection with a limit of one.

As noted previously, in test nine the future worth obtained from the version of the concurrent model that allows partial project selection with an upper limit of one was higher than the future worth obtained from the 0,1 version. There are two main causes for this. First, this partial project selection version of the concurrent model assigned the pre-production/post-support project a value of 0.33333. In contrast, the 0,1 version assigned the same project a value of one. Because of this, the future worth of the pre-production/post-support project cost is higher in the 0,1 version of the concurrent model. Second, while the version of the concurrent model that allows partial project selection with a limit of one assigned production project one a value of 1.0, the version also assigned production project two a value of 0.591317. In contrast, the version of the

---

[7.38] The future worth obtained from these versions was identical because the capital budgeting project values are identical.

concurrent model that restricts project values to zero and one assigned production projects zero and one values of one. Because of this, the future worth of production capital budgeting costs is higher in the 0,1 version of the concurrent model. Thus, the 0,1 version of the concurrent model incurred more capital budgeting costs than were incurred by either version of the concurrent model that allow partial project selection.

7.3.3. Comparison of Future Worth Obtained from Three Versions of the Sequential Model

Table 7.3 and Figure 7.3 showed the difference in the future worth obtained from the three versions of the sequential model when nine combinations of demand were considered. In every test, the two versions of the sequential model that allow partial project selection provided a higher future worth than the version that restricts projects to values of zero and one. For low demand levels (e.g., combination one where product sub-group demand equals (50, 50)), the two versions of the sequential model that allow partial project selection usually provided the same future worth. This future worth was higher than the future worth provided by the version of the sequential model that restricts projects to values of zero and one. Further, for high demand levels (e.g., combination nine where product sub-group demand equal (150, 150)), the version of the sequential model that allows partial unbounded project selection provided the highest future worth in three tests[7.39]. In comparison nine, the version of the sequential model that allows partial project selection with a limit of one provided a future worth that was greater than the future worth provided by the 0,1 version of the sequential model.

From these comparisons of the results obtained from the three versions of the sequential model, it is clear that the type of project constraint (i.e., partial unbounded, partial bounded at one, and 0,1) may have a great impact on the future worth provided. In the case of comparison one, both versions that allow partial project selection funded the capital budgeting projects at identical values[7.40]. Both versions of the sequential model that allow partial project selection assigned the pre-production/post-support project a

---

[7.39] In three of nine comparisons, the version of the sequential model that allows partial project selection with an upper limit of one provided a higher future worth than was provided by the version of the sequential model that allows unconstrained partial project selection. In three comparisons, these two versions provided the identical future worth. In the remaining three comparisons, the version of the sequential model that allows unconstrained partial project selection provided the highest future worth.

[7.40] The future worth obtained from these versions was identical because the capital budgeting project values are identical.

value of 0.33333 and production project 1 a value of 0.122751. All other projects were assigned a value of zero. In the same comparison, the version of the sequential model that restricts projects to values of zero and one assigned the pre-production/post-support project a value of one, and production project one a value of one. All other projects were assigned a value of zero.

In the preceding comparison, the future worth obtained from the two versions that allow partial project selection was higher than was obtained from the 0,1 version because the versions that allow partial project selection selected a smaller percentage of the capital budgeting projects. Thus, these versions have a lower future worth of costs because less costs are incurred due to the funding of capital budgeting projects.

In comparison nine, the version of the sequential model that allows unconstrained partial project selection provided the highest future worth, and outperformed the version that allows partial project selection with an upper limit of one because projects could be assigned values that were higher than one. In fact, the version that allows unconstrained partial project selection assigned the support project a value of 1.25 and production projects one and two values of 1.506723 and 1.869946 respectively. In this same comparison, the version that allows partial project selection with an upper limit of one assigned the support project a value of one, production project one a value of one, and production project two a value of 0.999985. Thus, the upper limit of one had the effect of reducing production quantity and, thus, also reducing the future worth that could be obtained from this version of the sequential model.

As noted previously, in comparison nine the future worth obtained from the version of the sequential model that allows partial project selection with an upper limit of one was higher than the future worth obtained from the 0,1 version of the sequential model. There are two main causes for this. First, the partial project selection version of the sequential model assigned the pre-production/post-support project a value of 0.33333. In contrast, the 0,1 version assigned the pre-production/post-support project a value of one. Because of this, the future worth of the pre-production/post-support project cost is higher in the 0,1 version of the sequential model. Second, while the version of the sequential model that allows partial project selection with a limit of one assigned production project one a value of 1.0, the version also assigned production project two a value of 0.999985. In contrast, the version of the sequential model that restricts project values to zero and one assigned production projects zero and one values of one. Because of this, the future worth of production capital budgeting costs is higher in the 0,1 version

of the sequential model. Thus, the 0,1 version of the sequential model incurred more capital budgeting costs than were incurred by either version of the sequential model that allow partial project selection.

7.3.4. Comparison of Support Project Values

Table 7.6 and Figure 7.12 showed the difference in the values of the support project provided by all versions of the concurrent and sequential models. In general, both models provided higher values of the support project as the product sub-group demand increased. During three tests of the version of the concurrent model that allows unconstrained partial project selection, the support project was assigned a value of $e_{0_v} = 1.25$. These were tests three, five, and six. In addition, during test nine of the version of the sequential model that allows unconstrained partial project selection, the support project was assigned a value of 1.25. When the support project value reached 1.25, the maximum cost due to support projects constraint was binding[7.41]. This is because the support capital budgeting project had a cost of $c_{m_v t} = \$80,000$ in each period, t. Further, the maximum that could be spent on support capital budgeting projects in any period, t, was $E_{t_v} = \$100,000$.

The equation that describes this constraint is given by:

$$c_{m_v t} e_{m_v} \leq E_{t_v} \quad \forall\ t \in T; e_{m_v} \in 0,1$$

Substituting the values of $c_{m_v t}$, E_{t_v}, and e_{0_v} yields the following relationship:

$$\frac{\$100,000}{\$80,000} = 1.25$$

Because the value of the support projects cannot be increased in these tests, it follows that the future worth provided by the versions of the concurrent and sequential models that allow unconstrained partial project selection may be affected by the maximum amount that can be spent on support projects in any period, t. Thus, the inability to provide support capacity in excess of $e_{0_v} = 1.25$ has resulted in a lower future worth than could be obtained if no support cost constraint existed.

[7.41] This constraint is given by Equation (4.47).

The effect the aforementioned constraint has on the future worth that can be provided by the version of the concurrent model that allows unconstrained partial project selection was tested by first removing the constraint, and then re-running the model. Tests 3, 5, and 6 were re-run in this fashion. The results of the tests are shown in Table 7.8 below.

As shown in Table 7.8, in all tests, when the support project expense constraint is removed, the support project value increases. Further, the increase in the support project value also allows for an increase in production quantities. Because of this increase in production quantities, the future worth provided in each test also increases. This leads to the conclusion that, for this example, when the demand is high, removal of this constraint may cause an increase in the future worth.

Table 7.8. Comparison of Future Worth Obtained from Re-Tests of the Concurrent Model Where the Support Value was 1.25.

			Future Worth		Project Values				
No.	One	Two	Future Worth Obtained	Improvement in Future Worth	e_{0_n}	e_{0_l}	e_{1_l}	e_{2_l}	e_{0_v}
3	50	150	$91,142,592		0.33333	0	0.846553	0.659405	1.25
3	50	150	92,058,544	$915,952	0.33333	0	0.887197	0.699942	1.312703
5	100	100	91,309,376		0.33333	0	1.084948	0.602345	1.25
5	100	100	91,549,984	240,608	0.33333	0	1.092832	0.607402	1.267979
6	100	150	99,539,656		0.33333	0	1.091546	0.607981	1.25
6	100	150	99,774,288	234,632	0.33333	0	1.087539	0.609784	1.266585

It should be further noted that inclusion of this type of maximum budget constraint for any of the three project types (i.e., pre-production/post-support, production, and support) may result in a limit on production quantities and, thus, the future worth that could have been obtained had the constraint not existed. Further, this limit on future worth and production quantities may not be dependent upon the product sub-group demands. This is because the constraint depends upon the funding level of the project, and the resulting cost. Thus, production quantities are bounded indirectly.

7.3.5. Comparison of Pre-Production/Post-Support Values

Table 7.4 and Figure 7.4 showed the difference in the values of the pre-production/post-support project provided by all versions of the concurrent and sequential models. Note that both versions of the concurrent and sequential models that allow partial project selection always assigned the pre-production/post-support project a value of 0.33333. In contrast, the 0,1 version of the concurrent and sequential models always assigned the pre-production/post-support project a value of one. This leads to three conclusions. First, the optimal quantity of the pre-production/post-support project is 0.33333. A value greater than this will result in the acquisition of excess pre-production/post-support capacity[7.42]. Second, since the 0,1 version of both models assigned the pre-production/post-support project a value of one, it follows that a higher future worth was obtained from both models when excess pre-production/post-support capacity is acquired. Third, the additional pre-production/post-support capacity provided by the pre-production/post-support capital budgeting project is necessary to fund products 2 and 4[7.43]. If this additional capacity were not provided, then these products could not be produced. This would result in a significant decrease in future worth.

7.3.6. 0,1 Version of the Concurrent Model Versus Partial Versions Adjusted to 0,1 Values

In the following section, two decision rules are used to compare three results obtained from the 0,1 version of the concurrent model to the corresponding three results obtained from the two versions of the concurrent model that allow partial project selection.

In the following comparisons, it is assumed that the decision maker may only accept a capital budgeting project (i.e., e = 1) or reject a project (i.e., e = 0). Thus, the results obtained from the versions of the concurrent model that allow partial project selection were converted to 0,1 values.

Tests of the versions of the concurrent model that allow partial project selection were run again using LINDO/386, but with constraints specifying values of zero and one for the capital budgeting projects. Comparisons of these adjusted results were conducted on tests 1 and 10, 5 and 14, and 9 and 18. These results were compared to the future

---

[7.42] Note that this follows the definition of excess capacity given in Chapter 2.

[7.43] See the discussion of decision rule two that follows in Section 7.3.6.2.

worth obtained from tests 19, 23, and 27 respectively of the 0,1 version of the concurrent model.

7.3.6.1. Decision Rule One

The first decision rule states that: (1) if a capital budgeting project was assigned a value greater than zero (i.e., $e_a > 0$), then the project is set equal to one (i.e., $e_b = 1$), and (2) if a project was assigned a value of zero (i.e., $e_a = 0$), then the project retains that value (i.e., $e_b = 0$). This decision rule can be stated mathematically as: $\begin{cases} e_b = 0 \text{ if } e_a = 0 \\ e_b = 1 \text{ if } e_a > 0 \end{cases}$.

Table 7.9 shows the adjusted values for the capital budgeting projects using decision rule one. In addition, the table shows the future worth obtained from the 0,1 version of the concurrent model along with the future worth obtained when decision rule one is applied to the two versions of the concurrent model that allow partial project selection. Finally, the table shows the difference in future worth that results when project values obtained from versions of the concurrent model that allow partial project selection are converted to 0,1 values using decision rule one.

Table 7.9. Comparison of Future Worth Obtained from Tests of the Concurrent Model Using Decision Rule One.

				Future Worth		Project Values				
Test No.	Sub-Group One	Sub-Group Two	Constraint Type	Future Worth Obtained	Difference From 0,1 Version	e_{0_n}	e_{0_l}	e_{1_l}	e_{2_l}	e_{0_v}
1	50	50	$0 \le P$	$25,815,182	$3,186,662	1	0	1	0	0
10	50	50	$0 \le P \le 1$	25,815,182	3,186,662	1	0	1	0	0
19	50	50	$P \in 0,1$	29,001,844		1	0	0	0	0
5	100	100	$0 \le P$	83,303,192	870,168	1	0	1	1	1
14	100	100	$0 \le P \le 1$	83,303,192	870,168	1	0	1	1	1
23	100	100	$P \in 0,1$	84,173,360		1	1	1	0	1
9	150	150	$0 \le P$	91,081,770	551,254	1	0	1	1	1
18	150	150	$0 \le P \le 1$	91,081,770	551,254	1	0	1	1	1
27	150	150	$P \in 0,1$	91,633,024		1	1	1	0	1

In tests 1 and 10, since production project one was set equal to 0.003259, this project value was rounded to one. Since the remaining two production projects were set equal to zero by the two versions of the concurrent model that allow partial project selection, the project retained a value of zero. In addition, both versions of the concurrent model set the support project value equal to zero. This value was retained in the subsequent analysis. Finally, the value of the pre-production/post support project provided by the two versions of the concurrent model was set equal to 0.33333. Following the aforementioned decision rule, this value was rounded to one.

In test 5, since production project one was set equal to 1.084948, this project value was rounded to one. Further, in test 14, since production project one was set equal to 0.941208, this project value was rounded to one. In test 5, since production project one was set equal to 0.602345, this project value was rounded to one. In test 14, since production project two was set equal to 0.484802, this project value was assigned a value of one. In addition, in test 5, the concurrent model set the support project value equal to 1.25. Thus, the support project was assigned a value of one. Further, in test 14, the concurrent model set the support project equal to one. Thus, this value was retained in the subsequent analysis. Finally, the value of the pre-production/post-support project provided by the two versions of the concurrent model that allow partial project selection was set equal to 0.33333. Following the aforementioned decision rule, this value was rounded to one.

In test 9, since production project one was set equal to 1.299854, this project value was rounded to one. Further, in test 18, since production project one was set equal to 1.0, this project value was retained. In test 9, since production project one was set equal to 0.654854, this project value was rounded to one. In test 18, since production project two was set equal to 0.591317, this project value was assigned a value of one. In addition, in test 9, the concurrent model set the support project value equal to 1.235136. Thus, the support project was assigned a value of one. Further, in test 18, the concurrent model set the support project equal to one. Thus, this value was retained in the subsequent analysis. Finally, the value of the pre-production/post support project provided by the two versions of the concurrent model that allow partial project selection was set equal to 0.33333. Following the aforementioned decision rule, the value of the support project for both versions of the concurrent model was rounded to one.

Following optimization using the adjusted values, it was found that the future worth provided using decision rule one is dependent upon demand. That is, as the demand

increased, so too did the future worth obtained. In addition, as the demand increased, the difference between the future worth obtained from the 0,1 version of the concurrent model and the future worth obtained from the adjusted values of the partial models decreased. Thus, as the demand increased, the difference in future worth obtained from the 0,1 adjustment of the partial versions of the concurrent model using decision rule one decreased resulting in the improved performance of decision rule one.

There seem to be two main reasons for these outcomes. First, using decision rule one, each version of the concurrent model that allows partial project selection was required to fully accept the pre-production/post-support project (i.e., $e_{0_n} = 1$). Recall that both versions of the concurrent model that allow partial project selection always assigned the pre-production/post-support project a value of 0.33333. Thus, since a greater percentage of the pre-production/post-support project must be accepted, the cost of the project will also increase. Therefore, the future worth obtained from these versions using the adjusted values will be less than that obtained when partial project selection is allowed. In the second case, the decreasing difference in future worth most likely stems from an increase in production volume compensating for the increased cost of full acceptance of the pre-production/post-support project. Note that as the production volume increases, the total revenues also increase. Thus, the difference between the future worth obtained from the 0,1 version of the concurrent model and the future worth obtained from the adjusted partial versions should also decrease. This relationship is demonstrated in the previous comparisons.

7.3.6.2. Decision Rule Two

The second decision rule states that: (1) if a project was assigned a value greater than or equal 0.5 (i.e., $e_a \geq 0.5$), then the project is set equal to one (i.e., $e_b = 1$), and (2) if a project was assigned a value less than 0.5 (i.e., $e_a < 0.5$), then the project is set equal to zero (i.e., $e_b = 0$). This decision rule can be stated mathematically as:

$$\begin{cases} e_b = 0 \text{ if } e_a < 0.5 \\ e_b = 1 \text{ if } e_a \geq 0.5 \end{cases}.$$

Table 7.10 shows the adjusted values for the capital budgeting projects using decision rule two. In addition, the table shows the future worth obtained from the 0,1 version of the concurrent model along with the future worth obtained when decision rule two is applied to the two versions of the concurrent model that allow partial project selection. Finally, the table shows the difference in future worth that results when project

values obtained from versions of the concurrent model that allow partial project selection are converted to 0,1 values using decision rule two.

Table 7.10. Comparison of Future Worth Obtained from Tests of the Concurrent Model Using Decision Rule Two.

				Future Worth		Project Values				
Test No.	Sub-Group One	Sub-Group Two	Constraint Type	Future Worth Obtained	Difference From 0,1 Version	e_{0_n}	e_{0_l}	e_{1_l}	e_{2_l}	e_{0_v}
1	50	50	$0 \leq P$	$11,471,057	$17,530,787	0	0	0	0	0
10	50	50	$0 \leq P \leq 1$	11,471,057	17,530,787	0	0	0	0	0
19	50	50	$P \in 0,1$	29,001,844		1	0	0	0	0
5	100	100	$0 \leq P$	43,203,836	40,969,524	0	0	1	1	1
14	100	100	$0 \leq P \leq 1$	26,512,398	57,660,962	0	0	1	0	1
23	100	100	$P \in 0,1$	84,173,360		1	1	1	0	1
9	150	150	$0 \leq P$	55,365,500	36,267,524	0	0	1	1	1
18	150	150	$0 \leq P \leq 1$	55,365,500	36,267,524	0	0	1	1	1
27	150	150	$P \in 0,1$	91,633,024		1	1	1	0	1

In tests 1 and 10, since production project one was set equal to 0.003259, this project value was rounded down to zero. Since the remaining two production projects were set equal to zero by the two versions of the concurrent model that allow partial project selection, the project retained a value of zero. In addition, both versions of the concurrent model set the support project value equal to zero. This value was retained in the subsequent analysis. Finally, the value of the pre-production/post-support project provided by the two versions of the concurrent model was set equal to 0.33333. Following the aforementioned decision rule, this value was rounded down to zero.

In test 5, since production project one was set equal to 1.084948, this project value was rounded down to one. Further, in test 14, since production project one was set equal to 0.941208, this project value was rounded up to one. In test 5, since production project one was set equal to 0.602345, this project value was rounded up to one. In test 14, since production project two was set equal to 0.484802, this project value was

assigned a value of zero. In addition, in test 5, the concurrent model set the support project value equal to 1.25. Thus, the support project was assigned a value of one. Further, in test 14, the concurrent model set the support project equal to one. Thus, this value was retained in the subsequent analysis. Finally, the value of the pre-production/post support project provided by the two versions of the concurrent model that allow partial project selection was set equal to 0.33333. Following the aforementioned decision rule, this value was rounded down to zero.

In test 9, since production project one was set equal to 1.299854, this project value was rounded down to one. Further, in test 18, since production project one was set equal to 1.0, this project value was retained. In test 9, since production project one was set equal to 0.654854, this project value was rounded up to one. In test 18, since production project two was set equal to 0.591317, this project value was assigned a value of one. In addition, in test 9, the concurrent model set the support project value equal to 1.235136. Thus, the support project was assigned a value of one. Further, in test 18, the concurrent model set the support project equal to one. Thus, this value was retained in the subsequent analysis. Finally, the value of the pre-production/post support project provided by the two versions of the concurrent model that allow partial project selection was set equal to 0.33333. Following the aforementioned decision rule, the value of the support project for both versions of the concurrent model was rounded down to zero.

Following optimization using the adjusted values, it was found that the future worth provided using decision rule two is dependent upon demand. That is, as the demand increased, so too did the future worth obtained. As the demand increased, the difference between the future worth obtained from the 0,1 version of the concurrent model and the future worth obtained from the adjusted values of the partial models initially increased. However, the difference between the future worth obtained in the tests in the second comparison (i.e., where the product sub-group demand equals 100, 100) and the future worth obtained in tests in the third comparison (i.e., where the product sub-group demand equals 150, 150) decreased. Further, the differences in future worth obtained in the third comparison are greater than the differences in future worth obtained in the first comparison. Because of this, as the demand increased, the 0,1 adjustment of the partial versions of the concurrent model using decision rule two resulted in an increasing difference in future worth. Thus, as the demand increased, decision rule two's performance worsened.

There appears to be two main reasons for these outcomes. First, using decision rule two, each version of the concurrent model that allows partial project selection must reject the pre-production/post-support project (i.e., $e_{0_n} = 0$). Since the pre-production/post-support project was rejected, no pre-production/post-support project costs were incurred. However, because the pre-production/post-support project using decision rule two is not funded, products 2 and 4 could not be funded. Thus, the product set funded by these versions of the concurrent model using the adjusted values of the pre-production/post-support project is different from the product set funded when partial project selection is allowed and when 0,1 optimization is conducted. As in the results obtained from both partial versions, product 0 was produced through the end of period 3. However, since product 2 cannot be produced, production in periods 4, 5, 6, and 7 for product sub-group 0 cannot occur. Thus, under decision rule two, the future worth is decreased because revenues are lost in these periods. Further, since product 4 cannot be produced, product 3 is produced through the end of period 7. The future worth is decreased because the future worth of the revenues less costs for product 3 is less than the future worth of the revenues less costs for product 4. Therefore, the future worth obtained from the partial selection versions of the concurrent model using the adjusted values provided by decision rule two is less than the future worth obtained from the 0,1 version of the concurrent model.

In the second case, the increase in future worth most likely stems from an increase in production volume. Note that as the production volume increases, the total revenues also increase. However, the difference between the future worth obtained from the 0,1 version of the concurrent model and the future worth obtained from the adjusted partial versions also increases because products 2 and 4 are not funded. Finally, the acceptance of production project two in test 5 versus the rejection of the same production project in test 14 demonstrates the significant effect that production projects have on the future worth that can be obtained.

7.3.6.3. Comparisons of Results Obtained from Each Decision Rule

As shown in Table 7.11 and Figure 7.15, for these three demand combinations and the hypothetical example shown in Chapter 6, decision rule one always provided a higher future worth than was obtained from decision rule two. Further, decision rule one provided a future worth that is closer to the future worth obtained from the 0,1 version of the concurrent model than to the future worth obtained when decision rule two was

applied. That is, as shown in Figure 7.15, the difference between the future worth obtained from the 0,1 version of the concurrent model and decision rule one is less than the difference between decision rule one and decision rule two.

Table 7.11. Comparison of Future Worth Obtained from Tests of the Concurrent Model Using Decision Rules One and Two.

				Future Worth				
No	**Sub-Group One**	**Sub-Group Two**	**Constraint Type**	**Future Worth Obtained: Rule 1**	**Difference From 0,1 Version**	**Future Worth Obtained: Rule 2**	**Difference From 0,1 Version**	**Difference Between Two Decision Rules**
1	50	50	$0 \leq P$	$25,815,182	$3,186,662	$11,471,057	$17,530,787	$14,344,125
10	50	50	$0 \leq P \leq 1$	25,815,182	3,186,662	11,471,057	17,530,787	14,344,125
5	100	100	$0 \leq P$	83,303,192	870,168	43,203,836	40,969,524	40,099,356
14	100	100	$0 \leq P \leq 1$	83,303,192	870,168	26,512,398	57,660,962	57,150,794
9	150	150	$0 \leq P$	91,081,770	551,254	55,365,500	36,267,524	35,716,270
18	150	150	$0 \leq P \leq 1$	91,081,770	551,254	55,365,500	36,267,524	35,716,270

As was shown, the difference between the future worth obtained from the 0,1 version of the concurrent model and the future worth obtained when decision rule one is applied decreases as the demand increases. Thus, the future worth generated by decision rule one may improve as the demand increases. This was not the case when decision rule two was applied to the same combinations of demand.

It is likely that the main reason for the difference in future worth obtained from these decision rules stems from the pre-production/post-support project funding decision. As was noted, the pre-production/post-support project was funded under decision rule one and was not funded under decision rule two. Because decision rule two specified that the pre-production/post-support project not be funded, products 2 and 4 were not produced. This failure to produce these two new products resulted in a significant decrease of the future worth obtained. Thus, since decision rule one enabled funding of the pre-production/post-support project, two new products were produced. These

products provided a significant increase in the future worth for the three demand sets studied using the hypothetical example.

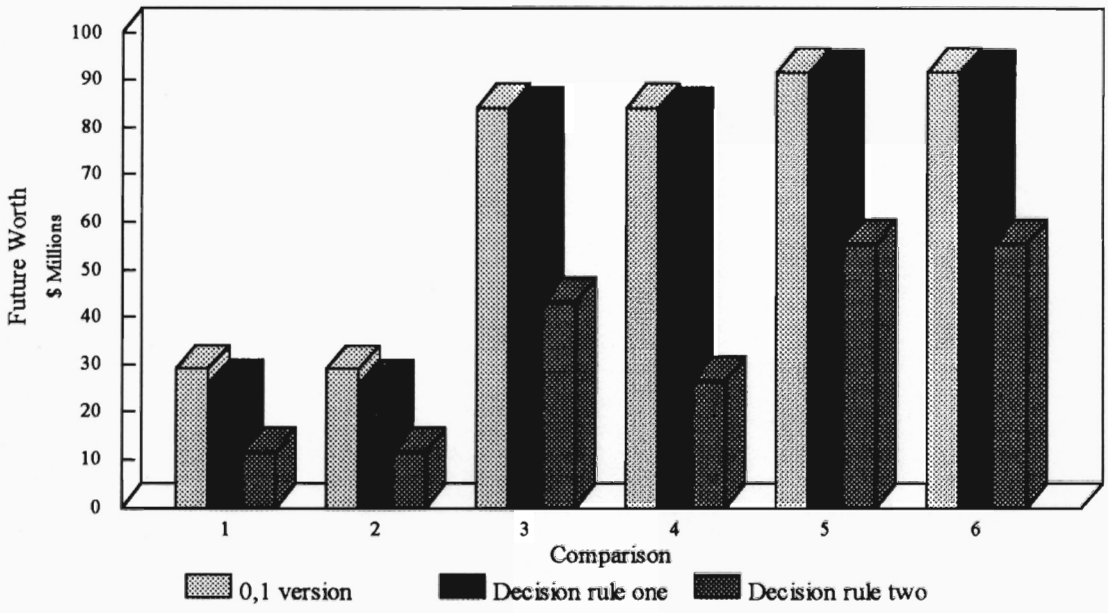

Figure 7.15. Comparison of Future Worth Obtained from the 0,1 Version of the Concurrent Model and the Partial Versions Adjusted to 0,1 Values.

7.3.7. Support Project Availability

In the following section, the results obtained from the sequential model when the support project is available to be funded and when the support project is not available are compared. This comparison is important because the first stage of the sequential model assumes that unlimited additional support capacity is available at a marginal cost equal to the marginal cost of the existing support capacity[7.44]. Thus, a question arises: How well

---

7.44 As noted in Chapter 5, the first stage of the sequential model estimates the cost of support capacity based upon the cost per period of existing support capacity. The marginal cost of base support capacity is determined by dividing the total cost of support capacity per period by the number of hours of base support capacity that exist. Thus, the marginal cost is based upon capacity support cost per hour per period. The second and third stages of the sequential model use the actual support projects that can be funded. Thus, it is possible that stage one may specify that additional support capacity be acquired, while stages two and three may not consider any support projects.

will the sequential model perform with respect to future worth when no support projects exist?

To facilitate the comparison, the three versions of the sequential model were run a second time. During these additional runs, the support project was set equal to zero. A run for each of the nine combinations of demand for each of three versions of the sequential model was conducted.

Table 7.12 and Figure 7.16 show the results of this comparison. In every case, the future worth obtained when the support project was available was greater than when the support project was not available. Further, the results indicate that, as the demand increased, so too did the difference between the future worth obtained[7.45]. This leads to two main conclusions. First, the availability of additional support capacity may have a significant impact on the production quantities that can be obtained. Second, as the demand increases, the availability of additional support capacity may have a more pronounced effect on future worth than when the demand is relatively low.

It is expected that the increasing loss in future worth as the demand increases is due to the fact that more nearly adequate support capacity existed for low demand levels. That is, when the demand was low, so too would be the production quantities. Because the production quantities were lower, less additional support capacity was required.

Table 7.6 shows that the sequential model specified additional support capacity of 0.697344 in tests 2 and 11 where product sub-group demand was (50, 100), and 0.681927 in tests 4 and 13 where product sub-group demand was (100, 50). The difference in future worth obtained in these tests was $22,374,712 and $20,156,112 respectively. In contrast, the sequential model specified additional support capacity of 1.181151 in test 8 where product sub-group demand was (150, 100), and 1.25 in test 9 where product sub-group demand was (150, 150). The difference in future worth obtained in these tests was $49,005,408 and $48,978,896 respectively.

[7.45] The difference in future worth ranged from $0 in tests 1, 10, and 19 where product sub-group demand was (50, 50) to $49,005,408 in test 8 where product sub-group demand was (150, 100).

Table 7.12. Paired Comparisons of Future Worth Obtained from Tests of the Sequential Model With and Without Support Project Availability.

Test No.	Demand for Products in Sub-Group One	Demand for Products in Sub-Group Two	Project (P) Constraint Type	Future Worth		
				Sequential Model: No Support Project Available	Sequential Model: Support Project Available	Difference
1	50	50	$0 \leq P$	$29,799,880	$29,799,880	$0
10	50	50	$0 \leq P \leq 1$	29,799,880	29,799,880	0
19	50	50	$P \in 0,1$	24,836,808	24,836,808	0
2	50	100	$0 \leq P$	40,751,600	63,126,312	22,374,712
11	50	100	$0 \leq P \leq 1$	40,751,600	63,126,312	22,374,712
20	50	100	$P \in 0,1$	36,564,560	58,473,264	21,908,704
3	50	150	$0 \leq P$	34,632,960	78,492,048	43,859,088
12	50	150	$0 \leq P \leq 1$	42,005,816	83,134,976	41,129,160
21	50	150	$P \in 0,1$	38,354,480	80,972,016	42,617,536
4	100	50	$0 \leq P$	43,411,952	63,568,064	20,156,112
13	100	50	$0 \leq P \leq 1$	43,411,952	63,568,064	20,156,112
22	100	50	$P \in 0,1$	39,152,280	58,864,408	19,712,128
5	100	100	$0 \leq P$	36,176,808	76,216,000	40,039,192
14	100	100	$0 \leq P \leq 1$	41,878,704	78,453,328	36,574,624
23	100	100	$P \in 0,1$	39,740,176	76,185,312	36,445,136
6	100	150	$0 \leq P$	38,130,872	82,053,184	43,922,312
15	100	150	$0 \leq P \leq 1$	45,413,016	81,567,776	36,154,760
24	100	150	$P \in 0,1$	41,422,520	79,429,264	38,006,744
7	150	50	$0 \leq P$	36,556,160	73,517,408	36,961,248
16	150	50	$0 \leq P \leq 1$	45,076,856	75,420,816	30,343,960
25	150	50	$P \in 0,1$	42,294,920	72,134,320	29,839,400
8	150	100	$0 \leq P$	36,882,208	85,887,616	49,005,408
17	150	100	$0 \leq P \leq 1$	45,692,776	81,966,608	36,273,832
26	150	100	$P \in 0,1$	42,027,880	79,828,064	37,800,184
9	150	150	$0 \leq P$	38,240,896	87,219,792	48,978,896
18	150	150	$0 \leq P \leq 1$	44,481,872	82,243,952	37,762,080
27	150	150	$P \in 0,1$	42,343,388	80,105,440	37,762,052

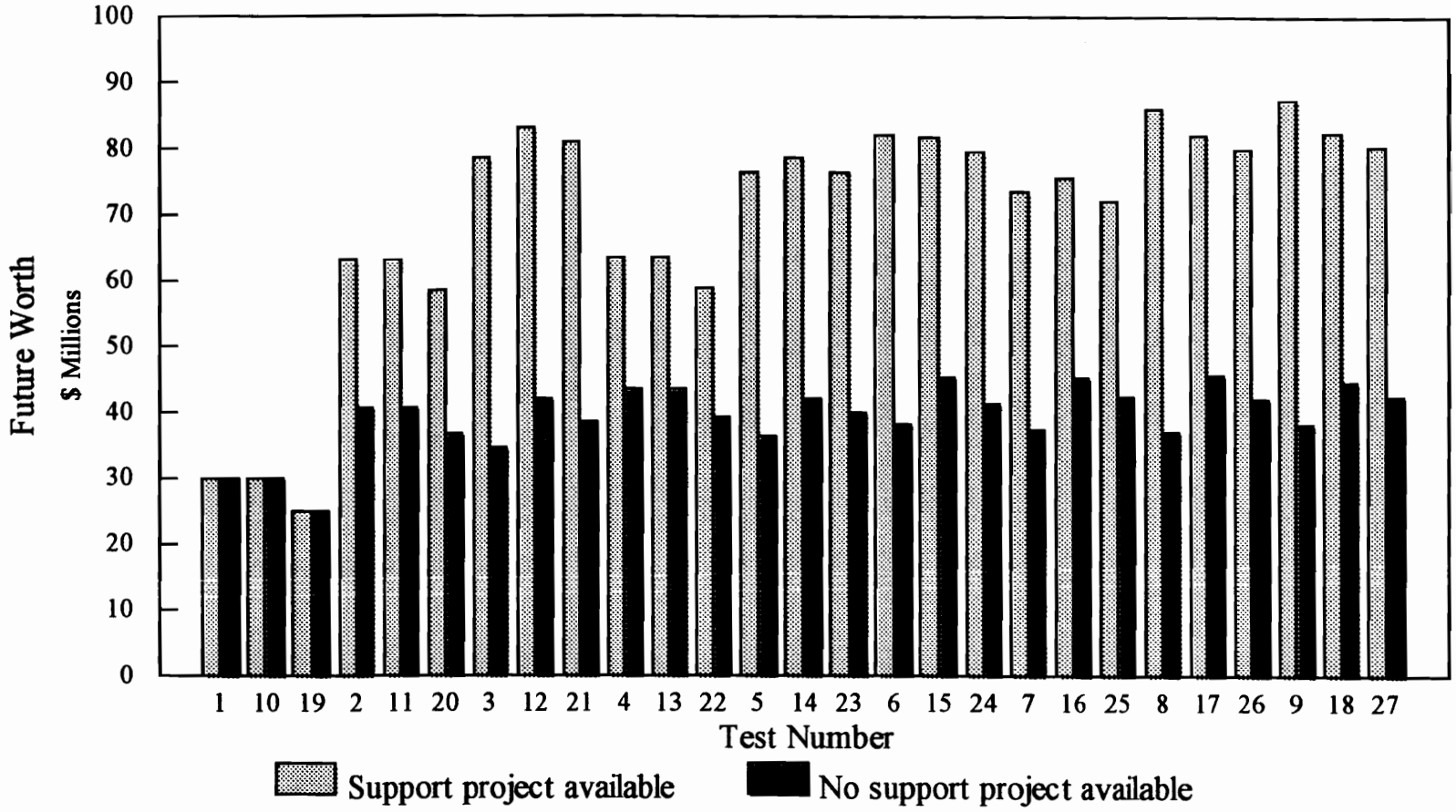

Figure 7.16. Chart Depicting the Future Worth Obtained from Tests of All Versions of the Sequential Model When the Availability of the Support Project was Varied.

8. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

This chapter is divided into three main sections. The first section contains a summary of the research presented herein. The second section contains conclusions drawn from the tests, and subsequent analyses of the results obtained from three versions of the concurrent and sequential models. Finally, the third section includes recommendations for further research.

8.1. Summary

This research was conducted to determine if the concurrent solution of product, production, and capacity planning problems will yield a higher future worth of all revenues less costs than if they were solved sequentially. To demonstrate that the simultaneous solution can be better than the sequential solution, three versions of the life-cycle complete concurrent model (simultaneous solution), and three versions of the life-cycle complete sequential model (consecutive solutions) were developed, and are presented and compared herein.

All versions of the concurrent and sequential models consider three types of work centers. Work centers that affect the conceptual design, detailed design, development, and phase-out activities of a product are called pre-production/post-support work centers. The concurrent and sequential models consider multiple pre-production/post-support work centers. Work centers that affect the production activities of a product are called production work centers. The concurrent and sequential models consider multiple production work centers. The work center that affects the support of a product is called the support work center. Only one support work center is considered by the concurrent and sequential models.

The models developed during this research consider three types of capital budgeting projects. These three types of capital budgeting projects correspond to the three types of work centers. Thus, capital budgeting projects are: (1) pre-production/post-support, (2) production, and (3) support. Each type of project can add capacity to only one type of work center. However, each pre-production/post-support project can add capacity to more than one pre-production/post-support work center. In addition, each production project can add capacity to more than one production work center.

During the research, it was shown that pre-production/post-support, production, and support capital budgeting projects may be constrained by three types of constraints. These are: (1) projects that may be partially accepted, and can be any value greater than zero, (2) projects that may be partially accepted, but cannot be greater than one, and (3) projects that are restricted to values of zero and one. Because there exist three types of project constraints, three versions of the concurrent and sequential models were constructed and compared. Version one of each model allows unconstrained partial project selection. Version two of each model allows partial project selection but requires that capital budgeting project values be less than or equal to one. Version three of each model restricts capital budgeting values to zero and one.

After the concurrent and sequential models were developed, a hypothetical example spanning twelve fiscal periods that included five products, two product sub-groups, one pre-production/post-support project, three production projects, and one support project was developed to enable comparison of the models. This hypothetical example included 873 inputs. Two of the production projects are mutually exclusive. Further, two products are mutually exclusive.

Tests of the models required the development of five computer programs written in C using Borland's C/C++ compiler. The output from four of these programs was interfaced with LINDO/386, a commercially available linear programming software package. The fifth program read output from LINDO/386, and provided the future worth obtained from each test of the sequential model.

The concurrent model required the development of one Borland C/C++ program. This program read the inputs contained in the hypothetical example, and created a MPS formatted file that was read by LINDO/386. LINDO/386 was then used to find the optimal solution.

The sequential model required the development of four Borland C/C++ programs. The first program partially automated stage one of the sequential model. This program read the inputs contained in the hypothetical example, and created an MPS formatted file that was read by LINDO/386. After creation of the MPS file, LINDO/386 was used to find the optimal stage one solution. The second program partially automated stage two of the sequential model. This program read the inputs contained in the hypothetical example, and the production quantities determined in the first stage of the sequential model. The program created an MPS formatted file that was read by LINDO/386. After creation of the MPS file, LINDO/386 was used to find the optimal stage two solution. The third

program partially automated stage three of the sequential model. This program read the inputs contained in the hypothetical example, and the production quantities determined in the second stage of the sequential model. The program created an MPS formatted file that was read by LINDO/386. After creation of the MPS file, LINDO/386 was used to find the optimal third stage solution. The fourth program read the inputs contained in the hypothetical example, and the value of the pre-production/post-support project determined in stage one, and all other decision variable values determined in stage three. This program determined the future worth provided by the sequential model.

Nine comparisons between each of the three versions of the concurrent and sequential models were made. Each comparison consisted of two levels of product sub-group demand. A total of three product sub-group demands (i.e., low = 50, medium = 100, and high = 150) were considered for each product sub-group. Further, all combinations of the three levels of demand for the two product sub-groups were considered. Hence, $3^2 = 9$ combinations of product sub-group demand were considered. Because three versions of each model were developed, a total of $3^2x3 = 27$ comparisons was made.

Results obtained from tests of the concurrent and sequential models were compared based upon future worth, capital budgeting projects funded, products funded, production and inventory quantities, and regular and overtime labor levels. It was found that, for every combination of product sub-group demand and capital budgeting constraint type, the concurrent model always provided a higher future worth than the sequential model. It is likely that the concurrent model provided the highest future worth for two main reasons. First, the sequential model estimates the cost of additional capacity based upon the cost of the present capacity. Thus, the production quantities selected by the first stage of the sequential model may not be optimal when specific capital budgeting projects are considered. Second, the objective of the third stage in the sequential model is to minimize the future worth of production, and production and support capital budgeting project costs rather than to maximize the future worth of revenues less costs. Thus, it is likely that the minimum cost solution obtained from the sequential model will not be as good as the maximum revenues less costs solution obtained from the concurrent model.

During comparisons of test results, it was found that the future worth obtained from all versions of the concurrent model is dependent upon the product sub-group demand. In addition, the version of the concurrent model that allows unconstrained partial project selection provided the highest future worth in six out of nine comparisons between

the three versions of the concurrent model. In the remaining three comparisons, this version provided the same future worth as the version of the concurrent model that allows partial project selection with an upper bound of one on project values. Finally, all versions of the concurrent model that allow partial project selection provided a higher future worth than the version that restricted capital budgeting projects to values of zero and one.

Following these comparisons, it was concluded that, for any type of capital budgeting constraint type and product sub-group demand, a concurrent model may provide better planning results than could be obtained from a sequential model. Further, when it is feasible to allow unconstrained partial capital budgeting project selection, the best results may be obtained from a version of the concurrent model that allows partial unconstrained project selection. When partial project selection is allowed, but projects may be no greater than one, the best planning results may be obtained from the version of the concurrent model that allows partial project selection that bounds projects by one. Finally, when it is not feasible to select partial capital budgeting projects, the version of the concurrent model that restricts capital budgeting projects to values of zero and one may provide the best results. Thus, a concurrent model may provide the best results when any type of capital budgeting constraint is encountered.

8.2. Conclusions

This research has resulted in nine conclusions. These are:

(1) The concurrent model provided the highest future worth of all revenues less costs in every comparison of the three versions of the concurrent and sequential models. Thus, it is concluded that a concurrent product, production, and capacity planning model may provide better results for any type of capital budgeting constraint type and product sub-group demand, than could be obtained from a similar sequential model.

(2) The version of the concurrent model that allows unconstrained partial capital budgeting selection provided the highest future worth in six out of nine comparisons between the three versions of the concurrent model. In the remaining three comparisons, the aforementioned version provided the same future worth as the version of the concurrent model that allows partial project selection with an upper bound of one on project values. Thus, it is concluded that, when unconstrained partial project selection is feasible, the best results may be obtained from a concurrent model that allows unconstrained partial project selection. Further, since the version of the concurrent model that allows unconstrained partial project selection always provided a higher future worth

than the similar version of the sequential model, it is concluded that the concurrent solution may be better than the sequential solution for these type of capital budgeting projects.

(3) In every comparison, the version of the concurrent model that allows partial capital budgeting project selection with an upper bound of one on projects provided a higher future worth than the version of the concurrent model that restricts capital budgeting projects to values of zero and one. Thus, it is concluded that, when partial project selection is allowed, and the values of projects can be no greater than one, the best results may be obtained from the concurrent model that allows partial project selection with an upper bound of one on projects. Further, since this version of the concurrent model always provided a higher future worth than the similar version of the sequential model, it is concluded that the concurrent solution may be better than the sequential solution for these type of capital budgeting projects.

(4) When partial project selection is not feasible, the best results may be obtained from a concurrent model that restricts project values to zero and one.

(5) In every test, the versions of the concurrent model that allow partial project selection provided a higher future worth than the version of the concurrent model that restricts project values to zero and one. In addition, of the five capital budgeting projects considered, no versions of the concurrent model that allow partial project selection ever assigned all five projects all values of zero or one. Because of this, it is concluded that the use of any model that allows partial capital budgeting project values when partial project selection is not feasible, may result in a higher future worth than can actually be obtained.

(6) In 15 out of 16 tests, the versions of the concurrent model that allow partial project selection selected one production project that was funded by the corresponding test of the 0,1 version, and also one project that was not. Because of this, it is concluded that when a situation exists where partial project selection is not feasible, the use of a model that allows partial project selection may result in inappropriate project selection.

(7) The availability of additional support capacity may have a significant impact on the production quantities that can be obtained. Because of this, the provision for additional support capacity can greatly affect the future worth.

(8) As the demand increases, the availability of additional support capacity may have a more pronounced effect on future worth than when the demand is relatively low.

Because of this, the loss in future worth due to insufficient provisions for additional capacity may be greater when demand is high than when the demand is relatively low.

(9) Project budgets (i.e., the maximum amount that can be spent on projects in a fiscal period) may adversely affect the future worth that can be obtained from the concurrent model. That is, the inclusion of maximum project expense constraints may reduce the production quantities and, thus, the future worth that can be obtained from any version of the concurrent model.

8.3. Recommendations

It is clear from the preceding sections that the concurrent model always provided better results for this hypothetical example. Because of this, all recommendations for future work are focused towards improvements that may be made to the concurrent model. These recommendations are presented below:

(1) The concurrent model can be modified so that a single capital budgeting project can affect more than one type of work center. In its current state, the concurrent model considered three types of work centers and three types of capital budgeting projects. Separation of projects by work center type was necessitated by the need to compare the concurrent model to a sequential model. This is because the first stage of the sequential model only considered pre-production/post-support capital budgeting projects while stages two and three only considered production and support projects. However, since the concurrent model simultaneously solves these three project types, it is possible in the concurrent model to consider one project that affects all three types of work centers.

(2) The concurrent model may be modified so that all three types of capital budgeting project constraints are allowed in the same model. During this research, three versions of the concurrent model were developed. Each version allowed only one type of capital budgeting project constraint. This segregation of constraints was required to demonstrate that the concurrent model may provide better results when any type of project constraint is considered. Since this has been shown, the concurrent model may be extended in the future to simultaneously consider capital budgeting project constraints of all three types.

(3) A hypothetical example may be constructed, and applied to the concurrent model over several periods. Simulation may be used to generate the actual values of inputs that were estimated during planning. Analysis of this type conducted over

several periods using simulated data can be used to show how the model performs in the dynamic decision environment.

(4) The concurrent model may be further extended to determine where the input values could be gathered in a real-world organization. As such, the concurrent model may form a framework for further research in the areas of product, production, and capacity planning.

(5) The concurrent model may be applied in an organization to determine how well and in what ways the model improves the product, production, and capacity planning process of the organization. The results of the concurrent model may be compared to the typical planning approach taken in the organization. It is recommended that a small study be conducted so that the problem is small enough and well defined to yield comparable results.

9. REFERENCES

9.1. References Cited

Ansoff, H. Igor **Strategic Management**, London, England: The Macmillan Press Ltd., 1979.

Anthony, Robert N. **Planning and Control Systems: A Framework for Analysis**, Boston, Massachusetts: Harvard University, 1965.

Baker, Michael J. and Ronald McTavish **Product Policy and Management**, London, England: The Macmillan Press Ltd., 1976.

Baker, Kenneth R. and William W. Damon *A Simultaneous Planning Model for Production and Working Capital*, **Decision Sciences**, Vol. 8 (January 1977), pp. 95 - 108.

Billington, Peter J. *The Classic Economic Production Quantity Model with Setup Cost as a Function of Capital Expenditure*, **Decision Sciences**, Vol. 18, No. 1 (Winter 1987), pp. 25 - 42.

Blanchard, Benjamin S. and Wolter J. Fabrycky **Systems Engineering and Analysis**, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1990.

Blank, Leland *The Changing Scene of Economic Analysis for the Evaluation of Manufacturing System Design and Operation*, **The Engineering Economist**, Vol. 30, No. 3 (Spring 1985), pp. 227 - 244.

Bulloch, James, Donald E. Keller, and Louis Vlasho **Accountants' Cost Handbook**, New York, NY: John Wiley & Sons, 1983.

Canada, John R. and William G. Sullivan **Economic and Multiattribute Evaluation of Advanced Manufacturing Systems**, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1989.

Chatfield, Michael and Denis Neilson **Cost Accounting**, New York, NY: Harcourt Brace Jovanovich, Inc., 1983.

Clark, John J., Thomas J. Hindelang, and Robert E. Pritchard **Capital Budgeting: Planning and Control of Capital Expenditures**, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1984.

Crowingshield, Gerald R. and Kenneth Gorman **Cost Accounting: Principles and Managerial Applications**, Boston, Mass: Houghton Mifflin Company, 1979.

Damon, William W. and Richard Schramm *A Simultaneous Decision Model for Production, Marketing, and Finance*, **Management Science**, Vol. 19, No. 2 (October 1972), pp. 161 - 172.

Dickey, Robert F. **Accountans' Cost Handbook**, New York, NY: The Ronald Press Company, 1964.

Evans, Dorla A. *Investment Decision Making Under Uncertainty: Potential Environmental/Social Impacts of New Products*, **The Engineering Economist**, Vol. 32, No. 4 (Summer 1987), pp. 263 - 275.

Fabrycky, Wolter J. and Benjamin S. Blanchard **Life-Cycle Cost and Economic Analysis**, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1991.

Federal Tax Regulations, *U.S. Code Congressional and Administrative News*, Volume 2, St. Paul, Minn.: West Publishing Company, 1993.

Fishburn, Peter **Nonlinear Preference and Utility Theory**, Baltimore, Maryland: The Johns Hopkins University Press, 1988.

Groover, Mikell P. and Emory W. Zimmers, Jr. **CAD/CAM: Computer Aided Design and Manufacturing**, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1984.

Gurnani, Chandan *Capital Budgeting: Theory and Practice*, **The Engineering Economist**, Vol. 30, No. 1 (Fall 1984), pp. 19 - 46.

Hax, Arnoldo C. and Dan Candea **Production and Inventory Management**, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1984.

Hax, Arnoldo C. and Harlan C. Meal *Hierarchical Integration of Production Planning and Scheduling* in **North-Holland/TIMS Studies in Management Sciences Vol.1, Logistics** edited by Murray A. Geisler, Amsterdam; North-Holland Publishing Company, 1975.

Hodder, James E. and Henry E. Riggs *Pitfalls in Evaluating Risky Projects*, **Harvard Business Review**, No. 1 (January-February 1985), pp. 128 - 135.

Holt, Charles C., Franco Modigliani, John F. Muth, and Herbert A. Simon **Planning Production, Inventories, and Work Force**, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1960.

Iglehart, Donald L. *Capital Accumulation and Production for the Firm: Optimal Dynamic Policies*, **Management Science**, Vol. 12, No. 3 (November 1965), pp. 193 - 205.

Kim, Jay S., Larry P. Ritzman, W. C. Benton, and David L. Snyder *Linking Product Planning and Process Design Decisions*, **Decision Sciences**, Vol. 23 (1992), pp. 44 - 59.

Kim, Suk H. and Edward J. Farragher *Current Capital Budgeting Practices*, **Management Accounting**, June 1981, pp. 26 - 30.

Kim, Suk H. *An Empirical Study on the Relationship between Capital Budgeting Practices and Earnings Performance*, **The Engineering Economist**, Vol. 27, No. 3 (Spring 1982), pp. 185 - 196.

Lawrence, Kenneth D., Sheila M. Lawrence, and Gary R. Reeves *Aggregate Industrial Expansion: A Multiple-Objective Linear Programming Approach*, **The Engineering Economist**, Vol. 25, No. 3 (Spring 1980), pp. 197 - 207.

Luss, Hanan *Operations Research and Capacity Expansion Problems : A Survey*, **Operations Research**, Vol. 30, No. 5 (September - October 1982), pp. 907 - 947.

Malhotra, Manoj K. and Larry P. Ritzman *Resource Flexibility Issues in Multistage Manufacturing*, **Decision Sciences**, Vol. 21, No. 4 (Fall 1990), pp. 673 - 690.

Matz, Adolph and Milton F. Usry **Cost Accounting: Planning and Control**, Cincinnati, Ohio: South-Western Publishing Company, 1984.

Nunamaker, Thomas R. and Jack F. Truitt *Rationing Discretionary Economic Resources: A Multiobjective Approach*, **Decision Sciences**, Vol. 18, No. 4 (Fall 1987), pp. 524 - 534.

Park, Chan S. and Gunter P. Sharp-Bette **Advanced Engineering Economics**, New York, NY: John Wiley & Sons, Inc., 1990.

Park, Chan S. and Young K. Son *An Economic Evaluation Model for Advanced Manufacturing Systems*, **The Engineering Economist**, Vol. 34, No. 1 (Fall 1988), pp. 1 - 26.

Rosenblatt, Meir J. *A Survey and Analysis of Capital Budgeting Decision Processes in Multi-Divisional Firms*, **The Engineering Economist**, Vol. 25, No. 4 (Summer 1980), pp. 259 - 273.

Salazar, Rodolpho C. and Subrata K. Sen *A Simulation Model of Capital Budgeting Under Uncertainty*, **Management Science**, Vol. 15, No. 4 (December 1968), pp. B-161 - B-179.

Sherali, Hanif D. and C. M. Shetty **Optimization with Disjunctive Programming**, New York, NY: Springer-Verlag, 1980.

Sink, D. Scott and Thomas C. Tuttle **Planning and Measurement in Your Organization of the Future**, Norcross, Georgia: Industrial Engineering and Management Press, 1989.

Skipper, Lee R. **Development of a Microcomputer-Based Capital Budgeting Algorithm for the Dynamic Decision Environment**, Blacksburg, Virginia: Unpublished Master's Thesis, 1985.

Skipper, Lee R. and Wolter J. Fabrycky *The Life-Cycle Economic Evaluation of Interdependent Projects*, in the 1985 **Annual International Industrial Engineering Conference Proceedings**.

Smith, Robert L. *Optimal Expansion Policies for the Deterministic Capacity Problem*, **The Engineering Economist**, Vol. 25, No. 3 (Spring 1980), pp. 149 - 160.

Stiglitz, Joseph E. **Economies of the Public Sector**, New York, NY: W. W. Norton Company, 1988.

Stone, Merlin **Product Planning**, Plymouth, England: The Bowering Press Ltd, 1976.

Sullivan, William G. *Models IEs can use to Include Strategic, Non-Monetary Factors in Automation Decisions*, **Journal of Industrial Engineering**, March 1986, pp. 42 - 50.

Swalm, R. O. and J. L. Lopez-Leautaud **Engineering Economic Analysis: A Future Worth Approach**, New York, NY: John Wiley & Sons, Inc., 1984.

Sypkens, H. A. **Planning for Optimal Plant Capacity**, Cambridge, Massachusetts: Unpublished Master's Thesis, 1967.

Thompson, James D. **Organizations in Action**, New York, NY: McGraw-Hill Book Company, 1967.

Thuesen, G. J. and W. J. Fabrycky **Engineering Economy**, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1993.

Trzaskalik, T. *Multi-Objective, Multi-Period Planning for a Manufacturing Plant*, **Engineering Costs and Production Economics**, Vol. 20 (1990), pp. 113 - 120.

Vander Veen, David J. and William C. Jordan *Analyzing Trade-Offs Between Machine Investment and Utilization*, **Management Science**, Vol. 15, No. 10 (October 1989), pp. 1215 - 1227.

Vercellis, C. *Multi-Criteria Models for Capacity Analysis and Aggregate Planning in Manufacturing Systems*, **International Journal of Production Economics**, Vol. 23 (1991), pp. 261 - 272.

9.2. Additional References

Adler, Lee *Systems Approach to Marketing*, In **Product Planning** edited by A. Edward Spitz, Princeton, New Jersey: Auerbach Publishers, 1972.

Ashford, R. W., R. G. Dyson, and S. D. Hodges *The Capital-Investment Appraisal of New Technology: Problems, Misconceptions and Research Directions*, **Journal of the Operational Research Society**, Vol. 39, No. 7 (1988), pp. 637 - 642.

Axsäter, Sven *On the Design of the Aggregate Model in a Hierarchical Production Planning System*, **Engineering and Process Economics**, Vol. 4 (1979), pp. 89 - 97.

Bechtel, William and Robert C. Richardson **Discovering Complexity: Decomposition and Localization as Strategies in Scientific Research**, Princeton, New Jersey: Princeton University Press, 1993.

Bergstrom, Gary L. and Barnard E. Smith *Multi-Item Production Planning - An Extension of the HMMS Rules*, **Management Science**, Vol. 16, No. 10 (June 1970), pp. B-614 - B-629.

Bierman, Harold, Jr., and Warren H. Hausman *The Resolution of Investment Uncertainty Through Time*, **Management Science**, Vol. 18, No. 12 (August 1972), pp. B-654 - B-662.

Bitran, Gabriel R., Elizabeth A. Haas, and Arnoldo C. Hax *Hierarchical Production Planning: A Single Stage System*, **Operations Research**, Vol. 29, No. 4 (July - August 1981), pp. 717 - 743.

Chakravarty, Amiya K. *New Technology Investments in Multistage Production Systems*, **Decision Sciences**, Vol. 16, No. 3 (Summer 1985), pp. 248 - 264.

Clark, John J., Thomas J. Hindelang, and Robert E. Pritchard **Capital Budgeting: Planning and Control of Capital Expenditures**, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1984.

Cooper, Robin and Robert S. Kaplan *Activity-Based Systems: Measuring the Costs of Resource Usage*, **Accounting Horizons**, September 1992, pp. 1 - 12.

Daft, Richard L. **Organizational Theory and Design**, St. Paul, Minnesota: West Publishing Company, 1983.

Fine, Charles H. and Robert M. Freund *Optimal Investment in Product-Flexible Manufacturing Capacity*, **Management Science**, Vol. 36, No. 4 (April 1990), pp. 449 - 466.

Fotsch, Richard J. *Machine Tool Justification Policies: Their Effect on Productivity and Profitability*, **Journal of Manufacturing Systems**, Vol. 3, No. 2 (1983), pp. 169 - 189.

Geisler, Murray A. and Laurence D. Richards *The Use of Management Indicators in a Large Public System*, in **Studies in Operations Management** edited by Arnoldo C. Hax, New York, NY: North-Holland Publishing Company, 1978.

Gibson, James L., John M. Ivanevich, and James H. Donnelly, Jr. **Organizations: Behavior, Structure, Processes**, Dallas, Texas: Business Publication, Inc., 1992.

Graves, Stephen C. *A Review of Production Scheduling*, **Operations Research**, Vol. 29, No. 4 (July - August 1981), pp. 646 - 675.

Hanssmann, Fred and Sidney W. Hess *A Linear Programming Approach to Production and Employment Scheduling*, **Management Technology**, Vol. 1, No. 1 (January 1960), pp. 46 - 52.

Hax, Arnoldo C. and Nicolas S. Majuf *A Methodological Approach for the Development of Strategic Planning in Diversified Corporations*, in **Studies in Operations Management** edited by Arnoldo C. Hax, New York, NY: North-Holland Publishing Company, 1978.

Hrebiniak, L. G., W. F. Joyce, and C. C. Snow *Strategy, Structure, and Performance: Past and Future Research*, In **Strategy, Organizational Design, and Human Resource Management**, edited by Charles C. Snow, Greenwich, Connecticut: JAI Press, Inc., 1989.

Huang, Philip and Parviz Ghandforoush *Procedures Given for Evaluating, and Selecting Robots*, **Journal of Industrial Engineering**, April 1984, pp. 44 - 48.

Hussey, David E. **Corporate Planning Theory and Practice**, Oxford, England: Pergamon Press, 1982.

Karmarkar, Uday S. *Capacity Loading and Release Planning with Work-in Progress (WIP) and Leadtimes*, **Manufacturing Operations Management**, Vol. 2 (1989), pp. 105 - 123.

King, G. M. and R. E. Thomas *Selecting the Manufacturing Capacity of a New Product*, **Engineering and Process Economics**, Vol. 3 (1978), pp. 179 - 185.

Kistner, K. P. and M. Switalski *Hierarchical Production Planning Necessity, Problems, and Methods*, **Zeitschriff für Operations Research**, Vol. 33, No. 3 (1989), pp. 199 - 212.

Krinsky, Itzhak *Capital Budgeting and Plant Capacity*, **Engineering Costs and Production Economics**, Vol. 21 (1991), pp. 233 - 241.

Maggio, Ralph A. *Design or Production and Distribution Systems*, in **Industrial Engineering Handbook** edited by H. B. Maynard, New York, NY: McGraw-Hill Book Company, 1971.

Manne, Alan S. and Arthur F. Veinott, Jr. *Optimal Plant Size with Arbitrary Increasing Time Paths of Demand* in **Investments for Capacity Expansion** edited by Alan S. Manne, Cambridge, Massachusetts; The MIT Press, 1967.

Myers, Stewart C. *Notes on an Expert System for Capital Budgeting*, **Financial Management**, Autumn 1988, pp. 23 - 31.

Parker, R. Gary and Ronald L. Rardin **Discrete Optimization**, Boston, Massachusetts: Academic Press, Inc., 1988.

Petty, J. William, David F. Scott, Jr., and Monroe M. Bird *The Capital Expenditure Decision-Making Process of Large Corporations*, **The Engineering Economist**, Vol. 20, No. 3 (Spring 1975), pp. 159 - 172.

Pike, Richard *Do Sophisticated Capital Budgeting Approaches Improve Investment Decision Making?*, **The Engineering Economist**, Vol. 34, No. 2 (Winter 1989).

Pinkus, Charles E., Donald Gross, and Richard E. Soland *Optimal Design of Multiactivity Multifacility Systems by Branch and Bound*, **Operations Research**, Vol. 21, No. 1 (January-February 1973), pp. 270 - 283.

Rajagopalan, S. *Deterministic Capacity Expansion Under Deterioration*, **Management Science**, Vol. 38, No. 4 (April 1992), pp. 525 - 539.

Rocklin, Sol M., Arik Kashper, and George C. Varvaloucas *Capacity Expansion/Contraction of a Facility with Demand Augmentation Dynamics*, **Operations Research**, Vol. 32, No. 1 (January - February 1984), pp. 133 - 147.

Rossler, Paul E. and D. Scott Sink *A Roadmap for Quality and Productivity Management and Improvement*, **Engineering Management Journal**, Vol. 2, No. 3 (September 1990), pp. 17 - 22.

Scudder, Gary D. *Application of the Net Present Value Criterion in Random and Flow Shop Scheduling*, **Decision Sciences**, Vol. 20, No. 3 (Summer 1989), pp. 602 - 622.

Sink, D. Scott and George L. Smith, Jr. *The Role of Measurement in Managing Linkages between Individual, Group, and Organizational Performance*, prepared for The National Research Council Commission on Behavioral and Social Sciences and Education Committee on Human Factors Panel on Organizational Linkages Applications Subgroup, (no date).

Steiner, George A. **Strategic Planning**, New York, NY: The Free Press, 1979.

Strategy + Structure = Performance: The Strategic Planning Imperative, edited by Hans B. Thorelli, Bloomington, Indiana: The Indiana University Press, 1977.

Sung, C. S. and Y. S. Park *Dynamic Lot Sizing for a Single-Facility Multiproduct Problem in a Rolling-Horizon Environment*, **Decision Sciences**, Vol. 18, No. 2 (Spring 1987), pp. 266 - 278.

Van Horne, James C. *The Analysis of Uncertainty Resolution in Capital Budgeting for New Products*, **Management Science**, Vol. 15, No. 8 (April 1969), pp. B-376 - B386.

Veinott, Arthur F., Jr. and Harvey M. Wagner *Optimal Capacity Scheduling - 1*, **Operations Research**, Vol. 10, No. 4 (July - August 1962), pp. 518 - 532.

Veinott, Arthur F., Jr. and Harvey M. Wagner *Optimal Capacity Scheduling - 2*, **Operations Research**, Vol. 10, No. 4 (July - August 1962), pp. 533 - 546.

Wagner, Harvey M. *The Design of Production and Inventory Systems for Multifacility and MultiWharehouse Companies*, **Operations Research**, Vol. 22, No. 2 (March - April 1974), pp. 279 - 291.

Weingartner, H. Martin *Some New Views on the Payback Period and Capital Budgeting Decisions*, **Management Science**, Vol. 15, No. 12 (August 1969), pp. B-594 - B-642.

APPENDIX A: Input and Output Data Sets

input.dat

```
150 150 150 150 150 150 150 150
150 150 150 150 150 150 150 150
9
0 4 2 0 0
4 0 1
1 3 5 8 24
35 40 30 60 45
100 78 65 95 78
45 40 30 50 25
70 62 50 85 65
95 72 60 69 55
50 45 30 60 40
5000 5000 5000 5000 5000 5000 5000 5000
0 0 0 0 0 0 0 0
3000 3000 3000 3000 3000 3000 3000 3000
0 0 0 0 0 0 0 0
6240 6240 6240 6240 6240 6240 6240 6240
0 0 0 0 0 0 0 0
9000 9000 9000 9000 9000 9000 9000 9000
0 0 0 0 0 0 0 0
4000 4000 4000 4000 4000 4000 4000 4000
60000 61000 62000 63000 0 0 0 0
0 0 0 61500 62500 63500 64500 65500
0 0 0 0 62250 63250 64250 65250
59500 59500 59500 59500 59500 59500 59500 59500
0 0 0 62250 63250 64250 65250 66250
0.10
0.05
30.0 30.0 30.0 30.0 30.0 30.0 30.0 32.0
35.0 35.0 35.0 35.0 35.0 35.0 35.0 38.0
25.0 25.0 25.0 25.0 25.0 25.0 25.0 27.0
45.0 45.0 45.0 45.0 45.0 45.0 45.0 48.0
52.5 52.5 52.5 52.5 52.5 52.5 52.5 57.0
37.5 37.5 37.5 37.5 37.5 37.5 37.5 40.5
8320 8320 8320
20 30 15 50
50 50 50 50 50 50 50 50 50 50 50 50
75 75 75 75 75 75 75 75 75 75 75 75
50 50 50 50 50 50 50 50 50 50 50 50
30 30 30 30 30 30 30 30 30 30 30 30
180289 180289 180289 180289 180289 180289 180289 180289 180289 180289
```

180289 180289
187500 187500 187500 187500 187500 187500 187500 187500 187500 187500
187500 187500
250000 250000 250000 250000 250000 250000 250000 250000 250000 250000
250000 250000
150000 150000 150000 150000 150000 150000 150000 150000 150000 150000
150000 150000
80000 80000 80000 80000 80000 80000 80000 80000 80000 80000
80000 80000
750000 750000 750000 750000 750000 750000 750000 750000 750000 750000
750000 750000
200000 200000 200000 200000 200000 200000 200000 200000 200000 200000
200000 200000
100000 100000 100000 100000 100000 100000 100000 100000 100000 100000
100000 100000
0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
4 4 4 4 4
2 2 2 2 2
25.3 26.1 27.25 28.4
24.0 24.5 25.1 26.3
18.3 20.1 21.25 22.5
27.2 29.3 29.5 30.0
20.1 21.3 22.5 23.0
12000
3000000 3000000 3000000 3000000 3000000 3000000 3000000 3000000
200 250 300 350
150 200 250 300
75 100 200 250
225 275 300 350
125 175 250 290
225000 225000 225000 225000 225000 225000 225000 225000 225000 225000 225000
0 0 500 650
0 0 450 500
0 0 400 450
0 0 525 670
0 0 500 575
100 100 100 100 100 100 100 100 100 100 100
30 31 32 33 34 35 36 37 38 39 40
300000 250000 350000
200000 300000 125000 50000
125000

8500 8500 8500 8500 8500 8600 8700 9000
7500 7500 7500 7500 7600 7700 7700 7800
7000 7000 7000 7000 7100 7100 7100 7100
8500 8500 8500 8500 8500 8600 9000 9000
7750 7750 7750 7750 7750 7750 7800 7800
25.00 25.50 25.75 25.90 26.00 26.25 26.50 26.75 26.95 27.00 27.30
2.0 2.0 2.0 2.0 0 0 0 0 0 0 0 0
12.25 6.15 3.0 3.0 3.0 3.0 3.0 3.0 0 0 0 0
0 20.5 10.25 5.0 3.0 2.0 2.0 2.0 0 0 0 0
2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 0 0 0 0
12.3 6.15 3.0 2.0 2.0 2.0 2.0 2.0 0 0 0 0
3.0 3.0 3.0 3.0 0 0 0 0 0 0 0 0
0 12.0 6.0 3.0 3.0 3.0 3.0 3.0 0 0 0 0
0 0 12.0 6.0 3.0 3.0 3.0 3.0 0 0 0 0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 0 0 0 0
0 20.0 10.0 5.0 3.0 3.0 3.0 3.0 0 0 0 0
5.0 5.0 5.0 5.0 0 0 0 0 0 0 0 0
0 0 10.0 8.0 6.0 5.0 5.0 5.0 0 0 0 0
0 0 0 10.0 8.0 6.0 5.0 5.0 0 0 0 0
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 0 0 0 0
0 0 9.0 7.0 6.0 5.0 5.0 5.0 0 0 0 0
0 0 0 0 0 0 0 10.0 20.0 0 0 0
0 0 0 0 0 0 0 0 0 0 10.0 20.0
0 0 0 0 0 0 0 0 0 0 10.0 20.0
0 0 0 0 0 0 0 0 0 0 10.0 20.0
0 0 0 0 0 0 0 0 0 0 10.0 20.0
4 5 4 8 5
0 3 4 0 3
100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100
6000 6000 6000 6000 6000 6000 6000 6000 6000 6000 6000
1 0 0 1 0

output.dat

INPUT IS FROM: input.dat

LIST OF INPUTS AND OUTPUTS

A. Input Data

DEMAND FOR SUBGROUP ONE
150 150 150 150 150 150 150 150
DEMAND FOR SUBGROUP TWO
150 150 150 150 150 150 150 150
MUST FUND
9
MUTUALLY EXCLUSIVE PRODUCTS
0 4 2 0 0
MUTUALLY EXCLUSIVE PROJECTS
4 0 1
PRODUCT CONTINGENCY
1 3 5 8 24
PRINT SETUP HOURS
Setup Hours At Work Center 0 for Each Product
35 40 30 60 45
Setup Hours At Work Center 1 for Each Product
100 78 65 95 78
Setup Hours At Work Center 2 for Each Product
45 40 30 50 25
PRINT PRODUCTION HOURS
Production Hours At Work Center 0 for Each Product
70 62 50 85 65
Production Hours At Work Center 1 for Each Product
95 72 60 69 55
Production Hours At Work Center 2 for Each Product
50 45 30 60 40
PRODUCTION INCREASES FROM PROJECT 0
Production Increases At Work Center 0
5000 5000 5000 5000 5000 5000 5000 5000
Production Increases At Work Center 1
0 0 0 0 0 0 0 0
Production Increases At Work Center 2
3000 3000 3000 3000 3000 3000 3000 3000
PRODUCTION INCREASES FROM PROJECT 1

Production Increases At Work Center 0
0 0 0 0 0 0 0 0
Production Increases At Work Center 1
6240 6240 6240 6240 6240 6240 6240 6240
Production Increases At Work Center 2
0 0 0 0 0 0 0 0
PRODUCTION INCREASES FROM PROJECT 2
Production Increases At Work Center 0
9000 9000 9000 9000 9000 9000 9000 9000
Production Increases At Work Center 1
0 0 0 0 0 0 0 0
Production Increases At Work Center 2
4000 4000 4000 4000 4000 4000 4000 4000
REVENUE
Revenue from product 0
60000.00 61000.00 62000.00 63000.00 0.00 0.00 0.00 0.00
Revenue from product 1
0.00 0.00 0.00 61500.00 62500.00 63500.00 64500.00 65500.00
Revenue from product 2
0.00 0.00 0.00 0.00 62250.00 63250.00 64250.00 65250.00
Revenue from product 3
59500.00 59500.00 59500.00 59500.00 59500.00 59500.00 59500.00 59500.00
Revenue from product 4
0.00 0.00 0.00 62250.00 63250.00 64250.00 65250.00 66250.00
INTEREST RATE
ACC = 0.100
INFLATION RATE
inflation = 0.050
REGULAR TIME COSTS
Regular Time Cost for Production Work Center 0
30.00 30.00 30.00 30.00 30.00 30.00 30.00 32.00
Regular Time Cost for Production Work Center 1
35.00 35.00 35.00 35.00 35.00 35.00 35.00 38.00
Regular Time Cost for Production Work Center 2
25.00 25.00 25.00 25.00 25.00 25.00 25.00 27.00
PRINT OVER TIME COST
Over Time Cost for Production Work Center 0
45.00 45.00 45.00 45.00 45.00 45.00 45.00 48.00
Over Time Cost for Production Work Center 1
52.50 52.50 52.50 52.50 52.50 52.50 52.50 57.00
Over Time Cost for Production Work Center 2
37.50 37.50 37.50 37.50 37.50 37.50 37.50 40.50
BEGINNING HOURS AT PRODUCTION WC'S

8320.00 8320.00 8320.00
BEGINNING UNITS OF CAPACITY AT NON-PRODUCTION WC'S
20.00 30.00 15.00 50.00
INCREASES FROM NON-PRODUCTION CAPITAL BUDGTEING PROJECT ONE
Increases at WC0
50 50 50 50 50 50 50 50 50 50 50 50
Increases at WC1
75 75 75 75 75 75 75 75 75 75 75 75
Increases at WC2
50 50 50 50 50 50 50 50 50 50 50 50
Increases at WC3
30 30 30 30 30 30 30 30 30 30 30 30
COST OF PRODUCTION CAPITAL BUDGETING PROJECTS
Capital Budgeting Project 0
180289.00 180289.00 180289.00 180289.00 180289.00 180289.00 180289.00
180289.00 180289.00 180289.00 180289.00 180289.00
Capital Budgeting Project 1
187500.00 187500.00 187500.00 187500.00 187500.00 187500.00 187500.00
187500.00 187500.00 187500.00 187500.00 187500.00
Capital Budgeting Project 2
250000.00 250000.00 250000.00 250000.00 250000.00 250000.00 250000.00
250000.00 250000.00 250000.00 250000.00 250000.00
COST OF NON-PRODUCTION CAPITAL BUDGETING PROJECT
150000.00 150000.00 150000.00 150000.00 150000.00 150000.00 150000.00 150000.00
150000.00 150000.00 150000.00 150000.00
COST OF SUPPORT CAPITAL BUDGETING PROJECT
80000.00 80000.00 80000.00 80000.00 80000.00 80000.00 80000.00 80000.00 80000.00
80000.00 80000.00 80000.00
MAXIMUM ALLOWABLE PRODUCTION PROJECT EXPENSE
750000.00 750000.00 750000.00 750000.00 750000.00 750000.00 750000.00
750000.00 750000.00 750000.00 750000.00 750000.00
MAXIMUM ALLOWABLE NON-PRODUCTION PROJECT EXPENSE
200000.00 200000.00 200000.00 200000.00 200000.00 200000.00 200000.00 200000.00
200000.00 200000.00 200000.00 200000.00
MAXIMUM ALLOWABLE SUPPORT PROJECT EXPENSE
100000.00 100000.00 100000.00 100000.00 100000.00 100000.00 100000.00 100000.00
100000.00 100000.00 100000.00 100000.00
MAXIMUM FRACTION REGULAR TIME LABOR
WC 0
0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.50
WC 1
0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.50
WC 2

0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.50
PRODUCT LIFE LENGTH
4 4 4 4 4
PRODUCT WARRANTEE LENGTH
2 2 2 2 2
SUPPORT HOURS FOR PRODUCTS OF AGE TAU
Support Hours for Product Number 0
25.30 26.10 27.25 28.40
Support Hours for Product Number 1
24.00 24.50 25.10 26.30
Support Hours for Product Number 2
18.30 20.10 21.25 22.50
Support Hours for Product Number 3
27.20 29.30 29.50 30.00
Support Hours for Product Number 4
20.10 21.30 22.50 23.00
BEGINNING SUPPORT HOURS AVAILABLE
12000.00
MAXIMUM ALLOWABLE PRODUCTION COST
3000000.00 3000000.00 3000000.00 3000000.00 3000000.00 3000000.00 3000000.00 3000000.00
AVERAGE PARTS COST TO REPAIR PRODUCT OF AGE TAU
Average parts cost for Product Number 0
200.00 250.00 300.00 350.00
Average parts cost for Product Number 1
150.00 200.00 250.00 300.00
Average parts cost for Product Number 2
75.00 100.00 200.00 250.00
Average parts cost for Product Number 3
225.00 275.00 300.00 350.00
Average parts cost for Product Number 4
125.00 175.00 250.00 290.00
MAXIMUM REPAIR PARTS COST
225000.00 225000.00 225000.00 225000.00 225000.00 225000.00 225000.00 225000.00 225000.00 225000.00 225000.00
PARTS REVENUE FROM REPAIR OF PRODUCT OF AGE TAU
Average parts revenue for Product Number 0
0.00 0.00 500.00 650.00
Average parts revenue for Product Number 1
0.00 0.00 450.00 500.00
Average parts revenue for Product Number 2
0.00 0.00 400.00 450.00
Average parts revenue for Product Number 3

0.00 0.00 525.00 670.00
Average parts revenue for Product Number 4
0.00 0.00 500.00 575.00
MINIMUM ACCEPTABLE REVENUE
100.00 100.00 100.00 100.00 100.00 100.00 100.00 100.00 100.00 100.00 100.00
REPAIR LABOR REVENUE PER HOUR
30.00 31.00 32.00 33.00 34.00 35.00 36.00 37.00 38.00 39.00 40.00
BASE COST TO OPERATE PRODUCTION WC
300000.000000 250000.000000 350000.000000
BASE COST TO OPERATE NON-PRODUCTION WC
200000.000000 300000.000000 125000.000000 50000.000000
BASE COST TO OPERATE SUPPORT WC
125000.000000
MATERIAL COST FOR PRODUCTION
Material cost for Product Number 0
8500.00 8500.00 8500.00 8500.00 8500.00 8600.00 8700.00 9000.00
Material cost for Product Number 1
7500.00 7500.00 7500.00 7500.00 7600.00 7700.00 7700.00 7800.00
Material cost for Product Number 2
7000.00 7000.00 7000.00 7000.00 7100.00 7100.00 7100.00 7100.00
Material cost for Product Number 3
8500.00 8500.00 8500.00 8500.00 8500.00 8600.00 9000.00 9000.00
Material cost for Product Number 4
7750.00 7750.00 7750.00 7750.00 7750.00 7750.00 7800.00 7800.00
REPAIR LABOR COST
25.00 25.50 25.75 25.90 26.00 26.25 26.50 26.75 26.95 27.00 27.30
CONSISTENT UNITS REQUIRED BY PRODUCT AT NON_PRODUCTION WC
Units required at wc 0
Consistent non-production units for product number 0
2.00 2.00 2.00 2.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Consistent non-production units for product number 1
12.25 6.15 3.00 3.00 3.00 3.00 3.00 3.00 0.00 0.00 0.00 0.00
Consistent non-production units for product number 2
0.00 20.50 10.25 5.00 3.00 2.00 2.00 2.00 0.00 0.00 0.00 0.00
Consistent non-production units for product number 3
2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00 0.00 0.00 0.00 0.00
Consistent non-production units for product number 4

12.30 6.15 3.00 2.00 2.00 2.00 2.00 2.00 0.00 0.00 0.00 0.00
Units required at wc 1
Consistent non-production units for product number 0
3.00 3.00 3.00 3.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Consistent non-production units for product number 1
0.00 12.00 6.00 3.00 3.00 3.00 3.00 3.00 0.00 0.00 0.00 0.00
Consistent non-production units for product number 2
0.00 0.00 12.00 6.00 3.00 3.00 3.00 3.00 0.00 0.00 0.00 0.00
Consistent non-production units for product number 3
3.00 3.00 3.00 3.00 3.00 3.00 3.00 3.00 0.00 0.00 0.00 0.00
Consistent non-production units for product number 4
0.00 20.00 10.00 5.00 3.00 3.00 3.00 3.00 0.00 0.00 0.00 0.00
Units required at wc 2
Consistent non-production units for product number 0
5.00 5.00 5.00 5.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Consistent non-production units for product number 1
0.00 0.00 10.00 8.00 6.00 5.00 5.00 5.00 0.00 0.00 0.00 0.00
Consistent non-production units for product number 2
0.00 0.00 0.00 10.00 8.00 6.00 5.00 5.00 0.00 0.00 0.00 0.00
Consistent non-production units for product number 3
5.00 5.00 5.00 5.00 5.00 5.00 5.00 5.00 0.00 0.00 0.00 0.00
Consistent non-production units for product number 4
0.00 0.00 9.00 7.00 6.00 5.00 5.00 5.00 0.00 0.00 0.00 0.00
Units required at wc 3
Consistent non-production units for product number 0
0.00 0.00 0.00 0.00 0.00 0.00 0.00 10.00 20.00 0.00 0.00 0.00
Consistent non-production units for product number 1
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 10.00 20.00
Consistent non-production units for product number 2
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 10.00 20.00
Consistent non-production units for product number 3
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 10.00 20.00
Consistent non-production units for product number 4
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 10.00 20.00
MAXIMUM PRODUCTION PERIODS FOR EACH PRODUCT
4 5 4 8 5
EARLIEST PERIOD PRODUCTION OF A PRODUCT MAY COMMENCE
0 3 4 0 3
INVENTORY HOLDING COSTS FOR PRODUCTS IN EACH PERIOD
PRODUCT 0
100.00 100.00 100.00 100.00 100.00 100.00 100.00 100.00
PRODUCT 1
100.00 100.00 100.00 100.00 100.00 100.00 100.00 100.00

PRODUCT 2
100.00 100.00 100.00 100.00 100.00 100.00 100.00 100.00
PRODUCT 3
100.00 100.00 100.00 100.00 100.00 100.00 100.00 100.00
PRODUCT 4
100.00 100.00 100.00 100.00 100.00 100.00 100.00 100.00
IMPROVMENT IN HOURS FROM SUPPORT CAP BUD PROJECT
6000 6000 6000 6000 6000 6000 6000 6000 6000 6000 6000
BEGINNING INVENTORY FOR EACH PRODUCT
1 0 0 1 0

APPENDIX B: Program Listing for the Concurrent Model

```
/* sinmain.h */
/* header file for sinmain.c */

#include <stdio.h>
#include <stdlib.h>
#define NUM_PERIODS                     12
#define HORIZON_PERIOD                  11
#define NUM_SUPPORT_PERIODS             11
#define NUM_PROD_PERIODS                8
#define NUM_PRODUCTS                    5
#define NUM_PROD_WC                     3
#define NUM_NON_PROD_WC                 4
#define NUM_CAP_BUD_PROJECTS            3
#define NUM_NON_PROD_CAP_BUD_PROJ       1
#define AGE                             5 /* maximum life length of any product */

typedef unsigned int BitVector;

int k;
        /* INPUT PARAMETERS */
/* DEMAND */
int d1 [NUM_PROD_PERIODS]; /* Demand for subgroup one */
int d2 [NUM_PROD_PERIODS]; /* Demand for subgroup two */

/* PRODUCT AND PROJECT INTERACTION */
BitVector Mutually_Exclusive_Prod[NUM_PRODUCTS]; /* Mutually exclusive products */
BitVector Mutually_Exclusive_Proj[NUM_CAP_BUD_PROJECTS]; /* " " projects */
BitVector Contingency[NUM_PRODUCTS]; /* Contingency between products */
BitVector Must_Fund; /* Must fund a set of products */

/* INTEREST RATES */
float ACC;  /* Interest Rate */
float inflation; /* Inflation Rate */

/* PRODUCT REVENUES */
float REVENUE [NUM_PRODUCTS][NUM_PROD_PERIODS]; /* Product revenue */

/* PRE-PRODUCTION/POST-SUPPORT PRODUCT INPUTS */
float G[NUM_NON_PROD_WC][NUM_PRODUCTS][NUM_PERIODS]; /* No. consistent units req by
product at non-production wc's  */

/* LABOR INPUTS */
float WLT [NUM_PROD_WC] [NUM_PROD_PERIODS]; /* reg time cost by wc */
float OLT [NUM_PROD_WC] [NUM_PROD_PERIODS]; /* ot time cost by wc */
float LAMBDA[NUM_PROD_WC][NUM_PROD_PERIODS]; /* Fraction labor regular time */

/* PRODUCTION INPUTS */
int SETUP_HOURS[NUM_PROD_WC] [NUM_PRODUCTS]; /* setup hours for prod by wc */
int PRODUCTION_HOURS[NUM_PROD_WC] [NUM_PRODUCTS]; /* production hrs by prod & wc */
float PC[NUM_PROD_PERIODS]; /* Maximum allowable production cost */
float MA[NUM_PRODUCTS][NUM_PROD_PERIODS]; /* Material cost for production */
```

```
/* SUPPORT VARIABLES */
float SJT[NUM_PRODUCTS][AGE]; /* Support hours required for product of different ages */
int PLJ[NUM_PRODUCTS]; /* Product life length */
int PWJ[NUM_PRODUCTS]; /* Warrantee length */
float pj[NUM_PRODUCTS][AGE]; /* Average parts cost */
float PT[NUM_PERIODS]; /* Maximum repair parts cost */
float PR[NUM_PRODUCTS][AGE]; /* Parts revenue */
float RT[NUM_PERIODS]; /* Minimum acceptable revenue */
float ft[NUM_PERIODS]; /* Repair labor revenue per hour */
float RC[NUM_PERIODS]; /* Repair labor cost */

/* INVENTORY INPUTS */
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS]; /* inventory holding costs */
int BEGIN_INVENTORY[NUM_PRODUCTS];

/* CAPITAL BUDGETING INPUTS */
float Bl[NUM_PROD_WC]; /* Beginning hours at production wc's */
float Bn[NUM_NON_PROD_WC]; /* Beginning units of capacity at non-production wc's */
float S0; /* Beginning support hours available */
float cl[NUM_CAP_BUD_PROJECTS][NUM_PERIODS]; /* cost of production cap budget projects */
float cn[NUM_PERIODS]; /* cost of non-prod cap budget projects */
float cv[NUM_PERIODS]; /* cost of support capital budgeting project in each period */
float El[NUM_PERIODS]; /* maximum allowable prod project expense */
float En[NUM_PERIODS]; /* maximum allowable pre-production project expense */
float Ev[NUM_PERIODS]; /* maximum allowable post support wc project expense */
float BASE_COST_L[NUM_PROD_WC]; /* Base cost to operate production wc's */
float BASE_COST_N[NUM_NON_PROD_WC]; /* Base cost to operate non-prod wc's */
float BASE_COST_V; /* Base support cost */
int CL [NUM_CAP_BUD_PROJECTS][NUM_PROD_WC][NUM_PROD_PERIODS]; /* improvement at
WC's due to cap bud proj 1 */
int CV [NUM_PERIODS - 1]; /* improvement in support hours from support cap bud project */
int CN[NUM_NON_PROD_WC][NUM_PERIODS]; /* Increase at non-prod wc's from cap bud project */

/* STARTING AND ENDING PERIOD INPUTS */
int max_periods[NUM_PRODUCTS]; /* maximum number of periods a product may be produced */
int start_period[NUM_PRODUCTS]; /* earliest period a product can be produced */

char TITLE[15];

/* prototype of mps function */
void single( int [NUM_PROD_PERIODS], int [NUM_PROD_PERIODS],
int [NUM_CAP_BUD_PROJECTS][NUM_PROD_WC][NUM_PROD_PERIODS],
int [NUM_PROD_WC] [NUM_PRODUCTS], int [NUM_PROD_WC] [NUM_PRODUCTS],
float [NUM_PRODUCTS][NUM_PROD_PERIODS], float [NUM_PROD_WC]
[NUM_PROD_PERIODS],
float [NUM_PROD_WC] [NUM_PROD_PERIODS], float [NUM_PROD_WC],
float [NUM_NON_PROD_WC], int [NUM_NON_PROD_WC][NUM_PERIODS],
float [NUM_CAP_BUD_PROJECTS] [NUM_PERIODS], float [NUM_PERIODS],
float [NUM_PERIODS], float El [NUM_PERIODS], float En [NUM_PERIODS],
float Ev [NUM_PERIODS],
```

```
float [NUM_PROD_WC][NUM_PROD_PERIODS],
float [NUM_PRODUCTS][AGE], float S0, float PC [NUM_PROD_PERIODS],
int PLJ [NUM_PRODUCTS],
int PWJ [NUM_PRODUCTS], float pj [NUM_PRODUCTS][AGE], float PT [NUM_PERIODS],
float PR [NUM_PRODUCTS][AGE], float RT [NUM_PERIODS], float ft [NUM_PERIODS],
float MA [NUM_PRODUCTS][NUM_PROD_PERIODS], float RC [NUM_PERIODS],
float G [NUM_NON_PROD_WC][NUM_PRODUCTS][NUM_PERIODS], int [NUM_PRODUCTS],
int [NUM_PRODUCTS],
float BASE_COST_L[NUM_PROD_WC], float BASE_COST_N[NUM_NON_PROD_WC],
float BASE_COST_V, float ACC,
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS], int CV [NUM_PERIODS - 1],
float inflation, int BEGIN_INVENTORY[NUM_PRODUCTS] );
```

```
/* sinmain.c */
/* main routine for the concurrent model model */

/* reads data from input.dat */
/* writes data to output.dat */

#include "sinmain.h"

main(void)

{

        FILE *fptri, *fptro;
        register i, j;

        /* read inputs from a file and print back to output file */

        if ( (fptri = fopen("input.dat","r")) == NULL ) /* open input file */
                {
                 printf("Can't open the input file.");
                 exit(0);
                }

        if ( ( fptro = fopen("output.dat", "w") ) == NULL ) /* open output file */
                {
                printf("Can't open the output file.");
                exit(0);
                }
        fprintf(fptro, "output.dat\n\n");
        fscanf(fptri, "%s", TITLE );
        fprintf(fptro, "INPUT IS FROM: %s\n", TITLE );

        fprintf(fptro,"\n\n                LIST OF INPUTS AND OUTPUTS\n");
        fprintf(fptro,"                _____________________________\n\n\n");
        fprintf(fptro,"A.  Input Data\n");

        fprintf(fptro, "\n\n");


        /* Read data from input file */
        fprintf(fptro, "DEMAND FOR SUBGROUP ONE\n");
        for(i = 0; i < NUM_PROD_PERIODS; i++) /* read demand for group 1 */
                {
                fscanf(fptri, "%d", &d1[i]);
                fprintf(fptro, "%1d\t", d1[i]);
                }

        fprintf(fptro, "\nDEMAND FOR SUBGROUP TWO\n");
        for(i = 0; i < NUM_PROD_PERIODS; i++) /* read demand for group 2 */
                {
                fscanf(fptri, "%d", &d2[i]);
```

```
        fprintf(fptro, "%1d\t", d2[i]);
        }

fprintf(fptro, "\n\nMUST FUND\n");
fscanf(fptri, "%d", &Must_Fund );
fprintf(fptro, "%4d", Must_Fund );

fprintf(fptro, "\n\nMUTUALLY EXCLUSIVE PRODUCTS\n");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read mutually exclusive */
        {
        fscanf(fptri, "%d", &Mutually_Exclusive_Prod[i]);
        fprintf(fptro, "%4d\t", Mutually_Exclusive_Prod[i]);
        }


fprintf(fptro, "\n\nMUTUALLY EXCLUSIVE PROJECTS\n");
for ( i = 0; i < NUM_CAP_BUD_PROJECTS; ++i ) /* read mut ex cap budget */
        {
        fscanf(fptri, "%d", &Mutually_Exclusive_Proj[i]);
        fprintf(fptro, "%4d\t", Mutually_Exclusive_Proj[i]);
        }

fprintf(fptro, "\n\nPRODUCT CONTINGENCY\n");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read in product contingency */
        {
        fscanf(fptri, "%d", &Contingency[i] );
        fprintf(fptro, "%4d\t", Contingency[i] );
        }

fprintf(fptro, "\n\nPRINT SETUP HOURS");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read setup hours */
        {
        fprintf(fptro,"\nSetup Hours At Work Center %d for Each Product\n", i );
        for ( j = 0; j < NUM_PRODUCTS; j++ )
                {
                fscanf(fptri, "%d", &SETUP_HOURS[i] [j] );
                fprintf(fptro, "%d\t", SETUP_HOURS[i] [j]);
                }
        }


fprintf(fptro, "\n\nPRINT PRODUCTION HOURS");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read production hours */
        {
        fprintf(fptro,"\nProduction Hours At Work Center %d for Each Product\n", i );
        for ( j = 0; j < NUM_PRODUCTS; j++ )
                {
                fscanf(fptri, "%d", &PRODUCTION_HOURS[i] [j] );
                fprintf(fptro, "%d\t", PRODUCTION_HOURS[i] [j] );
                }
        }
```

```
for ( k = 0; k < NUM_CAP_BUD_PROJECTS; k++ )
{
fprintf(fptro, "\n\nPRODUCTION INCREASES FROM PROJECT %d", k );
for ( i = 0; i < NUM_PROD_WC; i++ )
        {
        fprintf(fptro,"\nProduction Increases At Work Center %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                {
                fscanf(fptri, "%d", &CL[k][i][j] );
                fprintf(fptro, "%d\t", CL[k][i][j] );
                }
        }
}

fprintf(fptro, "\n\nREVENUE");
for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
        fprintf(fptro, "\nRevenue from product %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
        {
        fscanf(fptri, "%f", &REVENUE[i][j] );
        fprintf(fptro, "%8.2f ", REVENUE[i][j] );
        }
        }

fprintf(fptro, "\n\nINTEREST RATE\n");
fscanf(fptri, "%f", &ACC );
fprintf(fptro, "ACC = %4.3f", ACC );

fprintf(fptro, "\n\nINFLATION RATE\n" );
fscanf(fptri, "%f", &inflation );
fprintf(fptro, "inflation = %4.3f", inflation );

fprintf(fptro, "\n\nREGULAR TIME COSTS");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read regular time cost */
        {
        fprintf(fptro,"\nRegular Time Cost for Production Work Center %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                {
                fscanf(fptri, "%f", &WLT[i] [j] );
                fprintf(fptro, "%4.2f\t", WLT[i] [j] );
                }
        }


fprintf(fptro, "\n\nPRINT OVER TIME COST");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read overtime cost */
        {
        fprintf(fptro,"\nOver Time Cost for Production Work Center %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
```

```
                    {
                    fscanf(fptri, "%f", &OLT [i] [j] );
                    fprintf(fptro, "%4.2f\t", OLT [i] [j] );
                    }
            }


    fprintf(fptro, "\n\nBEGINNING HOURS AT PRODUCTION WC'S\n");
    for ( i = 0; i < NUM_PROD_WC; i++ ) /* read beginning production time */
            {
            fscanf(fptri, "%f", &Bl [i] );
            fprintf(fptro, "%4.2f\t", Bl [i] );
            }

    fprintf(fptro, "\n\nBEGINNING UNITS OF CAPACITY AT NON-PRODUCTION WC'S\n");
    for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read beginning non-production time */
            {
            fscanf(fptri, "%f", &Bn [i] );
            fprintf(fptro, "%4.2f\t", Bn [i] );
            }

    fprintf(fptro, "\n\nINCREASES FROM NON-PRODUCTION CAPITAL BUDGTEING
PROJECT ONE\n");
    for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read increase from non-production cap bud
project 1 */
    {
    fprintf(fptro, "\nIncreases at WC%d\n", i );
            for ( j = 0; j < NUM_PERIODS; j++ )
            {
            fscanf(fptri, "%d", &CN [i][j] );
            fprintf(fptro, "%d  ", CN [i][j] );
            }
    }

    fprintf(fptro, "\n\nCOST OF PRODUCTION CAPITAL BUDGETING PROJECTS");
    for ( i = 0; i < NUM_CAP_BUD_PROJECTS; i++ ) /* read cost of each capital project */
            {
            fprintf(fptro,"\nCapital Budgeting Project %d\n", i );
            for ( j = 0; j < NUM_PERIODS; j++ )
                    {
                    fscanf(fptri, "%f", &cl [i] [j] );
                    fprintf(fptro, "%8.2f  ", cl [i] [j] );
                    }
            }

    fprintf(fptro, "\n\nCOST OF NON-PRODUCTION CAPITAL BUDGETING PROJECT\n");
    for ( i = 0; i < NUM_PERIODS; i++ ) /* read cost of capital project */
            {
            fscanf(fptri, "%f", &cn [i] );
            fprintf(fptro, "%8.2f ", cn [i] );
            }
```

```
        fprintf(fptro, "\n\nCOST OF SUPPORT CAPITAL BUDGETING PROJECT\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read cost of capital project */
                {
                fscanf(fptri, "%f", &cv [i] );
                fprintf(fptro, "%8.2f ", cv [i] );
                }


        fprintf(fptro, "\n\nMAXIMUM ALLOWABLE PRODUCTION PROJECT EXPENSE\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable production capital project
expense */
                {
                fscanf(fptri, "%f", &El [i] );
                fprintf(fptro, "%8.2f  ", El [i] );
                }

        fprintf(fptro, "\n\nMAXIMUM ALLOWABLE NON-PRODUCTION PROJECT EXPENSE\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable non-production capital project
expense */
                {
                fscanf(fptri, "%f", &En [i] );
                fprintf(fptro, "%8.2f ", En [i] );
                }

        fprintf(fptro, "\n\nMAXIMUM ALLOWABLE SUPPORT PROJECT EXPENSE\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable support capital project
expense */
                {
                fscanf(fptri, "%f", &Ev [i] );
                fprintf(fptro, "%8.2f ", Ev [i] );
                }

        fprintf(fptro, "\n\nMAXIMUM FRACTION REGULAR TIME LABOR\n");
        for ( i = 0; i < NUM_PROD_WC; i++ ) /* read maximum fraction regular time labor */
                {
                fprintf(fptro, "\nWC %d\n", i );
                for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                        {
                        fscanf(fptri, "%f", &LAMBDA[i][j] );
                        fprintf(fptro, "%4.2f\t", LAMBDA[i][j] );
                        }
                }

        fprintf(fptro, "\n\nPRODUCT LIFE LENGTH\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read product life lengths */
                {
                fscanf(fptri, "%d", &PLJ [i] );
                fprintf(fptro, "%d\t", PLJ [i] );
                }
```

```
        fprintf(fptro, "\n\nPRODUCT WARRANTEE LENGTH\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read product warrantee lengths */
                {
                fscanf(fptri, "%d", &PWJ [i] );
                fprintf(fptro, "%d\t", PWJ [i] );
                }

        fprintf(fptro, "\n\nSUPPORT HOURS FOR PRODUCTS OF AGE TAU");
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read support hours required for differring age
products */
                {
                fprintf(fptro,"\nSupport Hours for Product Number %d\n", i );
                for ( j = 0; j < PLJ[i]; j++ )
                        {
                        fscanf(fptri, "%f", &SJT [i] [j] );
                        fprintf(fptro, "%4.2f\t", SJT [i] [j] );
                        }
                }

        fprintf( fptro, "\n\nBEGINNING SUPPORT HOURS AVAILABLE\n" );
        fscanf(fptri, "%f", &S0 ); /* Read beginning support hours */
        fprintf(fptro, "%4.2f", S0 );

        fprintf(fptro, "\n\nMAXIMUM ALLOWABLE PRODUCTION COST\n");
        for ( i = 0; i < NUM_PROD_PERIODS; i++ ) /* read maximum allowable production cost */
                {
                fscanf(fptri, "%f", &PC [i] );
                fprintf(fptro, "%8.2f ", PC [i] );
                }

        fprintf(fptro, "\n\nAVERAGE PARTS COST TO REPAIR PRODUCT OF AGE TAU");
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read average parts cost to repair product of age tau */
                {
                fprintf(fptro,"\nAverage parts cost for Product Number %d\n", i );
                for ( j = 0; j < PLJ[i]; j++ )
                        {
                        fscanf(fptri, "%f", &pj [i] [j] );
                        fprintf(fptro, "%4.2f\t", pj [i] [j] );
                        }
                }

        fprintf(fptro, "\n\nMAXIMUM REPAIR PARTS COST\n");
        for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read maximum repair parts cost */
                {
                fscanf(fptri, "%f", &PT [i] );
                fprintf(fptro, "%8.2f ", PT [i] );
                }

        fprintf(fptro, "\n\nPARTS REVENUE FROM REPAIR OF PRODUCT OF AGE TAU");
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read average parts cost to repair product of age tau */
                {
```

```
        fprintf(fptro,"\nAverage parts revenue for Product Number %d\n", i );
        for ( j = 0; j < PLJ[i]; j++ )
                {
                fscanf(fptri, "%f", &PR [i] [j] );
                fprintf(fptro, "%4.2f\t", PR [i] [j] );
                }
        }

fprintf(fptro, "\n\nMINIMUM ACCEPTABLE REVENUE\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read minimum acceptable revenue */
        {
        fscanf(fptri, "%f", &RT [i] );
        fprintf(fptro, "%8.2f ", RT [i] );
        }

fprintf(fptro, "\n\nREPAIR LABOR REVENUE PER HOUR\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read repair labor revenue per hour */
        {
        fscanf(fptri, "%f", &ft [i] );
        fprintf(fptro, "%4.2f  ", ft [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE PRODUCTION WC\n");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read base cost to operate production wc's */
        {
        fscanf(fptri, "%f", &BASE_COST_L [i] );
        fprintf(fptro, "%f\t", BASE_COST_L [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE NON-PRODUCTION WC\n");
for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read base cost to operate non-production wc's */
        {
        fscanf(fptri, "%f", &BASE_COST_N [i] );
        fprintf(fptro, "%f\t", BASE_COST_N [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE SUPPORT WC\n");
/* read base cost to operate support wc */
fscanf(fptri, "%f", &BASE_COST_V );
fprintf(fptro, "%f\t", BASE_COST_V );

fprintf(fptro, "\n\nMATERIAL COST FOR PRODUCTION");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read material cost for production */
        {
        fprintf(fptro,"\nMaterial cost for Product Number %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                {
                fscanf(fptri, "%f", &MA [i] [j] );
                fprintf(fptro, "%4.2f\t", MA [i] [j] );
                }
        }
```

```
        fprintf(fptro, "\n\nREPAIR LABOR COST\n");
        for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read repair labor cost */
                {
                fscanf(fptri, "%f", &RC [i] );
                fprintf(fptro, "%4.2f  ", RC [i] );
                }

        fprintf(fptro, "\n\nCONSISTENT UNITS REQUIRED BY PRODUCT AT NON_PRODUCTION
WC");
        for ( k = 0; k < NUM_NON_PROD_WC; k++ )
        {
        fprintf(fptro, "\n\nUnits required at wc %d", k );
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read units required for non-production */
                {
                fprintf(fptro,"\n  Consistent non-production units for product number %d\n", i );
                fprintf(fptro, "  ");
                for ( j = 0; j < NUM_PERIODS; j++ )
                        {
                        fscanf(fptri, "%f", &G [k] [i] [j] );
                        fprintf(fptro, "%4.2f  ", G [k] [i] [j] );
                        }
                }
        }

        fprintf(fptro, "\n\nMAXIMUM PRODUCTION PERIODS FOR EACH PRODUCT\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                {
                fscanf(fptri, "%d", &max_periods[i] );
                fprintf(fptro, "%d\t", max_periods[i] );
                }

        fprintf(fptro, "\n\nEARLIEST PERIOD PRODUCTION OF A PRODUCT MAY
COMMENCE\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                {
                fscanf(fptri, "%d", &start_period[i] );
                fprintf(fptro, "%d\t", start_period[i] );
                }

        fprintf(fptro, "\n\nINVENTORY HOLDING COSTS FOR PRODUCTS IN EACH PERIOD\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
                fprintf(fptro, "\nPRODUCT %d\n", i );
                for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                {
                fscanf(fptri, "%f", &INVENTORY[i][j] );
                fprintf(fptro, "%8.2f  ", INVENTORY[i][j] );
                }
        }
```

```
        fprintf(fptro, "\n\nIMPROVMENT IN HOURS FROM SUPPORT CAP BUD PROJECT\n");
        for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ )
                {
                fscanf(fptri, "%d", &CV[i] );
                fprintf(fptro, "%d  ", CV[i] );
                }

        fprintf(fptro, "\n\nBEGINNING INVENTORY FOR EACH PRODUCT\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                {
                fscanf(fptri, "%d", &BEGIN_INVENTORY[i] );
                fprintf(fptro, "%d\t", BEGIN_INVENTORY[i] );
                }

        fclose(fptri);

        single( d1, d2, CL, SETUP_HOURS, PRODUCTION_HOURS, REVENUE,
        WLT, OLT, Bl, Bn, CN, cl, cn, cv,
        El, En, Ev, LAMBDA, SJT, S0, PC, PLJ, PWJ, pj, PT,
        PR, RT, ft, MA, RC, G, max_periods, start_period,
        BASE_COST_L, BASE_COST_N, BASE_COST_V, ACC, INVENTORY, CV,
        inflation, BEGIN_INVENTORY );

        fclose(fptro);


}
```

```
/* constrnt.h */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define NUM_PERIODS                     12
#define HORIZON_PERIOD                  11
#define NUM_SUPPORT_PERIODS             11
#define NUM_PROD_PERIODS                8
#define NUM_PRODUCTS                    5
#define NUM_PROD_WC                     3
#define NUM_NON_PROD_WC                 4
#define NUM_CAP_BUD_PROJECTS            3
#define NUM_NON_PROD_CAP_BUD_PROJ       1
#define AGE                             5
#define TRUE                            1
#define FALSE                           0


int project, i, j;

int setup_constraint;
int dec_var;
int age;
int inv_lim;
int must_fund;
int nprdwc;


/* base cost variables */
float TOT_BASE_COST;
float TOT_BASE_DISCOUNT;

/* discounted revenues from repair labor */
float ADJ_LABOR_REP_REV[NUM_PRODUCTS][NUM_PROD_PERIODS];

/* discounted cost from repair labor */
float ADJ_LABOR_REP_COST[NUM_PRODUCTS][NUM_PERIODS];

/* discounted parts revenues less costs */
float ADJ_PART[NUM_PRODUCTS][NUM_PERIODS];

/* production capital budgeting costs */
float TOTAL_CAP_BUD_EXPENSE[NUM_CAP_BUD_PROJECTS];

/* non-production capital budgeting expense */
float TOTAL_NP_CAP_BUD_EXPENSE;

/* adjusted inventory cost; includes stocking fee, loss of interest on
  revenues and repair costs */
float ACTUAL_INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS - 1];
```

```
/* total support capital budgeting cost */
float TOTAL_SUPPORT_CAP_BUD_COST;
```

```
/* constrnt.c */
#include "constrnt.h"

/* Builds an MPS file for the concurrent model */
void single( int d1[NUM_PROD_PERIODS], int d2[NUM_PROD_PERIODS],
int CL[NUM_CAP_BUD_PROJECTS][NUM_PROD_WC][NUM_PROD_PERIODS],
int SETUP_HOURS[NUM_PROD_WC] [NUM_PRODUCTS],
int PRODUCTION_HOURS[NUM_PROD_WC] [NUM_PRODUCTS],
float REVENUE [NUM_PRODUCTS][NUM_PROD_PERIODS],
float WLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float OLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float Bl [NUM_PROD_WC], float Bn [NUM_NON_PROD_WC],
int CN [NUM_NON_PROD_WC][NUM_PERIODS],
float cl [NUM_CAP_BUD_PROJECTS] [NUM_PERIODS], float cn [NUM_PERIODS],
float cv [NUM_PERIODS],
float El [NUM_PERIODS], float En [NUM_PERIODS], float Ev [NUM_PERIODS],
float LAMBDA [NUM_PROD_WC][NUM_PROD_PERIODS], float SJT [NUM_PRODUCTS][AGE],
float S0, float PC [NUM_PROD_PERIODS], int PLJ [NUM_PRODUCTS],
int PWJ [NUM_PRODUCTS], float pj [NUM_PRODUCTS][AGE], float PT [NUM_PERIODS],
float PR [NUM_PRODUCTS][AGE], float RT [NUM_PERIODS], float ft [NUM_PERIODS],
float MA [NUM_PRODUCTS][NUM_PROD_PERIODS], float RC [NUM_PERIODS],
float G [NUM_NON_PROD_WC][NUM_PRODUCTS][NUM_PERIODS],
int max_periods[NUM_PRODUCTS],
int start_period[NUM_PRODUCTS], float BASE_COST_L[NUM_PROD_WC],
float BASE_COST_N[NUM_NON_PROD_WC],
float BASE_COST_V, float ACC,
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS], int CV [NUM_PERIODS - 1],
float inflation, int BEGIN_INVENTORY[NUM_PRODUCTS] )

{
FILE *mps;

register product, period, wc;

        /* open MPS output file */

        if ( (mps = fopen("mps.dat","w")) == NULL ) /* open mps output file */
                {
                 printf("Can't open the mps output file.");
                 exit(0);
                }

        /* build total non-production capital budgeting costs */
        TOTAL_NP_CAP_BUD_EXPENSE = 0;
        for ( period = 0; period < NUM_PERIODS; period++ )
                TOTAL_NP_CAP_BUD_EXPENSE += cn[period] * pow( 1 + ACC,
HORIZON_PERIOD - period );


        /* build total production capital budgeting costs */
        for ( project = 0; project < NUM_CAP_BUD_PROJECTS; project++ )
```

```
        {
                for ( period = 0; period < NUM_PERIODS; period++ )
                {
                TOTAL_CAP_BUD_EXPENSE[project] = 0;
                }
        }
        for ( project = 0; project < NUM_CAP_BUD_PROJECTS; project++ )
        {
                for ( period = 0; period < NUM_PERIODS; period++ )
                {
                TOTAL_CAP_BUD_EXPENSE[project] += cl[project][period] * pow( 1 + ACC,
HORIZON_PERIOD - period );
                }
        }

        /* build total future worth of repair revenues and costs  for each product by
        period it was produced */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                ADJ_LABOR_REP_REV[product][period] = 0;
                ADJ_LABOR_REP_COST[product][period] = 0;
                ADJ_PART[product][period] = 0;
                }
        }
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period =  start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                for ( age = PWJ[product]; age < PLJ[product]; age++ )
                ADJ_LABOR_REP_REV[product][period] += SJT[product][age]*ft[age] * pow( 1 +
ACC, HORIZON_PERIOD - period - age );
                }
        }
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period =  start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                for ( age = 0; age < PLJ[product]; age++ )
                ADJ_LABOR_REP_COST[product][period] += SJT[product][age]*RC[age] * pow( 1 +
ACC, HORIZON_PERIOD - period - age );
                }
        }
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period =  start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
```

```
                for ( age = 0; age < PLJ[product]; age++ )
                {
                if ( age < PWJ[product] )
                        ADJ_PART[product][period] += - pj[product][age] * pow( 1 + inflation, period
+ age ) * pow( 1 + ACC, HORIZON_PERIOD - ( period + age ) );
                if ( age >= PWJ[product] )
                        ADJ_PART[product][period] += ( PR[product][age] - pj[product][age] ) * pow(
1 + inflation, period + age ) * pow( 1 + ACC, HORIZON_PERIOD - ( period + age ) );
                }
                }
        }

        /* actual inventory cost - includes stocking fee, loss of interest on
        revenues, and adjusts repair costs and revenues */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product] - 1; period++ )
                {
                ACTUAL_INVENTORY[product][period] = ( INVENTORY[product][period]
                + REVENUE[product][period] ) * pow( 1 + ACC, HORIZON_PERIOD - period )
                + ADJ_LABOR_REP_REV[product][period] -
                ADJ_LABOR_REP_COST[product][period] +
                ADJ_PART[product][period] -
                ( REVENUE[product][period + 1] * pow( 1 + ACC, HORIZON_PERIOD - period - 1)
                + ADJ_LABOR_REP_REV[product][period + 1 ] -
                ADJ_LABOR_REP_COST[product][period + 1] +
                ADJ_PART[product][period + 1] );
                }
        }

        /* determine total support capital budget cost */
        TOTAL_SUPPORT_CAP_BUD_COST = 0;
        for ( period = 0; period < NUM_PERIODS; period++ )
                TOTAL_SUPPORT_CAP_BUD_COST += cv[period] * pow( 1 + ACC,
HORIZON_PERIOD - period );

        /* calculate total base costs at production and non-production wc's */
        TOT_BASE_COST = 0;
        TOT_BASE_DISCOUNT = 0;
        /* determine total base production cost in each period */
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                TOT_BASE_COST += BASE_COST_L[wc];
        /* add total base non-production cost to base production cost */
        for ( wc = 0; wc < NUM_NON_PROD_WC; wc++ )
                TOT_BASE_COST += BASE_COST_N[wc];

        /* add base cost of support to other base costs */
        TOT_BASE_COST += BASE_COST_V;

        /* Determine total discounted constant dollar revenues less costs */
```

```
        for ( period = 0; period < NUM_PERIODS; period++ )
        {
        TOT_BASE_DISCOUNT += TOT_BASE_COST * pow( 1 + inflation, period ) * pow( 1 +
ACC, HORIZON_PERIOD - period );
        }


        /* print header to name LP */
        fprintf(mps, "NAME   CONCURRENT MODEL MPS FILE (MAX)\n");
        /* print header to start rows */
        fprintf(mps, "ROWS\n");
        /* develop row for the objective function */
        fprintf(mps, " N  obj\n");

        /* production capital budgeting project mutually exclusive row */
        fprintf(mps, " L  capmutex\n");

        /* non-production capital budget maximum cost per period */
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, " L  mxbudn%d\n", period );

        /* production capital budget maximum cost per period */
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, " L  mxbudl%d\n", period );

        /* support capital budget maximum cost per period */
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, " L  mxbudv%d\n", period );

        /* develop rows for product 0 - 1 (Zi) decision variable constraints */
        dec_var = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                dec_var++;
                fprintf(mps, " L  decvar%d\n", dec_var);
                }
        }

        /* develop rows for pre-production/post-support wc capacity */
        product = 0;
        for ( period = 0; period < NUM_PERIODS; period++ )
                {
                for ( wc = 0; wc < NUM_NON_PROD_WC; wc++ )
                        {
                        while ( product < NUM_PRODUCTS )
                        {
                        if ( G[wc][product][period] > 0 )
                                {
```

```
                        fprintf(mps, " L  npdwc%d%d\n", period, wc );
                        product = NUM_PRODUCTS;
                        }
                else
                        product++;
                }
                product = 0;
                }
        }

        /* develop rows for setup constraints */
        setup_constraint = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                setup_constraint++;
                fprintf(mps, " L  setup%d\n", setup_constraint);
                }
        }

        /* develop rows to set upper limits on inventory */
        inv_lim = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product] - 1; period++ )
                {
                inv_lim++;
                fprintf(mps, " L  invlim%d\n", inv_lim);
                }
        }

        /* develop rows for product demand */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
        fprintf(mps, " L  demand1%d\n",  period );
        }
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
        fprintf(mps, " L  demand2%d\n", period );
        }

        /* develop rows for prodwc constraint */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                {
                fprintf(mps, " L  prodwc%d%d\n", period, wc );
                }
```

```
}

/* develop values of WLT and OLT */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
{
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
        {
        fprintf(mps, " L  time%d%d\n", period, wc );
        }
}

/* develop maximum values of WLT */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
{
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
        {
        fprintf(mps, " L  regtm%d%d\n", period, wc );
        }
}

/* develop maximum values of OLT */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
{
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
        {
        fprintf(mps, " L  otime%d%d\n", period, wc );
        }
}

/* develop maximum production cost in each period */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        fprintf(mps, " L  mxprcst%d\n", period );

/* develop maximum support hours in each period rows */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, " L  prdsup%d\n", period );

/* develop maximum support parts cost in each period */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period ++ )
        fprintf(mps, " L  prtsup%d\n", period );

/* develop minimum revenue for each period */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, " G  minrev%d\n", period );

/* develop product must fund rows */
for ( must_fund = 0; must_fund < 2; must_fund++ )
        fprintf(mps, " E  mustfnd%d\n", must_fund );

/* develop rows for product mutual exclusivity */
fprintf(mps, " L  prodmutx\n");
```

```
        /* print basecost constraint row */
        fprintf(mps, " E  basecost\n");

        /* set base capacity to Bl */
        fprintf(mps, " E  basecap\n");

        /* set beginning inventory variables equal to one */
        i = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
                {
                if ( BEGIN_INVENTORY[product] > 0 )
                        {
                        i++;
                        fprintf(mps, " E  beginv%d\n", i );
                        }
                }



        /* print header to start columns */
        fprintf(mps, "COLUMNS\n");

        /* print product funding variables */
        dec_var = 0;
        must_fund = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {

                fprintf(mps, "    Z%d       obj       -0.001\n", product );
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                dec_var++;
                if ( dec_var < 10 && product < 3 )
                {
                fprintf(mps, "    Z%d       decvar%d    -%d\n", product, dec_var, d1[period]*3 );
                }
                if ( dec_var >= 10 && product < 3)
                {
                fprintf(mps, "    Z%d       decvar%d    -%d\n", product, dec_var, d1[period]*3 );
                }
                if ( dec_var < 10 && product >= 3 )
                {
                fprintf(mps, "    Z%d       decvar%d    -%d\n", product, dec_var, d2[period]*3 );
                }
                if ( dec_var >= 10 && product >= 3)
                {
                fprintf(mps, "    Z%d       decvar%d    -%d\n", product, dec_var, d2[period]*3 );
                }
                }
```

```
                /* develop rows for pre-production/post-support wc capacity */
                for ( period = 0; period < NUM_PERIODS; period++ )
                        {
                        for ( wc = 0; wc < NUM_NON_PROD_WC; wc++ )
                                {
                                if ( G[wc][product][period] > 0 )
                                        {
                                        fprintf(mps, "    Z%d       npdwc%d%d       %-12.2f\n",
product, period, wc, G[wc][product][period] );
                                        }
                                }
                        }


                /* develop must fund columns */
                if ( product == 0 || product == 3 )
                        {
                        fprintf(mps, "    Z%d       mustfnd%d    1.0\n", product, must_fund );
                        must_fund++;
                        }
                if ( product == 1 || product == 2 )
                        fprintf(mps, "    Z%d       prodmutx     1.0\n", product );
        }

        /* print production work center setup variables */
        setup_constraint = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                fprintf(mps, "    DELTA%d%-4dobj       -0.001\n", product, period );
                setup_constraint++;
                if ( setup_constraint < 10 && product < 3 )
                {
                fprintf(mps, "    DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d1[period] * 3 );
                }
                if ( setup_constraint >= 10 && product < 3)
                {
                fprintf(mps, "    DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d1[period] * 3 );
                }
                if ( setup_constraint < 10 && product >= 3 )
                {
                fprintf(mps, "    DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d2[period] * 3 );
                }
                if ( setup_constraint >= 10 && product >= 3)
                {
```

```
                fprintf(mps, "    DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d2[period] * 3 );
                }
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "    DELTA%d%-4dprodwc%d%d%12d\n", product, period,
period, wc, SETUP_HOURS[wc][product] );
                        }
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "    DELTA%d%-4dtime%d%d  %12d\n", product, period, period,
wc, SETUP_HOURS[wc][product] );
                        }
                }
        }

        /* print non-production capital budgeting variable */
        fprintf(mps, "    EN       obj       -%-12.2f\n", TOTAL_NP_CAP_BUD_EXPENSE );
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, "    EN       mxbudn%d     %-12.2f\n", period, cn[period] );
        product = 0;
        for ( period = 0; period < NUM_PERIODS; period++ )
        {
                for ( wc = 0; wc < NUM_NON_PROD_WC; wc++ )
                {
                while ( product < NUM_PRODUCTS )
                        {
                        if ( G[wc][product][period] > 0 )
                                {
                                fprintf(mps, "    EN       npdwc%d%d     -%-12d\n", period, wc,
CN[wc][period] );
                                product = NUM_PRODUCTS;
                                }
                        else
                                product++;
                        }
                product = 0;
                }
        }

        /* print production capital budegting variables */
        for ( project = 0; project < NUM_CAP_BUD_PROJECTS; project++ )
        {
                fprintf(mps, "    EL%d     obj       -%-12.2f\n", project,
TOTAL_CAP_BUD_EXPENSE[project] );
                if ( project == 0 || project == 2 )
                fprintf(mps, "    EL%d     capmutex       1.0\n", project );
                for ( period = 0; period < NUM_PERIODS; period++ )
                        fprintf(mps, "    EL%d     mxbudl%d     %-12.2f\n", project, period,
cl[project][period] );
                for ( period = 0; period < NUM_PROD_PERIODS; period++ )
```

```
                {
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "   EL%d      prodwc%d%d    -%-12d\n", project, period, wc,
CL[project][wc][period] );
                        fprintf(mps, "   EL%d      regtm%d%d     -%-12d\n", project, period, wc,
CL[project][wc][period] );
                        fprintf(mps, "   EL%d      otime%d%d     -%-12d\n", project, period, wc,
CL[project][wc][period] );
                        }
                }
        }

        /* print support capital budgeting project variable */
        fprintf(mps, "   EV       obj       -%-12.2f\n", TOTAL_SUPPORT_CAP_BUD_COST );
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, "   EV       mxbudv%d     %-12.2f\n", period, cv[period] );

        for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
                fprintf(mps, "   EV       prdsup%d     -%-12d\n", period, CV[period] );

        /* print production variables */
        setup_constraint = 0;
        dec_var = 0;
        inv_lim = 1;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
                {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                fprintf(mps, "   X%d%-4d   obj       %-12.2f\n", product, period, (
REVENUE[product][period] - MA[product][period] ) * pow( 1 + ACC, HORIZON_PERIOD - period ) +
ADJ_LABOR_REP_REV[product][period] - ADJ_LABOR_REP_COST[product][period] +
ADJ_PART[product][period] );
                dec_var++;
                fprintf(mps, "   X%d%-4d   decvar%d          1.0\n", product, period, dec_var );
                setup_constraint++;
                if ( setup_constraint < 10 )
                {
                fprintf(mps, "   X%d%-4d   setup%d           1.0\n", product, period,
setup_constraint);
                }
                if ( setup_constraint >= 10 )
                {
                fprintf(mps, "   X%d%-4d   setup%d           1.0\n", product, period, setup_constraint);
                }
                if ( inv_lim >= 10 && period < start_period[product] + max_periods[product] - 1 )
                {
                fprintf(mps, "   X%d%-4d   invlim%d      -1.0\n", product, period, inv_lim);
                inv_lim++;
                }
```

```
                if ( inv_lim < 10 && period < start_period[product] + max_periods[product] - 1 )
                {
                fprintf(mps, "    X%d%-4d    invlim%d        -1.0\n", product, period, inv_lim);
                inv_lim++;
                }
                if ( product < 3 )
                {
                fprintf(mps, "    X%d%-4d    demand1%d          1.0\n", product, period, period );
                }
                if ( product >= 3 )
                {
                fprintf(mps, "    X%d%-4d    demand2%d          1.0\n", product, period, period );
                }
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "    X%d%-4d    prodwc%d%d%12d\n", product, period, period,
wc, PRODUCTION_HOURS[wc][product] );
                        }
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "    X%d%-4d    time%d%d  %12d\n", product, period, period, wc,
PRODUCTION_HOURS[wc][product] );
                        }
                fprintf(mps, "    X%d%-4d    mxprcst%d      %-12.2f\n", product, period, period,
MA[product][period] );
                for ( age = 0; age < PLJ[product]; age++ ) /* support hours in each period */
                        fprintf(mps, "    X%d%-4d    prdsup%d       %-12.2f\n", product, period, period
+ age, SJT[product][age] );
                for ( age = 0; age < PLJ[product]; age++ ) /* support cost in each period */
                        fprintf(mps, "    X%d%-4d    prtsup%d       %-12.2f\n", product, period, period
+ age, pj[product][age] * pow( 1 + inflation, period + age ) );
                for ( age = 0; age < PLJ[product]; age++ ) /* revenues in each period */
                        {
                        if ( age == 0 )
                                fprintf(mps, "    X%d%-4d    minrev%d       %-12.2f\n", product,
period, period, PR[product][age] * pow( 1 + inflation, period ) + ft[period] * SJT[product][age] +
REVENUE[product][period] );
                        else
                                fprintf(mps, "    X%d%-4d    minrev%d       %-12.2f\n", product,
period, period + age, PR[product][age] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age] );
                        }
                }
        }
        /* print inventory variables */
        inv_lim = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
                {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product] - 1; period++ )
                {
```

```
                fprintf(mps, "   I%d%-4d   obj          -%-12.2f\n", product, period,
ACTUAL_INVENTORY[product][period] );
                inv_lim++;
                if ( inv_lim < 10 )
                {
                fprintf(mps, "   I%d%-4d   invlim%d        1.0\n", product, period, inv_lim);
                }
                if ( inv_lim >= 10 )
                {
                fprintf(mps, "   I%d%-4d   invlim%d       1.0\n", product, period, inv_lim);
                }
                if ( product < 3 )
                {
                fprintf(mps, "   I%d%-4d   demand1%d         -1.0\n", product, period, period );
                fprintf(mps, "   I%d%-4d   demand1%d          1.0\n", product, period, period + 1 );
                }
                if ( product >= 3 )
                {
                fprintf(mps, "   I%d%-4d   demand2%d         -1.0\n", product, period, period );
                fprintf(mps, "   I%d%-4d   demand2%d          1.0\n", product, period, period + 1 );
                }
                for ( age = 0; age < PLJ[product]; age++ ) /* adj period support hours */
                {
                        if ( age == 0 )
                                {
                                fprintf(mps, "   I%d%-4d   prdsup%d     -%-12.2f\n", product,
period, period, SJT[product][age] );
                                }
                        if ( age > 0 && age < PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   prdsup%d      %-12.2f\n", product,
period, period + age, SJT[product][age - 1] - SJT[product][age] );
                                }
                        if ( age == PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   prdsup%d      %-12.2f\n", product,
period, period + age, SJT[product][age - 1] - SJT[product][age] );
                                fprintf(mps, "   I%d%-4d   prdsup%d      %-12.2f\n", product,
period, period + age + 1, SJT[product][age] );
                                }
                }
                for ( age = 0; age < PLJ[product]; age++ ) /* adj period part sup cost */
                {
                        if ( age == 0 )
                                {
                                fprintf(mps, "   I%d%-4d   prtsup%d    -%-12.2f\n", product, period,
period, pj[product][age] * pow( 1 + inflation, period ) );
                                }
                        if ( age > 0 && age < PLJ[product] - 1 )
                                {
```

```
                                fprintf(mps, "    I%d%-4d    prtsup%d     %-12.2f\n", product, period,
period + age, pj[product][age - 1] * pow( 1 + inflation, period + age ) - pj[product][age] * pow( 1 +
inflation, period + age ) );
                                }
                        if ( age == PLJ[product] - 1 )
                                {
                                fprintf(mps, "    I%d%-4d    prtsup%d     %-12.2f\n", product, period,
period + age, pj[product][age - 1] * pow( 1 + inflation, period + age ) - pj[product][age] * pow( 1 +
inflation, period + age ) );
                                fprintf(mps, "    I%d%-4d    prtsup%d     %-12.2f\n", product, period,
period + age + 1, pj[product][age] * pow( 1 + inflation, period + age + 1 ) );
                                }
                }
                for ( age = 0; age < PLJ[product]; age++ ) /* adj period revenue */
                        {
                        if ( age == 0 )
                                {
                                fprintf(mps, "    I%d%-4d    minrev%d      -%-12.2f\n", product,
period, period, PR[product][age] * pow( 1 + inflation, period ) + ft[period] * SJT[product][age] +
REVENUE[product][period] );
                                }
                        if ( age == 1 )
                                {
                                fprintf(mps, "    I%d%-4d    minrev%d       %-12.2f\n", product,
period, period + age, PR[product][age - 1] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age - 1] + REVENUE[product][period] * pow( 1 + inflation, period ) - PR[product][age] *
pow( 1 + inflation, period + age ) - ft[period + age] * SJT[product][age] );
                                }
                        if ( age > 1 && age < PLJ[product] - 1 )
                                {
                                fprintf(mps, "    I%d%-4d    minrev%d       %-12.2f\n", product,
period, period + age, PR[product][age - 1] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age - 1] - PR[product][age] * pow( 1 + inflation, period + age ) - ft[period + age] *
SJT[product][age] );
                                }
                        if ( age == PLJ[product] - 1 )
                                {
                                fprintf(mps, "    I%d%-4d    minrev%d       %-12.2f\n", product,
period, period + age, PR[product][age - 1] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age - 1] - PR[product][age] * pow( 1 + inflation, period + age ) - ft[period + age] *
SJT[product][age] );
                                fprintf(mps, "    I%d%-4d    minrev%d       %-12.2f\n", product,
period, period + age + 1, PR[product][age] * pow( 1 + inflation, period + age + 1) + ft[period + age + 1] *
SJT[product][age] );
                                }
                        }

                }
        }

        /* print regular time labor variables */
```

```
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                {
                fprintf(mps, "   WLT%d%-4d obj        -%-12.2f\n", period, wc, WLT[wc][period] *
pow( 1 + ACC, HORIZON_PERIOD - period ) );
                fprintf(mps, "   WLT%d%-4d time%d%d      -1.0\n", period, wc, period, wc );
                fprintf(mps, "   WLT%d%-4d regtm%d%d      %-12.4f\n", period, wc, period, wc,
1/LAMBDA[wc][period] );
                fprintf(mps, "   WLT%d%-4d mxprcst%d     %-12.2f\n", period, wc, period,
WLT[wc][period] );
                }
        }

        /* print over time labor variables */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                {
                fprintf(mps, "   OLT%d%-4d obj        -%-12.2f\n", period, wc, OLT[wc][period] *
pow( 1 + ACC, HORIZON_PERIOD - period ) );
                fprintf(mps, "   OLT%d%-4d time%d%d      -1.0\n", period, wc, period, wc );
                fprintf(mps, "   OLT%d%-4d otime%d%d      %-12.4f\n", period, wc, period, wc, 1/( 1
- LAMBDA[wc][period] ) );
                fprintf(mps, "   OLT%d%-4d mxprcst%d     %-12.2f\n", period, wc, period,
OLT[wc][period] );
                }
        }


        /* print base costs constraint */
        fprintf(mps, "   BASECOST obj       -%-12.3f\n", TOT_BASE_DISCOUNT );
        fprintf(mps, "   BASECOST basecost      1.0\n");

        /* print base production capacity */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                {
                fprintf(mps, "   Bl      regtm%d%d   -%-12.3f\n", period, wc, Bl[wc] );
                fprintf(mps, "   Bl      otime%d%d   -%-12.3f\n", period, wc, Bl[wc] );
                }
        }
        fprintf(mps, "   Bl      basecap       1.0\n");

        /* print beginning inventory variables */
        i = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                if ( BEGIN_INVENTORY[product] > 0 )
                {
```

```
                if ( product < 3 )
                {
                fprintf(mps, "   BI%d      obj       %-12.2f\n", product, REVENUE[product][0] * pow(
1 + ACC, HORIZON_PERIOD ) );
                fprintf(mps, "   BI%d      demand10   %-12d\n", product,
BEGIN_INVENTORY[product] );
                }
                if ( product >= 3 )
                {
                fprintf(mps, "   BI%d      obj       %-12.2f\n", product, REVENUE[product][0] * pow(
1 + ACC, HORIZON_PERIOD ) );
                fprintf(mps, "   BI%d      demand20   %-12d\n", product,
BEGIN_INVENTORY[product] );
                }
                i++;
                fprintf(mps, "   BI%d      beginv%d   1.0\n", product, i );
                }
        }



        /* print header to start rows */
        fprintf(mps, "RHS\n");

        /* print setup right hand sides */
        fprintf(mps, "   RHS      capmutex      1.0\n");

        /* print non-production capital budget maximum cost per period */
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, "   RHS      mxbudn%d   %-12.2f\n", period, En[period] );

        /* print maximum production capital budget expense RHS */
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, "   RHS      mxbudl%d   %-12.2f\n", period, El[period] );

        /* print support capital budget maximum cost per period */
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, "   RHS      mxbudv%d   %-12.2f\n", period, Ev[period] );

        /* develop maximum pre-production/post-support wc capacity RHS */
        product = 0;
        for ( period = 0; period < NUM_PERIODS; period++ )
                {
                for ( wc = 0; wc < NUM_NON_PROD_WC; wc++ )
                        {
                        while ( product < NUM_PRODUCTS )
                        {
                        if ( G[wc][product][period] > 0 )
                                {
                                fprintf(mps, "   RHS      npdwc%d%d    %-12.2f\n", period, wc,
Bn[wc] );
```

```
                        product = NUM_PRODUCTS;
                        }
                else
                        product++;
                }
                product = 0;
                }
        }

/* print maximum production (demand) RHS */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
{
fprintf(mps, "   RHS     demand1%d       %d\n", period, d1[period] );
}
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
{
fprintf(mps, "   RHS     demand2%d       %d\n", period, d2[period] );
}

/* print production work center RHS */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
{
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
        {
        fprintf(mps, "   RHS     prodwc%d%d      %-13.2f\n", period, wc, Bl [wc] );
        }
}

/* print maximum production cost RHS */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        fprintf(mps, "   RHS     mxprcst%d     %-12.2f\n", period, PC[period] );

/* print maximum support hours RHS */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, "   RHS     prdsup%d      %-12.2f\n", period, S0 );

/* print maximum support parts cost RHS */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, "   RHS     prtsup%d      %-12.2f\n", period, PT[period] );

/* print minimum revenue for each period RHS */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, "   RHS     minrev%d      %-12.2f\n", period, RT[period] );

/* print must fund RHS */
for ( must_fund = 0; must_fund < 2; must_fund++ )
        fprintf(mps, "   RHS     mustfnd%d     1.0\n", must_fund );

fprintf(mps, "   RHS     prodmutx      1.0\n");
fprintf(mps, "   RHS     basecost      1.0\n");
fprintf(mps, "   RHS     basecap       1.0\n");
```

```
        /* print beginning inventory RHS */
        i = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
        if ( BEGIN_INVENTORY[product] > 0 )
                {
                i++;
                fprintf(mps, "    RHS       beginv%d       1.0\n", i );
                }
        }

        fprintf(mps, "ENDATA");
        fclose(mps);

} /* end main */
```

APPENDIX C: Program Listing for Stage One of the Sequential Model

```
/* frstmain.h */
/* header file for frstmain.c */

#include <stdio.h>
#include <stdlib.h>
#define NUM_PERIODS                     12
#define NUM_SUPPORT_PERIODS             11
#define NUM_PROD_PERIODS                8
#define NUM_PRODUCTS                    5
#define NUM_PROD_WC                     3
#define NUM_NON_PROD_WC                 4
#define NUM_CAP_BUD_PROJECTS            3
#define NUM_NON_PROD_CAP_BUD_PROJ       1
#define AGE                             5 /* maximum life length of any product */

typedef unsigned int BitVector;

int k;
        /* INPUT PARAMETERS */
/* DEMAND */
int d1 [NUM_PROD_PERIODS]; /* Demand for subgroup one */
int d2 [NUM_PROD_PERIODS]; /* Demand for subgroup two */

/* PRODUCT AND PROJECT INTERACTION */
BitVector Mutually_Exclusive_Prod[NUM_PRODUCTS]; /* Mutually exclusive products */
BitVector Mutually_Exclusive_Proj[NUM_CAP_BUD_PROJECTS]; /* " " projects */
BitVector Contingency[NUM_PRODUCTS]; /* Contingency between products */
BitVector Must_Fund; /* Must fund a set of products */

/* INTEREST RATES */
float ACC;  /* Interest Rate */
float inflation; /* Inflation Rate */

/* PRODUCT REVENUES */
float REVENUE [NUM_PRODUCTS][NUM_PROD_PERIODS]; /* Product revenue */
float RT [NUM_SUPPORT_PERIODS]; /* Minimum acceptable revenue */

/* PRE-PRODUCTION/POST-SUPPORT PRODUCT INPUTS */
float BASE_COST_N[NUM_NON_PROD_WC]; /* Base cost to operate non-prod wc's */
float Bn[NUM_NON_PROD_WC]; /* Beginning units of capacity at non-production wc's */
float G[NUM_NON_PROD_WC][NUM_PRODUCTS][NUM_PERIODS]; /* No. consistent units req by
product at non-production wc's  */

/* LABOR INPUTS */
float WLT [NUM_PROD_WC] [NUM_PROD_PERIODS]; /* reg time cost by wc */
float OLT [NUM_PROD_WC] [NUM_PROD_PERIODS]; /* ot time cost by wc */
float LAMBDA[NUM_PROD_WC][NUM_PROD_PERIODS]; /* Fraction labor regular time */

/* PRODUCTION INPUTS */
float Bl[NUM_PROD_WC]; /* Beginning hours at production wc's */
float BASE_COST_L[NUM_PROD_WC]; /* Base cost to operate production wc's */
```

```
int SETUP_HOURS[NUM_PROD_WC] [NUM_PRODUCTS]; /* setup hours for prod by wc */
int PRODUCTION_HOURS[NUM_PROD_WC] [NUM_PRODUCTS]; /* production hrs by prod & wc */
float PC[NUM_PROD_PERIODS]; /* Maximum allowable production cost */
float MA[NUM_PRODUCTS][NUM_PROD_PERIODS]; /* Material cost for production */

/* SUPPORT VARIABLES */
float BASE_COST_V; /* Base support cost */
float S0; /* Beginning support hours available */
float SJT[NUM_PRODUCTS][AGE]; /* Support hours required for product of different ages */
int PLJ[NUM_PRODUCTS]; /* Product life length */
int PWJ[NUM_PRODUCTS]; /* Warrantee length */
float pj[NUM_PRODUCTS][AGE]; /* Average parts cost */
float PT[NUM_SUPPORT_PERIODS]; /* Maximum repair parts cost */
float PR[NUM_PRODUCTS][AGE]; /* Parts revenue */
float ft[NUM_SUPPORT_PERIODS]; /* Repair labor revenue per hour */
float RC[NUM_SUPPORT_PERIODS]; /* Repair labor cost */

/* INVENTORY INPUTS */
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS]; /* inventory holding costs */
int BEGIN_INVENTORY[NUM_PRODUCTS];

/* CAPITAL BUDGETING INPUTS */
float cl[NUM_CAP_BUD_PROJECTS][NUM_PERIODS]; /* cost of production cap budget projects */
float cn[NUM_PERIODS]; /* cost of non-prod cap budget projects */
float cv[NUM_PERIODS]; /* cost of support capital budgeting project in each period */
float El[NUM_PERIODS]; /* maximum allowable prod project expense */
float En[NUM_PERIODS]; /* maximum allowable pre-production project expense */
float Ev[NUM_PERIODS]; /* maximum allowable post support wc project expense */
int CL [NUM_CAP_BUD_PROJECTS][NUM_PROD_WC][NUM_PROD_PERIODS]; /* improvement at
WC's due to cap bud proj 1 */
int CV [NUM_SUPPORT_PERIODS]; /* improvement in support hours from support cap bud project */
int CN[NUM_NON_PROD_WC][NUM_PERIODS]; /* Increase at non-prod wc's from cap bud project */

/* STARTING AND ENDING PERIOD INPUTS */
int max_periods[NUM_PRODUCTS]; /* maximum number of periods a product may be produced */
int start_period[NUM_PRODUCTS]; /* earliest period a product can be produced */

/* title of input file */
char TITLE[15];

/* prototype of mpsf function */
void first( int d1[NUM_PROD_PERIODS], int d2[NUM_PROD_PERIODS],
int SETUP_HOURS[NUM_PROD_WC] [NUM_PRODUCTS],
int PRODUCTION_HOURS[NUM_PROD_WC] [NUM_PRODUCTS],
float REVENUE [NUM_PRODUCTS][NUM_PROD_PERIODS],
float WLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float OLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float Bn [NUM_NON_PROD_WC], int CN [NUM_NON_PROD_WC][NUM_PERIODS],
float cn [NUM_PERIODS], float En [NUM_PERIODS],
float MA [NUM_PRODUCTS][NUM_PROD_PERIODS],
float G [NUM_NON_PROD_WC][NUM_PRODUCTS][NUM_PERIODS],
```

```
int max_periods[NUM_PRODUCTS], int start_period[NUM_PRODUCTS],
float BASE_COST_N[NUM_NON_PROD_WC], float ACC,
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS],
int BEGIN_INVENTORY[NUM_PRODUCTS], float BASE_COST_L[NUM_PROD_WC],
float Bl[NUM_PROD_WC], float S0, float BASE_COST_V,
int PLJ[NUM_PRODUCTS], float SJT[NUM_PRODUCTS][AGE],
int PWJ[NUM_PRODUCTS], float pj[NUM_PRODUCTS][AGE],
float PT[NUM_SUPPORT_PERIODS], float PR[NUM_PRODUCTS][AGE],
float RT[NUM_SUPPORT_PERIODS], float ft[NUM_SUPPORT_PERIODS],
float RC[NUM_SUPPORT_PERIODS], float inflation );
```

```
/* frstmain.c */
/* main routine to build an MPS file for the first stage of the
        two stage model */

/* inputs from input.dat */
/* outputs data to outputf.dat and mpfs.dat */
/* calls function: first.c */

#include "frstmain.h"

main(void)

{

        FILE *fptri, *fptro;
        register i, j;

        /* read inputs from a file and print back to output file */

        if ( (fptri = fopen("input.dat","r")) == NULL ) /* open input file */
                {
                 printf("Can't open the input file.");
                 exit(0);
                }

        if ( ( fptro = fopen("outputf.dat", "w") ) == NULL ) /* open output file */
                {
                printf("Can't open the output file.");
                exit(0);
                }

        fprintf(fptro, "outputf.dat\n\n");
        fscanf(fptri, "%s", TITLE );
        fprintf(fptro, "INPUT IS FROM FILE: %s\n", TITLE );

        fprintf(fptro,"\n\n                LIST OF INPUTS AND OUTPUTS\n");
        fprintf(fptro,"                _____________________________\n\n\n");
        fprintf(fptro,"A.  Input Data\n");

        fprintf(fptro, "\n\n");


        /* Read data from input file */
        fprintf(fptro, "DEMAND FOR SUBGROUP ONE\n");
        for(i = 0; i < NUM_PROD_PERIODS; i++) /* read demand for group 1 */
                {
                fscanf(fptri, "%d", &d1[i]);
                fprintf(fptro, "%1d\t", d1[i]);
                }

        fprintf(fptro, "\nDEMAND FOR SUBGROUP TWO\n");
```

```
for(i = 0; i < NUM_PROD_PERIODS; i++) /* read demand for group 2 */
        {
        fscanf(fptri, "%d", &d2[i]);
        fprintf(fptro, "%1d\t", d2[i]);
        }

fprintf(fptro, "\n\nMUST FUND\n");
fscanf(fptri, "%d", &Must_Fund );
fprintf(fptro, "%4d", Must_Fund );

fprintf(fptro, "\n\nMUTUALLY EXCLUSIVE PRODUCTS\n");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read mutually exclusive */
        {
        fscanf(fptri, "%d", &Mutually_Exclusive_Prod[i]);
        fprintf(fptro, "%4d\t", Mutually_Exclusive_Prod[i]);
        }


fprintf(fptro, "\n\nMUTUALLY EXCLUSIVE PROJECTS\n");
for ( i = 0; i < NUM_CAP_BUD_PROJECTS; ++i ) /* read mut ex cap budget */
        {
        fscanf(fptri, "%d", &Mutually_Exclusive_Proj[i]);
        fprintf(fptro, "%4d\t", Mutually_Exclusive_Proj[i]);
        }

fprintf(fptro, "\n\nPRODUCT CONTINGENCY\n");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read in product contingency */
        {
        fscanf(fptri, "%d", &Contingency[i] );
        fprintf(fptro, "%4d\t", Contingency[i] );
        }

fprintf(fptro, "\n\nPRINT SETUP HOURS");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read setup hours */
        {
        fprintf(fptro,"\nSetup Hours At Work Center %d for Each Product\n", i );
        for ( j = 0; j < NUM_PRODUCTS; j++ )
                {
                fscanf(fptri, "%d", &SETUP_HOURS[i] [j] );
                fprintf(fptro, "%d\t", SETUP_HOURS[i] [j]);
                }
        }


fprintf(fptro, "\n\nPRINT PRODUCTION HOURS");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read production hours */
        {
        fprintf(fptro,"\nProduction Hours At Work Center %d for Each Product\n", i );
        for ( j = 0; j < NUM_PRODUCTS; j++ )
                {
                fscanf(fptri, "%d", &PRODUCTION_HOURS[i] [j] );
```

```
                fprintf(fptro, "%d\t", PRODUCTION_HOURS[i] [j] );
                }
        }

for ( k = 0; k < NUM_CAP_BUD_PROJECTS; k++ )
{
fprintf(fptro, "\n\nPRODUCTION INCREASES FROM PROJECT %d", k );
for ( i = 0; i < NUM_PROD_WC; i++ )
        {
        fprintf(fptro,"\nProduction Increases At Work Center %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                {
                fscanf(fptri, "%d", &CL[k][i][j] );
                fprintf(fptro, "%d\t", CL[k][i][j] );
                }
        }
 }

fprintf(fptro, "\n\nREVENUE");
for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
        fprintf(fptro, "\nRevenue from product %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
        {
        fscanf(fptri, "%f", &REVENUE[i][j] );
        fprintf(fptro, "%8.2f ", REVENUE[i][j] );
        }
        }

fprintf(fptro, "\n\nINTEREST RATE\n");
fscanf(fptri, "%f", &ACC );
fprintf(fptro, "ACC = %4.3f", ACC );

fprintf(fptro, "\n\nINFLATION RATE\n" );
fscanf(fptri, "%f", &inflation );
fprintf(fptro, "inflation = %4.3f", inflation );

fprintf(fptro, "\n\nREGULAR TIME COSTS");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read regular time cost */
        {
        fprintf(fptro,"\nRegular Time Cost for Production Work Center %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                {
                fscanf(fptri, "%f", &WLT[i] [j] );
                fprintf(fptro, "%4.2f\t", WLT[i] [j] );
                }
        }


fprintf(fptro, "\n\nPRINT OVER TIME COST");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read overtime cost */
```

```
                {
                fprintf(fptro,"\nOver Time Cost for Production Work Center %d\n", i );
                for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                        {
                        fscanf(fptri, "%f", &OLT [i] [j] );
                        fprintf(fptro, "%4.2f\t", OLT [i] [j] );
                        }
                }


        fprintf(fptro, "\n\nBEGINNING HOURS AT PRODUCTION WC'S\n");
        for ( i = 0; i < NUM_PROD_WC; i++ ) /* read beginning production time */
                {
                fscanf(fptri, "%f", &Bl [i] );
                fprintf(fptro, "%4.2f\t", Bl [i] );
                }

        fprintf(fptro, "\n\nBEGINNING UNITS OF CAPACITY AT NON-PRODUCTION WC'S\n");
        for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read beginning non-production time */
                {
                fscanf(fptri, "%f", &Bn [i] );
                fprintf(fptro, "%4.2f\t", Bn [i] );
                }

        fprintf(fptro, "\n\nINCREASES FROM NON-PRODUCTION CAPITAL BUDGTEING
PROJECT ONE\n");
        for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read increase from non-production cap bud
project 1 */
        {
        fprintf(fptro, "\nIncreases at WC%d\n", i );
                for ( j = 0; j < NUM_PERIODS; j++ )
                {
                fscanf(fptri, "%d", &CN [i][j] );
                fprintf(fptro, "%d  ", CN [i][j] );
                }
        }

        fprintf(fptro, "\n\nCOST OF PRODUCTION CAPITAL BUDGETING PROJECTS");
        for ( i = 0; i < NUM_CAP_BUD_PROJECTS; i++ ) /* read cost of each capital project */
                {
                fprintf(fptro,"\nCapital Budgeting Project %d\n", i );
                for ( j = 0; j < NUM_PERIODS; j++ )
                        {
                        fscanf(fptri, "%f", &cl [i] [j] );
                        fprintf(fptro, "%8.2f  ", cl [i] [j] );
                        }
                }

        fprintf(fptro, "\n\nCOST OF NON-PRODUCTION CAPITAL BUDGETING PROJECT\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read cost of capital project */
                {
```

```
                fscanf(fptri, "%f", &cn [i] );
                fprintf(fptro, "%8.2f ", cn [i] );
                }

        fprintf(fptro, "\n\nCOST OF SUPPORT CAPITAL BUDGETING PROJECT\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read cost of capital project */
                {
                fscanf(fptri, "%f", &cv [i] );
                fprintf(fptro, "%8.2f ", cv [i] );
                }


        fprintf(fptro, "\n\nMAXIMUM ALLOWABLE PRODUCTION PROJECT EXPENSE\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable production capital project
expense */
                {
                fscanf(fptri, "%f", &El [i] );
                fprintf(fptro, "%8.2f  ", El [i] );
                }

        fprintf(fptro, "\n\nMAXIMUM ALLOWABLE NON-PRODUCTION PROJECT EXPENSE\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable non-production capital project
expense */
                {
                fscanf(fptri, "%f", &En [i] );
                fprintf(fptro, "%8.2f ", En [i] );
                }

        fprintf(fptro, "\n\nMAXIMUM ALLOWABLE SUPPORT PROJECT EXPENSE\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable support capital project
expense */
                {
                fscanf(fptri, "%f", &Ev [i] );
                fprintf(fptro, "%8.2f ", Ev [i] );
                }

        fprintf(fptro, "\n\nMAXIMUM FRACTION REGULAR TIME LABOR\n");
        for ( i = 0; i < NUM_PROD_WC; i++ ) /* read maximum fraction regular time labor */
                {
                fprintf(fptro, "\nWC %d\n", i );
                for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                        {
                        fscanf(fptri, "%f", &LAMBDA[i][j] );
                        fprintf(fptro, "%4.2f\t", LAMBDA[i][j] );
                        }
                }

        fprintf(fptro, "\n\nPRODUCT LIFE LENGTH\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read product life lengths */
                {
                fscanf(fptri, "%d", &PLJ [i] );
```

```
                fprintf(fptro, "%d\t", PLJ [i] );
                }

        fprintf(fptro, "\n\nPRODUCT WARRANTEE LENGTH\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read product warrantee lengths */
                {
                fscanf(fptri, "%d", &PWJ [i] );
                fprintf(fptro, "%d\t", PWJ [i] );
                }

        fprintf(fptro, "\n\nSUPPORT HOURS FOR PRODUCTS OF AGE TAU");
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read support hours required for differring age
products */
                {
                fprintf(fptro,"\nSupport Hours for Product Number %d\n", i );
                for ( j = 0; j < PLJ[i]; j++ )
                        {
                        fscanf(fptri, "%f", &SJT [i] [j] );
                        fprintf(fptro, "%4.2f\t", SJT [i] [j] );
                        }
                }

        fprintf( fptro, "\n\nBEGINNING SUPPORT HOURS AVAILABLE\n" );
        fscanf(fptri, "%f", &S0 ); /* Read beginning support hours */
        fprintf(fptro, "%4.2f", S0 );

        fprintf(fptro, "\n\nMAXIMUM ALLOWABLE PRODUCTION COST\n");
        for ( i = 0; i < NUM_PROD_PERIODS; i++ ) /* read maximum allowable production cost */
                {
                fscanf(fptri, "%f", &PC [i] );
                fprintf(fptro, "%8.2f ", PC [i] );
                }

        fprintf(fptro, "\n\nAVERAGE PARTS COST TO REPAIR PRODUCT OF AGE TAU");
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read average parts cost to repair product of age tau */
                {
                fprintf(fptro,"\nAverage parts cost for Product Number %d\n", i );
                for ( j = 0; j < PLJ[i]; j++ )
                        {
                        fscanf(fptri, "%f", &pj [i] [j] );
                        fprintf(fptro, "%4.2f\t", pj [i] [j] );
                        }
                }

        fprintf(fptro, "\n\nMAXIMUM REPAIR PARTS COST\n");
        for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read maximum repair parts cost */
                {
                fscanf(fptri, "%f", &PT [i] );
                fprintf(fptro, "%8.2f ", PT [i] );
                }
```

```
fprintf(fptro, "\n\nPARTS REVENUE FROM REPAIR OF PRODUCT OF AGE TAU");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read average parts cost to repair product of age tau */
        {
        fprintf(fptro,"\nAverage parts revenue for Product Number %d\n", i );
        for ( j = 0; j < PLJ[i]; j++ )
                {
                fscanf(fptri, "%f", &PR [i] [j] );
                fprintf(fptro, "%4.2f\t", PR [i] [j] );
                }
        }

fprintf(fptro, "\n\nMINIMUM ACCEPTABLE REVENUE\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read minimum acceptable revenue */
        {
        fscanf(fptri, "%f", &RT [i] );
        fprintf(fptro, "%8.2f ", RT [i] );
        }

fprintf(fptro, "\n\nREPAIR LABOR REVENUE PER HOUR\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read repair labor revenue per hour */
        {
        fscanf(fptri, "%f", &ft [i] );
        fprintf(fptro, "%4.2f  ", ft [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE PRODUCTION WC\n");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read base cost to operate production wc's */
        {
        fscanf(fptri, "%f", &BASE_COST_L [i] );
        fprintf(fptro, "%f\t", BASE_COST_L [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE NON-PRODUCTION WC\n");
for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read base cost to operate non-production wc's */
        {
        fscanf(fptri, "%f", &BASE_COST_N [i] );
        fprintf(fptro, "%f\t", BASE_COST_N [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE SUPPORT WC\n");
/* read base cost to operate support wc */
fscanf(fptri, "%f", &BASE_COST_V );
fprintf(fptro, "%f\t", BASE_COST_V );

fprintf(fptro, "\n\nMATERIAL COST FOR PRODUCTION");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read material cost for production */
        {
        fprintf(fptro,"\nMaterial cost for Product Number %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                {
                fscanf(fptri, "%f", &MA [i] [j] );
```

```
                    fprintf(fptro, "%4.2f\t", MA [i] [j] );
                    }
            }


    fprintf(fptro, "\n\nREPAIR LABOR COST\n");
    for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read repair labor cost */
            {
            fscanf(fptri, "%f", &RC [i] );
            fprintf(fptro, "%4.2f  ", RC [i] );
            }

    fprintf(fptro, "\n\nCONSISTENT UNITS REQUIRED BY PRODUCT AT NON_PRODUCTION
WC");
    for ( k = 0; k < NUM_NON_PROD_WC; k++ )
    {
    fprintf(fptro, "\n\nUnits required at wc %d", k );
    for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read units required for non-production */
            {
            fprintf(fptro,"\n  Consistent non-production units for product number %d\n", i );
            fprintf(fptro, "  ");
            for ( j = 0; j < NUM_PERIODS; j++ )
                    {
                    fscanf(fptri, "%f", &G [k] [i] [j] );
                    fprintf(fptro, "%4.2f  ", G [k] [i] [j] );
                    }
            }
    }

    fprintf(fptro, "\n\nMAXIMUM PRODUCTION PERIODS FOR EACH PRODUCT\n");
    for ( i = 0; i < NUM_PRODUCTS; i++ )
            {
            fscanf(fptri, "%d", &max_periods[i] );
            fprintf(fptro, "%d\t", max_periods[i] );
            }

    fprintf(fptro, "\n\nEARLIEST PERIOD PRODUCTION OF A PRODUCT MAY
COMMENCE\n");
    for ( i = 0; i < NUM_PRODUCTS; i++ )
            {
            fscanf(fptri, "%d", &start_period[i] );
            fprintf(fptro, "%d\t", start_period[i] );
            }

    fprintf(fptro, "\n\nINVENTORY HOLDING COSTS FOR PRODUCTS IN EACH PERIOD\n");
    for ( i = 0; i < NUM_PRODUCTS; i++ )
    {
            fprintf(fptro, "\nPRODUCT %d\n", i );
            for ( j = 0; j < NUM_PROD_PERIODS; j++ )
            {
            fscanf(fptri, "%f", &INVENTORY[i][j] );
```

```
                fprintf(fptro, "%8.2f  ", INVENTORY[i][j] );
                }
        }

        fprintf(fptro, "\n\nIMPROVMENT IN HOURS FROM SUPPORT CAP BUD PROJECT\n");
        for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ )
                {
                fscanf(fptri, "%d", &CV[i] );
                fprintf(fptro, "%d  ", CV[i] );
                }

        fprintf(fptro, "\n\nBEGINNING INVENTORY FOR EACH PRODUCT\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                {
                fscanf(fptri, "%d", &BEGIN_INVENTORY[i] );
                fprintf(fptro, "%d\t", BEGIN_INVENTORY[i] );
                }

        fclose(fptri);

        first( d1, d2, SETUP_HOURS, PRODUCTION_HOURS, REVENUE, WLT, OLT, Bn,
        CN, cn, En, MA, G, max_periods, start_period, BASE_COST_N, ACC,
        INVENTORY, BEGIN_INVENTORY, BASE_COST_L, Bl, S0, BASE_COST_V, PLJ, SJT,
        PWJ, pj, PT, PR, RT, ft, RC, inflation );

        fclose(fptro);

} /* end main */
```

```
/* first.h */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define NUM_PERIODS                     12
#define NUM_SUPPORT_PERIODS             11
#define HORIZON_PERIOD                  11
#define NUM_PROD_PERIODS                8
#define NUM_PRODUCTS                    5
#define NUM_PROD_WC                     3
#define NUM_NON_PROD_WC                 4
#define NUM_CAP_BUD_PROJECTS            3
#define NUM_NON_PROD_CAP_BUD_PROJ       1
#define AGE                             5
#define TRUE                            1
#define FALSE                           0


int project, i, j;

int setup_constraint;
int dec_var;
int age;
int inv_lim;
int must_fund;
int nprdwc;


/* base cost variables */
float TOT_BASE_COST;
float TOT_BASE_DISCOUNT;

/* discounted revenues from repair labor */
float ADJ_LABOR_REP_REV[NUM_PRODUCTS][NUM_PERIODS];

/* discounted cost from repair labor */
float ADJ_LABOR_REP_COST[NUM_PRODUCTS][NUM_PERIODS];

/* discounted parts revenues less costs */
float ADJ_PART[NUM_PRODUCTS][NUM_PERIODS];

/* production capital budgeting costs */
float TOTAL_CAP_BUD_EXPENSE[NUM_CAP_BUD_PROJECTS];

/* non-production capital budgeting expense */
float TOTAL_NP_CAP_BUD_EXPENSE;

/* adjusted inventory cost; includes stocking fee, loss of interest on
  revenues and repair costs */
float ACTUAL_INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS - 1];
```

```
/* total support capital budgeting cost */
float TOTAL_SUPPORT_CAP_BUD_COST;

/* estimated production cost per hour */
float EST_PROD_COST_PER_HOUR[NUM_PROD_WC];

/* estimated cost to prossess a single product a specific work center */
float PRODUCTION_WC_EST_PRODUCT_COST[NUM_PRODUCTS][NUM_PROD_PERIODS];

/* cost to setup for production at a specific work center and for a specific product */
float PRODUCTION_WC_EST_SETUP_COST[NUM_PRODUCTS][NUM_PROD_PERIODS];

/* estimated support cost per hour */
float EST_SUPPORT_COST_PER_HOUR;

/* estimated cost to support one product built in aspecific period */
float SUPPORT_WC_EST_PRODUCT_COST[NUM_PRODUCTS][NUM_PERIODS];

/* disccounted production work center cost */
float DISCOUNT_BASE_COST_L;

/* total cost of all production work centers */
float TOT_BASE_COST_L;

/* total cost of support work center */
float TOT_BASE_COST_V;
```

```
/* first.c */
#include "first.h"

/* Builds an MPS file for the first stage of the two stage model */
void first( int d1[NUM_PROD_PERIODS], int d2[NUM_PROD_PERIODS],
int SETUP_HOURS[NUM_PROD_WC] [NUM_PRODUCTS],
int PRODUCTION_HOURS[NUM_PROD_WC] [NUM_PRODUCTS],
float REVENUE [NUM_PRODUCTS][NUM_PROD_PERIODS],
float WLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float OLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float Bn [NUM_NON_PROD_WC], int CN [NUM_NON_PROD_WC][NUM_PERIODS],
float cn [NUM_PERIODS], float En [NUM_PERIODS],
float MA [NUM_PRODUCTS][NUM_PROD_PERIODS],
float G [NUM_NON_PROD_WC][NUM_PRODUCTS][NUM_PERIODS],
int max_periods[NUM_PRODUCTS], int start_period[NUM_PRODUCTS],
float BASE_COST_N[NUM_NON_PROD_WC], float ACC,
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS],
int BEGIN_INVENTORY[NUM_PRODUCTS], float BASE_COST_L[NUM_PROD_WC],
float Bl[NUM_PROD_WC], float S0, float BASE_COST_V,
int PLJ[NUM_PRODUCTS], float SJT[NUM_PRODUCTS][AGE],
int PWJ[NUM_PRODUCTS], float pj[NUM_PRODUCTS][AGE],
float PT[NUM_SUPPORT_PERIODS], float PR[NUM_PRODUCTS][AGE],
float RT[NUM_SUPPORT_PERIODS], float ft[NUM_SUPPORT_PERIODS],
float RC[NUM_SUPPORT_PERIODS], float inflation )

{
FILE *mps;

register product, period, wc;

        /* open MPS output file */

        if ( (mps = fopen("mpfs.dat","w")) == NULL ) /* open mps output file */
                {
                 printf("Can't open the mps output file.");
                 exit(0);
                }


        /* build total non-production capital budgeting costs */
        TOTAL_NP_CAP_BUD_EXPENSE = 0;
        for ( period = 0; period < NUM_PERIODS; period++ )
                TOTAL_NP_CAP_BUD_EXPENSE +=  cn[period] * pow( 1 + ACC,
HORIZON_PERIOD - period );

        /* Determine average per period cost per hour to operate each production work center */
        /*  NOTE: in constant dollars */
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                EST_PROD_COST_PER_HOUR[wc] = BASE_COST_L[wc]/Bl[wc];

        /* Determine future worth of the average production cost of time
```

```
        consumed on production work centers per product per period */
        /* initialize variable values */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                PRODUCTION_WC_EST_PRODUCT_COST[product][period] = 0;
                }
        }
        /* determine values */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        PRODUCTION_WC_EST_PRODUCT_COST[product][period] +=
PRODUCTION_HOURS[wc][product] * EST_PROD_COST_PER_HOUR[wc] * pow( 1 + inflation,
period) * pow( 1 + ACC, HORIZON_PERIOD - period );
                        }
                }
        }

        /* Determine future worth of the average cost of setup time consumed
        on production work centers per product per period */
        /* initialize variable values */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = 0; period < NUM_PERIODS; period++ )
                {
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        PRODUCTION_WC_EST_SETUP_COST[product][period] = 0;
                        }
                }
        }
        /* determine values */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        PRODUCTION_WC_EST_SETUP_COST[product][period] +=
SETUP_HOURS[wc][product] * EST_PROD_COST_PER_HOUR[wc] * pow( 1 + inflation, period) *
pow( 1 + ACC, HORIZON_PERIOD - period );
                        }
                }
        }
```

```
        /* Determine average per period cost per hour to operate support work center */
        /*  NOTE: in constant dollars */
        EST_SUPPORT_COST_PER_HOUR = BASE_COST_V/S0;

        /* Determine future worth of the average cost of time
        consumed on support work center per product per period */
        /* initialize variable values */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = 0; period < NUM_PERIODS; period++ )
                {
                SUPPORT_WC_EST_PRODUCT_COST[product][period] = 0;
                }
        }
        /* Determine variable values */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
          for ( period = start_period[product]; period < start_period[product] + max_periods[product];
period++ )
          {
                for ( age = 0; age <  PLJ[product]; age++ )
                {
                SUPPORT_WC_EST_PRODUCT_COST[product][period] += SJT[product][age] *
EST_SUPPORT_COST_PER_HOUR * pow( 1 + inflation, period ) * pow( 1 + ACC,
HORIZON_PERIOD - ( period + age ) );
                }
          }
        }


        /* build total future worth of repair revenues and costs for each product by
        period it was produced */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = 0; period < NUM_PERIODS; period++ )
                {
                ADJ_LABOR_REP_REV[product][period] = 0;
                ADJ_LABOR_REP_COST[product][period] = 0;
                ADJ_PART[product][period] = 0;
                }
        }
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period =  start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                for ( age = PWJ[product]; age < PLJ[product]; age++ )
                ADJ_LABOR_REP_REV[product][period] += SJT[product][age]*ft[age] * pow( 1 +
ACC, HORIZON_PERIOD - ( period + age ) );
                }
```

```
        }
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period =  start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                for ( age = 0; age < PLJ[product]; age++ )
                ADJ_LABOR_REP_COST[product][period] += SJT[product][age]*RC[age] * pow( 1 +
ACC, HORIZON_PERIOD - ( period + age ) );
                }
        }
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period =  start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                for ( age = 0; age < PLJ[product]; age++ )
                {
                if ( age < PWJ[product] )
                        ADJ_PART[product][period] += - pj[product][age] * pow( 1 + inflation, period
+ age ) * pow( 1 + ACC, HORIZON_PERIOD - ( period + age ) );
                if ( age >= PWJ[product] )
                        ADJ_PART[product][period] += ( PR[product][age] - pj[product][age] ) * pow(
1 + inflation, period + age ) * pow( 1 + ACC, HORIZON_PERIOD - ( period + age ) );
                }
                }
        }

        /* actual inventory cost - includes stocking cost, loss of interest on
        revenues, adjusts repair costs and revenues, and adjusts period in
        which support capacity is required */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product] - 1; period++ )
                {
                ACTUAL_INVENTORY[product][period] = ( INVENTORY[product][period]
                + REVENUE[product][period] ) * pow( 1 + ACC, HORIZON_PERIOD - period )
                + ADJ_LABOR_REP_REV[product][period] -
                ADJ_LABOR_REP_COST[product][period] +
                ADJ_PART[product][period] -
SUPPORT_WC_EST_PRODUCT_COST[product][period]
                - ( REVENUE[product][period + 1] * pow( 1 + ACC, HORIZON_PERIOD - ( period +
1 ) )
                + ADJ_LABOR_REP_REV[product][period + 1 ] -
                ADJ_LABOR_REP_COST[product][period + 1] +
                ADJ_PART[product][period + 1] +
SUPPORT_WC_EST_PRODUCT_COST[product][period + 1] );
                }
        }
```

```
        /* calculate total base costs at non-production wc's */
        TOT_BASE_COST = 0;
        TOT_BASE_DISCOUNT = 0;
        /*  sum base non-production cost */
        for ( wc = 0; wc < NUM_NON_PROD_WC; wc++ )
                TOT_BASE_COST += BASE_COST_N[wc];

        /* Determine total discounted constant dollar costs */
        for ( period = 0; period < NUM_PERIODS; period++ )
                TOT_BASE_DISCOUNT += TOT_BASE_COST * pow( 1 + inflation, period ) * pow(
1 + ACC, HORIZON_PERIOD - period );

        /* Determine residual production future worth base cost */
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                TOT_BASE_COST_L = BASE_COST_L[wc];
        DISCOUNT_BASE_COST_L = 0;
        for ( period = NUM_PROD_PERIODS; period < NUM_PERIODS; period++ )
                DISCOUNT_BASE_COST_L += TOT_BASE_COST_L * pow( 1 + inflation, period) *
pow( 1 + ACC, HORIZON_PERIOD - period );

        /* Determine residual support future worth base cost */
        TOT_BASE_COST_V = BASE_COST_V * pow( 1 + inflation, HORIZON_PERIOD);

        /* print header to name LP */
        fprintf(mps, "NAME    TWO STAGE MODEL: FIRST STAGE MPS FILE (MAX)\n");
        /* print header to start rows */
        fprintf(mps, "ROWS\n");
        /* develop row for the objective function */
        fprintf(mps, " N  obj\n");

        /* non-production capital budget maximum cost per period */
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, " L  mxbudn%d\n", period );

        /* develop rows for product 0 - 1 (Zi) decision variable constraints */
        dec_var = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                dec_var++;
                fprintf(mps, " L  decvar%d\n", dec_var);
                }
        }

        /* develop rows for pre-production/post-support wc capacity */
        product = 0;
        for ( period = 0; period < NUM_PERIODS; period++ )
                {
                for ( wc = 0; wc < NUM_NON_PROD_WC; wc++ )
```

```
                        {
                        while ( product < NUM_PRODUCTS )
                        {
                        if ( G[wc][product][period] > 0 )
                                {
                                fprintf(mps, " L  npdwc%d%d\n", period, wc );
                                product = NUM_PRODUCTS;
                                }
                        else
                                product++;
                        }
                        product = 0;
                        }
                }

        /* develop rows for setup constraints */
        setup_constraint = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                setup_constraint++;
                fprintf(mps, " L  setup%d\n", setup_constraint);
                }
        }

        /* develop rows to set upper limits on inventory */
        inv_lim = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product] - 1; period++ )
                {
                inv_lim++;
                fprintf(mps, " L  invlim%d\n", inv_lim);
                }
        }

        /* develop rows for product demand */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
        fprintf(mps, " L  demand1%d\n",  period );
        }
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
        fprintf(mps, " L  demand2%d\n", period );
        }

        /* develop values of WLT and OLT */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
```

```
{
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
        {
        fprintf(mps, " L  time%d%d\n", period, wc );
        }
}

/* develop rows for minimum production support cost in each period */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
{
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
        {
        fprintf(mps, " E  minprd%d%d\n", period, wc );
        }
}

/* develop maximum values of WLT: assume that maximum WLT is 3/4
        of total labor time spent during production*/
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
{
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
        {
        fprintf(mps, " L  regtm%d%d\n", period, wc );
        }
}

/* develop rows for minimum support hours in each period */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, " E  prdsup%d\n", period );

/* develop maximum support parts cost in each period */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period ++ )
        fprintf(mps, " L  prtsup%d\n", period );

/* develop minimum revenue for each period */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, " G  minrev%d\n", period );

/* develop product must fund rows */
for ( must_fund = 0; must_fund < 2; must_fund++ )
        fprintf(mps, " E  mustfnd%d\n", must_fund );

/* print product mutual exclusivity */
fprintf(mps, " L  prodmutx\n");

/* print basecost constraint row */
fprintf(mps, " E  basecstn\n");

/* print residual production basecost constraint row */
fprintf(mps, " E  basecstl\n");
```

```
        /* print residual support basecost constraint row */
        fprintf(mps, " E  basecstv\n");

        /* set beginning inventory variables equal to one */
        i = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
                {
                if ( BEGIN_INVENTORY[product] > 0 )
                        {
                        i++;
                        fprintf(mps, " E  beginv%d\n", i );
                        }
                }



        /* print header to start columns */
        fprintf(mps, "COLUMNS\n");

        /* print product funding variables */
        dec_var = 0;
        must_fund = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {

                fprintf(mps, "    Z%d       obj       -1.0\n", product );
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                dec_var++;
                if ( dec_var < 10 && product < 3 )
                {
                fprintf(mps, "    Z%d       decvar%d    -%d\n", product, dec_var, d1[period]*3 );
                }
                if ( dec_var >= 10 && product < 3)
                {
                fprintf(mps, "    Z%d       decvar%d    -%d\n", product, dec_var, d1[period]*3 );
                }
                if ( dec_var < 10 && product >= 3 )
                {
                fprintf(mps, "    Z%d       decvar%d    -%d\n", product, dec_var, d2[period]*3 );
                }
                if ( dec_var >= 10 && product >= 3)
                {
                fprintf(mps, "    Z%d       decvar%d    -%d\n", product, dec_var, d2[period]*3 );
                }
                }
                /* develop rows for pre-production/post-support wc capacity */
                for ( period = 0; period < NUM_PERIODS; period++ )
                        {
                        for ( wc = 0; wc < NUM_NON_PROD_WC; wc++ )
```

```
                                {
                                if ( G[wc][product][period] > 0 )
                                        {
                                        fprintf(mps, "   Z%d       npdwc%d%d      %-12.2f\n",
product, period, wc, G[wc][product][period] );
                                        }
                                }
                        }


                /* develop must fund columns */
                if ( product == 0 || product == 3 )
                        {
                        fprintf(mps, "   Z%d       mustfnd%d    1.0\n", product, must_fund );
                        must_fund++;
                        }
                if ( product == 1 || product == 2 )
                        fprintf(mps, "   Z%d       prodmutx     1.0\n", product );
        }

        /* print production work center setup variables */
        setup_constraint = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                fprintf(mps, "   DELTA%d%-4dobj       -%-12.2f\n", product, period,
PRODUCTION_WC_EST_SETUP_COST[product][period] );
                setup_constraint++;
                if ( setup_constraint < 10 && product < 3 )
                {
                fprintf(mps, "   DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d1[period] * 3 );
                }
                if ( setup_constraint >= 10 && product < 3)
                {
                fprintf(mps, "   DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d1[period] * 3 );
                }
                if ( setup_constraint < 10 && product >= 3 )
                {
                fprintf(mps, "   DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d2[period] * 3 );
                }
                if ( setup_constraint >= 10 && product >= 3)
                {
                fprintf(mps, "   DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d2[period] * 3 );
                }
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
```

```
                        {
                        fprintf(mps, "    DELTA%d%-4dtime%d%d  %12d\n", product, period, period,
wc, SETUP_HOURS[wc][product] );
                        }
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "    DELTA%d%-4dminprd%d%d%12d\n", product, period,
period, wc, SETUP_HOURS[wc][product] );
                        }
                }
        }

        /* print non-production capital budgeting variable */
        fprintf(mps, "    EN      obj       -%-12.2f\n", TOTAL_NP_CAP_BUD_EXPENSE );
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, "    EN      mxbudn%d    %-12.2f\n", period, cn[period] );
        product = 0;
        for ( period = 0; period < NUM_PERIODS; period++ )
        {
                for ( wc = 0; wc < NUM_NON_PROD_WC; wc++ )
                {
                while ( product < NUM_PRODUCTS )
                        {
                        if ( G[wc][product][period] > 0 )
                                {
                                fprintf(mps, "    EN      npdwc%d%d     -%-12d\n", period, wc,
CN[wc][period] );
                                product = NUM_PRODUCTS;
                                }
                        else
                                product++;
                        }
                product = 0;
                }
        }

        /* print production variables */
        setup_constraint = 0;
        dec_var = 0;
        inv_lim = 1;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
                {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                fprintf(mps, "    X%d%-4d   obj       %-12.2f\n", product, period,
                ( REVENUE[product][period]
                - MA[product][period] ) * pow( 1 + ACC, HORIZON_PERIOD - period )
                + ADJ_LABOR_REP_REV[product][period]
                - ADJ_LABOR_REP_COST[product][period]
                + ADJ_PART[product][period]
```

```
                - PRODUCTION_WC_EST_PRODUCT_COST[product][period]
                - SUPPORT_WC_EST_PRODUCT_COST[product][period] );
                dec_var++;
                fprintf(mps, "   X%d%-4d   decvar%d          1.0\n", product, period, dec_var );
                setup_constraint++;
                if ( setup_constraint < 10 )
                {
                fprintf(mps, "   X%d%-4d   setup%d           1.0\n", product, period,
setup_constraint);
                }
                if ( setup_constraint >= 10 )
                {
                fprintf(mps, "   X%d%-4d   setup%d           1.0\n", product, period, setup_constraint);
                }
                if ( inv_lim >= 10 && period < start_period[product] + max_periods[product] - 1 )
                {
                fprintf(mps, "   X%d%-4d   invlim%d      -1.0\n", product, period, inv_lim);
                inv_lim++;
                }
                if ( inv_lim < 10 && period < start_period[product] + max_periods[product] - 1 )
                {
                fprintf(mps, "   X%d%-4d   invlim%d      -1.0\n", product, period, inv_lim);
                inv_lim++;
                }
                if ( product < 3 )
                {
                fprintf(mps, "   X%d%-4d   demand1%d         1.0\n", product, period, period );
                }
                if ( product >= 3 )
                {
                fprintf(mps, "   X%d%-4d   demand2%d         1.0\n", product, period, period );
                }
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "   X%d%-4d   time%d%d  %12d\n", product, period, period, wc,
PRODUCTION_HOURS[wc][product] );
                        }
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "   X%d%-4d   minprd%d%d%12d\n", product, period, period,
wc, PRODUCTION_HOURS[wc][product] );
                        }
                for ( age = 0; age < PLJ[product]; age++ ) /* support hours in each period */
                        fprintf(mps, "   X%d%-4d   prdsup%d      %-12.2f\n", product, period, period
+ age, SJT[product][age] );
                for ( age = 0; age < PLJ[product]; age++ ) /* support cost in each period */
                        fprintf(mps, "   X%d%-4d   prtsup%d      %-12.2f\n", product, period, period
+ age, pj[product][age] * pow( 1 + inflation, period + age ) );
                for ( age = 0; age < PLJ[product]; age++ ) /* revenues in each period */
                        {
                        if ( age == 0 )
```

```
                                fprintf(mps, "   X%d%-4d   minrev%d      %-12.2f\n", product,
period, period, PR[product][age] * pow( 1 + inflation, period ) + ft[period] * SJT[product][age] +
REVENUE[product][period] );
                        else
                                fprintf(mps, "   X%d%-4d   minrev%d      %-12.2f\n", product,
period, period + age, PR[product][age] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age] );
                        }

                }
        }
        /* print inventory variables */
        inv_lim = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
                {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product] - 1; period++ )
                {
                fprintf(mps, "   I%d%-4d   obj          -%-12.2f\n", product, period,
ACTUAL_INVENTORY[product][period] );
                inv_lim++;
                if ( inv_lim < 10 )
                {
                fprintf(mps, "   I%d%-4d   invlim%d        1.0\n", product, period, inv_lim);
                }
                if ( inv_lim >= 10 )
                {
                fprintf(mps, "   I%d%-4d   invlim%d       1.0\n", product, period, inv_lim);
                }
                if ( product < 3 )
                {
                fprintf(mps, "   I%d%-4d   demand1%d         -1.0\n", product, period, period );
                fprintf(mps, "   I%d%-4d   demand1%d          1.0\n", product, period, period + 1 );
                }
                if ( product >= 3 )
                {
                fprintf(mps, "   I%d%-4d   demand2%d         -1.0\n", product, period, period );
                fprintf(mps, "   I%d%-4d   demand2%d         1.0\n", product, period, period + 1 );
                }
                for ( age = 0; age < PLJ[product]; age++ ) /* adj period support hours */
                {
                        if ( age == 0 )
                                {
                                fprintf(mps, "   I%d%-4d   prdsup%d     -%-12.2f\n", product,
period, period, SJT[product][age] );
                                }
                        if ( age > 0 && age < PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   prdsup%d      %-12.2f\n", product,
period, period + age, SJT[product][age - 1] - SJT[product][age] );
                                }
```

```
                        if ( age == PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   prdsup%d      %-12.2f\n", product,
period, period + age, SJT[product][age - 1] - SJT[product][age] );
                                fprintf(mps, "   I%d%-4d   prdsup%d      %-12.2f\n", product,
period, period + age + 1, SJT[product][age] );
                                }
                }
                for ( age = 0; age < PLJ[product]; age++ ) /* adj period part sup cost */
                {
                        if ( age == 0 )
                                {
                                fprintf(mps, "   I%d%-4d   prtsup%d     -%-12.2f\n", product, period,
period, pj[product][age] * pow( 1 + inflation, period ) );
                                }
                        if ( age > 0 && age < PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   prtsup%d     %-12.2f\n", product, period,
period + age, pj[product][age - 1] * pow( 1 + inflation, period + age ) - pj[product][age] * pow( 1 +
inflation, period + age ) );
                                }
                        if ( age == PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   prtsup%d     %-12.2f\n", product, period,
period + age, pj[product][age - 1] * pow( 1 + inflation, period + age ) - pj[product][age] * pow( 1 +
inflation, period + age ) );
                                fprintf(mps, "   I%d%-4d   prtsup%d     %-12.2f\n", product, period,
period + age + 1, pj[product][age] * pow( 1 + inflation, period + age + 1 ) );
                                }
                }
                for ( age = 0; age < PLJ[product]; age++ ) /* adj period revenue */
                        {
                        if ( age == 0 )
                                {
                                fprintf(mps, "   I%d%-4d   minrev%d      -%-12.2f\n", product,
period, period, PR[product][age] * pow( 1 + inflation, period ) + ft[period] * SJT[product][age] +
REVENUE[product][period] );
                                }
                        if ( age == 1 )
                                {
                                fprintf(mps, "   I%d%-4d   minrev%d      %-12.2f\n", product,
period, period + age, PR[product][age - 1] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age - 1] + REVENUE[product][period] * pow( 1 + inflation, period + age ) -
PR[product][age] * pow( 1 + inflation, period + age ) - ft[period + age] * SJT[product][age] );
                                }
                        if ( age > 1 && age < PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   minrev%d      %-12.2f\n", product,
period, period + age, PR[product][age - 1] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age - 1] - PR[product][age] * pow( 1 + inflation, period + age ) - ft[period + age] *
SJT[product][age] );
```

```
                        }
                    if ( age == PLJ[product] - 1 )
                        {
                        fprintf(mps, "   I%d%-4d   minrev%d       %-12.2f\n", product,
period, period + age, PR[product][age - 1] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age - 1] - PR[product][age] * pow( 1 + inflation, period + age ) - ft[period + age] *
SJT[product][age] );
                        fprintf(mps, "   I%d%-4d   minrev%d       %-12.2f\n", product,
period, period + age + 1, PR[product][age] * pow( 1 + inflation, period + age + 1) + ft[period + age + 1] *
SJT[product][age] );
                        }
                    }

            }
        }

        /* print regular time labor variables: maximum is 3/4 of total
                production labor time */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
            for ( wc = 0; wc < NUM_PROD_WC; wc++ )
            {
            fprintf(mps, "   WLT%d%-4d  obj        -%-12.2f\n", period, wc, WLT[wc][period] *
pow( 1 + ACC, HORIZON_PERIOD - period ) );
            fprintf(mps, "   WLT%d%-4d  time%d%d     -1.0\n", period, wc, period, wc );
            fprintf(mps, "   WLT%d%-4d  regtm%d%d     0.25\n", period, wc, period, wc );
            }
        }


        /* print over time labor variables */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
            for ( wc = 0; wc < NUM_PROD_WC; wc++ )
            {
            fprintf(mps, "   OLT%d%-4d  obj        -%-12.2f\n", period, wc, OLT[wc][period] *
pow( 1 + ACC, HORIZON_PERIOD - period ) );
            fprintf(mps, "   OLT%d%-4d  time%d%d     -1.0\n", period, wc, period, wc );
            fprintf(mps, "   OLT%d%-4d  regtm%d%d    -1.0\n", period, wc, period, wc );
            }
        }

        /* print pre-production/post-support base costs constraint */
        fprintf(mps, "   BASECSTN  obj       -%-12.3f\n", TOT_BASE_DISCOUNT );
        fprintf(mps, "   BASECSTN  basecstn      1.0\n");

        /* print residual production base costs constraint */
        fprintf(mps, "   BASECSTL  obj       -%-12.3f\n", DISCOUNT_BASE_COST_L );
        fprintf(mps, "   BASECSTL  basecstl      1.0\n");

        /* print residual support base costs constraint */
```

```
        fprintf(mps, "    BASECSTV  obj       -%-12.3f\n", TOT_BASE_COST_V );
        fprintf(mps, "    BASECSTV  basecstv      1.0\n");

        /* print beginning inventory variables */
        i = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                if ( BEGIN_INVENTORY[product] > 0 )
                {
                if ( product < 3 )
                {
                fprintf(mps, "    BI%d      obj       %-12.2f\n", product, REVENUE[product][0] * pow(
1 + ACC, HORIZON_PERIOD ) );
                fprintf(mps, "    BI%d      demand10    %-12d\n", product,
BEGIN_INVENTORY[product] );
                }
                if ( product >= 3 )
                {
                fprintf(mps, "    BI%d      obj       %-12.2f\n", product, REVENUE[product][0] *
pow(1 + ACC, HORIZON_PERIOD ) );
                fprintf(mps, "    BI%d      demand20    %-12d\n", product,
BEGIN_INVENTORY[product] );
                }
                i++;
                fprintf(mps, "    BI%d      beginv%d    1.0\n", product, i );
                }
        }

        /* develop cost variables for minimum production support cost in
                each period */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                {
                fprintf(mps, "    UNDERP%d%d  obj       -%-12.2f\n", period, wc,
EST_PROD_COST_PER_HOUR[wc] * pow( 1 + inflation, period ) * pow( 1 + ACC,
HORIZON_PERIOD - period ) );
                fprintf(mps, "    UNDERP%d%d  minprd%d%d   1.0\n", period, wc, period, wc );
                }
        }

        /* develop slack variables for minimum production support cost in
                each period */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                {
                fprintf(mps, "    OVERP%d%d   minprd%d%d   -1.0\n", period, wc, period, wc );
                }
        }
```

```
        /* develop cost variables for minimum support cost in each period */
        for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        {
          if ( period < 10 )
                {
                fprintf(mps, "    UNDRS%d    obj          -%-12.2f\n", period,
EST_SUPPORT_COST_PER_HOUR * pow( 1 + inflation, period ) * pow( 1 + ACC,
HORIZON_PERIOD - period ) );
                fprintf(mps, "    UNDRS%d    prdsup%d       1.0\n", period, period );
                }
          else
                {
                fprintf(mps, "    UNDRS%d   obj          -%-12.2f\n", period,
EST_SUPPORT_COST_PER_HOUR * pow( 1 + inflation, period ) * pow( 1 + ACC,
HORIZON_PERIOD - period ) );
                fprintf(mps, "    UNDRS%d   prdsup%d       1.0\n", period, period );
                }
        }

        /* develop slack variables for minimum support cost in each period */
        for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        {
          if ( period < 10 )
                fprintf(mps, "    OVERS%d    prdsup%d      -1.0\n", period, period );
          else
                fprintf(mps, "    OVERS%d   prdsup%d      -1.0\n", period, period );
        }


        /* print header to start rows */
        fprintf(mps, "RHS\n");

        /* print non-production capital budget maximum cost per period */
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, "    RHS       mxbudn%d   %-12.2f\n", period, En[period] );

        /* develop maximum pre-production/post-support wc capacity RHS */
        product = 0;
        for ( period = 0; period < NUM_PERIODS; period++ )
                {
                for ( wc = 0; wc < NUM_NON_PROD_WC; wc++ )
                        {
                        while ( product < NUM_PRODUCTS )
                        {
                        if ( G[wc][product][period] > 0 )
                                {
                                fprintf(mps, "    RHS       npdwc%d%d     %-12.2f\n", period, wc,
Bn[wc] );
                                product = NUM_PRODUCTS;
                                }
                        else
```

```
                        product++;
                }
                product = 0;
                }
        }

/* print maximum production (demand) RHS */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
{
fprintf(mps, "    RHS      demand1%d        %d\n", period, d1[period] );
}
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
{
fprintf(mps, "    RHS      demand2%d        %d\n", period, d2[period] );
}

/* develop rows for minimum production support cost in each period */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
{
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
        {
        fprintf(mps, "    RHS      minprd%d%d    %-8.2f\n", period, wc, Bl[wc] );
        }
}

/* print maximum support hours RHS */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, "    RHS      prdsup%d      %-12.2f\n", period, S0 );

/* print maximum support parts cost RHS */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, "    RHS      prtsup%d      %-12.2f\n", period, PT[period] );

/* print minimum revenue for each period RHS */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, "    RHS      minrev%d      %-12.2f\n", period, RT[period] );

/* print must fund RHS */
for ( must_fund = 0; must_fund < 2; must_fund++ )
        fprintf(mps, "    RHS      mustfnd%d     1.0\n", must_fund );

/* print product mutual exclusivity */
fprintf(mps, "    RHS      prodmutx      1.0\n");

/* set pre-production/post-support base cost to 1 */
fprintf(mps, "    RHS      basecstn      1.0\n");

/* set residual production base cost to 1 */
fprintf(mps, "    RHS      basecstl      1.0\n");

/* set residual support base cost to 1 */
```

```
        fprintf(mps, "    RHS       basecstv      1.0\n");

        /* print beginning inventory RHS */
        i = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
        if ( BEGIN_INVENTORY[product] > 0 )
                {
                i++;
                fprintf(mps, "    RHS       beginv%d      1.0\n", i );
                }
        }

        fprintf(mps, "ENDATA");
        fclose(mps);
}
```

APPENDIX D: Program Listing for Stage Two of the Sequential Model

```
/* midlmain.h */
/* header file for midlmain.c */

#include <stdio.h>
#include <stdlib.h>
#define NUM_PERIODS                    12
#define NUM_SUPPORT_PERIODS            11
#define HORIZON_PERIOD                 11
#define NUM_PROD_PERIODS                8
#define NUM_PRODUCTS                    5
#define NUM_PROD_WC                     3
#define NUM_NON_PROD_WC                 4
#define NUM_CAP_BUD_PROJECTS            3
#define NUM_NON_PROD_CAP_BUD_PROJ       1
#define AGE                             5 /* maximum life length of any product */

typedef unsigned int BitVector;

int k;
        /* INPUT PARAMETERS */
/* DEMAND */
int d1 [NUM_PROD_PERIODS]; /* Demand for subgroup one */
int d2 [NUM_PROD_PERIODS]; /* Demand for subgroup two */

/* PRODUCT AND PROJECT INTERACTION */
BitVector Mutually_Exclusive_Prod[NUM_PRODUCTS]; /* Mutually exclusive products */
BitVector Mutually_Exclusive_Proj[NUM_CAP_BUD_PROJECTS]; /* " " projects */
BitVector Contingency[NUM_PRODUCTS]; /* Contingency between products */
BitVector Must_Fund; /* Must fund a set of products */

/* INTEREST RATES */
float ACC;  /* Interest Rate */
float inflation; /* Inflation Rate */

/* PRODUCT REVENUES */
float REVENUE [NUM_PRODUCTS][NUM_PROD_PERIODS]; /* Product revenue */

/* PRE-PRODUCTION/POST-SUPPORT PRODUCT INPUTS */
float G[NUM_NON_PROD_WC][NUM_PRODUCTS][NUM_PERIODS]; /* No. consistent units req by
product at non-production wc's  */

/* LABOR INPUTS */
float WLT [NUM_PROD_WC] [NUM_PROD_PERIODS]; /* reg time cost by wc */
float OLT [NUM_PROD_WC] [NUM_PROD_PERIODS]; /* ot time cost by wc */
float LAMBDA[NUM_PROD_WC][NUM_PROD_PERIODS]; /* Fraction labor regular time */

/* PRODUCTION INPUTS */
int SETUP_HOURS[NUM_PROD_WC] [NUM_PRODUCTS]; /* setup hours for prod by wc */
int PRODUCTION_HOURS[NUM_PROD_WC] [NUM_PRODUCTS]; /* production hrs by prod & wc */
float PC[NUM_PROD_PERIODS]; /* Maximum allowable production cost */
float MA[NUM_PRODUCTS][NUM_PROD_PERIODS]; /* Material cost for production */
```

```
/* SUPPORT VARIABLES */
float SJT[NUM_PRODUCTS][AGE]; /* Support hours required for product of different ages */
int PLJ[NUM_PRODUCTS]; /* Product life length */
int PWJ[NUM_PRODUCTS]; /* Warrantee length */
float pj[NUM_PRODUCTS][AGE]; /* Average parts cost */
float PT[NUM_PERIODS]; /* Maximum repair parts cost */
float PR[NUM_PRODUCTS][AGE]; /* Parts revenue */
float RT[NUM_PERIODS]; /* Minimum acceptable revenue */
float ft[NUM_PERIODS]; /* Repair labor revenue per hour */
float RC[NUM_PERIODS]; /* Repair labor cost */

/* INVENTORY INPUTS */
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS]; /* inventory holding costs */
int BEGIN_INVENTORY[NUM_PRODUCTS];

/* CAPITAL BUDGETING INPUTS */
float Bl[NUM_PROD_WC]; /* Beginning hours at production wc's */
float Bn[NUM_NON_PROD_WC]; /* Beginning units of capacity at non-production wc's */
float S0; /* Beginning support hours available */
float cl[NUM_CAP_BUD_PROJECTS][NUM_PERIODS]; /* cost of production cap budget projects */
float cn[NUM_PERIODS]; /* cost of non-prod cap budget projects */
float cv[NUM_PERIODS]; /* cost of support capital budgeting project in each period */
float El[NUM_PERIODS]; /* maximum allowable prod project expense */
float En[NUM_PERIODS]; /* maximum allowable pre-production project expense */
float Ev[NUM_PERIODS]; /* maximum allowable post support wc project expense */
float BASE_COST_L[NUM_PROD_WC]; /* Base cost to operate production wc's */
float BASE_COST_N[NUM_NON_PROD_WC]; /* Base cost to operate non-prod wc's */
float BASE_COST_V; /* Base support cost */
int CL [NUM_CAP_BUD_PROJECTS][NUM_PROD_WC][NUM_PROD_PERIODS]; /* improvement at
WC's due to cap bud proj 1 */
int CV [NUM_PERIODS - 1]; /* improvement in support hours from support cap bud project */
int CN[NUM_NON_PROD_WC][NUM_PERIODS]; /* Increase at non-prod wc's from cap bud project */

/* STARTING AND ENDING PERIOD INPUTS */
int max_periods[NUM_PRODUCTS]; /* maximum number of periods a product may be produced */
int start_period[NUM_PRODUCTS]; /* earliest period a product can be produced */

/* VARIABLE VALUES FROM mpfs.rep, i.e., first stage optimization */
char TITLE[15];
char ZNAME[2];
float Z[NUM_PRODUCTS];
float ZJUNK;
char DELTANAME[7];
float DELTA[NUM_PRODUCTS][NUM_PROD_PERIODS];
float DELTAJUNK;
char ENNAME[2];
float EN;
float ENJUNK;
char ELNAME[3];
float EL[NUM_CAP_BUD_PROJECTS];
```

```
float ELJUNK;
char EVNAME[2];
float EV;
float EVJUNK;
char XNAME[3];
float X[NUM_PRODUCTS][NUM_PROD_PERIODS];
float XJUNK;
char INAME[3];
float I[NUM_PRODUCTS][NUM_PROD_PERIODS - 1];
float IJUNK;
char wltNAME[5];
float wlt[NUM_PROD_PERIODS][NUM_PROD_WC];
float wltJUNK;
char oltNAME[5];
float olt[NUM_PROD_PERIODS][NUM_PROD_WC];
float oltJUNK;

/* string variables to strip mpss.rep off header information */
char PROJECT[8];
char CONSTRAINT[14];
char DEMAND[2];
char EQUAL[1];
char DEMAND_VALUE[3];
char VAR_NAME[8];
char VAR_VALUE[5];
char REDUCED[7];
char COST[4];


/* periods that production begins and ends */
int begin_period;
int end_period;

/* prototype of mpss function */
void last( int d1[NUM_PROD_PERIODS], int d2[NUM_PROD_PERIODS],
int CL[NUM_CAP_BUD_PROJECTS][NUM_PROD_WC][NUM_PROD_PERIODS],
int SETUP_HOURS[NUM_PROD_WC] [NUM_PRODUCTS],
int PRODUCTION_HOURS[NUM_PROD_WC] [NUM_PRODUCTS],
float REVENUE [NUM_PRODUCTS][NUM_PROD_PERIODS],
float WLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float OLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float Bl [NUM_PROD_WC], float cl [NUM_CAP_BUD_PROJECTS] [NUM_PERIODS],
float cv [NUM_PERIODS], float El [NUM_PERIODS],
float Ev [NUM_PERIODS], float LAMBDA [NUM_PROD_WC][NUM_PROD_PERIODS],
float SJT [NUM_PRODUCTS][AGE], float S0, float PC [NUM_PROD_PERIODS],
int PLJ [NUM_PRODUCTS], int PWJ [NUM_PRODUCTS], float pj [NUM_PRODUCTS][AGE],
float PT [NUM_PERIODS], float PR [NUM_PRODUCTS][AGE], float RT [NUM_PERIODS],
float ft [NUM_PERIODS], float MA [NUM_PRODUCTS][NUM_PROD_PERIODS],
float RC [NUM_PERIODS], int max_periods[NUM_PRODUCTS],
int start_period[NUM_PRODUCTS], float BASE_COST_L[NUM_PROD_WC],
float BASE_COST_V, float ACC,
```

```
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS], int CV [NUM_PERIODS - 1],
float inflation, int BEGIN_INVENTORY[NUM_PRODUCTS],
float Z [NUM_PRODUCTS],
float X[NUM_PRODUCTS][NUM_PROD_PERIODS] );
```

```
/* midlmain.c */
/* main routine to build an MPS file for the middle stage of the
        two stage model */

/* input data from input.dat and mpfs.rep */
/* outputs data to outputm.dat and mpms.dat */
/* calls function: middle.c */

#include "midlmain.h"

main(void)

{

        FILE *fptri, *fptri2, *fptro;
        register i, j;

        /* read inputs from a file and print back to output file */

        if ( (fptri = fopen("input.dat","r")) == NULL ) /* open input file */
                {
                 printf("Can't open the input file.");
                 exit(0);
                }

        if ( (fptri2 = fopen("mpfs.rep","r")) == NULL ) /* open input file */
                {
                 printf("Can't open the input file.");
                 exit(0);
                }


        if ( ( fptro = fopen("outputm.dat", "w") ) == NULL ) /* open output file */
                {
                printf("Can't open the output file.");
                exit(0);
                }


        fprintf(fptro, "FILE NAME: outputs.dat\n\n");
        fscanf(fptri, "%s", TITLE );
        fprintf(fptro, "INPUT FILE IS: %s\n", TITLE );

        fprintf(fptro,"\n\n                LIST OF INPUTS AND OUTPUTS\n");
        fprintf(fptro,"                ___________________________\n\n\n");
        fprintf(fptro,"A.  Input Data\n");

        fprintf(fptro, "\n\n");


        /* Read data from input file */
```

```
fprintf(fptro, "DEMAND FOR SUBGROUP ONE\n");
for(i = 0; i < NUM_PROD_PERIODS; i++) /* read demand for group 1 */
        {
        fscanf(fptri, "%d", &d1[i]);
        fprintf(fptro, "%1d\t", d1[i]);
        }

fprintf(fptro, "\nDEMAND FOR SUBGROUP TWO\n");
for(i = 0; i < NUM_PROD_PERIODS; i++) /* read demand for group 2 */
        {
        fscanf(fptri, "%d", &d2[i]);
        fprintf(fptro, "%1d\t", d2[i]);
        }

fprintf(fptro, "\n\nMUST FUND\n");
fscanf(fptri, "%d", &Must_Fund );
fprintf(fptro, "%4d", Must_Fund );

fprintf(fptro, "\n\nMUTUALLY EXCLUSIVE PRODUCTS\n");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read mutually exclusive */
        {
        fscanf(fptri, "%d", &Mutually_Exclusive_Prod[i]);
        fprintf(fptro, "%4d\t", Mutually_Exclusive_Prod[i]);
        }


fprintf(fptro, "\n\nMUTUALLY EXCLUSIVE PROJECTS\n");
for ( i = 0; i < NUM_CAP_BUD_PROJECTS; ++i ) /* read mut ex cap budget */
        {
        fscanf(fptri, "%d", &Mutually_Exclusive_Proj[i]);
        fprintf(fptro, "%4d\t", Mutually_Exclusive_Proj[i]);
        }

fprintf(fptro, "\n\nPRODUCT CONTINGENCY\n");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read in product contingency */
        {
        fscanf(fptri, "%d", &Contingency[i] );
        fprintf(fptro, "%4d\t", Contingency[i] );
        }

fprintf(fptro, "\n\nPRINT SETUP HOURS");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read setup hours */
        {
        fprintf(fptro,"\nSetup Hours At Work Center %d for Each Product\n", i );
        for ( j = 0; j < NUM_PRODUCTS; j++ )
                {
                fscanf(fptri, "%d", &SETUP_HOURS[i] [j] );
                fprintf(fptro, "%d\t", SETUP_HOURS[i] [j]);
                }
        }
```

```
fprintf(fptro, "\n\nPRINT PRODUCTION HOURS");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read production hours */
        {
        fprintf(fptro,"\nProduction Hours At Work Center %d for Each Product\n", i );
        for ( j = 0; j < NUM_PRODUCTS; j++ )
                {
                fscanf(fptri, "%d", &PRODUCTION_HOURS[i] [j] );
                fprintf(fptro, "%d\t", PRODUCTION_HOURS[i] [j] );
                }
        }

for ( k = 0; k < NUM_CAP_BUD_PROJECTS; k++ )
{
fprintf(fptro, "\n\nPRODUCTION INCREASES FROM PROJECT %d", k );
for ( i = 0; i < NUM_PROD_WC; i++ )
        {
        fprintf(fptro,"\nProduction Increases At Work Center %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                {
                fscanf(fptri, "%d", &CL[k][i][j] );
                fprintf(fptro, "%d\t", CL[k][i][j] );
                }
        }
}

fprintf(fptro, "\n\nREVENUE");
for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
        fprintf(fptro, "\nRevenue from product %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
        {
        fscanf(fptri, "%f", &REVENUE[i][j] );
        fprintf(fptro, "%8.2f ", REVENUE[i][j] );
        }
        }

fprintf(fptro, "\n\nINTEREST RATE\n");
fscanf(fptri, "%f", &ACC );
fprintf(fptro, "ACC = %4.3f", ACC );

fprintf(fptro, "\n\nINFLATION RATE\n" );
fscanf(fptri, "%f", &inflation );
fprintf(fptro, "inflation = %4.3f", inflation );

fprintf(fptro, "\n\nREGULAR TIME COSTS");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read regular time cost */
        {
        fprintf(fptro,"\nRegular Time Cost for Production Work Center %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                {
```

```
                    fscanf(fptri, "%f", &WLT[i] [j] );
                    fprintf(fptro, "%4.2f\t", WLT[i] [j] );
                    }
            }


        fprintf(fptro, "\n\nPRINT OVER TIME COST");
        for ( i = 0; i < NUM_PROD_WC; i++ ) /* read overtime cost */
                {
                fprintf(fptro,"\nOver Time Cost for Production Work Center %d\n", i );
                for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                        {
                        fscanf(fptri, "%f", &OLT [i] [j] );
                        fprintf(fptro, "%4.2f\t", OLT [i] [j] );
                        }
                }


        fprintf(fptro, "\n\nBEGINNING HOURS AT PRODUCTION WC'S\n");
        for ( i = 0; i < NUM_PROD_WC; i++ ) /* read beginning production time */
                {
                fscanf(fptri, "%f", &Bl [i] );
                fprintf(fptro, "%4.2f\t", Bl [i] );
                }

        fprintf(fptro, "\n\nBEGINNING UNITS OF CAPACITY AT NON-PRODUCTION WC'S\n");
        for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read beginning non-production time */
                {
                fscanf(fptri, "%f", &Bn [i] );
                fprintf(fptro, "%4.2f\t", Bn [i] );
                }

        fprintf(fptro, "\n\nINCREASES FROM NON-PRODUCTION CAPITAL BUDGTEING
PROJECT ONE\n");
        for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read increase from non-production cap bud
project 1 */
        {
        fprintf(fptro, "\nIncreases at WC%d\n", i );
                for ( j = 0; j < NUM_PERIODS; j++ )
                {
                fscanf(fptri, "%d", &CN [i][j] );
                fprintf(fptro, "%d  ", CN [i][j] );
                }
        }

        fprintf(fptro, "\n\nCOST OF PRODUCTION CAPITAL BUDGETING PROJECTS");
        for ( i = 0; i < NUM_CAP_BUD_PROJECTS; i++ ) /* read cost of each capital project */
                {
                fprintf(fptro,"\nCapital Budgeting Project %d\n", i );
                for ( j = 0; j < NUM_PERIODS; j++ )
                        {
```

```
                    fscanf(fptri, "%f", &cl [i] [j] );
                    fprintf(fptro, "%8.2f  ", cl [i] [j] );
                    }
            }

        fprintf(fptro, "\n\nCOST OF NON-PRODUCTION CAPITAL BUDGETING PROJECT\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read cost of capital project */
                {
                fscanf(fptri, "%f", &cn [i] );
                fprintf(fptro, "%8.2f ", cn [i] );
                }

        fprintf(fptro, "\n\nCOST OF SUPPORT CAPITAL BUDGETING PROJECT\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read cost of capital project */
                {
                fscanf(fptri, "%f", &cv [i] );
                fprintf(fptro, "%8.2f ", cv [i] );
                }


        fprintf(fptro, "\n\nMAXIMUM ALLOWABLE PRODUCTION PROJECT EXPENSE\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable production capital project
expense */
                {
                fscanf(fptri, "%f", &El [i] );
                fprintf(fptro, "%8.2f  ", El [i] );
                }

        fprintf(fptro, "\n\nMAXIMUM ALLOWABLE NON-PRODUCTION PROJECT EXPENSE\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable non-production capital project
expense */
                {
                fscanf(fptri, "%f", &En [i] );
                fprintf(fptro, "%8.2f ", En [i] );
                }

        fprintf(fptro, "\n\nMAXIMUM ALLOWABLE SUPPORT PROJECT EXPENSE\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable support capital project
expense */
                {
                fscanf(fptri, "%f", &Ev [i] );
                fprintf(fptro, "%8.2f ", Ev [i] );
                }

        fprintf(fptro, "\n\nMAXIMUM FRACTION REGULAR TIME LABOR\n");
        for ( i = 0; i < NUM_PROD_WC; i++ ) /* read maximum fraction regular time labor */
                {
                fprintf(fptro, "\nWC %d\n", i );
                for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                        {
                        fscanf(fptri, "%f", &LAMBDA[i][j] );
```

```
                    fprintf(fptro, "%4.2f\t", LAMBDA[i][j] );
                    }
            }

    fprintf(fptro, "\n\nPRODUCT LIFE LENGTH\n");
    for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read product life lengths */
            {
            fscanf(fptri, "%d", &PLJ [i] );
            fprintf(fptro, "%d\t", PLJ [i] );
            }

    fprintf(fptro, "\n\nPRODUCT WARRANTEE LENGTH\n");
    for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read product warrantee lengths */
            {
            fscanf(fptri, "%d", &PWJ [i] );
            fprintf(fptro, "%d\t", PWJ [i] );
            }

    fprintf(fptro, "\n\nSUPPORT HOURS FOR PRODUCTS OF AGE TAU");
    for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read support hours required for differring age
products */
            {
            fprintf(fptro,"\nSupport Hours for Product Number %d\n", i );
            for ( j = 0; j < PLJ[i]; j++ )
                    {
                    fscanf(fptri, "%f", &SJT [i] [j] );
                    fprintf(fptro, "%4.2f\t", SJT [i] [j] );
                    }
            }

    fprintf( fptro, "\n\nBEGINNING SUPPORT HOURS AVAILABLE\n" );
    fscanf(fptri, "%f", &S0 ); /* Read beginning support hours */
    fprintf(fptro, "%4.2f", S0 );

    fprintf(fptro, "\n\nMAXIMUM ALLOWABLE PRODUCTION COST\n");
    for ( i = 0; i < NUM_PROD_PERIODS; i++ ) /* read maximum allowable production cost */
            {
            fscanf(fptri, "%f", &PC [i] );
            fprintf(fptro, "%8.2f ", PC [i] );
            }

    fprintf(fptro, "\n\nAVERAGE PARTS COST TO REPAIR PRODUCT OF AGE TAU");
    for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read average parts cost to repair product of age tau */
            {
            fprintf(fptro,"\nAverage parts cost for Product Number %d\n", i );
            for ( j = 0; j < PLJ[i]; j++ )
                    {
                    fscanf(fptri, "%f", &pj [i] [j] );
                    fprintf(fptro, "%4.2f\t", pj [i] [j] );
                    }
            }
```

```
fprintf(fptro, "\n\nMAXIMUM REPAIR PARTS COST\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read maximum repair parts cost */
        {
        fscanf(fptri, "%f", &PT [i] );
        fprintf(fptro, "%8.2f ", PT [i] );
        }

fprintf(fptro, "\n\nPARTS REVENUE FROM REPAIR OF PRODUCT OF AGE TAU");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read average parts cost to repair product of age tau */
        {
        fprintf(fptro,"\nAverage parts revenue for Product Number %d\n", i );
        for ( j = 0; j < PLJ[i]; j++ )
                {
                fscanf(fptri, "%f", &PR [i] [j] );
                fprintf(fptro, "%4.2f\t", PR [i] [j] );
                }
        }

fprintf(fptro, "\n\nMINIMUM ACCEPTABLE REVENUE\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read minimum acceptable revenue */
        {
        fscanf(fptri, "%f", &RT [i] );
        fprintf(fptro, "%8.2f ", RT [i] );
        }

fprintf(fptro, "\n\nREPAIR LABOR REVENUE PER HOUR\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read repair labor revenue per hour */
        {
        fscanf(fptri, "%f", &ft [i] );
        fprintf(fptro, "%4.2f  ", ft [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE PRODUCTION WC\n");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read base cost to operate production wc's */
        {
        fscanf(fptri, "%f", &BASE_COST_L [i] );
        fprintf(fptro, "%f\t", BASE_COST_L [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE NON-PRODUCTION WC\n");
for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read base cost to operate non-production wc's */
        {
        fscanf(fptri, "%f", &BASE_COST_N [i] );
        fprintf(fptro, "%f\t", BASE_COST_N [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE SUPPORT WC\n");
/* read base cost to operate support wc */
fscanf(fptri, "%f", &BASE_COST_V );
fprintf(fptro, "%f\t", BASE_COST_V );
```

```
        fprintf(fptro, "\n\nMATERIAL COST FOR PRODUCTION");
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read material cost for production */
                {
                fprintf(fptro,"\nMaterial cost for Product Number %d\n", i );
                for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                        {
                        fscanf(fptri, "%f", &MA [i] [j] );
                        fprintf(fptro, "%4.2f\t", MA [i] [j] );
                        }
                }


        fprintf(fptro, "\n\nREPAIR LABOR COST\n");
        for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read repair labor cost */
                {
                fscanf(fptri, "%f", &RC [i] );
                fprintf(fptro, "%4.2f  ", RC [i] );
                }

        fprintf(fptro, "\n\nCONSISTENT UNITS REQUIRED BY PRODUCT AT NON_PRODUCTION
WC");
        for ( k = 0; k < NUM_NON_PROD_WC; k++ )
        {
        fprintf(fptro, "\n\nUnits required at wc %d", k );
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read units required for non-production */
                {
                fprintf(fptro,"\n  Consistent non-production units for product number %d\n", i );
                fprintf(fptro, "  ");
                for ( j = 0; j < NUM_PERIODS; j++ )
                        {
                        fscanf(fptri, "%f", &G [k] [i] [j] );
                        fprintf(fptro, "%4.2f  ", G [k] [i] [j] );
                        }
                }
        }

        fprintf(fptro, "\n\nMAXIMUM PRODUCTION PERIODS FOR EACH PRODUCT\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                {
                fscanf(fptri, "%d", &max_periods[i] );
                fprintf(fptro, "%d\t", max_periods[i] );
                }

        fprintf(fptro, "\n\nEARLIEST PERIOD PRODUCTION OF A PRODUCT MAY
COMMENCE\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                {
                fscanf(fptri, "%d", &start_period[i] );
                fprintf(fptro, "%d\t", start_period[i] );
                }
```

```
fprintf(fptro, "\n\nINVENTORY HOLDING COSTS FOR PRODUCTS IN EACH PERIOD\n");
for ( i = 0; i < NUM_PRODUCTS; i++ )
{
        fprintf(fptro, "\nPRODUCT %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
        {
        fscanf(fptri, "%f", &INVENTORY[i][j] );
        fprintf(fptro, "%8.2f  ", INVENTORY[i][j] );
        }
}

fprintf(fptro, "\n\nIMPROVMENT IN HOURS FROM SUPPORT CAP BUD PROJECT\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ )
        {
        fscanf(fptri, "%d", &CV[i] );
        fprintf(fptro, "%d  ", CV[i] );
        }

fprintf(fptro, "\n\nBEGINNING INVENTORY FOR EACH PRODUCT\n");
for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
        fscanf(fptri, "%d", &BEGIN_INVENTORY[i] );
        fprintf(fptro, "%d\t", BEGIN_INVENTORY[i] );
        }

/* start read from mpss.rep file */
fprintf(fptro, "\n\n\nOUTPUT FROM STAGE TWO OPTIMIZATION");
fscanf(fptri2, "%s", TITLE );
fprintf(fptro,"\n\nINPUT IS FROM:  %s", TITLE );

/* strip off header information */
fscanf(fptri2, "%s", PROJECT );
fprintf(fptro, "\n\n%s ", PROJECT );
fscanf(fptri2, "%s", CONSTRAINT );
fprintf(fptro, "%s\n", CONSTRAINT );
for ( i = 0; i < 2; i++ )
        {
        fscanf(fptri2, "%s", DEMAND );
        fprintf(fptro, "%s ", DEMAND );
        fscanf(fptri2, "%s", EQUAL );
        fprintf(fptro, "%s ", EQUAL );
        fscanf(fptri2, "%s", DEMAND_VALUE );
        fprintf(fptro, "%s\n", DEMAND_VALUE );
        }
fscanf(fptri2, "%s", VAR_NAME );
fscanf(fptri2, "%s", VAR_VALUE );
fscanf(fptri2, "%s", REDUCED );
fscanf(fptri2, "%s", COST );

fprintf(fptro, "\n\nVALUES OF Z FROM STAGE TWO OPTIMIZATION\n");
```

```
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                {
                fscanf(fptri2, "%s %f %f", ZNAME, &Z[i], &ZJUNK );
                fprintf(fptro, "\n\n Z%d    %1.0f ", i, Z[i] );
                }

        fprintf(fptro, "\n\nVALUES OF SETUP VARIABLES FROM STAGE TWO
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
                for ( j = start_period[i]; j < start_period[i] + max_periods[i]; j++ )
                {
                fscanf(fptri2, "%s %f %f", DELTANAME, &DELTA[i][j], &DELTAJUNK );
                fprintf(fptro, "%s   %1.0f\n", DELTANAME, DELTA[i][j] );
                }
        }

        fprintf(fptro, "\n\nVALUE OF NON_PRODUCTION CAPITAL BUDGET VARIABLE FROM
STAGE TWO OPTIMIZATION\n");
        fscanf(fptri2, "%s %f %f", ENNAME, &EN, &ENJUNK );
        fprintf(fptro, "\n\nEN    %1.6f", EN );

        fprintf(fptro, "\n\nVALUES OF PRODUCTION VARIABLES FROM STAGE TWO
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
                for ( j = start_period[i]; j < start_period[i] + max_periods[i]; j++ )
                {
                fscanf(fptri2, "%s %f %f", XNAME, &X[i][j], &XJUNK );
                fprintf(fptro, "%s    %12.6f\n", XNAME, X[i][j] );
                }
        }

        fprintf(fptro, "\n\nVALUES OF INVENTORY VARIABLES FROM STAGE TWO
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
                for ( j = start_period[i]; j < start_period[i] + max_periods[i] - 1; j++ )
                {
                fscanf(fptri2, "%s %f %f", INAME, &I[i][j], &IJUNK );
                fprintf(fptro, "%s    %12.6f\n", INAME, I[i][j] );
                }
        }

        fprintf(fptro, "\n\nVALUES OF REGULAR TIME LABOR VARIABLES FROM STAGE TWO
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PROD_PERIODS; i++ )
        {
                for ( j = 0; j < NUM_PROD_WC; j++ )
                {
                fscanf(fptri2, "%s %f %f", wltNAME, &wlt[i][j], &wltJUNK );
```

```
                fprintf(fptro, "%s    %12.6f\n", wltNAME, wlt[i][j] );
                }
        }

        fprintf(fptro, "\n\nVALUES OF OVERTIME LABOR VARIABLES FROM STAGE TWO
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PROD_PERIODS; i++ )
        {
                for ( j = 0; j < NUM_PROD_WC; j++ )
                {
                fscanf(fptri2, "%s %f %f", oltNAME, &olt[i][j], &oltJUNK );
                fprintf(fptro, "%s    %12.6f\n", oltNAME, olt[i][j] );
                }
        }

        fclose(fptri2);
        fclose(fptri);

        last( d1, d2, CL, SETUP_HOURS, PRODUCTION_HOURS, REVENUE, WLT, OLT,
        Bl, cl, cv, El, Ev, LAMBDA, SJT, S0, PC, PLJ, PWJ, pj, PT, PR, RT,
        ft, MA, RC, max_periods, start_period, BASE_COST_L, BASE_COST_V, ACC,
        INVENTORY, CV, inflation, BEGIN_INVENTORY, Z, X );

        fclose(fptro);

} /* end main */
```

```
/* middle.h */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define NUM_PERIODS                     12
#define NUM_SUPPORT_PERIODS             11
#define HORIZON_PERIOD                  11
#define NUM_PROD_PERIODS                8
#define NUM_PRODUCTS                    5
#define NUM_PROD_WC                     3
#define NUM_NON_PROD_WC                 4
#define NUM_CAP_BUD_PROJECTS            3
#define NUM_NON_PROD_CAP_BUD_PROJ       1
#define AGE                             5
#define TRUE                            1
#define FALSE                           0


int project, i, j;

int setup_constraint;
int dec_var;
int age;
int inv_lim;
int must_fund;
int nprdwc;


/* base cost variables */
float TOT_BASE_COST;
float TOT_BASE_DISCOUNT;

/* discounted cost from repair labor */
float ADJ_LABOR_REP_COST[NUM_PRODUCTS][NUM_PERIODS];

/* discounted parts costs */
float ADJ_PART[NUM_PRODUCTS][NUM_PERIODS];

/* discounted labor revenue */
float ADJ_LABOR_REP_REV[NUM_PRODUCTS][NUM_SUPPORT_PERIODS];

/* production capital budgeting costs */
float TOTAL_CAP_BUD_EXPENSE[NUM_CAP_BUD_PROJECTS];

/* adjusted inventory cost; includes stocking fee, loss of interest on
  revenues and repair costs */
float ACTUAL_INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS - 1];

/* total support capital budgeting cost */
float TOTAL_SUPPORT_CAP_BUD_COST;
```

```
/* middle.c */
#include "middle.h"

/* Builds an MPS file for the middle stage of the two stage model */
void last( int d1[NUM_PROD_PERIODS], int d2[NUM_PROD_PERIODS],
int CL[NUM_CAP_BUD_PROJECTS][NUM_PROD_WC][NUM_PROD_PERIODS],
int SETUP_HOURS[NUM_PROD_WC] [NUM_PRODUCTS],
int PRODUCTION_HOURS[NUM_PROD_WC] [NUM_PRODUCTS],
float REVENUE [NUM_PRODUCTS][NUM_PROD_PERIODS],
float WLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float OLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float Bl [NUM_PROD_WC], float cl [NUM_CAP_BUD_PROJECTS] [NUM_PERIODS],
float cv [NUM_PERIODS], float El [NUM_PERIODS],
float Ev [NUM_PERIODS], float LAMBDA [NUM_PROD_WC][NUM_PROD_PERIODS],
float SJT [NUM_PRODUCTS][AGE], float S0, float PC [NUM_PROD_PERIODS],
int PLJ [NUM_PRODUCTS], int PWJ [NUM_PRODUCTS], float pj [NUM_PRODUCTS][AGE],
float PT [NUM_PERIODS], float PR [NUM_PRODUCTS][AGE], float RT [NUM_PERIODS],
float ft [NUM_PERIODS], float MA [NUM_PRODUCTS][NUM_PROD_PERIODS],
float RC [NUM_PERIODS], int max_periods[NUM_PRODUCTS],
int start_period[NUM_PRODUCTS], float BASE_COST_L[NUM_PROD_WC],
float BASE_COST_V, float ACC,
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS], int CV [NUM_PERIODS - 1],
float inflation, int BEGIN_INVENTORY[NUM_PRODUCTS],
float Z [NUM_PRODUCTS],
float X[NUM_PRODUCTS][NUM_PROD_PERIODS] )

{
FILE *mps;

register product, period, wc;

        /* open MPS output file */

        if ( (mps = fopen("mpms.dat","w")) == NULL ) /* open mps output file */
                {
                 printf("Can't open the mps output file.");
                 exit(0);
                }

        /* build total production capital budgeting costs */
        for ( project = 0; project < NUM_CAP_BUD_PROJECTS; project++ )
        {
                for ( period = 0; period < NUM_PERIODS; period++ )
                {
                TOTAL_CAP_BUD_EXPENSE[project] = 0;
                }
        }
        for ( project = 0; project < NUM_CAP_BUD_PROJECTS; project++ )
        {
                for ( period = 0; period < NUM_PERIODS; period++ )
                {
```

```
                TOTAL_CAP_BUD_EXPENSE[project] += cl[project][period] * pow( 1 + ACC,
HORIZON_PERIOD - period );
                }
        }

        /* build total future worth of repair revenues and costs  for each product by
        period it was produced */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                ADJ_LABOR_REP_REV[product][period] = 0;
                ADJ_LABOR_REP_COST[product][period] = 0;
                ADJ_PART[product][period] = 0;
                }
        }
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period =  start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                for ( age = PWJ[product]; age < PLJ[product]; age++ )
                ADJ_LABOR_REP_REV[product][period] += SJT[product][age]*ft[age] * pow( 1 +
ACC, HORIZON_PERIOD - period - age );
                }
        }
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period =  start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                for ( age = 0; age < PLJ[product]; age++ )
                ADJ_LABOR_REP_COST[product][period] += SJT[product][age]*RC[age] * pow( 1 +
ACC, HORIZON_PERIOD - period - age );
                }
        }
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period =  start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                for ( age = 0; age < PLJ[product]; age++ )
                {
                if ( age < PWJ[product] )
                        ADJ_PART[product][period] += - pj[product][age] * pow( 1 + inflation, period
+ age ) * pow( 1 + ACC, HORIZON_PERIOD - ( period + age ) );
                if ( age >= PWJ[product] )
                        ADJ_PART[product][period] += ( PR[product][age] - pj[product][age] ) * pow(
1 + inflation, period + age ) * pow( 1 + ACC, HORIZON_PERIOD - ( period + age ) );
                }
                }
```

```
        }

        /* actual inventory cost - includes loss of interest on revenues and adjusts repair costs and
revenues */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
                {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product] - 1; period++ )
                {
                ACTUAL_INVENTORY[product][period] =
                 ( INVENTORY[product][period]
                + REVENUE[product][period] ) * pow( 1 + ACC, HORIZON_PERIOD - period )
                + ADJ_LABOR_REP_REV[product][period]
                - ADJ_LABOR_REP_COST[product][period] +
                ADJ_PART[product][period] -
                ( REVENUE[product][period + 1] * pow( 1 + ACC, HORIZON_PERIOD - period - 1)
                + ADJ_LABOR_REP_REV[product][period + 1 ]
                - ADJ_LABOR_REP_COST[product][period + 1] +
                ADJ_PART[product][period + 1] );
                }
        }

        /* determine total support capital budget cost */
        TOTAL_SUPPORT_CAP_BUD_COST = 0;
        for ( period = 0; period < NUM_PERIODS; period++ )
                TOTAL_SUPPORT_CAP_BUD_COST += cv[period] * pow( 1 + ACC,
HORIZON_PERIOD - period );

        /* calculate total base costs at production and non-production wc's */
        TOT_BASE_COST = 0;
        TOT_BASE_DISCOUNT = 0;
        /* determine total base production cost in each period */
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                TOT_BASE_COST += BASE_COST_L[wc];

        /* add base cost of support to other base costs */
        TOT_BASE_COST += BASE_COST_V;

        /* Determine total discounted constant dollar revenues less costs */
        for ( period = 0; period < NUM_PERIODS; period++ )
        {
        TOT_BASE_DISCOUNT += TOT_BASE_COST * pow( 1 + inflation, period ) * pow( 1 +
ACC, HORIZON_PERIOD - period );
        }


        /* print header to name LP */
        fprintf(mps, "NAME    MIDDLE STAGE OF TWO STAGE MODEL MPS FILE (MAX)\n");

        /* products funded: as a remark */
        fprintf(mps, "*\t");
```

```
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                fprintf(mps, "Z%d = %1.0f   ", i, Z[i] );
        fprintf(mps, "\n");

        /* print header to start rows */
        fprintf(mps, "ROWS\n");
        /* develop row for the objective function */
        fprintf(mps, " N  obj\n");

        /* products funded */
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                fprintf(mps, " E  prod%d\n", i);

        /* production levels */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                fprintf(mps, " L  prodct%d%d\n", product, period );
                }
        }

        /* production capital budgeting project mutually exclusive row */
        fprintf(mps, " L  capmutex\n");

        /* production capital budget maximum cost per period */
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, " L  mxbudl%d\n", period );

        /* support capital budget maximum cost per period */
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, " L  mxbudv%d\n", period );

        /* develop rows for setup constraints */
        setup_constraint = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                setup_constraint++;
                fprintf(mps, " L  setup%d\n", setup_constraint);
                }
        }

        /* develop rows to set upper limits on inventory */
        inv_lim = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
```

```
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product] - 1; period++ )
                {
                inv_lim++;
                fprintf(mps, " L  invlim%d\n", inv_lim);
                }
        }

        /* develop rows for product demand */
        product = 0;
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                fprintf(mps, " L  demand1%d\n",  period );

        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                fprintf(mps, " L  demand2%d\n", period );

        /* develop rows for prodwc constraint */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, " L  prodwc%d%d\n", period, wc );
                        }
                }


        /* develop values of WLT and OLT */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, " L  time%d%d\n", period, wc );
                        }
                }

        /* develop maximum values of WLT */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, " L  regtm%d%d\n", period, wc );
                        }
                }

        /* develop maximum values of OLT */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, " L  otime%d%d\n", period, wc );
                        }
```

```
        }

/* develop maximum production cost in each period */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        fprintf(mps, " L  mxprcst%d\n", period );


/* develop maximum support hours in each period rows */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, " L  prdsup%d\n", period );

/* develop maximum support parts cost in each period */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, " L  prtsup%d\n", period );

/* develop minimum revenue for each period */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, " G  minrev%d\n", period );

/* print basecost constraint row */
fprintf(mps, " E  basecost\n");

/* set base capacity to Bl */
fprintf(mps, " E  basecap\n");

/* set beginning inventory variables equal to one */
i = 0;
for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
        if ( Z[product] == 1 )
        {
        if ( BEGIN_INVENTORY[product] > 0 )
                {
                i++;
                fprintf(mps, " E  beginv%d\n", i );
                }
        }
        }



/* print header to start columns */
fprintf(mps, "COLUMNS\n");

/* print product variables in obj f(x) */
for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
        fprintf(mps, "   Z%d      obj       1.0\n", i);
        fprintf(mps, "   Z%d      prod%d    1.0\n", i, i);
        }
```

```
        /* print production work center setup variables */
        setup_constraint = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                fprintf(mps, "    DELTA%d%-4dobj       -1.0\n", product, period );
                setup_constraint++;
                if ( setup_constraint < 10 && product < 3 )
                {
                fprintf(mps, "    DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d1[period] * 3 );
                }
                if ( setup_constraint >= 10 && product < 3)
                {
                fprintf(mps, "    DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d1[period] * 3 );
                }
                if ( setup_constraint < 10 && product >= 3 )
                {
                fprintf(mps, "    DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d2[period] * 3 );
                }
                if ( setup_constraint >= 10 && product >= 3)
                {
                fprintf(mps, "    DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d2[period] * 3 );
                }
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "    DELTA%d%-4dprodwc%d%d%12d\n", product, period,
period, wc, SETUP_HOURS[wc][product] );
                        }
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "    DELTA%d%-4dtime%d%d  %12d\n", product, period, period,
wc, SETUP_HOURS[wc][product] );
                        }
                }
        }

        /* print production capital budegting variables */
        for ( project = 0; project < NUM_CAP_BUD_PROJECTS; project++ )
                {
                fprintf(mps, "    EL%d      obj        -1.0\n", project);
                if ( project == 0 || project == 2 )
                fprintf(mps, "    EL%d      capmutex       1.0\n", project );
                for ( period = 0; period < NUM_PERIODS; period++ )
                        fprintf(mps, "    EL%d      mxbudl%d     %-12.2f\n", project, period,
cl[project][period] );
```

```
                for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                        {
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                                {
                                fprintf(mps, "   EL%d      prodwc%d%d    -%-12d\n",
project, period, wc, CL[project][wc][period] );
                                fprintf(mps, "   EL%d      regtm%d%d     -%-12d\n", project,
period, wc, CL[project][wc][period] );
                                fprintf(mps, "   EL%d      otime%d%d     -%-12d\n", project,
period, wc, CL[project][wc][period] );
                                }
                        }
                }

        /* print support capital budgeting project variable */
        product = 0;
        fprintf(mps, "   EV       obj       -1.0\n");
        for ( period = 0; period < NUM_PERIODS; period++ )
                                fprintf(mps, "   EV       mxbudv%d     %-12.2f\n", period, cv[period]
);

        for ( period = 0; period < NUM_PROD_PERIODS + 3; period++ )
                fprintf(mps, "   EV       prdsup%d     -%-12d\n", period, CV[period] );

        /* print production variables */
        setup_constraint = 0;
        inv_lim = 1;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
                {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                fprintf(mps, "   X%d%-4d   obj       %-12.2f\n", product, period, (
REVENUE[product][period] - MA[product][period] ) * pow( 1 + ACC, HORIZON_PERIOD - period ) +
ADJ_LABOR_REP_REV[product][period] - ADJ_LABOR_REP_COST[product][period] +
ADJ_PART[product][period] );
                fprintf(mps, "   X%d%-4d   prodct%d%d       1.0\n", product, period, product, period
);
                setup_constraint++;
                if ( setup_constraint < 10 )
                {
                fprintf(mps, "   X%d%-4d   setup%d          1.0\n", product, period,
setup_constraint);
                }
                if ( setup_constraint >= 10 )
                {
                fprintf(mps, "   X%d%-4d   setup%d          1.0\n", product, period, setup_constraint);
                }
                if ( inv_lim >= 10 && period < start_period[product] + max_periods[product] - 1 )
                {
                fprintf(mps, "   X%d%-4d   invlim%d     -1.0\n", product, period, inv_lim);
```

```
                inv_lim++;
                }
                if ( inv_lim < 10 && period < start_period[product] + max_periods[product] - 1 )
                {
                fprintf(mps, "    X%d%-4d    invlim%d       -1.0\n", product, period, inv_lim);
                inv_lim++;
                }
                if ( product < 3 )
                {
                fprintf(mps, "    X%d%-4d    demand1%d          1.0\n", product, period, period );
                }
                if ( product >= 3 )
                {
                fprintf(mps, "    X%d%-4d    demand2%d          1.0\n", product, period, period );
                }
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "    X%d%-4d    prodwc%d%d%12d\n", product, period, period,
wc, PRODUCTION_HOURS[wc][product] );
                        }
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "    X%d%-4d    time%d%d  %12d\n", product, period, period, wc,
PRODUCTION_HOURS[wc][product] );
                        }
                fprintf(mps, "    X%d%-4d    mxprcst%d     %-12.2f\n", product, period, period,
MA[product][period] );
                for ( age = 0; age < PLJ[product]; age++ ) /* support hours in each period */
                        fprintf(mps, "    X%d%-4d    prdsup%d      %-12.2f\n", product, period, period
+ age, SJT[product][age] );
                for ( age = 0; age < PLJ[product]; age++ ) /* support cost in each period */
                        fprintf(mps, "    X%d%-4d    prtsup%d      %-12.2f\n", product, period, period
+ age, pj[product][age] * pow( 1 + inflation, period + age ) );
                for ( age = 0; age < PLJ[product]; age++ ) /* revenues in each period */
                        {
                        if ( age == 0 )
                                fprintf(mps, "    X%d%-4d    minrev%d      %-12.2f\n", product,
period, period, PR[product][age] * pow( 1 + inflation, period ) + ft[period] * SJT[product][age] +
REVENUE[product][period] );
                        else
                                fprintf(mps, "    X%d%-4d    minrev%d      %-12.2f\n", product,
period, period + age, PR[product][age] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age] );
                        }
                }
        }
        /* print inventory variables */
        inv_lim = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
                {
```

```
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product] - 1; period++ )
                {
                fprintf(mps, "   I%d%-4d   obj           -1.0\n", product, period );
                inv_lim++;
                if ( inv_lim < 10 )
                {
                fprintf(mps, "   I%d%-4d   invlim%d       1.0\n", product, period, inv_lim);
                }
                if ( inv_lim >= 10 )
                {
                fprintf(mps, "   I%d%-4d   invlim%d      1.0\n", product, period, inv_lim);
                }
                if ( product < 3 )
                {
                fprintf(mps, "   I%d%-4d   demand1%d        -1.0\n", product, period, period );
                fprintf(mps, "   I%d%-4d   demand1%d         1.0\n", product, period, period + 1 );
                }
                if ( product >= 3 )
                {
                fprintf(mps, "   I%d%-4d   demand2%d        -1.0\n", product, period, period );
                fprintf(mps, "   I%d%-4d   demand2%d         1.0\n", product, period, period + 1 );
                }
                for ( age = 0; age < PLJ[product]; age++ ) /* adj period support hours */
                {
                        if ( age == 0 )
                                {
                                fprintf(mps, "   I%d%-4d   prdsup%d     -%-12.2f\n", product,
period, period, SJT[product][age] );
                                }
                        if ( age > 0 && age < PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   prdsup%d      %-12.2f\n", product,
period, period + age, SJT[product][age - 1] - SJT[product][age] );
                                }
                        if ( age == PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   prdsup%d      %-12.2f\n", product,
period, period + age, SJT[product][age - 1] - SJT[product][age] );
                                fprintf(mps, "   I%d%-4d   prdsup%d      %-12.2f\n", product,
period, period + age + 1, SJT[product][age] );
                                }
                }
                for ( age = 0; age < PLJ[product]; age++ ) /* adj period part sup cost */
                {
                        if ( age == 0 )
                                {
                                fprintf(mps, "   I%d%-4d   prtsup%d    -%-12.2f\n", product, period,
period, pj[product][age] * pow( 1 + inflation, period ) );
                                }
                        if ( age > 0 && age < PLJ[product] - 1 )
```

```
                                {
                                fprintf(mps, "   I%d%-4d   prtsup%d    %-12.2f\n", product, period,
period + age, pj[product][age - 1] * pow( 1 + inflation, period + age ) - pj[product][age] * pow( 1 +
inflation, period + age ) );
                                }
                        if ( age == PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   prtsup%d    %-12.2f\n", product, period,
period + age, pj[product][age - 1] * pow( 1 + inflation, period + age ) - pj[product][age] * pow( 1 +
inflation, period + age ) );
                                fprintf(mps, "   I%d%-4d   prtsup%d    %-12.2f\n", product, period,
period + age + 1, pj[product][age] * pow( 1 + inflation, period + age + 1 ) );
                                }
                }
                for ( age = 0; age < PLJ[product]; age++ ) /* adj period revenue */
                        {
                        if ( age == 0 )
                                {
                                fprintf(mps, "   I%d%-4d   minrev%d     -%-12.2f\n", product,
period, period, PR[product][age] * pow( 1 + inflation, period ) + ft[period] * SJT[product][age] +
REVENUE[product][period] );
                                }
                        if ( age == 1 )
                                {
                                fprintf(mps, "   I%d%-4d   minrev%d      %-12.2f\n", product,
period, period + age, PR[product][age - 1] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age - 1] + REVENUE[product][period] * pow( 1 + inflation, age ) - PR[product][age] *
pow( 1 + inflation, period + age ) - ft[period + age] * SJT[product][age] );
                                }
                        if ( age > 1 && age < PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   minrev%d      %-12.2f\n", product,
period, period + age, PR[product][age - 1] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age - 1] - PR[product][age] * pow( 1 + inflation, period + age ) - ft[period + age] *
SJT[product][age] );
                                }
                        if ( age == PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   minrev%d      %-12.2f\n", product,
period, period + age, PR[product][age - 1] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age - 1] - PR[product][age] * pow( 1 + inflation, period + age ) - ft[period + age] *
SJT[product][age] );
                                fprintf(mps, "   I%d%-4d   minrev%d      %-12.2f\n", product,
period, period + age + 1, PR[product][age] * pow( 1 + inflation, period + age + 1) + ft[period + age + 1] *
SJT[product][age] );
                                }
                        }

                }
        }
```

```
        /* print regular time labor variables */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                 for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "   WLT%d%-4d  obj        -1.0\n", period, wc );
                        fprintf(mps, "   WLT%d%-4d  time%d%d      -1.0\n", period, wc, period, wc );
                        fprintf(mps, "   WLT%d%-4d  regtm%d%d      %-12.4f\n", period, wc, period,
wc, 1/LAMBDA[wc][period] );
                        fprintf(mps, "   WLT%d%-4d  mxprcst%d     %-12.2f\n", period, wc, period,
WLT[wc][period] );
                        }
                }

        /* print over time labor variables */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "   OLT%d%-4d  obj        -1.0\n", period, wc );
                        fprintf(mps, "   OLT%d%-4d  time%d%d      -1.0\n", period, wc, period, wc );
                        fprintf(mps, "   OLT%d%-4d  otime%d%d      %-12.4f\n", period, wc, period,
wc, 1/( 1 - LAMBDA[wc][period] ) );
                        fprintf(mps, "   OLT%d%-4d  mxprcst%d     %-12.2f\n", period, wc, period,
OLT[wc][period] );
                        }
                }

        /* print base costs constraint */
        fprintf(mps, "   BASECOST  obj       -1.0\n" );
        fprintf(mps, "   BASECOST  basecost      1.0\n");

        /* print base production capacity */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "   Bl      regtm%d%d   -%-12.3f\n", period, wc, Bl[wc] );
                        fprintf(mps, "   Bl      otime%d%d   -%-12.3f\n", period, wc, Bl[wc] );
                        }
                }

        fprintf(mps, "   Bl      basecap       1.0\n");

        /* print beginning inventory variables */
        i = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                if ( Z[product] == 1 && BEGIN_INVENTORY[product] > 0 )
                {
                if ( product < 3 )
```

```
                {
                fprintf(mps, "    BI%d      obj       %-12.2f\n", product, REVENUE[product][0] * pow(
1 + ACC, HORIZON_PERIOD ) );
                fprintf(mps, "    BI%d      demand10    %-12d\n", product,
BEGIN_INVENTORY[product] );
                }
                if ( product >= 3 )
                {
                fprintf(mps, "    BI%d      obj       -%-12.2f\n", product, REVENUE[product][0] * pow(
1 + ACC, HORIZON_PERIOD ) );
                fprintf(mps, "    BI%d      demand20    %-12d\n", product,
BEGIN_INVENTORY[product] );
                }
                i++;
                fprintf(mps, "    BI%d      beginv%d    1.0\n", product, i );
                }
        }



        /* print header to start rows */
        fprintf(mps, "RHS\n");

    /* products funded */
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                fprintf(mps, "    RHS       prod%d        %1.0f\n", i, Z[i] );

        /* production levels */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                fprintf(mps, "    RHS       prodct%d%d    %12.6f\n", product, period, X[product][period]
- 0.00001 );
                }
        }

        /* print setup right hand sides */
        fprintf(mps, "    RHS       capmutex      1.0\n");

        /* print maximum production capital budget expense RHS */
        for ( period = 0; period < NUM_PERIODS; period++ )
                        fprintf(mps, "    RHS       mxbudl%d    %-12.2f\n", period, El[period] );

        /* print support capital budget maximum cost per period */
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, "    RHS       mxbudv%d    %-12.2f\n", period, Ev[period] );

        /* print maximum production (demand) RHS */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
```

```
            fprintf(mps, "   RHS      demand1%d        %d\n", period, d1[period] );

        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
            fprintf(mps, "   RHS      demand2%d        %d\n", period, d2[period] );

        /* print production work center RHS */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
            {
            for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                {
                fprintf(mps, "   RHS      prodwc%d%d       %-13.2f\n", period, wc, Bl [wc] );
                }
            }

        /* print maximum production cost RHS */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
            fprintf(mps, "   RHS      mxprcst%d     %-12.2f\n", period, PC[period] );

        /* print maximum support hours RHS */
        for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
            fprintf(mps, "   RHS      prdsup%d      %-12.2f\n", period, S0 );

        /* print maximum support parts cost RHS */
        for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
            fprintf(mps, "   RHS      prtsup%d      %-12.2f\n", period, PT[period] );

        /* print minimum revenue for each period RHS */
        for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
            fprintf(mps, "   RHS      minrev%d      %-12.2f\n", period, RT[period] );

        fprintf(mps, "   RHS      basecost      1.0\n");
        fprintf(mps, "   RHS      basecap       1.0\n");

        /* print beginning inventory RHS */
        i = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
        if ( Z[product] == 1 )
        {
        if ( BEGIN_INVENTORY[product] > 0 )
            {
            i++;
            fprintf(mps, "   RHS      beginv%d      1.0\n", i );
            }
        }
        }

        fprintf(mps, "ENDATA");
        fclose(mps);
}
```

APPENDIX E: Program Listing for Stage Three of the Sequential Model

```
/* lastmain.h */
/* header file for lastmain.c */

#include <stdio.h>
#include <stdlib.h>
#define NUM_PERIODS                     12
#define NUM_SUPPORT_PERIODS             11
#define HORIZON_PERIOD                  11
#define NUM_PROD_PERIODS                8
#define NUM_PRODUCTS                    5
#define NUM_PROD_WC                     3
#define NUM_NON_PROD_WC                 4
#define NUM_CAP_BUD_PROJECTS            3
#define NUM_NON_PROD_CAP_BUD_PROJ       1
#define AGE                             5 /* maximum life length of any product */

typedef unsigned int BitVector;

int k;
        /* INPUT PARAMETERS */
/* DEMAND */
int d1 [NUM_PROD_PERIODS]; /* Demand for subgroup one */
int d2 [NUM_PROD_PERIODS]; /* Demand for subgroup two */

/* PRODUCT AND PROJECT INTERACTION */
BitVector Mutually_Exclusive_Prod[NUM_PRODUCTS]; /* Mutually exclusive products */
BitVector Mutually_Exclusive_Proj[NUM_CAP_BUD_PROJECTS]; /* " " projects */
BitVector Contingency[NUM_PRODUCTS]; /* Contingency between products */
BitVector Must_Fund; /* Must fund a set of products */

/* INTEREST RATES */
float ACC;  /* Interest Rate */
float inflation; /* Inflation Rate */

/* PRODUCT REVENUES */
float REVENUE [NUM_PRODUCTS][NUM_PROD_PERIODS]; /* Product revenue */

/* PRE-PRODUCTION/POST-SUPPORT PRODUCT INPUTS */
float G[NUM_NON_PROD_WC][NUM_PRODUCTS][NUM_PERIODS]; /* No. consistent units req by
product at non-production wc's  */

/* LABOR INPUTS */
float WLT [NUM_PROD_WC] [NUM_PROD_PERIODS]; /* reg time cost by wc */
float OLT [NUM_PROD_WC] [NUM_PROD_PERIODS]; /* ot time cost by wc */
float LAMBDA[NUM_PROD_WC][NUM_PROD_PERIODS]; /* Fraction labor regular time */

/* PRODUCTION INPUTS */
int SETUP_HOURS[NUM_PROD_WC] [NUM_PRODUCTS]; /* setup hours for prod by wc */
int PRODUCTION_HOURS[NUM_PROD_WC] [NUM_PRODUCTS]; /* production hrs by prod & wc */
float PC[NUM_PROD_PERIODS]; /* Maximum allowable production cost */
float MA[NUM_PRODUCTS][NUM_PROD_PERIODS]; /* Material cost for production */
```

```
/* SUPPORT VARIABLES */
float SJT[NUM_PRODUCTS][AGE]; /* Support hours required for product of different ages */
int PLJ[NUM_PRODUCTS]; /* Product life length */
int PWJ[NUM_PRODUCTS]; /* Warrantee length */
float pj[NUM_PRODUCTS][AGE]; /* Average parts cost */
float PT[NUM_PERIODS]; /* Maximum repair parts cost */
float PR[NUM_PRODUCTS][AGE]; /* Parts revenue */
float RT[NUM_PERIODS]; /* Minimum acceptable revenue */
float ft[NUM_PERIODS]; /* Repair labor revenue per hour */
float RC[NUM_PERIODS]; /* Repair labor cost */

/* INVENTORY INPUTS */
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS]; /* inventory holding costs */
int BEGIN_INVENTORY[NUM_PRODUCTS];

/* CAPITAL BUDGETING INPUTS */
float Bl[NUM_PROD_WC]; /* Beginning hours at production wc's */
float Bn[NUM_NON_PROD_WC]; /* Beginning units of capacity at non-production wc's */
float S0; /* Beginning support hours available */
float cl[NUM_CAP_BUD_PROJECTS][NUM_PERIODS]; /* cost of production cap budget projects */
float cn[NUM_PERIODS]; /* cost of non-prod cap budget projects */
float cv[NUM_PERIODS]; /* cost of support capital budgeting project in each period */
float El[NUM_PERIODS]; /* maximum allowable prod project expense */
float En[NUM_PERIODS]; /* maximum allowable pre-production project expense */
float Ev[NUM_PERIODS]; /* maximum allowable post support wc project expense */
float BASE_COST_L[NUM_PROD_WC]; /* Base cost to operate production wc's */
float BASE_COST_N[NUM_NON_PROD_WC]; /* Base cost to operate non-prod wc's */
float BASE_COST_V; /* Base support cost */
int CL [NUM_CAP_BUD_PROJECTS][NUM_PROD_WC][NUM_PROD_PERIODS]; /* improvement at
WC's due to cap bud proj 1 */
int CV [NUM_PERIODS - 1]; /* improvement in support hours from support cap bud project */
int CN[NUM_NON_PROD_WC][NUM_PERIODS]; /* Increase at non-prod wc's from cap bud project */

/* STARTING AND ENDING PERIOD INPUTS */
int max_periods[NUM_PRODUCTS]; /* maximum number of periods a product may be produced */
int start_period[NUM_PRODUCTS]; /* earliest period a product can be produced */

/* VARIABLE VALUES FROM mpfs.rep, i.e., first stage optimization */
char TITLE[15];
char ZNAME[2];
float Z[NUM_PRODUCTS];
float ZJUNK;
char DELTANAME[7];
float DELTA[NUM_PRODUCTS][NUM_PROD_PERIODS];
float DELTAJUNK;
char ENNAME[2];
float EN;
float ENJUNK;
char ELNAME[3];
float EL[NUM_CAP_BUD_PROJECTS];
```

```
float ELJUNK;
char EVNAME[2];
float EV;
float EVJUNK;
char XNAME[3];
float X[NUM_PRODUCTS][NUM_PROD_PERIODS];
float XJUNK;
char INAME[3];
float I[NUM_PRODUCTS][NUM_PROD_PERIODS - 1];
float IJUNK;
char wltNAME[5];
float wlt[NUM_PROD_PERIODS][NUM_PROD_WC];
float wltJUNK;
char oltNAME[5];
float olt[NUM_PROD_PERIODS][NUM_PROD_WC];
float oltJUNK;

/* string variables to strip mpss.rep off header information */
char PROJECT[8];
char CONSTRAINT[14];
char DEMAND[2];
char EQUAL[1];
char DEMAND_VALUE[3];
char VAR_NAME[8];
char VAR_VALUE[5];
char REDUCED[7];
char COST[4];


/* periods that production begins and ends */
int begin_period;
int end_period;

/* prototype of mpss function */
void last( int d1[NUM_PROD_PERIODS], int d2[NUM_PROD_PERIODS],
int CL[NUM_CAP_BUD_PROJECTS][NUM_PROD_WC][NUM_PROD_PERIODS],
int SETUP_HOURS[NUM_PROD_WC] [NUM_PRODUCTS],
int PRODUCTION_HOURS[NUM_PROD_WC] [NUM_PRODUCTS],
float REVENUE [NUM_PRODUCTS][NUM_PROD_PERIODS],
float WLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float OLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float Bl [NUM_PROD_WC], float cl [NUM_CAP_BUD_PROJECTS] [NUM_PERIODS],
float cv [NUM_PERIODS], float El [NUM_PERIODS],
float Ev [NUM_PERIODS], float LAMBDA [NUM_PROD_WC][NUM_PROD_PERIODS],
float SJT [NUM_PRODUCTS][AGE], float S0, float PC [NUM_PROD_PERIODS],
int PLJ [NUM_PRODUCTS], int PWJ [NUM_PRODUCTS], float pj [NUM_PRODUCTS][AGE],
float PT [NUM_PERIODS], float PR [NUM_PRODUCTS][AGE], float RT [NUM_PERIODS],
float ft [NUM_PERIODS], float MA [NUM_PRODUCTS][NUM_PROD_PERIODS],
float RC [NUM_PERIODS], int max_periods[NUM_PRODUCTS],
int start_period[NUM_PRODUCTS], float BASE_COST_L[NUM_PROD_WC],
float BASE_COST_V, float ACC,
```

```
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS], int CV [NUM_PERIODS - 1],
float inflation, int BEGIN_INVENTORY[NUM_PRODUCTS],
float Z [NUM_PRODUCTS],
float X[NUM_PRODUCTS][NUM_PROD_PERIODS] );
```

```
/* lastmain.c */
/* main routine to build an MPS file for the final stage of the
   two stage model */

/* input data from input.dat and mpms.rep */
/* outputs data to outputl.dat and mpls.dat */

#include "lastmain.h"

main(void)

{

        FILE *fptri, *fptri2, *fptro;
        register i, j;

        /* read inputs from a file and print back to output file */

        if ( (fptri = fopen("input.dat","r")) == NULL ) /* open input file */
                {
                 printf("Can't open the input file.");
                 exit(0);
                }

        if ( (fptri2 = fopen("mpms.rep","r")) == NULL ) /* open input file */
                {
                 printf("Can't open the input file.");
                 exit(0);
                }


        if ( ( fptro = fopen("outputl.dat", "w") ) == NULL ) /* open output file */
                {
                printf("Can't open the output file.");
                exit(0);
                }


        fprintf(fptro, "FILE NAME: outputs.dat\n\n");
        fscanf(fptri, "%s", TITLE );
        fprintf(fptro, "INPUT FILE IS: %s\n", TITLE );

        fprintf(fptro,"\n\n                LIST OF INPUTS AND OUTPUTS\n");
        fprintf(fptro,"                _____________________________\n\n\n");
        fprintf(fptro,"A.  Input Data\n");

        fprintf(fptro, "\n\n");


        /* Read data from input file */
        fprintf(fptro, "DEMAND FOR SUBGROUP ONE\n");
```

```
for(i = 0; i < NUM_PROD_PERIODS; i++) /* read demand for group 1 */
        {
        fscanf(fptri, "%d", &d1[i]);
        fprintf(fptro, "%1d\t", d1[i]);
        }

fprintf(fptro, "\nDEMAND FOR SUBGROUP TWO\n");
for(i = 0; i < NUM_PROD_PERIODS; i++) /* read demand for group 2 */
        {
        fscanf(fptri, "%d", &d2[i]);
        fprintf(fptro, "%1d\t", d2[i]);
        }

fprintf(fptro, "\n\nMUST FUND\n");
fscanf(fptri, "%d", &Must_Fund );
fprintf(fptro, "%4d", Must_Fund );

fprintf(fptro, "\n\nMUTUALLY EXCLUSIVE PRODUCTS\n");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read mutually exclusive */
        {
        fscanf(fptri, "%d", &Mutually_Exclusive_Prod[i]);
        fprintf(fptro, "%4d\t", Mutually_Exclusive_Prod[i]);
        }


fprintf(fptro, "\n\nMUTUALLY EXCLUSIVE PROJECTS\n");
for ( i = 0; i < NUM_CAP_BUD_PROJECTS; ++i ) /* read mut ex cap budget */
        {
        fscanf(fptri, "%d", &Mutually_Exclusive_Proj[i]);
        fprintf(fptro, "%4d\t", Mutually_Exclusive_Proj[i]);
        }

fprintf(fptro, "\n\nPRODUCT CONTINGENCY\n");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read in product contingency */
        {
        fscanf(fptri, "%d", &Contingency[i] );
        fprintf(fptro, "%4d\t", Contingency[i] );
        }

fprintf(fptro, "\n\nPRINT SETUP HOURS");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read setup hours */
        {
        fprintf(fptro,"\nSetup Hours At Work Center %d for Each Product\n", i );
        for ( j = 0; j < NUM_PRODUCTS; j++ )
                {
                fscanf(fptri, "%d", &SETUP_HOURS[i] [j] );
                fprintf(fptro, "%d\t", SETUP_HOURS[i] [j]);
                }
        }
```

```
fprintf(fptro, "\n\nPRINT PRODUCTION HOURS");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read production hours */
        {
        fprintf(fptro,"\nProduction Hours At Work Center %d for Each Product\n", i );
        for ( j = 0; j < NUM_PRODUCTS; j++ )
                {
                fscanf(fptri, "%d", &PRODUCTION_HOURS[i] [j] );
                fprintf(fptro, "%d\t", PRODUCTION_HOURS[i] [j] );
                }
        }

for ( k = 0; k < NUM_CAP_BUD_PROJECTS; k++ )
{
fprintf(fptro, "\n\nPRODUCTION INCREASES FROM PROJECT %d", k );
for ( i = 0; i < NUM_PROD_WC; i++ )
        {
        fprintf(fptro,"\nProduction Increases At Work Center %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                {
                fscanf(fptri, "%d", &CL[k][i][j] );
                fprintf(fptro, "%d\t", CL[k][i][j] );
                }
        }
}

fprintf(fptro, "\n\nREVENUE");
for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
        fprintf(fptro, "\nRevenue from product %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
        {
        fscanf(fptri, "%f", &REVENUE[i][j] );
        fprintf(fptro, "%8.2f ", REVENUE[i][j] );
        }
        }

fprintf(fptro, "\n\nINTEREST RATE\n");
fscanf(fptri, "%f", &ACC );
fprintf(fptro, "ACC = %4.3f", ACC );

fprintf(fptro, "\n\nINFLATION RATE\n" );
fscanf(fptri, "%f", &inflation );
fprintf(fptro, "inflation = %4.3f", inflation );

fprintf(fptro, "\n\nREGULAR TIME COSTS");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read regular time cost */
        {
        fprintf(fptro,"\nRegular Time Cost for Production Work Center %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                {
                fscanf(fptri, "%f", &WLT[i] [j] );
```

```
                    fprintf(fptro, "%4.2f\t", WLT[i] [j] );
                    }
            }


    fprintf(fptro, "\n\nPRINT OVER TIME COST");
    for ( i = 0; i < NUM_PROD_WC; i++ ) /* read overtime cost */
            {
            fprintf(fptro,"\nOver Time Cost for Production Work Center %d\n", i );
            for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                    {
                    fscanf(fptri, "%f", &OLT [i] [j] );
                    fprintf(fptro, "%4.2f\t", OLT [i] [j] );
                    }
            }


    fprintf(fptro, "\n\nBEGINNING HOURS AT PRODUCTION WC'S\n");
    for ( i = 0; i < NUM_PROD_WC; i++ ) /* read beginning production time */
            {
            fscanf(fptri, "%f", &Bl [i] );
            fprintf(fptro, "%4.2f\t", Bl [i] );
            }

    fprintf(fptro, "\n\nBEGINNING UNITS OF CAPACITY AT NON-PRODUCTION WC'S\n");
    for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read beginning non-production time */
            {
            fscanf(fptri, "%f", &Bn [i] );
            fprintf(fptro, "%4.2f\t", Bn [i] );
            }

    fprintf(fptro, "\n\nINCREASES FROM NON-PRODUCTION CAPITAL BUDGTEING
PROJECT ONE\n");
    for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read increase from non-production cap bud
project 1 */
    {
    fprintf(fptro, "\nIncreases at WC%d\n", i );
            for ( j = 0; j < NUM_PERIODS; j++ )
            {
            fscanf(fptri, "%d", &CN [i][j] );
            fprintf(fptro, "%d  ", CN [i][j] );
            }
    }

    fprintf(fptro, "\n\nCOST OF PRODUCTION CAPITAL BUDGETING PROJECTS");
    for ( i = 0; i < NUM_CAP_BUD_PROJECTS; i++ ) /* read cost of each capital project */
            {
            fprintf(fptro,"\nCapital Budgeting Project %d\n", i );
            for ( j = 0; j < NUM_PERIODS; j++ )
                    {
                    fscanf(fptri, "%f", &cl [i] [j] );
```

```
                    fprintf(fptro, "%8.2f  ", cl [i] [j] );
                    }
            }

        fprintf(fptro, "\n\nCOST OF NON-PRODUCTION CAPITAL BUDGETING PROJECT\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read cost of capital project */
                {
                fscanf(fptri, "%f", &cn [i] );
                fprintf(fptro, "%8.2f ", cn [i] );
                }

        fprintf(fptro, "\n\nCOST OF SUPPORT CAPITAL BUDGETING PROJECT\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read cost of capital project */
                {
                fscanf(fptri, "%f", &cv [i] );
                fprintf(fptro, "%8.2f ", cv [i] );
                }


        fprintf(fptro, "\n\nMAXIMUM ALLOWABLE PRODUCTION PROJECT EXPENSE\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable production capital project
expense */
                {
                fscanf(fptri, "%f", &El [i] );
                fprintf(fptro, "%8.2f  ", El [i] );
                }

        fprintf(fptro, "\n\nMAXIMUM ALLOWABLE NON-PRODUCTION PROJECT EXPENSE\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable non-production capital project
expense */
                {
                fscanf(fptri, "%f", &En [i] );
                fprintf(fptro, "%8.2f ", En [i] );
                }

        fprintf(fptro, "\n\nMAXIMUM ALLOWABLE SUPPORT PROJECT EXPENSE\n");
        for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable support capital project
expense */
                {
                fscanf(fptri, "%f", &Ev [i] );
                fprintf(fptro, "%8.2f ", Ev [i] );
                }

        fprintf(fptro, "\n\nMAXIMUM FRACTION REGULAR TIME LABOR\n");
        for ( i = 0; i < NUM_PROD_WC; i++ ) /* read maximum fraction regular time labor */
                {
                fprintf(fptro, "\nWC %d\n", i );
                for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                        {
                        fscanf(fptri, "%f", &LAMBDA[i][j] );
                        fprintf(fptro, "%4.2f\t", LAMBDA[i][j] );
```

```
                }
        }

    fprintf(fptro, "\n\nPRODUCT LIFE LENGTH\n");
    for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read product life lengths */
        {
        fscanf(fptri, "%d", &PLJ [i] );
        fprintf(fptro, "%d\t", PLJ [i] );
        }

    fprintf(fptro, "\n\nPRODUCT WARRANTEE LENGTH\n");
    for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read product warrantee lengths */
        {
        fscanf(fptri, "%d", &PWJ [i] );
        fprintf(fptro, "%d\t", PWJ [i] );
        }

    fprintf(fptro, "\n\nSUPPORT HOURS FOR PRODUCTS OF AGE TAU");
    for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read support hours required for differring age
products */
        {
        fprintf(fptro,"\nSupport Hours for Product Number %d\n", i );
        for ( j = 0; j < PLJ[i]; j++ )
            {
            fscanf(fptri, "%f", &SJT [i] [j] );
            fprintf(fptro, "%4.2f\t", SJT [i] [j] );
            }
        }

    fprintf( fptro, "\n\nBEGINNING SUPPORT HOURS AVAILABLE\n" );
    fscanf(fptri, "%f", &S0 ); /* Read beginning support hours */
    fprintf(fptro, "%4.2f", S0 );

    fprintf(fptro, "\n\nMAXIMUM ALLOWABLE PRODUCTION COST\n");
    for ( i = 0; i < NUM_PROD_PERIODS; i++ ) /* read maximum allowable production cost */
        {
        fscanf(fptri, "%f", &PC [i] );
        fprintf(fptro, "%8.2f ", PC [i] );
        }

    fprintf(fptro, "\n\nAVERAGE PARTS COST TO REPAIR PRODUCT OF AGE TAU");
    for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read average parts cost to repair product of age tau */
        {
        fprintf(fptro,"\nAverage parts cost for Product Number %d\n", i );
        for ( j = 0; j < PLJ[i]; j++ )
            {
            fscanf(fptri, "%f", &pj [i] [j] );
            fprintf(fptro, "%4.2f\t", pj [i] [j] );
            }
        }
```

```
fprintf(fptro, "\n\nMAXIMUM REPAIR PARTS COST\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read maximum repair parts cost */
        {
        fscanf(fptri, "%f", &PT [i] );
        fprintf(fptro, "%8.2f ", PT [i] );
        }

fprintf(fptro, "\n\nPARTS REVENUE FROM REPAIR OF PRODUCT OF AGE TAU");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read average parts cost to repair product of age tau */
        {
        fprintf(fptro,"\nAverage parts revenue for Product Number %d\n", i );
        for ( j = 0; j < PLJ[i]; j++ )
                {
                fscanf(fptri, "%f", &PR [i] [j] );
                fprintf(fptro, "%4.2f\t", PR [i] [j] );
                }
        }

fprintf(fptro, "\n\nMINIMUM ACCEPTABLE REVENUE\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read minimum acceptable revenue */
        {
        fscanf(fptri, "%f", &RT [i] );
        fprintf(fptro, "%8.2f ", RT [i] );
        }

fprintf(fptro, "\n\nREPAIR LABOR REVENUE PER HOUR\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read repair labor revenue per hour */
        {
        fscanf(fptri, "%f", &ft [i] );
        fprintf(fptro, "%4.2f  ", ft [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE PRODUCTION WC\n");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read base cost to operate production wc's */
        {
        fscanf(fptri, "%f", &BASE_COST_L [i] );
        fprintf(fptro, "%f\t", BASE_COST_L [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE NON-PRODUCTION WC\n");
for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read base cost to operate non-production wc's */
        {
        fscanf(fptri, "%f", &BASE_COST_N [i] );
        fprintf(fptro, "%f\t", BASE_COST_N [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE SUPPORT WC\n");
/* read base cost to operate support wc */
fscanf(fptri, "%f", &BASE_COST_V );
fprintf(fptro, "%f\t", BASE_COST_V );
```

```
        fprintf(fptro, "\n\nMATERIAL COST FOR PRODUCTION");
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read material cost for production */
                {
                fprintf(fptro,"\nMaterial cost for Product Number %d\n", i );
                for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                        {
                        fscanf(fptri, "%f", &MA [i] [j] );
                        fprintf(fptro, "%4.2f\t", MA [i] [j] );
                        }
                }


        fprintf(fptro, "\n\nREPAIR LABOR COST\n");
        for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read repair labor cost */
                {
                fscanf(fptri, "%f", &RC [i] );
                fprintf(fptro, "%4.2f  ", RC [i] );
                }

        fprintf(fptro, "\n\nCONSISTENT UNITS REQUIRED BY PRODUCT AT NON_PRODUCTION
WC");
        for ( k = 0; k < NUM_NON_PROD_WC; k++ )
        {
        fprintf(fptro, "\n\nUnits required at wc %d", k );
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read units required for non-production */
                {
                fprintf(fptro,"\n  Consistent non-production units for product number %d\n", i );
                fprintf(fptro, "  ");
                for ( j = 0; j < NUM_PERIODS; j++ )
                        {
                        fscanf(fptri, "%f", &G [k] [i] [j] );
                        fprintf(fptro, "%4.2f  ", G [k] [i] [j] );
                        }
                }
        }

        fprintf(fptro, "\n\nMAXIMUM PRODUCTION PERIODS FOR EACH PRODUCT\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                {
                fscanf(fptri, "%d", &max_periods[i] );
                fprintf(fptro, "%d\t", max_periods[i] );
                }

        fprintf(fptro, "\n\nEARLIEST PERIOD PRODUCTION OF A PRODUCT MAY
COMMENCE\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                {
                fscanf(fptri, "%d", &start_period[i] );
                fprintf(fptro, "%d\t", start_period[i] );
                }
```

```
fprintf(fptro, "\n\nINVENTORY HOLDING COSTS FOR PRODUCTS IN EACH PERIOD\n");
for ( i = 0; i < NUM_PRODUCTS; i++ )
{
        fprintf(fptro, "\nPRODUCT %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
        {
        fscanf(fptri, "%f", &INVENTORY[i][j] );
        fprintf(fptro, "%8.2f  ", INVENTORY[i][j] );
        }
}

fprintf(fptro, "\n\nIMPROVMENT IN HOURS FROM SUPPORT CAP BUD PROJECT\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ )
        {
        fscanf(fptri, "%d", &CV[i] );
        fprintf(fptro, "%d  ", CV[i] );
        }

fprintf(fptro, "\n\nBEGINNING INVENTORY FOR EACH PRODUCT\n");
for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
        fscanf(fptri, "%d", &BEGIN_INVENTORY[i] );
        fprintf(fptro, "%d\t", BEGIN_INVENTORY[i] );
        }

/* start read from mpss.rep file */
fprintf(fptro, "\n\n\nOUTPUT FROM STAGE TWO OPTIMIZATION");
fscanf(fptri2, "%s", TITLE );
fprintf(fptro,"\n\nINPUT IS FROM:  %s", TITLE );

/* strip off header information */
fscanf(fptri2, "%s", PROJECT );
fprintf(fptro, "\n\n%s ", PROJECT );
fscanf(fptri2, "%s", CONSTRAINT );
fprintf(fptro, "%s\n", CONSTRAINT );
for ( i = 0; i < 2; i++ )
        {
        fscanf(fptri2, "%s", DEMAND );
        fprintf(fptro, "%s ", DEMAND );
        fscanf(fptri2, "%s", EQUAL );
        fprintf(fptro, "%s ", EQUAL );
        fscanf(fptri2, "%s", DEMAND_VALUE );
        fprintf(fptro, "%s\n", DEMAND_VALUE );
        }
fscanf(fptri2, "%s", VAR_NAME );
fscanf(fptri2, "%s", VAR_VALUE );
fscanf(fptri2, "%s", REDUCED );
fscanf(fptri2, "%s", COST );

fprintf(fptro, "\n\nVALUES OF Z FROM STAGE TWO OPTIMIZATION\n");
for ( i = 0; i < NUM_PRODUCTS; i++ )
```

```
                {
                fscanf(fptri2, "%s %f %f", ZNAME, &Z[i], &ZJUNK );
                fprintf(fptro, "\n\n Z%d    %1.0f ", i, Z[i] );
                }

        fprintf(fptro, "\n\nVALUES OF SETUP VARIABLES FROM STAGE TWO
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
                for ( j = start_period[i]; j < start_period[i] + max_periods[i]; j++ )
                {
                fscanf(fptri2, "%s %f %f", DELTANAME, &DELTA[i][j], &DELTAJUNK );
                fprintf(fptro, "%s   %1.0f\n", DELTANAME, DELTA[i][j] );
                }
        }

        fprintf(fptro, "\n\nVALUE OF PRODUCTION CAPITAL BUDGET VARIABLES FROM
STAGE TWO OPTIMIZATION\n");
        for ( i = 0; i < NUM_CAP_BUD_PROJECTS; i++ )
                {
                fscanf(fptri2, "%s %f %f", ELNAME, &EL[i], &ELJUNK );
                fprintf(fptro, "%s    %1.6f\n", ELNAME, EL[i] );
                }

        fprintf(fptro, "\n\nVALUE OF SUPPORT PRODUCTION CAPITAL BUDGET VARIABLE
FROM STAGE TWO OPTIMIZATION\n");
        fscanf(fptri2, "%s %f %f", EVNAME, &EV, &EVJUNK );
        fprintf(fptro, "EV    %1.6f\n", EV );

        fprintf(fptro, "\n\nVALUES OF PRODUCTION VARIABLES FROM STAGE TWO
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
                for ( j = start_period[i]; j < start_period[i] + max_periods[i]; j++ )
                {
                fscanf(fptri2, "%s %f %f", XNAME, &X[i][j], &XJUNK );
                fprintf(fptro, "%s    %12.6f\n", XNAME, X[i][j] );
                }
        }

        fprintf(fptro, "\n\nVALUES OF INVENTORY VARIABLES FROM STAGE TWO
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
                for ( j = start_period[i]; j < start_period[i] + max_periods[i] - 1; j++ )
                {
                fscanf(fptri2, "%s %f %f", INAME, &I[i][j], &IJUNK );
                fprintf(fptro, "%s    %12.6f\n", INAME, I[i][j] );
                }
        }
```

```
        fprintf(fptro, "\n\nVALUES OF REGULAR TIME LABOR VARIABLES FROM STAGE TWO
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PROD_PERIODS; i++ )
        {
                for ( j = 0; j < NUM_PROD_WC; j++ )
                {
                fscanf(fptri2, "%s %f %f", wltNAME, &wlt[i][j], &wltJUNK );
                fprintf(fptro, "%s    %12.6f\n", wltNAME, wlt[i][j] );
                }
        }

        fprintf(fptro, "\n\nVALUES OF OVERTIME LABOR VARIABLES FROM STAGE TWO
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PROD_PERIODS; i++ )
        {
                for ( j = 0; j < NUM_PROD_WC; j++ )
                {
                fscanf(fptri2, "%s %f %f", oltNAME, &olt[i][j], &oltJUNK );
                fprintf(fptro, "%s    %12.6f\n", oltNAME, olt[i][j] );
                }
        }

        fclose(fptri2);
        fclose(fptri);

        last( d1, d2, CL, SETUP_HOURS, PRODUCTION_HOURS, REVENUE, WLT, OLT,
        Bl, cl, cv, El, Ev, LAMBDA, SJT, S0, PC, PLJ, PWJ, pj, PT, PR, RT,
        ft, MA, RC, max_periods, start_period, BASE_COST_L, BASE_COST_V, ACC,
        INVENTORY, CV, inflation, BEGIN_INVENTORY, Z, X );

        fclose(fptro);

} /* end main */
```

```
/* last.h */
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define NUM_PERIODS                    12
#define NUM_SUPPORT_PERIODS            11
#define HORIZON_PERIOD                 11
#define NUM_PROD_PERIODS               8
#define NUM_PRODUCTS                   5
#define NUM_PROD_WC                    3
#define NUM_NON_PROD_WC                4
#define NUM_CAP_BUD_PROJECTS           3
#define NUM_NON_PROD_CAP_BUD_PROJ      1
#define AGE                            5
#define TRUE                           1
#define FALSE                          0


int project, i, j;

int setup_constraint;
int dec_var;
int age;
int inv_lim;
int must_fund;
int nprdwc;


/* base cost variables */
float TOT_BASE_COST;
float TOT_BASE_DISCOUNT;

/* discounted cost from repair labor */
float ADJ_LABOR_REP_COST[NUM_PRODUCTS][NUM_PERIODS];

/* discounted parts costs */
float ADJ_PART[NUM_PRODUCTS][NUM_PERIODS];

/* production capital budgeting costs */
float TOTAL_CAP_BUD_EXPENSE[NUM_CAP_BUD_PROJECTS];

/* adjusted inventory cost; includes stocking fee, loss of interest on
  revenues and repair costs */
float ACTUAL_INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS - 1];

/* total support capital budgeting cost */
float TOTAL_SUPPORT_CAP_BUD_COST;
```

```
/* last.c */
#include "last.h"

/* Builds an MPS file for the final stage of the two stage model */
void last( int d1[NUM_PROD_PERIODS], int d2[NUM_PROD_PERIODS],
int CL[NUM_CAP_BUD_PROJECTS][NUM_PROD_WC][NUM_PROD_PERIODS],
int SETUP_HOURS[NUM_PROD_WC] [NUM_PRODUCTS],
int PRODUCTION_HOURS[NUM_PROD_WC] [NUM_PRODUCTS],
float REVENUE [NUM_PRODUCTS][NUM_PROD_PERIODS],
float WLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float OLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float Bl [NUM_PROD_WC], float cl [NUM_CAP_BUD_PROJECTS] [NUM_PERIODS],
float cv [NUM_PERIODS], float El [NUM_PERIODS],
float Ev [NUM_PERIODS], float LAMBDA [NUM_PROD_WC][NUM_PROD_PERIODS],
float SJT [NUM_PRODUCTS][AGE], float S0, float PC [NUM_PROD_PERIODS],
int PLJ [NUM_PRODUCTS], int PWJ [NUM_PRODUCTS], float pj [NUM_PRODUCTS][AGE],
float PT [NUM_PERIODS], float PR [NUM_PRODUCTS][AGE], float RT [NUM_PERIODS],
float ft [NUM_PERIODS], float MA [NUM_PRODUCTS][NUM_PROD_PERIODS],
float RC [NUM_PERIODS], int max_periods[NUM_PRODUCTS],
int start_period[NUM_PRODUCTS], float BASE_COST_L[NUM_PROD_WC],
float BASE_COST_V, float ACC,
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS], int CV [NUM_PERIODS - 1],
float inflation, int BEGIN_INVENTORY[NUM_PRODUCTS],
float Z [NUM_PRODUCTS],
float X[NUM_PRODUCTS][NUM_PROD_PERIODS] )

{
FILE *mps;

register product, period, wc;

        /* open MPS output file */

        if ( (mps = fopen("mpls.dat","w")) == NULL ) /* open mps output file */
                {
                 printf("Can't open the mps output file.");
                 exit(0);
                }

        /* build total production capital budgeting costs */
        for ( project = 0; project < NUM_CAP_BUD_PROJECTS; project++ )
        {
                for ( period = 0; period < NUM_PERIODS; period++ )
                {
                TOTAL_CAP_BUD_EXPENSE[project] = 0;
                }
        }
        for ( project = 0; project < NUM_CAP_BUD_PROJECTS; project++ )
        {
                for ( period = 0; period < NUM_PERIODS; period++ )
                {
```

```
                TOTAL_CAP_BUD_EXPENSE[project] += cl[project][period] * pow( 1 + ACC,
HORIZON_PERIOD - period );
                }
        }

        /* build total future worth of repair revenues and costs  for each product by
        period it was produced */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                ADJ_LABOR_REP_COST[product][period] = 0;
                ADJ_PART[product][period] = 0;
                }
        }
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period =  start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                for ( age = 0; age < PLJ[product]; age++ )
                ADJ_LABOR_REP_COST[product][period] += SJT[product][age]*RC[age] * pow( 1 +
ACC, HORIZON_PERIOD - ( period + age ) );
                }
        }
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period =  start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                for ( age = 0; age < PLJ[product]; age++ )
                {
                ADJ_PART[product][period] += - pj[product][age] * pow( 1 + inflation, period ) * pow
( 1 + ACC, HORIZON_PERIOD - ( period + age ) );
                }
                }
        }

        /* actual inventory cost - includes loss of interest on revenues and adjusts repair costs and
revenues */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
                {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product] - 1; period++ )
                {
                ACTUAL_INVENTORY[product][period] =
                 ( INVENTORY[product][period]
                + REVENUE[product][period] ) * pow( 1 + ACC, HORIZON_PERIOD - period )
                - ADJ_LABOR_REP_COST[product][period] +
                ADJ_PART[product][period] -
                ( REVENUE[product][period + 1] * pow( 1 + ACC, HORIZON_PERIOD - period - 1)
```

```
                - ADJ_LABOR_REP_COST[product][period + 1] +
                ADJ_PART[product][period + 1] );
                }
        }

        /* determine total support capital budget cost */
        TOTAL_SUPPORT_CAP_BUD_COST = 0;
        for ( period = 0; period < NUM_PERIODS; period++ )
                TOTAL_SUPPORT_CAP_BUD_COST += cv[period] * pow( 1 + ACC,
HORIZON_PERIOD - period );

        /* calculate total base costs at production and non-production wc's */
        TOT_BASE_COST = 0;
        TOT_BASE_DISCOUNT = 0;
        /* determine total base production cost in each period */
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                TOT_BASE_COST += BASE_COST_L[wc];

        /* add base cost of support to other base costs */
        TOT_BASE_COST += BASE_COST_V;

        /* Determine total discounted constant dollar revenues less costs */
        for ( period = 0; period < NUM_PERIODS; period++ )
        {
        TOT_BASE_DISCOUNT += TOT_BASE_COST * pow( 1 + inflation, period ) * pow( 1 +
ACC, HORIZON_PERIOD - period );
        }


        /* print header to name LP */
        fprintf(mps, "NAME    LAST STAGE OF TWO STAGE MODEL MPS FILE (MIN)\n");

        /* products funded: as a remark */
        fprintf(mps, "*\t");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                 fprintf(mps, "Z%d = %1.0f   ", i, Z[i] );
        fprintf(mps, "\n");

        /* print header to start rows */
        fprintf(mps, "ROWS\n");
        /* develop row for the objective function */
        fprintf(mps, " N  obj\n");

        /* products funded */
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                fprintf(mps, " E  prod%d\n", i);

        /* production levels */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
```

```
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                fprintf(mps, " E  prodct%d%d\n", product, period );
                }
        }

        /* production capital budgeting project mutually exclusive row */
        fprintf(mps, " L  capmutex\n");

        /* production capital budget maximum cost per period */
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, " L  mxbudl%d\n", period );

        /* support capital budget maximum cost per period */
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, " L  mxbudv%d\n", period );

        /* develop rows for setup constraints */
        setup_constraint = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                setup_constraint++;
                fprintf(mps, " L  setup%d\n", setup_constraint);
                }
        }

        /* develop rows to set upper limits on inventory */
        inv_lim = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product] - 1; period++ )
                {
                inv_lim++;
                fprintf(mps, " L  invlim%d\n", inv_lim);
                }
        }

        /* develop rows for product demand */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                fprintf(mps, " L  demand1%d\n",  period );

        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                        fprintf(mps, " L  demand2%d\n", period );

        /* develop rows for prodwc constraint */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
```

```
        {
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                {
                fprintf(mps, " L  prodwc%d%d\n", period, wc );
                }
        }


/* develop values of WLT and OLT */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                {
                fprintf(mps, " L  time%d%d\n", period, wc );
                }
        }

/* develop maximum values of WLT */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                {
                fprintf(mps, " L  regtm%d%d\n", period, wc );
                }
        }

/* develop maximum values of OLT */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                {
                fprintf(mps, " L  otime%d%d\n", period, wc );
                }
         }

/* develop maximum production cost in each period */
for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        fprintf(mps, " L  mxprcst%d\n", period );


/* develop maximum support hours in each period rows */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, " L  prdsup%d\n", period );

/* develop maximum support parts cost in each period */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, " L  prtsup%d\n", period );

/* develop minimum revenue for each period */
for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
        fprintf(mps, " G  minrev%d\n", period );
```

```
        /* print basecost constraint row */
        fprintf(mps, " E  basecost\n");

        /* set base capacity to Bl */
        fprintf(mps, " E  basecap\n");

        /* set beginning inventory variables equal to one */
        i = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
                {
                if ( Z[product] == 1 )
                {
                if ( BEGIN_INVENTORY[product] > 0 )
                        {
                        i++;
                        fprintf(mps, " E  beginv%d\n", i );
                        }
                }
                }



        /* print header to start columns */
        fprintf(mps, "COLUMNS\n");

        /* print product variables in obj f(x) */
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                {
                fprintf(mps, "    Z%d       obj        1.0\n", i);
                fprintf(mps, "    Z%d       prod%d     1.0\n", i, i);
                }

        /* print production work center setup variables */
        setup_constraint = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                fprintf(mps, "    DELTA%d%-4dobj       1.0\n", product, period );
                setup_constraint++;
                if ( setup_constraint < 10 && product < 3 )
                {
                fprintf(mps, "    DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d1[period] * 3 );
                }
                if ( setup_constraint >= 10 && product < 3)
                {
                fprintf(mps, "    DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d1[period] * 3 );
```

```
                }
                if ( setup_constraint < 10 && product >= 3 )
                {
                fprintf(mps, "    DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d2[period] * 3 );
                }
                if ( setup_constraint >= 10 && product >= 3)
                {
                fprintf(mps, "    DELTA%d%-4dsetup%d    -%d\n", product, period, setup_constraint,
d2[period] * 3 );
                }
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "    DELTA%d%-4dprodwc%d%d%12d\n", product, period,
period, wc, SETUP_HOURS[wc][product] );
                        }
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "    DELTA%d%-4dtime%d%d  %12d\n", product, period, period,
wc, SETUP_HOURS[wc][product] );
                        }
                }
        }

        /* print production capital budegting variables */
        for ( project = 0; project < NUM_CAP_BUD_PROJECTS; project++ )
                {
                fprintf(mps, "    EL%d      obj       %-12.2f\n", project,
TOTAL_CAP_BUD_EXPENSE[project] );
                if ( project == 0 || project == 2 )
                fprintf(mps, "    EL%d      capmutex       1.0\n", project );
                for ( period = 0; period < NUM_PERIODS; period++ )
                        fprintf(mps, "    EL%d      mxbudl%d     %-12.2f\n", project, period,
cl[project][period] );
                for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                        {
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                                        {
                                        fprintf(mps, "    EL%d      prodwc%d%d    -%-12d\n",
project, period, wc, CL[project][wc][period] );
                                        fprintf(mps, "    EL%d      regtm%d%d     -%-12d\n", project,
period, wc, CL[project][wc][period] );
                                        fprintf(mps, "    EL%d      otime%d%d     -%-12d\n", project,
period, wc, CL[project][wc][period] );
                                        }
                        }
                }

        /* print support capital budgeting project variable */
        product = 0;
        fprintf(mps, "    EV       obj       %-12.2f\n", TOTAL_SUPPORT_CAP_BUD_COST );
```

```
        for ( period = 0; period < NUM_PERIODS; period++ )
                        fprintf(mps, "    EV       mxbudv%d     %-12.2f\n", period, cv[period]
);

        for ( period = 0; period < NUM_PROD_PERIODS + 3; period++ )
                fprintf(mps, "    EV       prdsup%d     -%-12d\n", period, CV[period] );

        /* print production variables */
        setup_constraint = 0;
        inv_lim = 1;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
                {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                fprintf(mps, "    X%d%-4d    obj        %-12.2f\n", product, period, MA[product][period]
* pow( 1 + .ACC, HORIZON_PERIOD - period ) + ADJ_LABOR_REP_COST[product][period] +
ADJ_PART[product][period] );
                fprintf(mps, "    X%d%-4d    prodct%d%d        1.0\n", product, period, product, period
);
                setup_constraint++;
                if ( setup_constraint < 10 )
                {
                fprintf(mps, "    X%d%-4d    setup%d            1.0\n", product, period,
setup_constraint);
                }
                if ( setup_constraint >= 10 )
                {
                fprintf(mps, "    X%d%-4d    setup%d            1.0\n", product, period, setup_constraint);
                }
                if ( inv_lim >= 10 && period < start_period[product] + max_periods[product] - 1 )
                {
                fprintf(mps, "    X%d%-4d    invlim%d      -1.0\n", product, period, inv_lim);
                inv_lim++;
                }
                if ( inv_lim < 10 && period < start_period[product] + max_periods[product] - 1 )
                {
                fprintf(mps, "    X%d%-4d    invlim%d       -1.0\n", product, period, inv_lim);
                inv_lim++;
                }
                if ( product < 3 )
                {
                fprintf(mps, "    X%d%-4d    demand1%d          1.0\n", product, period, period );
                }
                if ( product >= 3 )
                {
                fprintf(mps, "    X%d%-4d    demand2%d          1.0\n", product, period, period );
                }
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
```

```
                        fprintf(mps, "   X%d%-4d   prodwc%d%d%12d\n", product, period, period,
wc, PRODUCTION_HOURS[wc][product] );
                        }
                        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "   X%d%-4d   time%d%d  %12d\n", product, period, period, wc,
PRODUCTION_HOURS[wc][product] );
                        }
                fprintf(mps, "   X%d%-4d   mxprcst%d     %-12.2f\n", product, period, period,
MA[product][period] );
                for ( age = 0; age < PLJ[product]; age++ ) /* support hours in each period */
                        fprintf(mps, "   X%d%-4d   prdsup%d      %-12.2f\n", product, period, period
+ age, SJT[product][age] );
                for ( age = 0; age < PLJ[product]; age++ ) /* support cost in each period */
                        fprintf(mps, "   X%d%-4d   prtsup%d      %-12.2f\n", product, period, period
+ age, pj[product][age] * pow( 1 + inflation, period + age ) );
                for ( age = 0; age < PLJ[product]; age++ ) /* revenues in each period */
                        {
                        if ( age == 0 )
                                fprintf(mps, "   X%d%-4d   minrev%d      %-12.2f\n", product,
period, period, PR[product][age] * pow( 1 + inflation, period ) + ft[period] * SJT[product][age] +
REVENUE[product][period] );
                        else
                                fprintf(mps, "   X%d%-4d   minrev%d      %-12.2f\n", product,
period, period + age, PR[product][age] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age] );
                        }
                }
        }
        /* print inventory variables */
        inv_lim = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
                {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product] - 1; period++ )
                {
                fprintf(mps, "   I%d%-4d   obj          %-12.2f\n", product, period,
ACTUAL_INVENTORY[product][period] );
                inv_lim++;
                if ( inv_lim < 10 )
                {
                fprintf(mps, "   I%d%-4d   invlim%d        1.0\n", product, period, inv_lim);
                }
                if ( inv_lim >= 10 )
                {
                fprintf(mps, "   I%d%-4d   invlim%d       1.0\n", product, period, inv_lim);
                }
                if ( product < 3 )
                {
                fprintf(mps, "   I%d%-4d   demand1%d         -1.0\n", product, period, period );
                fprintf(mps, "   I%d%-4d   demand1%d          1.0\n", product, period, period + 1 );
```

```
                }
                if ( product >= 3 )
                {
                fprintf(mps, "   I%d%-4d   demand2%d          -1.0\n", product, period, period );
                fprintf(mps, "   I%d%-4d   demand2%d          1.0\n", product, period, period + 1 );
                }
                for ( age = 0; age < PLJ[product]; age++ ) /* adj period support hours */
                {
                        if ( age == 0 )
                                {
                                fprintf(mps, "   I%d%-4d   prdsup%d      -%-12.2f\n", product,
period, period, SJT[product][age] );
                                }
                        if ( age > 0 && age < PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   prdsup%d      %-12.2f\n", product,
period, period + age, SJT[product][age - 1] - SJT[product][age] );
                                }
                        if ( age == PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   prdsup%d      %-12.2f\n", product,
period, period + age, SJT[product][age - 1] - SJT[product][age] );
                                fprintf(mps, "   I%d%-4d   prdsup%d      %-12.2f\n", product,
period, period + age + 1, SJT[product][age] );
                                }
                }
                for ( age = 0; age < PLJ[product]; age++ ) /* adj period part sup cost */
                {
                        if ( age == 0 )
                                {
                                fprintf(mps, "   I%d%-4d   prtsup%d     -%-12.2f\n", product, period,
period, pj[product][age] * pow( 1 + inflation, period ) );
                                }
                        if ( age > 0 && age < PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   prtsup%d     %-12.2f\n", product, period,
period + age, pj[product][age - 1] * pow( 1 + inflation, period + age ) - pj[product][age] * pow( 1 +
inflation, period + age ) );
                                }
                        if ( age == PLJ[product] - 1 )
                                {
                                fprintf(mps, "   I%d%-4d   prtsup%d     %-12.2f\n", product, period,
period + age, pj[product][age - 1] * pow( 1 + inflation, period + age ) - pj[product][age] * pow( 1 +
inflation, period + age ) );
                                fprintf(mps, "   I%d%-4d   prtsup%d     %-12.2f\n", product, period,
period + age + 1, pj[product][age] * pow( 1 + inflation, period + age + 1 ) );
                                }
                }
                for ( age = 0; age < PLJ[product]; age++ ) /* adj period revenue */
                        {
                        if ( age == 0 )
```

```
                                {
                                fprintf(mps, "    I%d%-4d    minrev%d      -%-12.2f\n", product,
period, period, PR[product][age] * pow( 1 + inflation, period ) + ft[period] * SJT[product][age] +
REVENUE[product][period] );
                                }
                        if ( age == 1 )
                                {
                                fprintf(mps, "    I%d%-4d    minrev%d      %-12.2f\n", product,
period, period + age, PR[product][age - 1] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age - 1] + REVENUE[product][period] * pow( 1 + inflation, period + age ) -
PR[product][age] * pow( 1 + inflation, period + age ) - ft[period + age] * SJT[product][age] );
                                }
                        if ( age > 1 && age < PLJ[product] - 1 )
                                {
                                fprintf(mps, "    I%d%-4d    minrev%d      %-12.2f\n", product,
period, period + age, PR[product][age - 1] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age - 1] - PR[product][age] * pow( 1 + inflation, period + age ) - ft[period + age] *
SJT[product][age] );
                                }
                        if ( age == PLJ[product] - 1 )
                                {
                                fprintf(mps, "    I%d%-4d    minrev%d      %-12.2f\n", product,
period, period + age, PR[product][age - 1] * pow( 1 + inflation, period + age ) + ft[period + age] *
SJT[product][age - 1] - PR[product][age] * pow( 1 + inflation, period + age ) - ft[period + age] *
SJT[product][age] );
                                fprintf(mps, "    I%d%-4d    minrev%d      %-12.2f\n", product,
period, period + age + 1, PR[product][age] * pow( 1 + inflation, period + age + 1) + ft[period + age + 1] *
SJT[product][age] );
                                }
                        }

                }
        }

        /* print regular time labor variables */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                 for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "    WLT%d%-4d  obj        %-12.2f\n", period, wc,
WLT[wc][period] * pow( 1 + ACC, HORIZON_PERIOD - period ) );
                        fprintf(mps, "    WLT%d%-4d  time%d%d      -1.0\n", period, wc, period, wc );
                        fprintf(mps, "    WLT%d%-4d  regtm%d%d     %-12.4f\n", period, wc, period,
wc, 1/LAMBDA[wc][period] );
                        fprintf(mps, "    WLT%d%-4d  mxprcst%d     %-12.2f\n", period, wc, period,
WLT[wc][period] );
                        }
                }

        /* print over time labor variables */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
```

```
                {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "   OLT%d%-4d obj        %-12.2f\n", period, wc,
OLT[wc][period] * pow( 1 + ACC, HORIZON_PERIOD - period ) );
                        fprintf(mps, "   OLT%d%-4d time%d%d      -1.0\n", period, wc, period, wc );
                        fprintf(mps, "   OLT%d%-4d otime%d%d      %-12.4f\n", period, wc, period,
wc, 1/( 1 - LAMBDA[wc][period] ) );
                        fprintf(mps, "   OLT%d%-4d mxprcst%d     %-12.2f\n", period, wc, period,
OLT[wc][period] );
                        }
                }

        /* print base costs constraint */
        fprintf(mps, "   BASECOST  obj       %-12.3f\n", TOT_BASE_DISCOUNT );
        fprintf(mps, "   BASECOST  basecost      1.0\n");

        /* print base production capacity */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "   Bl      regtm%d%d   -%-12.3f\n", period, wc, Bl[wc] );
                        fprintf(mps, "   Bl      otime%d%d   -%-12.3f\n", period, wc, Bl[wc] );
                        }
                }

        fprintf(mps, "   Bl      basecap       1.0\n");

        /* print beginning inventory variables */
        i = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                if ( Z[product] == 1 && BEGIN_INVENTORY[product] > 0 )
                {
                if ( product < 3 )
                {
                fprintf(mps, "   BI%d      obj       -%-12.2f\n", product, REVENUE[product][0] * pow(
1 + ACC, HORIZON_PERIOD ) );
                fprintf(mps, "   BI%d      demand10    %-12d\n", product,
BEGIN_INVENTORY[product] );
                }
                if ( product >= 3 )
                {
                fprintf(mps, "   BI%d      obj       -%-12.2f\n", product, REVENUE[product][0] * pow(
1 + ACC, HORIZON_PERIOD ) );
                fprintf(mps, "   BI%d      demand20    %-12d\n", product,
BEGIN_INVENTORY[product] );
                }
                i++;
                fprintf(mps, "   BI%d      beginv%d    1.0\n", product, i );
```

```
                }
        }



        /* print header to start rows */
        fprintf(mps, "RHS\n");

    /* products funded */
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                fprintf(mps, "    RHS      prod%d        %1.0f\n", i, Z[i] );

        /* production levels */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                fprintf(mps, "    RHS      prodct%d%d    %12.6f\n", product, period, X[product][period]
- 0.00001 );
                }
        }

        /* print setup right hand sides */
        fprintf(mps, "    RHS      capmutex      1.0\n");

        /* print maximum production capital budget expense RHS */
        for ( period = 0; period < NUM_PERIODS; period++ )
                        fprintf(mps, "    RHS      mxbudl%d    %-12.2f\n", period, El[period] );

        /* print support capital budget maximum cost per period */
        for ( period = 0; period < NUM_PERIODS; period++ )
                fprintf(mps, "    RHS      mxbudv%d    %-12.2f\n", period, Ev[period] );

        /* print maximum production (demand) RHS */
        product = 0;
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                fprintf(mps, "    RHS      demand1%d        %d\n", period, d1[period] );

        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                fprintf(mps, "    RHS      demand2%d        %d\n", period, d2[period] );

        /* print production work center RHS */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        fprintf(mps, "    RHS      prodwc%d%d        %-13.2f\n", period, wc, Bl [wc] );
                        }
                }
```

```
        /* print maximum production cost RHS */
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                fprintf(mps, "    RHS      mxprcst%d     %-12.2f\n", period, PC[period] );

        /* print maximum support hours RHS */
        for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
                fprintf(mps, "    RHS      prdsup%d      %-12.2f\n", period, S0 );

        /* print maximum support parts cost RHS */
        for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
                fprintf(mps, "    RHS      prtsup%d      %-12.2f\n", period, PT[period] );

        /* print minimum revenue for each period RHS */
        for ( period = 0; period < NUM_SUPPORT_PERIODS; period++ )
                fprintf(mps, "    RHS      minrev%d      %-12.2f\n", period, RT[period] );

        fprintf(mps, "    RHS      basecost      1.0\n");
        fprintf(mps, "    RHS      basecap       1.0\n");

        /* print beginning inventory RHS */
        i = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
        if ( Z[product] == 1 )
        {
        if ( BEGIN_INVENTORY[product] > 0 )
                {
                i++;
                fprintf(mps, "    RHS      beginv%d      1.0\n", i );
                }
        }
        }

        fprintf(mps, "ENDATA");
        fclose(mps);
}
```

APPENDIX F: Program Listing for the Future Worth from the Sequential Model

```
/* fwssmain.h */
/* header file for fwssmain.c */

#include <stdio.h>
#include <stdlib.h>
#define NUM_PERIODS                     12
#define HORIZON_PERIOD                  11
#define NUM_SUPPORT_PERIODS             11
#define NUM_PROD_PERIODS                8
#define NUM_PRODUCTS                    5
#define NUM_PROD_WC                     3
#define NUM_NON_PROD_WC                 4
#define NUM_CAP_BUD_PROJECTS            3
#define NUM_NON_PROD_CAP_BUD_PROJ       1
#define AGE                             5 /* maximum life length of any product */

typedef unsigned int BitVector;

int k;
        /* INPUT PARAMETERS */
/* DEMAND */
int d1 [NUM_PROD_PERIODS]; /* Demand for subgroup one */
int d2 [NUM_PROD_PERIODS]; /* Demand for subgroup two */

/* PRODUCT AND PROJECT INTERACTION */
BitVector Mutually_Exclusive_Prod[NUM_PRODUCTS]; /* Mutually exclusive products */
BitVector Mutually_Exclusive_Proj[NUM_CAP_BUD_PROJECTS]; /* " " projects */
BitVector Contingency[NUM_PRODUCTS]; /* Contingency between products */
BitVector Must_Fund; /* Must fund a set of products */

/* INTEREST RATES */
float ACC;  /* Interest Rate */
float inflation; /* Inflation Rate */

/* PRODUCT REVENUES */
float REVENUE [NUM_PRODUCTS][NUM_PROD_PERIODS]; /* Product revenue */

/* PRE-PRODUCTION/POST-SUPPORT PRODUCT INPUTS */
float BASE_COST_N[NUM_NON_PROD_WC]; /* Base cost to operate non-prod wc's */
float Bn[NUM_NON_PROD_WC]; /* Beginning units of capacity at non-production wc's */
float G[NUM_NON_PROD_WC][NUM_PRODUCTS][NUM_PERIODS]; /* No. consistent units req by
product at non-production wc's  */

/* LABOR INPUTS */
float WLT [NUM_PROD_WC] [NUM_PROD_PERIODS]; /* reg time cost by wc */
float OLT [NUM_PROD_WC] [NUM_PROD_PERIODS]; /* ot time cost by wc */
float LAMBDA[NUM_PROD_WC][NUM_PROD_PERIODS]; /* Fraction labor regular time */

/* PRODUCTION INPUTS */
float BASE_COST_L[NUM_PROD_WC]; /* Base cost to operate production wc's */
float Bl[NUM_PROD_WC]; /* Beginning hours at production wc's */
```

```
int SETUP_HOURS[NUM_PROD_WC] [NUM_PRODUCTS]; /* setup hours for prod by wc */
int PRODUCTION_HOURS[NUM_PROD_WC] [NUM_PRODUCTS]; /* production hrs by prod & wc */
float PC[NUM_PROD_PERIODS]; /* Maximum allowable production cost */
float MA[NUM_PRODUCTS][NUM_PROD_PERIODS]; /* Material cost for production */

/* SUPPORT VARIABLES */
float BASE_COST_V; /* Base support cost */
float S0; /* Beginning support hours available */
float SJT[NUM_PRODUCTS][AGE]; /* Support hours required for product of different ages */
int PLJ[NUM_PRODUCTS]; /* Product life length */
int PWJ[NUM_PRODUCTS]; /* Warrantee length */
float pj[NUM_PRODUCTS][AGE]; /* Average parts cost */
float PT[NUM_PERIODS]; /* Maximum repair parts cost */
float PR[NUM_PRODUCTS][AGE]; /* Parts revenue */
float RT[NUM_PERIODS]; /* Minimum acceptable revenue */
float ft[NUM_PERIODS]; /* Repair labor revenue per hour */
float RC[NUM_PERIODS]; /* Repair labor cost */

/* INVENTORY INPUTS */
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS]; /* inventory holding costs */
int BEGIN_INVENTORY[NUM_PRODUCTS];

/* CAPITAL BUDGETING INPUTS */
float cl[NUM_CAP_BUD_PROJECTS][NUM_PERIODS]; /* cost of production cap budget projects */
float cn[NUM_PERIODS]; /* cost of non-prod cap budget projects */
float cv[NUM_PERIODS]; /* cost of support capital budgeting project in each period */
float El[NUM_PERIODS]; /* maximum allowable prod project expense */
float En[NUM_PERIODS]; /* maximum allowable pre-production project expense */
float Ev[NUM_PERIODS]; /* maximum allowable post support wc project expense */
int CL [NUM_CAP_BUD_PROJECTS][NUM_PROD_WC][NUM_PROD_PERIODS]; /* improvement at
WC's due tc cap bud proj 1 */
int CV [NUM_PERIODS - 1]; /* improvement in support hours from support cap bud project */
int CN[NUM_NON_PROD_WC][NUM_PERIODS]; /* Increase at non-prod wc's from cap bud project */

/* STARTING AND ENDING PERIOD INPUTS */
int max_periods[NUM_PRODUCTS]; /* maximum number of periods a product may be produced */
int start_period[NUM_PRODUCTS]; /* earliest period a product can be produced */

/* VARIABLE VALUES FROM mpls.rep, i.e., last stage optimization */
char TITLE[15];
char ZNAME[2];
float Z[NUM_PRODUCTS];
float ZJUNK;
char DELTANAME[7];
float DELTA[NUM_PRODUCTS][NUM_PROD_PERIODS];
float DELTAJUNK;
char ENNAME[2];
float EN;
char ELNAME[3];
float EL[NUM_CAP_BUD_PROJECTS];
float ELJUNK;
```

```
char EVNAME[2];
float EV;
float EVJUNK;
char XNAME[3];
float X[NUM_PRODUCTS][NUM_PROD_PERIODS];
float XJUNK;
char INAME[3];
float I[NUM_PRODUCTS][NUM_PROD_PERIODS - 1];
float IJUNK;
char wltNAME[5];
float wlt[NUM_PROD_PERIODS][NUM_PROD_WC];
float wltJUNK;
char oltNAME[5];
float olt[NUM_PROD_PERIODS][NUM_PROD_WC];
float oltJUNK;

/* string variables to strip mpls.rep off header information */
char PROJECT[8];
char CONSTRAINT[14];
char DEMAND[2];
char EQUAL[1];
char DEMAND_VALUE[3];
char VAR_NAME[8];
char VAR_VALUE[5];
char REDUCED[7];
char COST[4];

/* portfolio future worth */
float FW;

/* prototype of two stage model future worth function */
float fwsscon( float REVENUE [NUM_PRODUCTS][NUM_PROD_PERIODS],
float WLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float OLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float cl [NUM_CAP_BUD_PROJECTS] [NUM_PERIODS], float cn [NUM_PERIODS],
float cv [NUM_PERIODS], float SJT [NUM_PRODUCTS][AGE],
int PLJ [NUM_PRODUCTS], int PWJ [NUM_PRODUCTS], float pj [NUM_PRODUCTS][AGE],
float PR [NUM_PRODUCTS][AGE], float ft [NUM_PERIODS],
float MA [NUM_PRODUCTS][NUM_PROD_PERIODS], float RC [NUM_PERIODS],
int max_periods[NUM_PRODUCTS], int start_period[NUM_PRODUCTS],
float BASE_COST_L[NUM_PROD_WC], float BASE_COST_N[NUM_NON_PROD_WC],
float BASE_COST_V, float ACC, float inflation,
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS],
int BEGIN_INVENTORY[NUM_PRODUCTS], float EN,
float X[NUM_PRODUCTS][NUM_PROD_PERIODS],
float I[NUM_PRODUCTS][NUM_PROD_PERIODS - 1],
float wlt[NUM_PROD_PERIODS][NUM_PROD_WC],
float olt[NUM_PROD_PERIODS][NUM_PROD_WC], float EL[NUM_CAP_BUD_PROJECTS],
float EV );
```

```
/* fwssmain.c */
/* main routine to determine the future worth of the portfolio of funded
        products after optimization of the last stage of the two stage
        model is complete */

/* inputs data from input.dat and mpls.rep */
/* outputs data to fwssout.dat */
/* calls function: fwsscon.c */

#include "fwssmain.h"

main(void)

{
        FILE *fptri1, *fptro, *fptri2;
        register i, j;

        /* read inputs from a file and print back to output file */

        if ( (fptri1 = fopen("input.dat","r")) == NULL ) /* open input file */
                {
                 printf("Can't open the input file.");
                 exit(0);
                }

        if ( (fptri2 = fopen("mpls.rep","r")) == NULL ) /* open input file */
                {
                 printf("Can't open the input file.");
                 exit(0);
                }

        if ( ( fptro = fopen("fwssout.dat", "w") ) == NULL ) /* open output file */
                {
                printf("Can't open the output file.");
                exit(0);
                }

        fprintf(fptro, "FILE NAME: fwssout.dat\n\n");
        fscanf(fptri1, "%s", TITLE );
        fprintf(fptro, "INPUT FILE IS: %s\n", TITLE );

        fprintf(fptro,"\n\n                LIST OF INPUTS AND OUTPUTS\n");
        fprintf(fptro,"                ___________________________\n\n\n");
        fprintf(fptro,"A.  Input Data\n");

        fprintf(fptro, "\n\n");


        /* Read data from input file */
        fprintf(fptro, "DEMAND FOR SUBGROUP ONE\n");
        for(i = 0; i < NUM_PROD_PERIODS; i++) /* read demand for group 1 */
```

```
        {
        fscanf(fptri1, "%d", &d1[i]);
        fprintf(fptro, "%1d\t", d1[i]);
        }

fprintf(fptro, "\nDEMAND FOR SUBGROUP TWO\n");
for(i = 0; i < NUM_PROD_PERIODS; i++) /* read demand for group 2 */
        {
        fscanf(fptri1, "%d", &d2[i]);
        fprintf(fptro, "%1d\t", d2[i]);
        }

fprintf(fptro, "\n\nMUST FUND\n");
fscanf(fptri1, "%d", &Must_Fund );
fprintf(fptro, "%4d", Must_Fund );

fprintf(fptro, "\n\nMUTUALLY EXCLUSIVE PRODUCTS\n");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read mutually exclusive */
        {
        fscanf(fptri1, "%d", &Mutually_Exclusive_Prod[i]);
        fprintf(fptro, "%4d\t", Mutually_Exclusive_Prod[i]);
        }


fprintf(fptro, "\n\nMUTUALLY EXCLUSIVE PROJECTS\n");
for ( i = 0; i < NUM_CAP_BUD_PROJECTS; ++i ) /* read mut ex cap budget */
        {
        fscanf(fptri1, "%d", &Mutually_Exclusive_Proj[i]);
        fprintf(fptro, "%4d\t", Mutually_Exclusive_Proj[i]);
        }

fprintf(fptro, "\n\nPRODUCT CONTINGENCY\n");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read in product contingency */
        {
        fscanf(fptri1, "%d", &Contingency[i] );
        fprintf(fptro, "%4d\t", Contingency[i] );
        }

fprintf(fptro, "\n\nPRINT SETUP HOURS");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read setup hours */
        {
        fprintf(fptro,"\nSetup Hours At Work Center %d for Each Product\n", i );
        for ( j = 0; j < NUM_PRODUCTS; j++ )
                {
                fscanf(fptri1, "%d", &SETUP_HOURS[i] [j] );
                fprintf(fptro, "%d\t", SETUP_HOURS[i] [j]);
                }
        }


fprintf(fptro, "\n\nPRINT PRODUCTION HOURS");
```

```
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read production hours */
        {
        fprintf(fptro,"\nProduction Hours At Work Center %d for Each Product\n", i );
        for ( j = 0; j < NUM_PRODUCTS; j++ )
                {
                fscanf(fptri1, "%d", &PRODUCTION_HOURS[i] [j] );
                fprintf(fptro, "%d\t", PRODUCTION_HOURS[i] [j] );
                }
        }

for ( k = 0; k < NUM_CAP_BUD_PROJECTS; k++ )
{
fprintf(fptro, "\n\nPRODUCTION INCREASES FROM PROJECT %d", k );
for ( i = 0; i < NUM_PROD_WC; i++ )
        {
        fprintf(fptro,"\nProduction Increases At Work Center %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                {
                fscanf(fptri1, "%d", &CL[k][i][j] );
                fprintf(fptro, "%d\t", CL[k][i][j] );
                }
        }
}

fprintf(fptro, "\n\nREVENUE");
for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
        fprintf(fptro, "\nRevenue from product %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
        {
        fscanf(fptri1, "%f", &REVENUE[i][j] );
        fprintf(fptro, "%8.2f ", REVENUE[i][j] );
        }
        }

fprintf(fptro, "\n\nINTEREST RATE\n");
fscanf(fptri1, "%f", &ACC );
fprintf(fptro, "ACC = %4.3f", ACC );

fprintf(fptro, "\n\nINFLATION RATE\n" );
fscanf(fptri1, "%f", &inflation );
fprintf(fptro, "inflation = %4.3f", inflation );

fprintf(fptro, "\n\nREGULAR TIME COSTS");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read regular time cost */
        {
        fprintf(fptro,"\nRegular Time Cost for Production Work Center %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                {
                fscanf(fptri1, "%f", &WLT[i] [j] );
                fprintf(fptro, "%4.2f\t", WLT[i] [j] );
```

```
                }
        }


    fprintf(fptro, "\n\nPRINT OVER TIME COST");
    for ( i = 0; i < NUM_PROD_WC; i++ ) /* read overtime cost */
        {
        fprintf(fptro,"\nOver Time Cost for Production Work Center %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
            {
            fscanf(fptri1, "%f", &OLT [i] [j] );
            fprintf(fptro, "%4.2f\t", OLT [i] [j] );
            }
        }


    fprintf(fptro, "\n\nBEGINNING HOURS AT PRODUCTION WC'S\n");
    for ( i = 0; i < NUM_PROD_WC; i++ ) /* read beginning production time */
        {
        fscanf(fptri1, "%f", &Bl [i] );
        fprintf(fptro, "%4.2f\t", Bl [i] );
        }

    fprintf(fptro, "\n\nBEGINNING UNITS OF CAPACITY AT NON-PRODUCTION WC'S\n");
    for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read beginning non-production time */
        {
        fscanf(fptri1, "%f", &Bn [i] );
        fprintf(fptro, "%4.2f\t", Bn [i] );
        }

    fprintf(fptro, "\n\nINCREASES FROM NON-PRODUCTION CAPITAL BUDGTEING
PROJECT ONE\n");
    for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read increase from non-production cap bud
project 1 */
    {
    fprintf(fptro, "\nIncreases at WC%d\n", i );
        for ( j = 0; j < NUM_PERIODS; j++ )
        {
        fscanf(fptri1, "%d", &CN [i][j] );
        fprintf(fptro, "%d  ", CN [i][j] );
        }
    }

    fprintf(fptro, "\n\nCOST OF PRODUCTION CAPITAL BUDGETING PROJECTS");
    for ( i = 0; i < NUM_CAP_BUD_PROJECTS; i++ ) /* read cost of each capital project */
        {
        fprintf(fptro,"\nCapital Budgeting Project %d\n", i );
        for ( j = 0; j < NUM_PERIODS; j++ )
            {
            fscanf(fptri1, "%f", &cl [i] [j] );
            fprintf(fptro, "%8.2f  ", cl [i] [j] );
```

```
                }
        }

    fprintf(fptro, "\n\nCOST OF NON-PRODUCTION CAPITAL BUDGETING PROJECT\n");
    for ( i = 0; i < NUM_PERIODS; i++ ) /* read cost of capital project */
        {
        fscanf(fptri1, "%f", &cn [i] );
        fprintf(fptro, "%8.2f ", cn [i] );
        }

    fprintf(fptro, "\n\nCOST OF SUPPORT CAPITAL BUDGETING PROJECT\n");
    for ( i = 0; i < NUM_PERIODS; i++ ) /* read cost of capital project */
        {
        fscanf(fptri1, "%f", &cv [i] );
        fprintf(fptro, "%8.2f ", cv [i] );
        }


    fprintf(fptro, "\n\nMAXIMUM ALLOWABLE PRODUCTION PROJECT EXPENSE\n");
    for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable production capital project
expense */
        {
        fscanf(fptri1, "%f", &El [i] );
        fprintf(fptro, "%8.2f  ", El [i] );
        }

    fprintf(fptro, "\n\nMAXIMUM ALLOWABLE NON-PRODUCTION PROJECT EXPENSE\n");
    for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable non-production capital project
expense */
        {
        fscanf(fptri1, "%f", &En [i] );
        fprintf(fptro, "%8.2f ", En [i] );
        }

    fprintf(fptro, "\n\nMAXIMUM ALLOWABLE SUPPORT PROJECT EXPENSE\n");
    for ( i = 0; i < NUM_PERIODS; i++ ) /* read maximum allowable support capital project
expense */
        {
        fscanf(fptri1, "%f", &Ev [i] );
        fprintf(fptro, "%8.2f ", Ev [i] );
        }

    fprintf(fptro, "\n\nMAXIMUM FRACTION REGULAR TIME LABOR\n");
    for ( i = 0; i < NUM_PROD_WC; i++ ) /* read maximum fraction regular time labor */
        {
        fprintf(fptro, "\nWC %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
            {
            fscanf(fptri1, "%f", &LAMBDA[i][j] );
            fprintf(fptro, "%4.2f\t", LAMBDA[i][j] );
            }
```

```
        }

    fprintf(fptro, "\n\nPRODUCT LIFE LENGTH\n");
    for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read product life lengths */
        {
        fscanf(fptri1, "%d", &PLJ [i] );
        fprintf(fptro, "%d\t", PLJ [i] );
        }

    fprintf(fptro, "\n\nPRODUCT WARRANTEE LENGTH\n");
    for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read product warrantee lengths */
        {
        fscanf(fptri1, "%d", &PWJ [i] );
        fprintf(fptro, "%d\t", PWJ [i] );
        }

    fprintf(fptro, "\n\nSUPPORT HOURS FOR PRODUCTS OF AGE TAU");
    for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read support hours required for differring age
products */
        {
        fprintf(fptro,"\nSupport Hours for Product Number %d\n", i );
        for ( j = 0; j < PLJ[i]; j++ )
            {
            fscanf(fptri1, "%f", &SJT [i] [j] );
            fprintf(fptro, "%4.2f\t", SJT [i] [j] );
            }
        }

    fprintf( fptro, "\n\nBEGINNING SUPPORT HOURS AVAILABLE\n" );
    fscanf(fptri1, "%f", &S0 ); /* Read beginning support hours */
    fprintf(fptro, "%4.2f", S0 );

    fprintf(fptro, "\n\nMAXIMUM ALLOWABLE PRODUCTION COST\n");
    for ( i = 0; i < NUM_PROD_PERIODS; i++ ) /* read maximum allowable production cost */
        {
        fscanf(fptri1, "%f", &PC [i] );
        fprintf(fptro, "%8.2f ", PC [i] );
        }

    fprintf(fptro, "\n\nAVERAGE PARTS COST TO REPAIR PRODUCT OF AGE TAU");
    for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read average parts cost to repair product of age tau */
        {
        fprintf(fptro,"\nAverage parts cost for Product Number %d\n", i );
        for ( j = 0; j < PLJ[i]; j++ )
            {
            fscanf(fptri1, "%f", &pj [i] [j] );
            fprintf(fptro, "%4.2f\t", pj [i] [j] );
            }
        }

    fprintf(fptro, "\n\nMAXIMUM REPAIR PARTS COST\n");
```

```
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read maximum repair parts cost */
        {
        fscanf(fptri1, "%f", &PT [i] );
        fprintf(fptro, "%8.2f ", PT [i] );
        }

fprintf(fptro, "\n\nPARTS REVENUE FROM REPAIR OF PRODUCT OF AGE TAU");
for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read average parts cost to repair product of age tau */
        {
        fprintf(fptro,"\nAverage parts revenue for Product Number %d\n", i );
        for ( j = 0; j < PLJ[i]; j++ )
                {
                fscanf(fptri1, "%f", &PR [i] [j] );
                fprintf(fptro, "%4.2f\t", PR [i] [j] );
                }
        }

fprintf(fptro, "\n\nMINIMUM ACCEPTABLE REVENUE\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read minimum acceptable revenue */
        {
        fscanf(fptri1, "%f", &RT [i] );
        fprintf(fptro, "%8.2f ", RT [i] );
        }

fprintf(fptro, "\n\nREPAIR LABOR REVENUE PER HOUR\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read repair labor revenue per hour */
        {
        fscanf(fptri1, "%f", &ft [i] );
        fprintf(fptro, "%4.2f  ", ft [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE PRODUCTION WC\n");
for ( i = 0; i < NUM_PROD_WC; i++ ) /* read base cost to operate production wc's */
        {
        fscanf(fptri1, "%f", &BASE_COST_L [i] );
        fprintf(fptro, "%f\t", BASE_COST_L [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE NON-PRODUCTION WC\n");
for ( i = 0; i < NUM_NON_PROD_WC; i++ ) /* read base cost to operate non-production wc's */
        {
        fscanf(fptri1, "%f", &BASE_COST_N [i] );
        fprintf(fptro, "%f\t", BASE_COST_N [i] );
        }

fprintf(fptro, "\n\nBASE COST TO OPERATE SUPPORT WC\n");
/* read base cost to operate support wc */
fscanf(fptri1, "%f", &BASE_COST_V );
fprintf(fptro, "%f\t", BASE_COST_V );

fprintf(fptro, "\n\nMATERIAL COST FOR PRODUCTION");
```

```
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read material cost for production */
                {
                fprintf(fptro,"\nMaterial cost for Product Number %d\n", i );
                for ( j = 0; j < NUM_PROD_PERIODS; j++ )
                        {
                        fscanf(fptri1, "%f", &MA [i] [j] );
                        fprintf(fptro, "%4.2f\t", MA [i] [j] );
                        }
                }


        fprintf(fptro, "\n\nREPAIR LABOR COST\n");
        for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ ) /* read repair labor cost */
                {
                fscanf(fptri1, "%f", &RC [i] );
                fprintf(fptro, "%4.2f  ", RC [i] );
                }

        fprintf(fptro, "\n\nCONSISTENT UNITS REQUIRED BY PRODUCT AT NON_PRODUCTION
WC");
        for ( k = 0; k < NUM_NON_PROD_WC; k++ )
        {
        fprintf(fptro, "\n\nUnits required at wc %d", k );
        for ( i = 0; i < NUM_PRODUCTS; i++ ) /* read units required for non-production */
                {
                fprintf(fptro,"\n  Consistent non-production units for product number %d\n", i );
                fprintf(fptro, "  ");
                for ( j = 0; j < NUM_PERIODS; j++ )
                        {
                        fscanf(fptri1, "%f", &G [k] [i] [j] );
                        fprintf(fptro, "%4.2f  ", G [k] [i] [j] );
                        }
                }
        }

        fprintf(fptro, "\n\nMAXIMUM PRODUCTION PERIODS FOR EACH PRODUCT\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                {
                fscanf(fptri1, "%d", &max_periods[i] );
                fprintf(fptro, "%d\t", max_periods[i] );
                }

        fprintf(fptro, "\n\nEARLIEST PERIOD PRODUCTION OF A PRODUCT MAY
COMMENCE\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
                {
                fscanf(fptri1, "%d", &start_period[i] );
                fprintf(fptro, "%d\t", start_period[i] );
                }

        fprintf(fptro, "\n\nINVENTORY HOLDING COSTS FOR PRODUCTS IN EACH PERIOD\n");
```

```
for ( i = 0; i < NUM_PRODUCTS; i++ )
{
        fprintf(fptro, "\nPRODUCT %d\n", i );
        for ( j = 0; j < NUM_PROD_PERIODS; j++ )
        {
        fscanf(fptri1, "%f", &INVENTORY[i][j] );
        fprintf(fptro, "%8.2f  ", INVENTORY[i][j] );
        }
}

fprintf(fptro, "\n\nIMPROVMENT IN HOURS FROM SUPPORT CAP BUD PROJECT\n");
for ( i = 0; i < NUM_SUPPORT_PERIODS; i++ )
        {
        fscanf(fptri1, "%d", &CV[i] );
        fprintf(fptro, "%d  ", CV[i] );
        }

fprintf(fptro, "\n\nBEGINNING INVENTORY FOR EACH PRODUCT\n");
for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
        fscanf(fptri1, "%d", &BEGIN_INVENTORY[i] );
        fprintf(fptro, "%d\t", BEGIN_INVENTORY[i] );
        }

/* start read from mpsl.rep file */
fprintf(fptro, "\n\n\nOUTPUT FROM LAST STAGE OPTIMIZATION");
fscanf(fptri2, "%s", TITLE );
fprintf(fptro,"\n\nINPUT IS FROM:  %s", TITLE );

/* strip off header information */
fscanf(fptri2, "%s", PROJECT );
fprintf(fptro, "\n\n%s ", PROJECT );
fscanf(fptri2, "%s", CONSTRAINT );
fprintf(fptro, "%s\n", CONSTRAINT );
for ( i = 0; i < 2; i++ )
        {
        fscanf(fptri2, "%s", DEMAND );
        fprintf(fptro, "%s ", DEMAND );
        fscanf(fptri2, "%s", EQUAL );
        fprintf(fptro, "%s ", EQUAL );
        fscanf(fptri2, "%s", DEMAND_VALUE );
        fprintf(fptro, "%s\n", DEMAND_VALUE );
        }
fscanf(fptri2, "%s", VAR_NAME );
fscanf(fptri2, "%s", VAR_VALUE );
fscanf(fptri2, "%s", REDUCED );
fscanf(fptri2, "%s", COST );

fprintf(fptro, "\n\nVALUES OF Z FROM LAST STAGE OPTIMIZATION\n");
for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
```

```
                fscanf(fptri2, "%s %f %f", ZNAME, &Z[i], &ZJUNK );
                fprintf(fptro, "Z%d    %1.0f\n", i, Z[i] );
                }

        fprintf(fptro, "\n\nVALUE OF NON_PRODUCTION CAPITAL BUDGET VARIABLE FROM
LAST STAGE OPTIMIZATION\n");
        fscanf(fptri2, "%s %f", ENNAME, &EN );
        fprintf(fptro, "EN    %1.6f", EN );

        fprintf(fptro, "\n\nVALUES OF SETUP VARIABLES FROM LAST STAGE
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
                for ( j = start_period[i]; j < start_period[i] + max_periods[i]; j++ )
                {
                fscanf(fptri2, "%s %f %f", DELTANAME, &DELTA[i][j], &DELTAJUNK );
                fprintf(fptro, "%s   %1.0f\n", DELTANAME, DELTA[i][j] );
                }
        }

        fprintf(fptro, "\n\nVALUE OF PRODUCTION CAPITAL BUDGET VARIABLES FROM LAST
STAGE OPTIMIZATION\n");
        for ( i = 0; i < NUM_CAP_BUD_PROJECTS; i++ )
                {
                fscanf(fptri2, "%s %f %f", ELNAME, &EL[i], &ELJUNK );
                fprintf(fptro, "%s    %1.6f\n", ELNAME, EL[i] );
                }

        fprintf(fptro, "\n\nVALUE OF SUPPORT PRODUCTION CAPITAL BUDGET VARIABLE
FROM LAST STAGE OPTIMIZATION\n");
        fscanf(fptri2, "%s %f %f", EVNAME, &EV, &EVJUNK );
        fprintf(fptro, "EV    %1.6f\n", EV );

        fprintf(fptro, "\n\nVALUES OF PRODUCTION VARIABLES FROM LAST STAGE
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
                for ( j = start_period[i]; j < start_period[i] + max_periods[i]; j++ )
                {
                fscanf(fptri2, "%s %f %f", XNAME, &X[i][j], &XJUNK );
                fprintf(fptro, "%s    %12.6f\n", XNAME, X[i][j] );
                }
        }

        fprintf(fptro, "\n\nVALUES OF INVENTORY VARIABLES FROM LAST STAGE
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PRODUCTS; i++ )
        {
                for ( j = start_period[i]; j < start_period[i] + max_periods[i] - 1; j++ )
                {
                fscanf(fptri2, "%s %f %f", INAME, &I[i][j], &IJUNK );
```

```
                fprintf(fptro, "%s    %12.6f\n", INAME, I[i][j] );
                }
        }

        fprintf(fptro, "\n\nVALUES OF REGULAR TIME LABOR VARIABLES FROM LAST STAGE
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PROD_PERIODS; i++ )
        {
                for ( j = 0; j < NUM_PROD_WC; j++ )
                {
                fscanf(fptri2, "%s %f %f", wltNAME, &wlt[i][j], &wltJUNK );
                fprintf(fptro, "%s    %12.6f\n", wltNAME, wlt[i][j] );
                }
        }

        fprintf(fptro, "\n\nVALUES OF OVERTIME LABOR VARIABLES FROM LAST STAGE
OPTIMIZATION\n");
        for ( i = 0; i < NUM_PROD_PERIODS; i++ )
        {
                for ( j = 0; j < NUM_PROD_WC; j++ )
                {
                fscanf(fptri2, "%s %f %f", oltNAME, &olt[i][j], &oltJUNK );
                fprintf(fptro, "%s    %12.6f\n", oltNAME, olt[i][j] );
                }
        }

        fclose(fptri1);
        fclose(fptri2);

        FW = fwsscon( REVENUE, WLT, OLT, cl, cn, cv, SJT, PLJ, PWJ, pj,
        PR, ft, MA, RC, max_periods, start_period, BASE_COST_L, BASE_COST_N,
        BASE_COST_V, ACC, inflation, INVENTORY, BEGIN_INVENTORY, EN, X,
        I, wlt, olt, EL, EV );

        /* output portfolio future worth */
        fprintf(fptro, "\n\nOPTIMAL LAST STAGE PRODUCT PORTFOLIO FUTURE WORTH = $%-
13.2f\n", FW );

        fclose(fptro);
} /* end main */
```

```
/* fwsscon.h */
/* header file for fwsscon.c */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define NUM_PERIODS                    12
#define HORIZON_PERIOD                 11
#define NUM_SUPPORT_PERIODS            11
#define NUM_PROD_PERIODS               8
#define NUM_PRODUCTS                   5
#define NUM_PROD_WC                    3
#define NUM_NON_PROD_WC                4
#define NUM_CAP_BUD_PROJECTS           3
#define NUM_NON_PROD_CAP_BUD_PROJ      1
#define AGE                            5
#define TRUE                           1
#define FALSE                          0


int project, i, j;

/* counters for constraints */
int setup_constraint;
int dec_var;
int inv_lim;
int must_fund;
int nprdwc;

/* counter for product age */
int age;

/* base cost variables */
float TOT_BASE_COST;
float TOT_BASE_DISCOUNT;

/* discounted revenues from repair labor */
float ADJ_LABOR_REP_REV[NUM_PRODUCTS][NUM_PROD_PERIODS];

/* discounted cost from repair labor */
float ADJ_LABOR_REP_COST[NUM_PRODUCTS][NUM_PERIODS];

/* discounted parts revenues less costs */
float ADJ_PART[NUM_PRODUCTS][NUM_PERIODS];

/* production capital budgeting costs */
float TOTAL_CAP_BUD_EXPENSE[NUM_CAP_BUD_PROJECTS];

/* non-production capital budgeting expense */
float TOTAL_NP_CAP_BUD_EXPENSE;

/* adjusted inventory cost; includes stocking fee, loss of interest on
```

```
 revenues and repair costs */
float ACTUAL_INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS - 1];

/* total support capital budgeting cost */
float TOTAL_SUPPORT_CAP_BUD_COST;

/* total product hours per period */
float PRODUCT_HOURS[NUM_PRODUCTS][NUM_PROD_PERIODS][NUM_PROD_WC];

/* temporary future worth variables */
float FWW; /* reg labor */
float FWO; /* otime labor */
float FWX; /* product revenues less direct costs */
float FWI; /* inventory holding */
float FWBI; /* beginning inventory */
```

```
/* fwsscon.c */
#include "fwsscon.h"

/* Returns the future worth of each the portfolio of funded products after
        optimization of the last stage of the two stage model is complete */
float fwsscon( float REVENUE [NUM_PRODUCTS][NUM_PROD_PERIODS],
float WLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float OLT [NUM_PROD_WC] [NUM_PROD_PERIODS],
float cl [NUM_CAP_BUD_PROJECTS] [NUM_PERIODS], float cn [NUM_PERIODS],
float cv [NUM_PERIODS],
float SJT [NUM_PRODUCTS][AGE],
int PLJ [NUM_PRODUCTS],
int PWJ [NUM_PRODUCTS], float pj [NUM_PRODUCTS][AGE],
float PR [NUM_PRODUCTS][AGE], float ft [NUM_PERIODS],
float MA [NUM_PRODUCTS][NUM_PROD_PERIODS], float RC [NUM_PERIODS],
int max_periods[NUM_PRODUCTS],
int start_period[NUM_PRODUCTS], float BASE_COST_L[NUM_PROD_WC],
float BASE_COST_N[NUM_NON_PROD_WC],
float BASE_COST_V, float ACC, float inflation,
float INVENTORY[NUM_PRODUCTS][NUM_PROD_PERIODS],
int BEGIN_INVENTORY[NUM_PRODUCTS], float EN,
float X[NUM_PRODUCTS][NUM_PROD_PERIODS],
float I[NUM_PRODUCTS][NUM_PROD_PERIODS - 1],
float wlt[NUM_PROD_PERIODS][NUM_PROD_WC],
float olt[NUM_PROD_PERIODS][NUM_PROD_WC], float EL[NUM_CAP_BUD_PROJECTS],
float EV )

{
float FW;
register product, period, wc;

        /* build total non-production capital budgeting costs */
        TOTAL_NP_CAP_BUD_EXPENSE = 0;
        for ( period = 0; period < NUM_PERIODS; period++ )
                TOTAL_NP_CAP_BUD_EXPENSE +=  cn[period] * pow( 1 + ACC,
HORIZON_PERIOD - period );

        /* build total production capital budgeting costs */
        for ( project = 0; project < NUM_CAP_BUD_PROJECTS; project++ )
        {
                for ( period = 0; period < NUM_PERIODS; period++ )
                {
                TOTAL_CAP_BUD_EXPENSE[project] = 0;
                }
        }
        for ( project = 0; project < NUM_CAP_BUD_PROJECTS; project++ )
        {
                for ( period = 0; period < NUM_PERIODS; period++ )
                {
                TOTAL_CAP_BUD_EXPENSE[project] += cl[project][period] * pow( 1 + ACC,
HORIZON_PERIOD - period );
```

```
                }
        }

        /* build total future worth of repair revenues and costs  for each product by
        period it was produced */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = 0; period < NUM_PROD_PERIODS; period++ )
                {
                ADJ_LABOR_REP_REV[product][period] = 0;
                ADJ_LABOR_REP_COST[product][period] = 0;
                ADJ_PART[product][period] = 0;
                }
        }
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period =  start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                for ( age = PWJ[product]; age < PLJ[product]; age++ )
                ADJ_LABOR_REP_REV[product][period] += SJT[product][age]*ft[age] * pow( 1 +
ACC, HORIZON_PERIOD - ( period + age ) );
                }
        }
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period =  start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                for ( age = 0; age < PLJ[product]; age++ )
                ADJ_LABOR_REP_COST[product][period] += SJT[product][age]*RC[age] * pow( 1 +
ACC, HORIZON_PERIOD - ( period + age ) );
                }
        }
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period =  start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                for ( age = 0; age < PLJ[product]; age++ )
                {
                if ( age < PWJ[product] )
                        ADJ_PART[product][period] += - pj[product][age] * pow( 1 + inflation, period
+ age ) * pow( 1 + ACC, HORIZON_PERIOD - ( period + age ) );
                if ( age >= PWJ[product] )
                        ADJ_PART[product][period] += ( PR[product][age] - pj[product][age] ) * pow(
1 + inflation, period + age ) * pow( 1 + ACC, HORIZON_PERIOD - ( period + age ) );
                }
                }
        }
```

```
        /* actual inventory cost - includes loss of interest on revenues and adjusts repair costs and
revenues */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product] - 1; period++ )
                {
                ACTUAL_INVENTORY[product][period] = ( INVENTORY[product][period]
                + REVENUE[product][period] ) * pow( 1 + ACC, HORIZON_PERIOD - period )
                + ADJ_LABOR_REP_REV[product][period] -
                ADJ_LABOR_REP_COST[product][period] +
                ADJ_PART[product][period] -
                ( REVENUE[product][period + 1] * pow( 1 + ACC, HORIZON_PERIOD - period - 1)
                + ADJ_LABOR_REP_REV[product][period + 1 ] -
                ADJ_LABOR_REP_COST[product][period + 1] +
                ADJ_PART[product][period + 1] );
                }
        }

        /* determine total support capital budget cost */
        TOTAL_SUPPORT_CAP_BUD_COST = 0;
        for ( period = 0; period < NUM_PERIODS; period++ )
                TOTAL_SUPPORT_CAP_BUD_COST += cv[period] * pow( 1 + ACC,
HORIZON_PERIOD - period );

        /* calculate total base costs at production and non-production wc's */
        TOT_BASE_COST = 0;
        TOT_BASE_DISCOUNT = 0;
        /* determine total base production cost in each period */
        for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                TOT_BASE_COST += BASE_COST_L[wc];

        /* add total base non-production cost to base production cost */
        for ( wc = 0; wc < NUM_NON_PROD_WC; wc++ )
                TOT_BASE_COST += BASE_COST_N[wc];

        /* add base cost of support to other base costs */
        TOT_BASE_COST += BASE_COST_V;

        /* Determine total discounted constant dollar costs of base capacity */
        for ( period = 0; period < NUM_PERIODS; period++ )
                TOT_BASE_DISCOUNT += TOT_BASE_COST * pow( 1 + inflation, period ) * pow(
1 + ACC, HORIZON_PERIOD - period );

        /* initialize future worth variable */
        FW = 0;

        /* if EN = 1, subtract non-production capital budgeting costs */
        FW = FW - EN * TOTAL_NP_CAP_BUD_EXPENSE;

        /* if EL[project] = 1 subtract production capital budgeting cost */
```

```
        for ( project = 0; project < NUM_CAP_BUD_PROJECTS; project++ )
                FW = FW - EL[project] * TOTAL_CAP_BUD_EXPENSE[project];

        /* if EV = 1, subtract total capital budgeting support cost */
        FW = FW - TOTAL_SUPPORT_CAP_BUD_COST * EV;

        /* add discounted value of production */
        FWX = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product]; period++ )
                {
                FWX += X[product][period] * ( ( REVENUE[product][period] - MA[product][period] )
* pow( 1 + ACC, HORIZON_PERIOD - period ) + ADJ_LABOR_REP_REV[product][period] -
ADJ_LABOR_REP_COST[product][period] + ADJ_PART[product][period] );
                }
        }
        FW = FW + FWX;

        /* add discounted values of inventory variables */
        FWI = 0;
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                for ( period = start_period[product]; period < start_period[product] +
max_periods[product] - 1; period++ )
                {
                FWI += I[product][period] * ACTUAL_INVENTORY[product][period];
                }
        }
        FW = FW - FWI;

        /* subtract discounted labor costs for each product */
        FWW = 0;
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
                        FWW += wlt[period][wc] * WLT[wc][period] * pow( 1 + ACC,
HORIZON_PERIOD - period );
                        }
        }

        FW = FW - FWW;

        /* subtract discounted labor costs for each product */
        FWO = 0;
        for ( period = 0; period < NUM_PROD_PERIODS; period++ )
        {
                for ( wc = 0; wc < NUM_PROD_WC; wc++ )
                        {
```

```
                        FWO += olt[period][wc] * OLT[wc][period] * pow( 1 + ACC,
HORIZON_PERIOD - period );
                        }
        }

        FW = FW - FWO;

        /* subtract future worth of base capacity costs to FW */
        FW = FW - TOT_BASE_DISCOUNT;

        /* determine revenues from beginning inventory variables */
        for ( product = 0; product < NUM_PRODUCTS; product++ )
        {
                FWBI += BEGIN_INVENTORY[product] * REVENUE[product][0] * pow( 1 + ACC,
HORIZON_PERIOD);
        }

        FW = FW + FWBI;

        return ( FW );

} /* end main */
```

APPENDIX G: Comparison of Production and Inventory Quantities

Table G.1. Production and Inventory Levels Obtained from Both Models During Test One.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	50.150879	57.999802	i_{00}	1.150877	8.999802
x_{01}	49.424561	40.999802	i_{01}	0.575439	0.0
x_{02}	49.424561	57.420898	i_{02}	0.0	7.420898
x_{03}	50.0	42.57880			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	50.0	67.851997	i_{24}	0.0	17.851997
x_{25}	50.0	82.147697	i_{25}	0.0	49.999695
x_{26}	50.0	49.999802	i_{26}	0.0	49.999496
x_{27}	50.0	0.0			
x_{30}	49.0	48.999802	i_{30}	0.0	0.0
x_{31}	50.0	61.304199	i_{31}	0.0	11.304199
x_{32}	50.0	38.695499	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	50.0	74.49070	i_{43}	0.0	24.49070
x_{44}	50.0	74.6520	i_{44}	0.0	49.14270
x_{45}	50.0	50.85680	i_{45}	0.0	49.99950
x_{46}	50.0	49.999802	i_{46}	0.0	49.999302
x_{47}	50.0	0.0			

Table G.2. Production and Inventory Levels Obtained from Both Models During Test Two.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	49.666668	59.999802	i_{00}	0.666667	0.0
x_{01}	49.666668	44.499802	i_{01}	0.333333	0.0
x_{02}	49.666668	44.499802	i_{02}	0.0	0.0
x_{03}	50.0	49.999802			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	50.0	66.149803	i_{24}	0.0	25.499802
x_{25}	50.0	33.849800	i_{25}	0.0	49.999603
x_{26}	50.0	49.999802	i_{26}	0.0	0.0
x_{27}	50.0	49.999802			
x_{30}	99.666664	98.999802	i_{30}	0.666667	13.304199
x_{31}	99.666664	99.999802	i_{31}	0.333333	0.0
x_{32}	99.666664	99.999802	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	100.0	123.490700	i_{43}	0.0	0.0
x_{44}	100.0	76.508904	i_{44}	0.0	0.0
x_{45}	100.0	111.745300	i_{45}	0.0	11.74530
x_{46}	100.0	94.127098	i_{46}	0.0	5.872398
x_{47}	100.0	94.127098			

Table G.3. Production and Inventory Levels Obtained from Both Models During Test Three.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	49.0	25.927401	i_{00}	0.0	50.0
x_{01}	50.0	49.999802	i_{01}	0.0	49.999802
x_{02}	50.0	0.0	i_{02}	0.0	0.0
x_{03}	50.0	49.999802			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	50.0	75.499802	i_{24}	0.0	50.0
x_{25}	50.0	74.499802	i_{25}	0.0	49.999901
x_{26}	50.0	0.0	i_{26}	0.0	0.0
x_{27}	50.0	49.999802			
x_{30}	126.231125	162.304199	i_{30}	0.0	87.733101
x_{31}	125.407593	95.416496	i_{31}	0.0	50.62830
x_{32}	125.407593	137.833206	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	150.0	149.999802	i_{43}	0.0	0.0
x_{44}	150.0	149.999802	i_{44}	0.0	0.0
x_{45}	150.0	150.749802	i_{45}	0.0	0.749802
x_{46}	150.0	205.122604	i_{46}	0.0	55.872406
x_{47}	150.0	94.127098			

Table G.4. Production and Inventory Levels Obtained from Both Models During Test Four.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	99.666664	105.699799	i_{00}	0.666667	6.699799
x_{01}	99.666664	104.499802	i_{01}	0.333333	11.19960
x_{02}	99.666664	104.499802	i_{02}	0.0	15.699402
x_{03}	100.0	84.299797			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	100.0	100.299797	i_{24}	0.0	0.299797
x_{25}	100.0	99.899803	i_{25}	0.0	0.19960
x_{26}	100.0	99.899803	i_{26}	0.0	0.099403
x_{27}	100.0	99.899803			
x_{30}	49.666668	48.999802	i_{30}	0.666667	0.0
x_{31}	49.666668	49.999802	i_{31}	0.333333	0.0
x_{32}	49.666668	49.999802	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	50.0	50.307499	i_{43}	0.0	0.307499
x_{44}	50.0	49.692101	i_{44}	0.0	0.0
x_{45}	50.0	49.999802	i_{45}	0.0	0.0
x_{46}	50.0	49.999802	i_{46}	0.0	0.0
x_{47}	50.0	49.999802			

Table G.5. Production and Inventory Levels Obtained from Both Models During Test Five.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	99.0	120.781403	i_{00}	0.0	21.781403
x_{01}	100.0	0.0	i_{01}	0.0	0.0
x_{02}	103.830650	124.399803	i_{02}	3.830652	24.399803
x_{03}	96.169350	75.599800			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	100.0	157.833206	i_{24}	0.0	57.833206
x_{25}	100.0	142.166504	i_{25}	0.0	99.999710
x_{26}	100.0	0.0	i_{26}	0.0	0.0
x_{27}	100.0	99.999802			
x_{30}	79.013000	65.504799	i_{30}	0.0	0.0
x_{31}	78.189468	137.833206	i_{31}	0.0	37.833206
x_{32}	72.915382	33.416500	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	100.0	111.749802	i_{43}	0.0	11.749802
x_{44}	100.0	85.092796	i_{44}	0.0	0.0
x_{45}	100.0	99.326302	i_{45}	0.0	0.0
x_{46}	100.0	150.076797	i_{46}	0.0	50.076797
x_{47}	100.0	49.922901			

Table G.6. Production and Inventory Levels Obtained from Both Models During Test Six.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	99.0	185.492203	i_{00}	0.0	86.492203
x_{01}	100.0	0.0	i_{01}	0.0	0.0
x_{02}	105.821220	179.024704	i_{02}	5.821223	79.024704
x_{03}	55.534996	0.0			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	100.0	199.999802	i_{24}	0.0	100.0
x_{25}	100.0	99.999802	i_{25}	0.0	99.999802
x_{26}	100.0	0.0	i_{26}	0.0	0.0
x_{27}	100.0	99.999802			
x_{30}	79.609718	0.0	i_{30}	0.0	0.0
x_{31}	78.786186	137.833206	i_{31}	0.0	0.0
x_{32}	70.771461	0.0	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	150.0	207.374802	i_{43}	0.0	57.374802
x_{44}	119.968735	46.77140	i_{44}	38.972836	21.448517
x_{45}	127.972885	131.624802	i_{45}	51.290806	17.029221
x_{46}	127.549484	201.104401	i_{46}	28.840292	100.077103
x_{47}	121.159706	49.922901			

Table G.7. Production and Inventory Levels Obtained from Both Models During Test Seven.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	150.310089	185.492203	i_{00}	1.310096	36.492203
x_{01}	150.310089	79.064499	i_{01}	1.620192	0.0
x_{02}	150.310089	165.499802	i_{02}	1.930288	15.499802
x_{03}	137.040207	84.999802			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	150.0	251.937805	i_{24}	0.0	101.937805
x_{25}	150.0	57.966499	i_{25}	0.0	9.904305
x_{26}	150.0	165.799805	i_{26}	0.0	25.704109
x_{27}	150.0	99.899803			
x_{30}	27.748926	0.0	i_{30}	0.0	0.0
x_{31}	27.748926	71.19590	i_{31}	0.0	21.19590
x_{32}	27.748926	0.0	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	50.0	99.999802	i_{43}	0.0	49.999802
x_{44}	50.0	0.0	i_{44}	0.0	0.0
x_{45}	50.0	99.999802	i_{45}	0.0	49.999802
x_{46}	50.0	0.0	i_{46}	0.0	0.0
x_{47}	50.0	49.999802			

Table G.8. Production and Inventory Levels Obtained from Both Models During Test Eight.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	91.760292	185.492203	i_{00}	0.0	36.492203
x_{01}	150.0	86.160599	i_{01}	0.0	0.0
x_{02}	150.0	165.499802	i_{02}	0.0	15.499802
x_{03}	96.936996	50.167099			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	150.0	251.937805	i_{24}	0.0	101.937805
x_{25}	150.0	150.027695	i_{25}	0.0	101.96550
x_{26}	150.0	105.78640	i_{26}	0.0	57.75190
x_{27}	150.0	67.851997			
x_{30}	87.901016	0.0	i_{30}	0.0	0.0
x_{31}	28.653173	65.282501	i_{31}	0.0	0.0
x_{32}	28.653173	0.0	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	100.0	143.540802	i_{43}	0.0	43.540802
x_{44}	83.629616	0.0	i_{44}	0.0	0.0
x_{45}	83.629616	91.974403	i_{45}	32.404259	0.0
x_{46}	83.352928	125.347702	i_{46}	15.757188	25.347702
x_{47}	76.993240	74.6520			

Table G.9. Production and Inventory Levels Obtained from Both Models During Test Nine.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	78.845650	185.492203	i_{00}	0.0	36.492203
x_{01}	150.0	94.193497	i_{01}	0.0	0.0
x_{02}	150.0	165.499802	i_{02}	0.0	15.499802
x_{03}	56.383556	0.0			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	150.0	251.937805	i_{24}	0.0	124.377495
x_{25}	150.0	173.666504	i_{25}	0.0	148.044006
x_{26}	150.0	149.999802	i_{26}	0.0	149.999802
x_{27}	150.0	0.0			
x_{30}	101.169968	0.0	i_{30}	0.0	0.0
x_{31}	28.783932	58.588501	i_{31}	0.0	0.0
x_{32}	28.783932	0.0	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	150.0	203.314407	i_{43}	0.0	53.314407
x_{44}	83.787590	0.0	i_{44}	0.0	0.0
x_{45}	83.787590	70.591599	i_{45}	44.685806	48.197140
x_{46}	83.510376	91.695198	i_{46}	0.0	0.145599
x_{47}	77.155785	149.854401			

Table G.10. Production and Inventory Levels Obtained from Both Models During Test Ten.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	50.150879	57.999802	i_{00}	1.150877	8.999802
x_{01}	49.424561	40.999802	i_{01}	0.575439	0.0
x_{02}	49.424561	57.420898	i_{02}	0.0	7.420898
x_{03}	50.0	42.57880			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	50.0	67.851997	i_{24}	0.0	17.851997
x_{25}	50.0	82.147697	i_{25}	0.0	49.999695
x_{26}	50.0	49.999802	i_{26}	0.0	49.999496
x_{27}	50.0	0.0			
x_{30}	49.0	48.999802	i_{30}	0.0	0.0
x_{31}	50.0	61.304199	i_{31}	0.0	11.304199
x_{32}	50.0	38.695499	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	50.0	74.49070	i_{43}	0.0	24.49070
x_{44}	50.0	74.6520	i_{44}	0.0	49.14270
x_{45}	50.0	50.85680	i_{45}	44.685806	49.99950
x_{46}	50.0	49.999802	i_{46}	0.0	49.999302
x_{47}	50.0	0.0			

Table G.11. Production and Inventory Levels Obtained from Both Models During Test Eleven.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	49.666668	59.999802	i_{00}	0.666667	0.0
x_{01}	49.666668	44.499802	i_{01}	0.333333	0.0
x_{02}	49.666668	44.499802	i_{02}	0.0	0.0
x_{03}	50.0	49.999802			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	50.0	66.149803	i_{24}	0.0	25.499802
x_{25}	50.0	33.849800	i_{25}	0.0	49.999603
x_{26}	50.0	49.999802	i_{26}	0.0	0.0
x_{27}	50.0	49.999802			
x_{30}	99.666664	98.999802	i_{30}	0.666667	13.304199
x_{31}	99.666664	99.999802	i_{31}	0.333333	0.0
x_{32}	99.666664	99.999802	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	100.0	123.49070	i_{43}	0.0	0.0
x_{44}	100.0	76.508904	i_{44}	0.0	0.0
x_{45}	100.0	111.74530	i_{45}	44.685806	11.74530
x_{46}	100.0	94.127098	i_{46}	0.0	5.872398
x_{47}	100.0	94.127098			

Table G.12. Production and Inventory Levels Obtained from Both Models During Test Twelve.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	49.0	19.59170	i_{00}	0.0	50.0
x_{01}	50.0	49.999802	i_{01}	0.0	49.999802
x_{02}	55.899818	0.0	i_{02}	0.0	0.0
x_{03}	44.100182	49.999802			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	50.0	75.499802	i_{24}	25.499802	50.0
x_{25}	50.0	74.499802	i_{25}	49.999603	49.999901
x_{26}	50.0	0.0	i_{26}	0.0	0.0
x_{27}	50.0	49.999802			
x_{30}	112.019432	162.304199	i_{30}	0.0	13.304199
x_{31}	111.19590	95.416496	i_{31}	0.0	0.0
x_{32}	103.072968	137.833206	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	150.0	149.999802	i_{43}	0.0	0.0
x_{44}	150.0	149.999802	i_{44}	0.0	0.0
x_{45}	150.0	150.749802	i_{45}	0.0	0.749802
x_{46}	150.0	205.122604	i_{46}	0.0	55.872406
x_{47}	150.0	94.127098			

Table G.13. Production and Inventory Levels Obtained from Both Models During Test Thirteen.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	99.666664	105.699799	i_{00}	0.666667	6.699799
x_{01}	99.666664	104.499802	i_{01}	0.333333	11.19960
x_{02}	99.666664	104.499802	i_{02}	0.0	15.699402
x_{03}	100.0	84.299797			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	100.0	100.299797	i_{24}	0.0	0.299797
x_{25}	100.0	99.899803	i_{25}	0.0	0.19960
x_{26}	100.0	99.899803	i_{26}	0.0	0.099403
x_{27}	100.0	99.899803			
x_{30}	49.666668	48.999802	i_{30}	0.666667	0.0
x_{31}	49.666668	49.999802	i_{31}	0.333333	0.0
x_{32}	49.666668	49.999802	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	50.0	50.307499	i_{43}	0.0	0.307499
x_{44}	50.0	49.692101	i_{44}	0.0	0.0
x_{45}	50.0	49.999802	i_{45}	0.0	0.0
x_{46}	50.0	49.999802	i_{46}	0.0	0.0
x_{47}	50.0	49.999802			

Table G.14. Production and Inventory Levels Obtained from Both Models During Test Fourteen.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	99.0	103.633202	i_{00}	0.0	4.633202
x_{01}	100.0	0.0	i_{01}	0.0	0.0
x_{02}	112.811234	124.399803	i_{02}	12.811233	24.399803
x_{03}	87.188766	75.59980			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	100.0	157.833206	i_{24}	0.0	57.833206
x_{25}	100.0	142.166504	i_{25}	0.0	99.999710
x_{26}	100.0	0.0	i_{26}	0.0	0.0
x_{27}	100.0	99.999802			
x_{30}	66.567223	65.504799	i_{30}	0.0	0.0
x_{31}	65.190407	137.833206	i_{31}	0.0	37.833206
x_{32}	47.551754	33.41650	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	100.0	111.749802	i_{43}	0.0	11.749802
x_{44}	100.0	77.734497	i_{44}	0.0	0.0
x_{45}	100.0	91.905998	i_{45}	0.0	0.0
x_{46}	100.0	150.076797	i_{46}	0.0	50.076797
x_{47}	100.0	49.922901			

Table G.15. Production and Inventory Levels Obtained from Both Models During Test Fifteen.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	99.0	88.103302	i_{00}	0.0	0.0
x_{01}	100.0	0.0	i_{01}	0.0	0.0
x_{02}	110.579544	152.210495	i_{02}	10.579547	52.210495
x_{03}	42.142677	0.0			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	100.0	199.999802	i_{24}	0.0	100.0
x_{25}	100.0	99.999802	i_{25}	0.0	99.999802
x_{26}	100.0	0.0	i_{26}	0.0	0.0
x_{27}	100.0	99.999802			
x_{30}	67.705734	86.886597	i_{30}	0.0	0.0
x_{31}	66.328926	137.833206	i_{31}	0.0	0.0
x_{32}	51.762882	0.0	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	150.0	207.374802	i_{43}	0.0	57.374802
x_{44}	118.538269	39.592098	i_{44}	0.0	39.562946
x_{45}	118.538269	130.048492	i_{45}	0.0	69.401596
x_{46}	94.314064	201.104401	i_{46}	0.0	120.505997
x_{47}	118.538269	29.493999			

Table G.16. Production and Inventory Levels Obtained from Both Models During Test Sixteen.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	115.621056	145.746796	i_{00}	0.0	0.0
x_{01}	114.894737	79.064499	i_{01}	0.0	0.0
x_{02}	114.894737	152.210495	i_{02}	0.0	2.210495
x_{03}	122.442108	84.999802			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	150.0	241.583298	i_{24}	0.0	91.583298
x_{25}	150.0	57.966499	i_{25}	0.0	0.0
x_{26}	150.0	165.799805	i_{26}	0.0	15.799805
x_{27}	150.0	99.899803			
x_{30}	49.0	7.52230	i_{30}	0.0	0.0
x_{31}	50.0	71.19590	i_{31}	0.0	21.19590
x_{32}	50.0	0.0	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	50.0	99.999802	i_{43}	0.0	49.999802
x_{44}	50.0	0.0	i_{44}	0.0	0.0
x_{45}	50.0	99.999802	i_{45}	0.0	49.999802
x_{46}	50.0	0.0	i_{46}	0.0	0.0
x_{47}	50.0	49.999802			

Table G.17. Production and Inventory Levels Obtained from Both Models During Test Seventeen.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	99.223549	144.625198	i_{00}	0.0	0.0
x_{01}	108.719467	86.160599	i_{01}	0.0	0.0
x_{02}	152.210526	152.210495	i_{02}	2.210526	2.210495
x_{03}	93.494736	50.167099			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	150.0	241.583298	i_{24}	0.0	149.966415
x_{25}	150.0	150.027695	i_{25}	0.0	149.994110
x_{26}	150.0	105.78640	i_{26}	0.0	105.78640
x_{27}	150.0	44.213402			
x_{30}	71.576271	9.06650	i_{30}	0.0	0.0
x_{31}	58.502186	65.282501	i_{31}	0.0	0.0
x_{32}	0.0	0.0	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	100.0	143.540802	i_{43}	0.0	43.540802
x_{44}	81.957573	0.0	i_{44}	0.0	0.0
x_{45}	68.474403	84.794998	i_{45}	0.0	2.581497
x_{46}	36.413960	124.401199	i_{46}	0.0	26.982697
x_{47}	73.442017	73.017303			

Table G.18. Production and Inventory Levels Obtained from Both Models During Test Eighteen.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	88.225693	140.763107	i_{00}	0.0	0.0
x_{01}	120.631439	94.193497	i_{01}	0.0	0.0
x_{02}	152.210526	152.210495	i_{02}	2.210526	2.210495
x_{03}	54.454994	0.0			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	150.0	241.583298	i_{24}	0.0	140.659195
x_{25}	150.0	159.336304	i_{25}	0.0	149.995499
x_{26}	150.0	149.999802	i_{26}	0.0	149.999802
x_{27}	150.0	0.0			
x_{30}	86.718246	14.3840	i_{30}	0.0	0.0
x_{31}	42.101639	58.588501	i_{31}	0.0	0.0
x_{32}	0.0	0.0	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	150.0	203.314407	i_{43}	0.0	53.314407
x_{44}	82.310493	0.0	i_{44}	0.0	0.0
x_{45}	65.057274	76.374802	i_{45}	0.0	76.374802
x_{46}	37.586731	84.539703	i_{46}	0.0	41.638901
x_{47}	75.620529	108.361099			

Table G.19. Production and Inventory Levels Obtained from Both Models During Test Nineteen.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	49.936844	57.999802	i_{00}	0.936842	8.999802
x_{01}	49.210526	40.999802	i_{01}	0.147368	0.0
x_{02}	49.210526	57.420898	i_{02}	0.0	7.420898
x_{03}	50.0	42.57880			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	50.0	67.851997	i_{24}	0.0	17.851997
x_{25}	50.0	82.147697	i_{25}	0.0	49.999695
x_{26}	50.0	49.999802	i_{26}	0.0	49.999496
x_{27}	50.0	0.0			
x_{30}	49.0	48.999802	i_{30}	0.0	0.0
x_{31}	50.0	61.304199	i_{31}	0.0	11.304199
x_{32}	50.0	38.695499	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	50.0	74.49070	i_{43}	0.0	24.49070
x_{44}	50.0	74.6520	i_{44}	0.0	49.14270
x_{45}	50.0	50.85680	i_{45}	0.0	49.99950
x_{46}	50.0	49.999802	i_{46}	0.0	49.999302
x_{47}	50.0	0.0			

Table G.20. Production and Inventory Levels Obtained from Both Models During Test Twenty.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	49.0	59.999802	i_{00}	0.0	10.999802
x_{01}	50.0	44.499802	i_{01}	0.0	5.499603
x_{02}	50.0	44.499802	i_{02}	0.0	0.0
x_{03}	50.0	49.999802			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	50.0	66.149803	i_{24}	0.0	16.149803
x_{25}	50.0	33.849800	i_{25}	0.0	0.0
x_{26}	50.0	49.999802	i_{26}	0.0	0.0
x_{27}	50.0	49.999802			
x_{30}	99.0	98.999802	i_{30}	0.0	0.0
x_{31}	100.0	99.999802	i_{31}	0.0	0.0
x_{32}	100.0	99.999802	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	100.0	123.49070	i_{43}	0.0	23.49070
x_{44}	100.0	76.508904	i_{44}	0.0	0.0
x_{45}	100.0	111.74530	i_{45}	0.0	11.74530
x_{46}	100.0	94.127098	i_{46}	0.0	5.872398
x_{47}	100.0	94.127098			

Table G.21. Production and Inventory Levels Obtained from Both Models During Test Twenty One.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	49.0	19.59170	i_{00}	0.0	0.0
x_{01}	50.0	49.999802	i_{01}	0.0	0.0
x_{02}	50.142857	0.0	i_{02}	0.142857	0.0
x_{03}	49.857143	49.999802			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	50.0	75.499802	i_{24}	0.0	25.499802
x_{25}	50.0	74.499802	i_{25}	0.0	49.999603
x_{26}	50.0	0.0	i_{26}	0.0	0.0
x_{27}	50.0	49.999802			
x_{30}	115.235291	162.304199	i_{30}	0.0	13.304199
x_{31}	114.411766	95.416496	i_{31}	0.0	0.0
x_{32}	96.542465	137.833206	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	150.0	149.999802	i_{43}	0.0	0.0
x_{44}	150.0	149.999802	i_{44}	0.0	0.0
x_{45}	150.0	150.749802	i_{45}	0.0	0.749802
x_{46}	150.0	205.122604	i_{46}	0.0	55.872406
x_{47}	150.0	94.127098			

Table G.22. Production and Inventory Levels Obtained from Both Models During Test Twenty Two.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	99.0	105.699799	i_{00}	0.0	6.699799
x_{01}	100.0	104.499802	i_{01}	0.0	11.19960
x_{02}	100.0	104.499802	i_{02}	0.0	15.699402
x_{03}	100.0	84.299797			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	100.0	100.299797	i_{24}	0.0	0.299797
x_{25}	100.0	99.899803	i_{25}	0.0	0.19960
x_{26}	100.0	99.899803	i_{26}	0.0	0.099403
x_{27}	100.0	99.899803			
x_{30}	49.0	48.999802	i_{30}	0.0	0.0
x_{31}	50.0	49.999802	i_{31}	0.0	16.747816
x_{32}	50.0	49.999802	i_{32}	0.0	17.39370
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	50.0	50.307499	i_{43}	0.0	0.307499
x_{44}	50.0	49.692101	i_{44}	0.0	0.0
x_{45}	50.0	49.999802	i_{45}	0.0	0.0
x_{46}	50.0	49.999802	i_{46}	0.0	0.0
x_{47}	50.0	49.999802			

Table G.23. Production and Inventory Levels Obtained from Both Models During Test Twenty Three.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	95.069336	103.633202	i_{00}	0.0	4.633202
x_{01}	100.0	0.0	i_{01}	0.0	0.0
x_{02}	106.505264	124.399803	i_{02}	6.505263	24.399803
x_{03}	93.494736	75.59980			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	100.0	157.833206	i_{24}	0.0	57.833206
x_{25}	100.0	142.166504	i_{25}	0.0	99.999710
x_{26}	100.0	0.0	i_{26}	0.0	0.0
x_{27}	100.0	99.999802			
x_{30}	77.295837	65.504799	i_{30}	0.0	0.0
x_{31}	70.507248	137.833206	i_{31}	0.0	37.833206
x_{32}	35.023613	33.41650	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	100.0	111.749802	i_{43}	0.0	11.749802
x_{44}	100.0	77.734497	i_{44}	0.0	0.0
x_{45}	100.0	91.905998	i_{45}	0.0	0.0
x_{46}	100.0	150.076797	i_{46}	0.0	50.076797
x_{47}	100.0	49.922901			

Table G.24. Production and Inventory Levels Obtained from Both Models During Test Twenty Four.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	95.069336	88.103302	i_{00}	0.0	0.0
x_{01}	100.0	0.0	i_{01}	0.0	0.0
x_{02}	123.627037	152.210495	i_{02}	23.627039	52.210495
x_{03}	49.857143	0.0			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	100.0	199.999802	i_{24}	0.0	100.0
x_{25}	100.0	99.999802	i_{25}	0.0	99.999802
x_{26}	100.0	0.0	i_{26}	0.0	0.0
x_{27}	100.0	99.999802			
x_{30}	77.295837	86.886597	i_{30}	0.0	0.0
x_{31}	70.507248	137.833206	i_{31}	0.0	0.0
x_{32}	23.619183	0.0	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	150.0	207.374802	i_{43}	0.0	57.374802
x_{44}	126.846153	39.592098	i_{44}	0.0	39.562946
x_{45}	125.977989	130.048492	i_{45}	0.0	69.401596
x_{46}	76.977692	201.104401	i_{46}	29.182680	120.505997
x_{47}	120.817322	29.493999			

Table G.25. Production and Inventory Levels Obtained from Both Models During Test Twenty Five.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	115.621056	145.746796	i_{00}	0.0	0.0
x_{01}	114.894737	79.064499	i_{01}	0.0	0.0
x_{02}	114.894737	152.210495	i_{02}	0.0	2.210495
x_{03}	122.442108	84.999802			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	150.0	241.583298	i_{24}	0.0	91.583298
x_{25}	150.0	57.966499	i_{25}	0.0	0.0
x_{26}	150.0	165.799805	i_{26}	0.0	15.799805
x_{27}	150.0	99.899803			
x_{30}	49.0	7.52230	i_{30}	0.0	0.0
x_{31}	50.0	71.19590	i_{31}	0.0	21.19590
x_{32}	50.0	0.0	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	50.0	99.999802	i_{43}	0.0	49.999802
x_{44}	50.0	0.0	i_{44}	0.0	0.0
x_{45}	50.0	99.999802	i_{45}	0.0	49.999802
x_{46}	50.0	0.0	i_{46}	0.0	0.0
x_{47}	50.0	49.999802			

Table G.26. Production and Inventory Levels Obtained from Both Models During Test Twenty Six.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	95.069336	144.625198	i_{00}	0.0	0.0
x_{01}	112.730209	86.160599	i_{01}	0.0	0.0
x_{02}	152.210526	152.210495	i_{02}	2.210526	2.210495
x_{03}	93.494736	50.167099			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	150.0	241.583298	i_{24}	0.0	149.966415
x_{25}	150.0	150.027695	i_{25}	0.0	149.994110
x_{26}	150.0	105.78640	i_{26}	0.0	105.78640
x_{27}	150.0	44.213402			
x_{30}	77.295837	9.06650	i_{30}	0.0	0.0
x_{31}	52.980141	65.282501	i_{31}	0.0	0.0
x_{32}	0.0	0.0	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	100.0	143.540802	i_{43}	0.0	43.540802
x_{44}	82.413612	0.0	i_{44}	0.0	0.0
x_{45}	67.991135	84.794998	i_{45}	0.0	2.581497
x_{46}	36.415585	124.401199	i_{46}	0.0	26.982697
x_{47}	73.459427	73.017303			

Table G.27. Production and Inventory Levels Obtained from Both Models During Test Twenty Seven.

Production Variable	Production Quantity: Concurrent Model	Production Quantity: Sequential Model	Inventory Variable	Inventory Variable: Concurrent Model	Inventory Variable: Sequential Model
x_{00}	95.069336	140.763107	i_{00}	0.0	0.0
x_{01}	105.320999	94.193497	i_{01}	0.0	0.0
x_{02}	152.210526	152.210495	i_{02}	2.210526	2.210495
x_{03}	49.857143	0.0			
x_{13}	0.0	0.0	i_{13}	0.0	0.0
x_{14}	0.0	0.0	i_{14}	0.0	0.0
x_{15}	0.0	0.0	i_{15}	0.0	0.0
x_{16}	0.0	0.0	i_{16}	0.0	0.0
x_{17}	0.0	0.0			
x_{24}	150.0	241.583298	i_{24}	0.0	140.659195
x_{25}	150.0	159.336304	i_{25}	0.0	149.995499
x_{26}	150.0	149.999802	i_{26}	0.0	149.999802
x_{27}	150.0	0.0			
x_{30}	77.295837	14.3840	i_{30}	0.0	0.0
x_{31}	63.181229	58.588501	i_{31}	0.0	0.0
x_{32}	0.0	0.0	i_{32}	0.0	0.0
x_{33}	0.0	0.0	i_{33}	0.0	0.0
x_{34}	0.0	0.0	i_{34}	0.0	0.0
x_{35}	0.0	0.0	i_{35}	0.0	0.0
x_{36}	0.0	0.0	i_{36}	0.0	0.0
x_{37}	0.0	0.0			
x_{43}	150.0	203.314407	i_{43}	0.0	53.314407
x_{44}	82.413612	0.0	i_{44}	0.0	0.0
x_{45}	71.181404	76.374802	i_{45}	0.0	76.374802
x_{46}	37.478016	84.539703	i_{46}	0.0	41.638901
x_{47}	68.762367	108.361099			

APPENDIX H: Comparison of Regular Time and Overtime Labor Levels

Table H.1. Regular and Overtime Levels Obtained from Both Models During Test One.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	4160.0	4159.989746	o_{00}	3610.561523	4159.979492
w_{01}	4170.166504	4542.983887	o_{01}	4170.166504	4542.983887
w_{02}	4160.0	4159.995605	o_{02}	1382.543823	1774.982666
w_{10}	4160.0	4159.989746	o_{10}	3644.719238	4015.853027
w_{11}	4170.166504	4542.983887	o_{11}	4170.166504	3776.987061
w_{12}	4160.0	4159.995605	o_{12}	1406.228027	1663.246460
w_{20}	4160.0	4159.989746	o_{20}	3644.719238	3243.590332
w_{21}	4170.166504	4542.983887	o_{21}	4170.166504	3776.990967
w_{22}	4160.0	4159.995605	o_{22}	1406.228027	1127.779297
w_{30}	4160.0	4159.989746	o_{30}	2670.0	3742.421631
w_{31}	4170.166504	4542.983887	o_{31}	3507.833252	3776.990723
w_{32}	4160.0	4159.995605	o_{32}	410.0	1018.572449
w_{40}	4160.0	4159.989746	o_{40}	1665.0	4159.989746
w_{41}	4170.166504	4542.983887	o_{41}	1722.833374	3776.996094
w_{42}	3555.0	4159.995605	o_{42}	0.0	916.644409
w_{50}	4160.0	4159.989746	o_{50}	1665.0	3328.086914
w_{51}	4170.166504	4542.983887	o_{51}	1722.833374	3326.002197
w_{52}	3555.0	4159.995605	o_{52}	0.0	393.707397
w_{60}	4160.0	4159.989746	o_{60}	1665.0	1664.987183
w_{61}	4170.166504	4542.983887	o_{61}	1722.833374	1349.993408
w_{62}	3555.0	3554.986084	o_{62}	0.0	0.0
w_{70}	4160.0	0.0	o_{70}	1665.0	0.0
w_{71}	4170.166504	0.0	o_{71}	1722.833374	0.0
w_{72}	3555.0	0.0	o_{72}	0.0	0.0

Table H.2. Regular and Overtime Levels Obtained from Both Models During Test Two.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6021.666504	6354.984863	o_{00}	6021.666504	6354.984863
w_{01}	5895.166504	6362.983887	o_{01}	5895.166504	6362.983887
w_{02}	4987.407227	5135.548828	o_{02}	3570.926025	3899.429443
w_{10}	6021.666504	6354.984863	o_{10}	6021.666504	5354.984863
w_{11}	5895.166504	6362.983887	o_{11}	5895.166504	4959.483887
w_{12}	4987.407227	5135.548828	o_{12}	3570.926025	3184.429443
w_{20}	6021.666504	6354.984863	o_{20}	6021.666504	5354.984863
w_{21}	5895.166504	6362.983887	o_{21}	5895.166504	4959.483887
w_{22}	4987.407227	5135.548828	o_{22}	3570.926025	3184.429443
w_{30}	6021.666504	6354.984863	o_{30}	4058.333252	5251.896973
w_{31}	5895.166504	6362.983887	o_{31}	4532.833496	5356.985840
w_{32}	4987.407227	5135.548828	o_{32}	1582.592651	2374.069336
w_{40}	6021.666504	6354.984863	o_{40}	3053.333252	2000.584229
w_{41}	5895.166504	6362.983887	o_{41}	2747.833252	1956.994141
w_{42}	4987.407227	5099.850098	o_{42}	567.592590	0.0
w_{50}	6021.666504	6354.984863	o_{50}	3053.333252	2675.949951
w_{51}	5895.166504	6362.983887	o_{51}	2747.833252	1956.995850
w_{52}	4987.407227	5135.548828	o_{52}	567.592590	404.757294
w_{60}	6021.666504	6354.984863	o_{60}	3053.333252	2338.266846
w_{61}	5895.166504	6362.983887	o_{61}	2747.833252	1956.994751
w_{62}	4987.407227	5135.548828	o_{62}	567.592590	184.529251
w_{70}	6021.666504	6354.984863	o_{70}	3053.333252	2338.266846
w_{71}	5895.166504	6362.983887	o_{71}	2747.833252	1956.994751
w_{72}	4987.407227	5135.548828	o_{72}	567.592590	184.529251

Table H.3. Regular and Overtime Levels Obtained from Both Models During Test Three.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	7127.322754	14665.048828	o_{00}	7127.322754	1040.726562
w_{01}	6799.062012	6928.546387	o_{01}	6760.885254	6928.546387
w_{02}	5478.810059	8828.910156	o_{02}	4640.057129	2300.711670
w_{10}	7127.322754	11705.388672	o_{10}	7127.322754	0.0
w_{11}	6799.062012	6928.546387	o_{11}	6799.062012	4600.172852
w_{12}	5478.810059	8319.979492	o_{12}	4640.645508	0.0
w_{20}	7127.322754	11775.822266	o_{20}	7127.322754	0.0
w_{21}	6799.062012	6928.546387	o_{21}	6799.062012	2676.944824
w_{22}	5478.810059	8319.992188	o_{22}	4640.645508	0.0
w_{30}	7127.322754	13329.973633	o_{30}	6202.677246	0.0
w_{31}	6799.062012	6928.546387	o_{31}	6378.937988	6249.423828
w_{32}	5478.810059	8569.982422	o_{32}	3091.189941	0.0
w_{40}	7127.322754	13599.977539	o_{40}	5197.677246	0.0
w_{41}	6799.062012	6928.546387	o_{41}	4593.937988	5994.430664
w_{42}	5478.810059	8319.986328	o_{42}	2076.189941	0.0
w_{50}	7127.322754	13598.727539	o_{50}	5197.677246	0.0
w_{51}	6799.062012	6928.546387	o_{51}	4593.937988	5975.680664
w_{52}	5478.810059	8319.986328	o_{52}	2076.189941	0.0
w_{60}	7127.322754	13377.969727	o_{60}	5197.677246	0.0
w_{61}	6799.062012	6928.546387	o_{61}	4593.937988	4431.196777
w_{62}	5478.810059	8229.904297	o_{62}	2076.189941	0.0
w_{70}	7127.322754	8693.250977	o_{70}	5197.677246	0.0
w_{71}	6799.062012	6928.546387	o_{71}	4593.937988	1391.432129
w_{72}	5478.810059	5320.078125	o_{72}	2076.189941	0.0

Table H.4. Regular and Overtime Levels Obtained from Both Models During Test Four.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	5646.666504	5829.984863	o_{00}	5646.666504	5828.984375
w_{01}	6545.166504	6808.733398	o_{01}	6545.166504	6808.733398
w_{02}	4820.740723	4902.215332	o_{02}	3237.592529	3417.762695
w_{10}	5646.666504	5829.984863	o_{10}	5646.666504	5829.984863
w_{11}	6545.166504	6808.733398	o_{11}	6545.166504	6763.733887
w_{12}	4820.740723	4902.215332	o_{12}	3237.592529	3417.762695
w_{20}	5646.666504	5829.984863	o_{20}	5646.666504	5829.984863
w_{21}	6545.166504	6808.733398	o_{21}	6545.166504	6763.733887
w_{22}	4820.740723	4902.215332	o_{22}	3237.592529	3417.762695
w_{30}	5646.666504	5829.984863	o_{30}	4683.333496	3420.988525
w_{31}	6545.166504	6808.733398	o_{31}	5882.833496	4144.659668
w_{32}	4820.740723	4902.215332	o_{32}	2249.259277	1395.074463
w_{40}	5646.666504	5829.984863	o_{40}	2678.333252	2489.991699
w_{41}	6545.166504	6808.733398	o_{41}	2347.833252	2085.319824
w_{42}	4820.740723	4902.215332	o_{42}	234.259262	149.462540
w_{50}	5646.666504	5829.984863	o_{50}	2678.333252	2489.992676
w_{51}	6545.166504	6808.733398	o_{51}	2347.833252	2078.243652
w_{52}	4820.740723	4902.215332	o_{52}	234.259262	149.770767
w_{60}	5646.666504	5829.984863	o_{60}	2678.333252	2489.992676
w_{61}	6545.166504	6808.733398	o_{61}	2347.833252	2078.243652
w_{62}	4820.740723	4902.215332	o_{62}	234.259262	149.770767
w_{70}	5646.666504	5829.984863	o_{70}	2678.333252	2489.992676
w_{71}	6545.166504	6808.733398	o_{71}	2347.833252	2078.243652
w_{72}	4820.740723	4902.215332	o_{72}	234.259262	149.770767

Table H.5. Regular and Overtime Levels Obtained from Both Models During Test Five.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6870.552246	13403.813477	o_{00}	6870.552246	713.792664
w_{01}	7545.036621	8094.532227	o_{01}	7506.860352	8094.532227
w_{02}	5364.689941	8268.361328	o_{02}	4421.089844	1795.996582
w_{10}	6870.552246	11775.822266	o_{10}	6870.552246	0.0
w_{11}	7545.036621	8094.532227	o_{11}	7545.036621	1510.958984
w_{12}	5364.689941	8268.361328	o_{12}	4421.678223	51.630859
w_{20}	6870.552246	11643.388672	o_{20}	6690.400879	0.0
w_{21}	7545.036621	8094.532227	o_{21}	7545.036621	6224.18750
w_{22}	5364.689941	8268.361328	o_{22}	4296.765625	51.618652
w_{30}	6870.552246	12635.722656	o_{30}	6441.301758	0.0
w_{31}	7545.036621	8094.532227	o_{31}	7269.051270	5411.687988
w_{32}	5364.689941	8268.361328	o_{32}	3513.777344	51.620560
w_{40}	6870.552246	13403.813477	o_{40}	4704.447754	93.878670
w_{41}	7545.036621	8094.532227	o_{41}	4097.963379	6198.563965
w_{42}	5364.689941	8193.708008	o_{42}	1690.310059	0.0
w_{50}	6870.552246	13403.813477	o_{50}	4704.447754	235.721390
w_{51}	7545.036621	8094.532227	o_{51}	4097.963379	6041.404785
w_{52}	5364.689941	8268.361328	o_{52}	1690.310059	24.685667
w_{60}	6870.552246	9799.992188	o_{60}	4704.447754	0.0
w_{61}	7545.036621	8094.532227	o_{61}	4097.963379	237.691681
w_{62}	5364.689941	6028.071777	o_{62}	1690.310059	0.0
w_{70}	6870.552246	8319.978516	o_{70}	4704.447754	0.0
w_{71}	7545.036621	8094.532227	o_{71}	4097.963379	794.215454
w_{72}	5364.689941	5051.910156	o_{72}	1690.310059	0.0

Table H.6. Regular and Overtime Levels Obtained from Both Models During Test Six.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6895.913086	12574.830078	o_{00}	6895.913086	444.623962
w_{01}	7565.623535	8860.879883	o_{01}	7527.446777	8860.879883
w_{02}	5375.961426	7899.924316	o_{02}	4445.621582	1419.685547
w_{10}	6895.913086	11775.822266	o_{10}	6895.913086	0.0
w_{11}	7565.623535	8860.879883	o_{11}	7565.623535	744.611572
w_{12}	5375.961426	7899.924316	o_{12}	4446.209961	420.067841
w_{20}	6895.913086	12566.729492	o_{20}	6622.146484	0.0
w_{21}	7565.623535	8860.879883	o_{21}	7565.623535	8246.466797
w_{22}	5375.961426	7899.924316	o_{22}	4256.387207	1096.310669
w_{30}	6895.913086	12574.830078	o_{30}	6821.536621	949.531860
w_{31}	7565.623535	8860.879883	o_{31}	6138.201172	2622.734375
w_{32}	5375.961426	7899.924316	o_{32}	3470.788574	420.067535
w_{40}	6895.913086	12574.830078	o_{40}	5977.054688	540.300903
w_{41}	7565.623535	8860.879883	o_{41}	5175.656738	5854.535645
w_{42}	5375.961426	7899.924316	o_{42}	2477.788086	25.925526
w_{50}	6895.913086	12574.830078	o_{50}	6497.324707	1055.771973
w_{51}	7565.623535	8860.879883	o_{51}	5615.885254	4521.472656
w_{52}	5375.961426	7899.924316	o_{52}	2797.954102	420.061584
w_{60}	6895.913086	12574.830078	o_{60}	6469.803711	541.955811
w_{61}	7565.623535	8860.879883	o_{61}	5592.598145	2277.862305
w_{62}	5375.961426	7899.924316	o_{62}	2781.018066	169.251480
w_{70}	6895.913086	8319.978516	o_{70}	6054.468262	0.0
w_{71}	7565.623535	8860.879883	o_{71}	5241.160645	27.868031
w_{72}	5375.961426	5051.910156	o_{72}	2525.427002	0.0

Table H.7. Regular and Overtime Levels Obtained from Both Models During Test Seven.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6487.682617	12574.727539	o_{00}	6487.682617	444.726898
w_{01}	8194.567383	8861.000977	o_{01}	8194.567383	8860.758789
w_{02}	5194.525879	7899.878906	o_{02}	4080.914795	1419.731323
w_{10}	6487.682617	11681.166016	o_{10}	6487.682617	0.0
w_{11}	8194.567383	8861.000977	o_{11}	8194.567383	3757.644043
w_{12}	5194.525879	7899.878906	o_{12}	4080.914795	420.100159
w_{20}	6487.682617	11619.986328	o_{20}	6487.682617	0.0
w_{21}	8194.567383	8861.000977	o_{21}	8194.567383	6961.480469
w_{22}	5194.525879	7899.878906	o_{22}	4080.914795	420.111267
w_{30}	6487.682617	12529.973633	o_{30}	6435.131348	0.0
w_{31}	8194.567383	8861.000977	o_{31}	7752.251465	4891.969727
w_{32}	5194.525879	7899.878906	o_{32}	3727.484375	420.103333
w_{40}	6487.682617	12574.727539	o_{40}	4337.317383	52.162960
w_{41}	8194.567383	8861.000977	o_{41}	3698.432373	6320.267578
w_{42}	5194.525879	7588.134277	o_{42}	1360.474365	0.0
w_{50}	6487.682617	9473.31250	o_{50}	4337.317383	0.0
w_{51}	8194.567383	8861.000977	o_{51}	3698.432373	259.978516
w_{52}	5194.525879	5793.986816	o_{52}	1360.474365	0.0
w_{60}	6487.682617	8319.990234	o_{60}	4337.317383	0.0
w_{61}	8194.567383	8861.000977	o_{61}	3698.432373	1151.987793
w_{62}	5194.525879	5003.994141	o_{62}	1360.474365	0.0
w_{70}	6487.682617	8319.977539	o_{70}	4337.317383	0.0
w_{71}	8194.567383	8861.000977	o_{71}	3698.432373	25.976759
w_{72}	5194.525879	5051.986328	o_{72}	1360.474365	0.0

Table H.8. Regular and Overtime Levels Obtained from Both Models During Test Eight.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6994.903320	12574.727539	o_{00}	6994.903320	444.726898
w_{01}	8211.034180	8861.000977	o_{01}	6766.363281	8860.758789
w_{02}	5419.957031	7899.878906	o_{02}	4537.118652	1419.731323
w_{10}	6994.903320	11675.254883	o_{10}	6035.616211	0.0
w_{11}	8211.034180	8861.000977	o_{11}	8211.034180	4023.749023
w_{12}	5419.957031	7899.878906	o_{12}	3894.233398	420.101196
w_{20}	6994.903320	11619.986328	o_{20}	6035.616211	0.0
w_{21}	8211.034180	8861.000977	o_{21}	8211.034180	6961.480469
w_{22}	5419.957031	7899.878906	o_{22}	3894.233398	420.111267
w_{30}	6994.903320	12574.727539	o_{30}	6370.686035	347.121765
w_{31}	8211.034180	8861.000977	o_{31}	6675.979980	3977.617920
w_{32}	5419.957031	7899.878906	o_{32}	3496.892578	420.108215
w_{40}	6994.903320	12574.727539	o_{40}	6016.021973	52.162960
w_{41}	8211.034180	8861.000977	o_{41}	5531.594727	6320.267578
w_{42}	5419.957031	7588.134277	o_{42}	2480.227539	0.0
w_{50}	6994.903320	12574.727539	o_{50}	6016.021973	979.993652
w_{51}	8211.034180	8861.000977	o_{51}	5531.594727	5342.253418
w_{52}	5419.957031	7899.878906	o_{52}	2480.227539	334.928162
w_{60}	6994.903320	12574.727539	o_{60}	5998.037109	937.193298
w_{61}	8211.034180	8861.000977	o_{61}	5516.376465	4523.307129
w_{62}	5419.957031	7899.878906	o_{62}	2469.160156	342.621277
w_{70}	6994.903320	8319.979492	o_{70}	5584.657227	0.0
w_{71}	8211.034180	8319.979492	o_{71}	5166.593750	0.0
w_{72}	5419.957031	5076.640137	o_{72}	2214.772461	0.0

Table H.9. Regular and Overtime Levels Obtained from Both Models During Test Nine.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	7106.821289	12574.756836	o_{00}	7106.821289	444.697235
w_{01}	8215.545898	8860.975586	o_{01}	6450.519043	8860.784180
w_{02}	5469.698242	7899.892090	o_{02}	4637.782227	1419.718140
w_{10}	7106.821289	11668.567383	o_{10}	5934.812988	0.0
w_{11}	8215.545898	8860.975586	o_{11}	8215.545898	4325.013672
w_{12}	5469.698242	7899.892090	o_{12}	3852.337402	420.092926
w_{20}	7106.821289	11619.986328	o_{20}	5934.812988	0.0
w_{21}	8215.545898	8860.975586	o_{21}	8215.545898	6961.505859
w_{22}	5469.698242	7899.892090	o_{22}	3852.337402	420.098114
w_{30}	7106.821289	12574.756836	o_{30}	6670.027832	685.679504
w_{31}	8215.545898	8860.975586	o_{31}	5568.892090	2399.317383
w_{32}	5469.698242	7899.892090	o_{32}	3419.479492	257.684326
w_{40}	7106.821289	12574.756836	o_{40}	5914.372070	52.133312
w_{41}	8215.545898	8860.975586	o_{41}	5535.771973	6320.292969
w_{42}	5469.698242	7588.134277	o_{42}	2436.805176	0.0
w_{50}	7106.821289	12574.756836	o_{50}	5914.372070	772.022156
w_{51}	8215.545898	8860.975586	o_{51}	5535.771973	5584.553223
w_{52}	5469.698242	7899.892090	o_{52}	2436.805176	188.767075
w_{60}	7106.821289	12574.756836	o_{60}	5896.353027	960.421021
w_{61}	8215.545898	8860.975586	o_{61}	5520.525391	5325.249023
w_{62}	5469.698242	7899.892090	o_{62}	2425.716797	322.910004
w_{70}	7106.821289	9785.536133	o_{70}	5483.304688	0.0
w_{71}	8215.545898	8319.992188	o_{71}	5171.022461	0.0
w_{72}	5469.698242	6019.175781	o_{72}	2171.532959	0.0

Table H.10. Regular and Overtime Levels Obtained from Both Models During Test Ten.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	4160.0	4159.989746	o_{00}	3610.561523	4159.979492
w_{01}	4170.166504	4542.983887	o_{01}	4170.166504	4542.983887
w_{02}	4160.0	4159.995605	o_{02}	1382.543823	1774.982666
w_{10}	4160.0	4159.989746	o_{10}	3644.719238	4015.853027
w_{11}	4170.166504	4542.983887	o_{11}	4170.166504	3776.987061
w_{12}	4160.0	4159.995605	o_{12}	1406.228027	1663.246460
w_{20}	4160.0	4159.989746	o_{20}	3644.719238	3243.590332
w_{21}	4170.166504	4542.983887	o_{21}	4170.166504	3776.990967
w_{22}	4160.0	4159.995605	o_{22}	1406.228027	1127.779297
w_{30}	4160.0	4159.989746	o_{30}	2670.0	3742.421631
w_{31}	4170.166504	4542.983887	o_{31}	3507.833252	3776.990723
w_{32}	4160.0	4159.995605	o_{32}	410.0	1018.572449
w_{40}	4160.0	4159.989746	o_{40}	1665.0	4159.989746
w_{41}	4170.166504	4542.983887	o_{41}	1722.833374	3776.996094
w_{42}	3555.0	4159.995605	o_{42}	0.0	916.644409
w_{50}	4160.0	4159.989746	o_{50}	1665.0	3328.086914
w_{51}	4170.166504	4542.983887	o_{51}	1722.833374	3326.002197
w_{52}	3555.0	4159.995605	o_{52}	0.0	393.707397
w_{60}	4160.0	4159.989746	o_{60}	1665.0	1664.987183
w_{61}	4170.166504	4542.983887	o_{61}	1722.833374	1349.993408
w_{62}	3555.0	3554.986084	o_{62}	0.0	0.0
w_{70}	4160.0	0.0	o_{70}	1665.0	0.0
w_{71}	4170.166504	0.0	o_{71}	1722.833374	0.0
w_{72}	3555.0	0.0	o_{72}	0.0	0.0

Table H.11. Regular and Overtime Levels Obtained from Both Models During Test Eleven.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6021.666504	6354.984863	o_{00}	6021.666504	6354.984863
w_{01}	5895.166504	6362.983887	o_{01}	5895.166504	6362.983887
w_{02}	4987.407227	5135.548828	o_{02}	3570.926025	3899.429443
w_{10}	6021.666504	6354.984863	o_{10}	6021.666504	5354.984863
w_{11}	5895.166504	6362.983887	o_{11}	5895.166504	4959.483887
w_{12}	4987.407227	5135.548828	o_{12}	3570.926025	3184.429443
w_{20}	6021.666504	6354.984863	o_{20}	6021.666504	5354.984863
w_{21}	5895.166504	6362.983887	o_{21}	5895.166504	4959.483887
w_{22}	4987.407227	5135.548828	o_{22}	3570.926025	3184.429443
w_{30}	6021.666504	6354.984863	o_{30}	4058.333252	5251.896973
w_{31}	5895.166504	6362.983887	o_{31}	4532.833496	5356.985840
w_{32}	4987.407227	5135.548828	o_{32}	1582.592651	2374.069336
w_{40}	6021.666504	6354.984863	o_{40}	3053.333252	2000.584229
w_{41}	5895.166504	6362.983887	o_{41}	2747.833252	1956.994141
w_{42}	4987.407227	5099.850098	o_{42}	567.592590	0.0
w_{50}	6021.666504	6354.984863	o_{50}	3053.333252	2675.949951
w_{51}	5895.166504	6362.983887	o_{51}	2747.833252	1956.995850
w_{52}	4987.407227	5135.548828	o_{52}	567.592590	404.757294
w_{60}	6021.666504	6354.984863	o_{60}	3053.333252	2338.266846
w_{61}	5895.166504	6362.983887	o_{61}	2747.833252	1956.994751
w_{62}	4987.407227	5135.548828	o_{62}	567.592590	184.529251
w_{70}	6021.666504	6354.984863	o_{70}	3053.333252	2338.266846
w_{71}	5895.166504	6362.983887	o_{71}	2747.833252	1956.994751
w_{72}	4987.407227	5135.548828	o_{72}	567.592590	184.529251

Table H.12. Regular and Overtime Levels Obtained from Both Models During Test Twelve.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6523.325684	8660.0	o_{00}	6523.325684	6602.275879
w_{01}	6308.758789	7279.828613	o_{01}	6270.582031	5975.372559
w_{02}	5210.367188	6160.0	o_{02}	4055.798828	4652.836914
w_{10}	6523.325684	8660.0	o_{10}	6523.325684	3045.388184
w_{11}	6308.758789	7279.828613	o_{11}	6308.758789	4248.890625
w_{12}	5210.367188	6160.0	o_{12}	4056.387207	2159.979980
w_{20}	6523.325684	8660.0	o_{20}	6245.863281	3115.822510
w_{21}	6308.758789	7279.828613	o_{21}	6308.758789	2325.662354
w_{22}	5210.367188	6160.0	o_{22}	3864.001709	2159.992432
w_{30}	6523.325684	8660.0	o_{30}	6393.687012	4669.973145
w_{31}	6308.758789	7279.828613	o_{31}	6308.758789	5898.141602
w_{32}	5210.367188	6160.0	o_{32}	3064.642090	2409.982178
w_{40}	6523.325684	8660.0	o_{40}	5801.674316	4939.977051
w_{41}	6308.758789	7279.828613	o_{41}	5084.241211	5643.148438
w_{42}	5210.367188	6160.0	o_{42}	2344.632812	2159.986084
w_{50}	6523.325684	8660.0	o_{50}	5801.674316	4938.727051
w_{51}	6308.758789	7279.828613	o_{51}	5084.241211	5624.398438
w_{52}	5210.367188	6160.0	o_{52}	2344.632812	2159.986084
w_{60}	6523.325684	8660.0	o_{60}	5801.674316	4717.969238
w_{61}	6308.758789	7279.828613	o_{61}	5084.241211	4079.914551
w_{62}	5210.367188	6160.0	o_{62}	2344.632812	2069.904297
w_{70}	6523.325684	8660.0	o_{70}	5801.674316	33.251457
w_{71}	6308.758789	7279.828613	o_{71}	5084.241211	1040.149658
w_{72}	5210.367188	5320.078125	o_{72}	2344.632812	0.0

Table H.13. Regular and Overtime Levels Obtained from Both Models During Test Thirteen.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	5646.666504	5829.984863	o_{00}	5646.666504	5828.984375
w_{01}	6545.166504	6808.733398	o_{01}	6545.166504	6808.733398
w_{02}	4820.740723	4902.215332	o_{02}	3237.592529	3417.762695
w_{10}	5646.666504	5829.984863	o_{10}	5646.666504	5829.984863
w_{11}	6545.166504	6808.733398	o_{11}	6545.166504	6763.733887
w_{12}	4820.740723	4902.215332	o_{12}	3237.592529	3417.762695
w_{20}	5646.666504	5829.984863	o_{20}	5646.666504	5829.984863
w_{21}	6545.166504	6808.733398	o_{21}	6545.166504	6763.733887
w_{22}	4820.740723	4902.215332	o_{22}	3237.592529	3417.762695
w_{30}	5646.666504	5829.984863	o_{30}	4683.333496	3420.988525
w_{31}	6545.166504	6808.733398	o_{31}	5882.833496	4144.659668
w_{32}	4820.740723	4902.215332	o_{32}	2249.259277	1395.074463
w_{40}	5646.666504	5829.984863	o_{40}	2678.333252	2489.991699
w_{41}	6545.166504	6808.733398	o_{41}	2347.833252	2085.319824
w_{42}	4820.740723	4902.215332	o_{42}	234.259262	149.462540
w_{50}	5646.666504	5829.984863	o_{50}	2678.333252	2489.992676
w_{51}	6545.166504	6808.733398	o_{51}	2347.833252	2078.243652
w_{52}	4820.740723	4902.215332	o_{52}	234.259262	149.770767
w_{60}	5646.666504	5829.984863	o_{60}	2678.333252	2489.992676
w_{61}	6545.166504	6808.733398	o_{61}	2347.833252	2078.243652
w_{62}	4820.740723	4902.215332	o_{62}	234.259262	149.770767
w_{70}	5646.666504	5829.984863	o_{70}	2678.333252	2489.992676
w_{71}	6545.166504	6808.733398	o_{71}	2347.833252	2078.243652
w_{72}	4820.740723	4902.215332	o_{72}	234.259262	149.770767

Table H.14. Regular and Overtime Levels Obtained from Both Models During Test Fourteen.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6341.606934	8659.922852	o_{00}	6341.606934	4257.309082
w_{01}	7096.568848	7279.992676	o_{01}	7096.568848	7279.992676
w_{02}	5129.603027	6159.965820	o_{02}	3909.430176	3046.981934
w_{10}	6341.606934	8659.922852	o_{10}	6294.577637	3115.899170
w_{11}	7096.568848	7279.992676	o_{11}	7096.568848	2325.498535
w_{12}	5129.603027	6159.965820	o_{12}	3876.821533	2160.026367
w_{20}	6341.606934	8659.922852	o_{20}	5692.078613	2983.465332
w_{21}	7096.568848	7279.992676	o_{21}	7096.568848	7038.727051
w_{22}	5129.603027	6159.965820	o_{22}	3459.063965	2160.014160
w_{30}	6341.606934	8659.922852	o_{30}	6341.606934	3975.799805
w_{31}	7096.568848	7279.992676	o_{31}	6864.363770	6226.227539
w_{32}	5129.603027	6159.965820	o_{32}	3299.835205	2160.016113
w_{40}	6341.606934	8659.922852	o_{40}	5233.393066	4359.479492
w_{41}	7096.568848	7279.992676	o_{41}	4546.431152	6608.396973
w_{42}	5129.603027	6159.965820	o_{42}	1925.396973	1739.410156
w_{50}	6341.606934	8659.922852	o_{50}	5233.393066	4497.291992
w_{51}	7096.568848	7279.992676	o_{51}	4546.431152	6447.827637
w_{52}	5129.603027	6159.965820	o_{52}	1925.396973	1836.269165
w_{60}	6341.606934	8659.922852	o_{60}	5233.393066	1140.068481
w_{61}	7096.568848	7279.992676	o_{61}	4546.431152	1052.231201
w_{62}	5129.603027	6028.071777	o_{62}	1925.396973	0.0
w_{70}	6341.606934	8319.978516	o_{70}	5233.393066	0.0
w_{71}	7096.568848	7279.992676	o_{71}	4546.431152	1608.755005
w_{72}	5129.603027	5051.910156	o_{72}	1925.396973	0.0

Table H.15. Regular and Overtime Levels Obtained from Both Models During Test Fifteen.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6389.993652	8659.917969	o_{00}	6389.993652	4987.674805
w_{01}	7135.847656	7279.998535	o_{01}	7135.847656	7279.990234
w_{02}	5151.108398	6159.963379	o_{02}	3956.235840	3553.397705
w_{10}	6389.993652	8659.917969	o_{10}	6342.964844	3115.905029
w_{11}	7135.847656	7279.998535	o_{11}	7135.847656	2325.492676
w_{12}	5151.108398	6159.963379	o_{12}	3923.627197	2160.029053
w_{20}	6389.993652	8659.917969	o_{20}	5845.419434	2029.817139
w_{21}	7135.847656	7279.998535	o_{21}	7135.847656	7279.998535
w_{22}	5151.108398	6159.963379	o_{22}	3578.641846	1495.561401
w_{30}	6389.993652	8659.917969	o_{30}	6389.993652	4864.444824
w_{31}	7135.847656	7279.998535	o_{31}	5295.706543	4203.615723
w_{32}	5151.108398	6159.963379	o_{32}	3026.025635	2160.028809
w_{40}	6389.993652	8659.917969	o_{40}	6389.993652	3988.559082
w_{41}	7135.847656	7279.998535	o_{41}	5526.756836	7040.555176
w_{42}	5151.108398	6159.963379	o_{42}	2645.422363	1478.714722
w_{50}	6389.993652	8659.917969	o_{50}	6389.993652	4868.224609
w_{51}	7135.847656	7279.998535	o_{51}	5526.756836	6015.656738
w_{52}	5151.108398	6159.963379	o_{52}	2645.422363	2096.970459
w_{60}	6389.993652	8659.917969	o_{60}	4815.420410	4456.868652
w_{61}	7135.847656	7279.998535	o_{61}	4194.425781	3858.743408
w_{62}	5151.108398	6159.963379	o_{62}	1676.454346	1909.212769
w_{70}	6389.993652	6992.100098	o_{70}	6389.993652	0.0
w_{71}	7135.847656	7279.998535	o_{71}	5526.756836	485.159546
w_{72}	5151.108398	4234.753906	o_{72}	2645.422363	0.0

Table H.16. Regular and Overtime Levels Obtained from Both Models During Test Sixteen.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6660.0	6867.781250	o_{00}	5693.473633	4068.889893
w_{01}	7280.0	7279.999023	o_{01}	7280.0	7279.985352
w_{02}	5660.0	5363.458496	o_{02}	3156.052734	2470.219482
w_{10}	6660.0	6867.781250	o_{10}	5727.631348	4813.385254
w_{11}	7280.0	7279.999023	o_{11}	7280.0	5338.645508
w_{12}	5660.0	5363.458496	o_{12}	3179.736816	2956.520508
w_{20}	6660.0	6867.781250	o_{20}	5727.631348	3821.953369
w_{21}	7280.0	7279.999023	o_{21}	7280.0	7279.998047
w_{22}	5660.0	5363.458496	o_{22}	3179.736816	2292.066406
w_{30}	6660.0	6867.781250	o_{30}	5240.947266	5662.191895
w_{31}	7280.0	7279.999023	o_{31}	7280.0	6472.971191
w_{32}	5660.0	5363.458496	o_{32}	2532.105225	2956.523682
w_{40}	6660.0	6867.781250	o_{40}	4165.0	5241.383301
w_{41}	7280.0	7279.999023	o_{41}	4613.0	7279.999023
w_{42}	5660.0	5363.458496	o_{42}	895.0	1914.040527
w_{50}	6660.0	6867.781250	o_{50}	4165.0	2605.530762
w_{51}	7280.0	7279.999023	o_{51}	4613.0	1840.980103
w_{52}	5660.0	5363.458496	o_{52}	895.0	430.528656
w_{60}	6660.0	6867.781250	o_{60}	4165.0	1452.208862
w_{61}	7280.0	7279.999023	o_{61}	4613.0	2732.989258
w_{62}	5660.0	5003.994141	o_{62}	895.0	0.0
w_{70}	6660.0	6867.781250	o_{70}	4165.0	1452.195923
w_{71}	7280.0	7279.999023	o_{71}	4613.0	1606.978394
w_{72}	5660.0	5051.986328	o_{72}	895.0	0.0

Table H.17. Regular and Overtime Levels Obtained from Both Models During Test Seventeen.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6562.315918	8659.894531	o_{00}	6562.315918	2329.521484
w_{01}	7280.0	7279.999023	o_{01}	7280.0	7279.983398
w_{02}	5227.695801	6159.953125	o_{02}	4123.058105	1710.296509
w_{10}	6562.315918	8659.894531	o_{10}	6115.732422	3015.359619
w_{11}	7280.0	7279.999023	o_{11}	7280.0	5604.750488
w_{12}	5227.695801	6159.953125	o_{12}	3813.408447	2160.026611
w_{20}	6562.315918	8659.894531	o_{20}	4127.420898	2029.839600
w_{21}	7280.0	7279.999023	o_{21}	7280.0	7279.998047
w_{22}	5227.695801	6159.953125	o_{22}	2427.830322	1495.571411
w_{30}	6562.315918	8659.894531	o_{30}	6562.315918	4261.954102
w_{31}	7280.0	7279.999023	o_{31}	7280.0	5558.619629
w_{32}	5227.695801	6159.953125	o_{32}	3517.041016	2160.033691
w_{40}	6562.315918	8659.894531	o_{40}	6339.926270	3449.269775
w_{41}	7280.0	7279.999023	o_{41}	6370.666504	7279.999023
w_{42}	5227.695801	6159.953125	o_{42}	2605.606934	1117.545654
w_{50}	6562.315918	8659.894531	o_{50}	5463.520020	4428.164551
w_{51}	7280.0	7279.999023	o_{51}	5629.091797	6528.387695
w_{52}	5227.695801	6159.953125	o_{52}	2066.280029	1787.677490
w_{60}	6562.315918	8659.894531	o_{60}	3379.591553	4790.502930
w_{61}	7280.0	7279.999023	o_{61}	3865.767822	6052.250977
w_{62}	5227.695801	6159.953125	o_{62}	783.862427	2044.686646
w_{70}	6562.315918	7031.794922	o_{70}	5786.415527	0.0
w_{71}	7280.0	6811.755859	o_{71}	5902.311035	0.0
w_{72}	5227.695801	4302.094238	o_{72}	2264.984863	0.0

Table H.18. Regular and Overtime Levels Obtained from Both Models During Test Eighteen.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6820.924805	8659.930664	o_{00}	6820.924805	2511.126465
w_{01}	7280.0	7279.999023	o_{01}	7280.0	7279.992188
w_{02}	5342.633301	6159.969238	o_{02}	4366.746094	1836.225952
w_{10}	6820.924805	8659.930664	o_{10}	5296.915527	3008.636230
w_{11}	7280.0	7279.999023	o_{11}	7280.0	5905.989746
w_{12}	5342.633301	6159.969238	o_{12}	3310.037109	2160.015625
w_{20}	6820.924805	8659.930664	o_{20}	3868.812012	2029.803589
w_{21}	7280.0	7279.999023	o_{21}	7280.0	7279.998047
w_{22}	5342.633301	6159.969238	o_{22}	2312.893066	1495.555420
w_{30}	6820.924805	8659.930664	o_{30}	6820.924805	4600.505371
w_{31}	7280.0	7279.999023	o_{31}	6321.224609	3980.293457
w_{32}	5342.633301	6159.969238	o_{32}	3450.116455	1997.606934
w_{40}	6820.924805	8659.930664	o_{40}	6104.257324	3449.233887
w_{41}	7280.0	7279.999023	o_{41}	6390.077148	7279.999023
w_{42}	5342.633301	6159.969238	o_{42}	2504.786377	1117.529541
w_{50}	6820.924805	8659.930664	o_{50}	4982.797852	4346.246094
w_{51}	7280.0	7279.999023	o_{51}	5441.149902	6623.793457
w_{52}	5342.633301	6159.969238	o_{52}	1814.657715	1730.111816
w_{60}	6820.924805	8659.930664	o_{60}	3197.212646	4410.139648
w_{61}	7280.0	7279.999023	o_{61}	3930.270264	6512.672852
w_{62}	5342.633301	6159.969238	o_{62}	715.835999	1776.612793
w_{70}	6820.924805	7088.471680	o_{70}	5669.409668	0.0
w_{71}	7280.0	6037.860352	o_{71}	6022.129395	0.0
w_{72}	5342.633301	4359.443848	o_{72}	2237.187988	0.0

Table H.19. Regular and Overtime Levels Obtained from Both Models During Test Nineteen.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	4160.0	4160.0	o_{00}	3595.578857	4159.969238
w_{01}	4160.0	7280.0	o_{01}	4160.0	1805.967529
w_{02}	4160.0	4160.0	o_{02}	1371.842163	1774.978149
w_{10}	4160.0	4160.0	o_{10}	3629.736816	4015.843018
w_{11}	4160.0	7280.0	o_{11}	4160.0	1039.970947
w_{12}	4160.0	4160.0	o_{12}	1395.526367	1663.242065
w_{20}	4160.0	4160.0	o_{20}	3629.736816	3243.580322
w_{21}	4160.0	7280.0	o_{21}	4160.0	1039.974854
w_{22}	4160.0	4160.0	o_{22}	1395.526367	1127.774902
w_{30}	4160.0	4160.0	o_{30}	2670.0	3742.411621
w_{31}	4160.0	7280.0	o_{31}	3518.0	1039.974487
w_{32}	4160.0	4160.0	o_{32}	410.0	1018.567993
w_{40}	4160.0	4160.0	o_{40}	1665.0	4159.979980
w_{41}	4160.0	7280.0	o_{41}	1733.0	1039.979858
w_{42}	3555.0	4160.0	o_{42}	0.0	916.639954
w_{50}	4160.0	4160.0	o_{50}	1665.0	3328.076904
w_{51}	4160.0	7280.0	o_{51}	1733.0	588.985840
w_{52}	3555.0	4160.0	o_{52}	0.0	393.702942
w_{60}	4160.0	4160.0	o_{60}	1665.0	1664.977173
w_{61}	4160.0	5892.977051	o_{61}	1733.0	0.0
w_{62}	3555.0	3554.986084	o_{62}	0.0	0.0
w_{70}	4160.0	0.0	o_{70}	1665.0	0.0
w_{71}	4160.0	0.0	o_{71}	1733.0	0.0
w_{72}	3555.0	0.0	o_{72}	0.0	0.0

Table H.20. Regular and Overtime Levels Obtained from Both Models During Test Twenty.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6660.0	6660.0	o_{00}	5280.0	6049.969238
w_{01}	7280.0	7280.0	o_{01}	4401.0	5445.967285
w_{02}	5660.0	5660.0	o_{02}	2825.0	3374.978271
w_{10}	6660.0	6660.0	o_{10}	5435.0	5049.969238
w_{11}	7280.0	7280.0	o_{11}	4565.0	4042.467529
w_{12}	5660.0	5660.0	o_{12}	2935.0	2659.978271
w_{20}	6660.0	6660.0	o_{20}	5435.0	5049.969238
w_{21}	7280.0	7280.0	o_{21}	4565.0	4042.467529
w_{22}	5660.0	5660.0	o_{22}	2935.0	2659.978271
w_{30}	6660.0	6660.0	o_{30}	3420.0	4946.881836
w_{31}	7280.0	7280.0	o_{31}	3148.0	4439.969727
w_{32}	5660.0	5660.0	o_{32}	910.0	1849.618042
w_{40}	6660.0	6660.0	o_{40}	2415.0	1695.568848
w_{41}	7280.0	7280.0	o_{41}	1363.0	1039.977905
w_{42}	5555.0	5099.850098	o_{42}	0.0	0.0
w_{50}	6660.0	6660.0	o_{50}	2415.0	2370.934570
w_{51}	7280.0	7280.0	o_{51}	1363.0	1039.979492
w_{52}	5555.0	5540.306152	o_{52}	0.0	0.0
w_{60}	6660.0	6660.0	o_{60}	2415.0	2033.251465
w_{61}	7280.0	7280.0	o_{61}	1363.0	1039.978516
w_{62}	5555.0	5320.078125	o_{62}	0.0	0.0
w_{70}	6660.0	6660.0	o_{70}	2415.0	2033.251465
w_{71}	7280.0	7280.0	o_{71}	1363.0	1039.978516
w_{72}	5555.0	5320.078125	o_{72}	0.0	0.0

Table H.21. Regular and Overtime Levels Obtained from Both Models During Test Twenty One.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6660.0	8660.0	o_{00}	6660.0	6602.275879
w_{01}	7280.0	7280.0	o_{01}	5521.235352	5975.201172
w_{02}	5660.0	6160.0	o_{02}	3799.117676	4652.836914
w_{10}	6660.0	8660.0	o_{10}	6660.0	3045.388184
w_{11}	7280.0	7280.0	o_{11}	5559.411621	4248.719238
w_{12}	5660.0	6160.0	o_{12}	3799.705811	2159.979980
w_{20}	6660.0	8660.0	o_{20}	5151.109375	3115.822510
w_{21}	7280.0	7280.0	o_{21}	4340.001465	2325.491211
w_{22}	5660.0	6160.0	o_{22}	2734.690674	2159.992432
w_{30}	6660.0	8660.0	o_{30}	6660.0	4669.973145
w_{31}	7280.0	7280.0	o_{31}	5884.428711	5897.970215
w_{32}	5660.0	6160.0	o_{32}	2902.857178	2409.982178
w_{40}	6660.0	8660.0	o_{40}	5665.0	4939.977051
w_{41}	7280.0	7280.0	o_{41}	4113.0	5642.977051
w_{42}	5660.0	6160.0	o_{42}	1895.0	2159.986084
w_{50}	6660.0	8660.0	o_{50}	5665.0	4938.727051
w_{51}	7280.0	7280.0	o_{51}	4113.0	5624.227051
w_{52}	5660.0	6160.0	o_{52}	1895.0	2159.986084
w_{60}	6660.0	8660.0	o_{60}	5665.0	4717.969238
w_{61}	7280.0	7280.0	o_{61}	4113.0	4079.743164
w_{62}	5660.0	6160.0	o_{62}	1895.0	2069.904297
w_{70}	6660.0	8660.0	o_{70}	5665.0	33.251457
w_{71}	7280.0	7280.0	o_{71}	4113.0	1039.978516
w_{72}	5660.0	5320.078125	o_{72}	1895.0	0.0

Table H.22. Regular and Overtime Levels Obtained from Both Models During Test Twenty Two.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6660.0	6660.0	o_{00}	4530.0	4998.969238
w_{01}	7280.0	7280.0	o_{01}	5701.0	6337.467285
w_{02}	5660.0	5660.0	o_{02}	2325.0	2659.978027
w_{10}	6660.0	6660.0	o_{10}	4685.0	4999.969238
w_{11}	7280.0	7280.0	o_{11}	5865.0	6292.467285
w_{12}	5660.0	5660.0	o_{12}	2435.0	2659.978271
w_{20}	6660.0	6660.0	o_{20}	4685.0	4999.969238
w_{21}	7280.0	7280.0	o_{21}	5865.0	6292.467285
w_{22}	5660.0	5660.0	o_{22}	2435.0	2659.978271
w_{30}	6660.0	6660.0	o_{30}	3670.0	2590.973145
w_{31}	7280.0	7280.0	o_{31}	5148.0	3673.393066
w_{32}	5660.0	5660.0	o_{32}	1410.0	637.289795
w_{40}	6660.0	6660.0	o_{40}	1665.0	1659.976440
w_{41}	7280.0	7280.0	o_{41}	1613.0	1614.053345
w_{42}	5055.0	5051.677734	o_{42}	0.0	0.0
w_{50}	6660.0	6660.0	o_{50}	1665.0	1659.977295
w_{51}	7280.0	7280.0	o_{51}	1613.0	1606.977295
w_{52}	5055.0	5051.986328	o_{52}	0.0	0.0
w_{60}	6660.0	6660.0	o_{60}	1665.0	1659.977295
w_{61}	7280.0	7280.0	o_{61}	1613.0	1606.977295
w_{62}	5055.0	5051.986328	o_{62}	0.0	0.0
w_{70}	6660.0	6660.0	o_{70}	1665.0	1659.977295
w_{71}	7280.0	7280.0	o_{71}	1613.0	1606.977295
w_{72}	5055.0	5051.986328	o_{72}	0.0	0.0

Table H.23. Regular and Overtime Levels Obtained from Both Models During Test Twenty Three.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6660.0	8660.0	o_{00}	6660.0	4257.231934
w_{01}	7280.0	7280.0	o_{01}	7280.0	7279.985352
w_{02}	5660.0	6160.0	o_{02}	3826.217285	3046.947998
w_{10}	6660.0	8660.0	o_{10}	6428.115723	3115.822510
w_{11}	7280.0	7280.0	o_{11}	7280.0	2325.491211
w_{12}	5660.0	6160.0	o_{12}	3665.434814	2159.992432
w_{20}	6660.0	8660.0	o_{20}	3867.375732	2983.388672
w_{21}	7280.0	7280.0	o_{21}	5449.629395	7038.719727
w_{22}	5660.0	6160.0	o_{22}	1861.680054	2159.980225
w_{30}	6660.0	8660.0	o_{30}	6464.631348	3975.723145
w_{31}	7280.0	7280.0	o_{31}	7280.0	6226.220215
w_{32}	5660.0	6160.0	o_{32}	3084.736816	2159.982178
w_{40}	6660.0	8660.0	o_{40}	4915.0	4359.402832
w_{41}	7280.0	7280.0	o_{41}	4363.0	6608.389648
w_{42}	5660.0	6160.0	o_{42}	1395.0	1739.376099
w_{50}	6660.0	8660.0	o_{50}	4915.0	4497.214844
w_{51}	7280.0	7280.0	o_{51}	4363.0	6447.820312
w_{52}	5660.0	6160.0	o_{52}	1395.0	1836.235107
w_{60}	6660.0	8660.0	o_{60}	4915.0	1139.991821
w_{61}	7280.0	7280.0	o_{61}	4363.0	1052.223877
w_{62}	5660.0	6028.071777	o_{62}	1395.0	0.0
w_{70}	6660.0	8319.978516	o_{70}	4915.0	0.0
w_{71}	7280.0	7280.0	o_{71}	4363.0	1608.747681
w_{72}	5660.0	5051.910156	o_{72}	1395.0	0.0

Table H.24. Regular and Overtime Levels Obtained from Both Models During Test Twenty Four.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6660.0	8660.0	o_{00}	6660.0	4987.591797
w_{01}	7280.0	7280.0	o_{01}	7280.0	7279.988770
w_{02}	5660.0	6160.0	o_{02}	3826.217285	3553.360840
w_{10}	6660.0	8660.0	o_{10}	6428.115723	3115.822510
w_{11}	7280.0	7280.0	o_{11}	7280.0	2325.491211
w_{12}	5660.0	6160.0	o_{12}	3665.434814	2159.992432
w_{20}	6660.0	8660.0	o_{20}	4096.523438	2029.734619
w_{21}	7280.0	7280.0	o_{21}	6289.292480	7279.997070
w_{22}	5660.0	6160.0	o_{22}	2033.502930	1495.524780
w_{30}	6660.0	8660.0	o_{30}	6660.000000	4864.362305
w_{31}	7280.0	7280.0	o_{31}	5884.428711	4203.614258
w_{32}	5660.0	6160.0	o_{32}	2902.857178	2159.992188
w_{40}	6660.0	8660.0	o_{40}	6660.0	3988.476562
w_{41}	7280.0	7280.0	o_{41}	5839.538574	7040.553711
w_{42}	5660.0	6160.0	o_{42}	2468.846191	1478.677979
w_{50}	6660.0	8660.0	o_{50}	6603.569336	4868.142090
w_{51}	7280.0	7280.0	o_{51}	5791.789551	6015.655273
w_{52}	5660.0	6160.0	o_{52}	2434.119629	2096.933838
w_{60}	6660.0	8660.0	o_{60}	3418.549805	4456.786133
w_{61}	7280.0	7280.0	o_{61}	3096.772949	3858.741943
w_{62}	5660.0	6160.0	o_{62}	474.107574	1909.176025
w_{70}	6660.0	6992.100098	o_{70}	6268.125977	0.0
w_{71}	7280.0	7280.0	o_{71}	5507.952637	485.158081
w_{72}	5660.0	4234.753906	o_{72}	2227.692871	0.0

Table H.25. Regular and Overtime Levels Obtained from Both Models During Test Twenty Five.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6660.0	6660.0	o_{00}	5693.473633	4276.671387
w_{01}	7280.0	7280.0	o_{01}	7280.0	7279.984375
w_{02}	5660.0	5660.0	o_{02}	3156.052734	2173.677734
w_{10}	6660.0	6660.0	o_{10}	5727.631348	5021.166504
w_{11}	7280.0	7280.0	o_{11}	7280.0	5338.644531
w_{12}	5660.0	5660.0	o_{12}	3179.736816	2659.979004
w_{20}	6660.0	6660.0	o_{20}	5727.631348	4029.734619
w_{21}	7280.0	7280.0	o_{21}	7280.0	7279.997070
w_{22}	5660.0	5660.0	o_{22}	3179.736816	1995.524780
w_{30}	6660.0	6660.0	o_{30}	5240.947266	5869.973145
w_{31}	7280.0	7280.0	o_{31}	7280.0	6472.970215
w_{32}	5660.0	5660.0	o_{32}	2532.105225	2659.982178
w_{40}	6660.0	6660.0	o_{40}	4165.0	5449.165039
w_{41}	7280.0	7280.0	o_{41}	4613.0	7279.998047
w_{42}	5660.0	5660.0	o_{42}	895.0	1617.498901
w_{50}	6660.0	6660.0	o_{50}	4165.0	2813.312012
w_{51}	7280.0	7280.0	o_{51}	4613.0	1840.979004
w_{52}	5660.0	5660.0	o_{52}	895.0	133.987045
w_{60}	6660.0	6660.0	o_{60}	4165.0	1659.990234
w_{61}	7280.0	7280.0	o_{61}	4613.0	2732.988281
w_{62}	5660.0	5003.994141	o_{62}	895.0	0.0
w_{70}	6660.0	6660.0	o_{70}	4165.0	1659.977295
w_{71}	7280.0	7280.0	o_{71}	4613.0	1606.977295
w_{72}	5660.0	5051.986328	o_{72}	895.0	0.0

Table H.26. Regular and Overtime Levels Obtained from Both Models During Test Twenty Six.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6660.0	8660.0	o_{00}	6660.0	2329.416260
w_{01}	7280.0	7280.0	o_{01}	7280.0	7279.982422
w_{02}	5660.0	6160.0	o_{02}	3826.217285	1710.249878
w_{10}	6660.0	8660.0	o_{10}	5829.426758	3015.254395
w_{11}	7280.0	7280.0	o_{11}	7280.0	5604.749512
w_{12}	5660.0	6160.0	o_{12}	3250.319092	2159.979980
w_{20}	6660.0	8660.0	o_{20}	4029.736816	2029.734619
w_{21}	7280.0	7280.0	o_{21}	7280.0	7279.997070
w_{22}	5660.0	6160.0	o_{22}	1995.526367	1495.524780
w_{30}	6660.0	8660.0	o_{30}	6464.631348	4261.849121
w_{31}	7280.0	7280.0	o_{31}	7280.0	5558.618652
w_{32}	5660.0	6160.0	o_{32}	3084.736816	2159.987061
w_{40}	6660.0	8660.0	o_{40}	6271.884766	3449.164795
w_{41}	7280.0	7280.0	o_{41}	6395.748535	7279.998047
w_{42}	5660.0	6160.0	o_{42}	2191.544434	1117.498901
w_{50}	6660.0	8660.0	o_{50}	5334.423828	4428.059570
w_{51}	7280.0	7280.0	o_{51}	5602.512695	6528.386719
w_{52}	5660.0	6160.0	o_{52}	1614.645508	1787.630737
w_{60}	6660.0	8660.0	o_{60}	3282.012939	4790.397949
w_{61}	7280.0	7280.0	o_{61}	3865.857178	6052.250
w_{62}	5660.0	6160.0	o_{62}	351.623352	2044.640015
w_{70}	6660.0	7031.794922	o_{70}	5689.862793	0.0
w_{71}	7280.0	6811.755859	o_{71}	5903.268555	0.0
w_{72}	5660.0	4302.094238	o_{72}	1833.377197	0.0

Table H.27. Regular and Overtime Levels Obtained from Both Models During Test Twenty Seven.

Regular Time Labor Variable	Regular Time Hours: Concurrent Model	Regular Time Hours: Sequential Model	Overtime Labor Variable	Overtime Hours: Concurrent Model	Overtime Hours: Sequential Model
w_{00}	6660.0	8660.0	o_{00}	6660.0	2511.057617
w_{01}	7280.0	7280.0	o_{01}	7280.0	7279.991211
w_{02}	5660.0	6160.0	o_{02}	3826.217285	1836.195312
w_{10}	6660.0	8660.0	o_{10}	6177.874512	3008.567383
w_{11}	7280.0	7280.0	o_{11}	7280.0	5905.988770
w_{12}	5660.0	6160.0	o_{12}	3491.923828	2159.984863
w_{20}	6660.0	8660.0	o_{20}	4029.736816	2029.734619
w_{21}	7280.0	7280.0	o_{21}	7280.0	7279.997070
w_{22}	5660.0	6160.0	o_{22}	1995.526367	1495.524780
w_{30}	6660.0	8660.0	o_{30}	6660.0	4600.436523
w_{31}	7280.0	7280.0	o_{31}	5884.428711	3980.292480
w_{32}	5660.0	6160.0	o_{32}	2902.857178	1997.576294
w_{40}	6660.0	8660.0	o_{40}	6271.884766	3449.164795
w_{41}	7280.0	7280.0	o_{41}	6395.748535	7279.998047
w_{42}	5660.0	6160.0	o_{42}	2191.544434	1117.498901
w_{50}	6660.0	8660.0	o_{50}	5541.791504	4346.177246
w_{51}	7280.0	7280.0	o_{51}	5777.977539	6623.792480
w_{52}	5660.0	6160.0	o_{52}	1742.256226	1730.081177
w_{60}	6660.0	8660.0	o_{60}	3351.071045	4410.070801
w_{61}	7280.0	7280.0	o_{61}	3924.291016	6512.671875
w_{62}	5660.0	6160.0	o_{62}	394.120636	1776.582153
w_{70}	6660.0	7088.471680	o_{70}	5384.554199	0.0
w_{71}	7280.0	6037.860352	o_{71}	5644.930176	0.0
w_{72}	5660.0	4359.443848	o_{72}	1645.494751	0.0

VITA

William Kenneth Hoehn was born on July 24, 1962 in Huntington, New York. He attended Walt Whitman High School in Bethesda Maryland, and graduated Summa Cum Laude from SUNY College of Technology in 1988. Following graduation from SUNY, Mr. Hoehn worked as a manufacturing Engineer for the Chicago Pneumatic Tool Company in Utica, New York.

In the Fall of 1989, Mr. Hoehn was accepted and began graduate study at the Virginia Polytechnic Institute and State University. He earned a Master's degree in Industrial and Systems Engineering in June of 1992.

While at Virginia Tech, Mr. Hoehn worked in the capacity of Graduate Teaching Assistant for two semesters. Further, for two years, Mr. Hoehn was funded by NASA, and worked as a Graduate Research Assistant. As a Graduate Project Assistant funded by BNR/Northern Telecom, Mr. Hoehn coordinated the work of several graduate and undergraduate students.