R & D Project Selection and Scheduling

by

Mark Anthony Coffin

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

Management Science

APPROVED:

[illegible]

Bernard W. Taylor, III, Chairman

[illegible]

Joanna R. Baker

[illegible]

Edward R. Clayton

Terry R. Rakes

Terry R. Rakes

Loren P. Rees

Loren P. Rees

August 28, 1992

Blacksburg, Virginia

c.2

LD
5655
V856
1992
C635
c.2

R & D Project Selection and Scheduling

by

Mark Anthony Coffin

Bernard W. Taylor, III, Chairman

Management Science

(ABSTRACT)

Research and Development (R&D) project selection has been the focus of much research attention and the literature on this subject is extensive. Numerous models have been proposed to facilitate the selection of R&D projects, and these models have been both qualitative and quantitative in nature. Various project goals are typically included in the selection process. For goals that can be developed as mathematical functions, models have been developed that determine optimal solutions efficiently. However, when project goals cannot be structured as linear or nonlinear functions, or to do so would make the solution of the model complex or even impossible, the models presented in the literature either make simplifying assumptions about these goals, or do not include them in the model.

One goal that cannot be structured in an optimization format, is scheduling of projects through the R&D process. The nature of scheduling is such that it is difficult to include in any of the existing quantitative selection models, and the result is that scheduling goals are either considered after the projects have been selected, or they are not included at all.

In this research an R&D selection model is developed that incorporates the scheduling of individual projects as one of several goals. The model is designed to reflect a realistic R&D situation, in terms of problem size, goals, objectives, and, constraints. In order to incorporate a scheduling goal, the model uses a heuristic approach to determine a solution. The validity of the solutions, as well as the computational complexity of the model are explored.

Acknowledgements

I would like to take this opportunity to thank the many individuals who helped make the completion of this dissertation possible. Without their guidance, assistance and support I would not have been able to completed this work.

First, I would like to thank my chairman, Bernard W. Taylor, III, for encouraging me to pursue a doctoral degree. His guidance and support has been instrumental in the completion of this research, and I am grateful for the opportunity to work with him.

I would like to thank the other members of my committee, Joanna R. Baker, Edward R. Clayton, Terry R. Rakes, and Loren P. Rees for their guidance, support, and friendship. Their inspiration and enthusiasm for management science has been a great influence on me, and I am indebted to them as I begin my career.

I would like to thank my many friends in the Ph.D. program for their support through good times and bad during my odyssey through graduate school. To Barry Wray, Ina Samanta, Ingrid Crouch, Gerald Kohers, Wendy Ceccuci, Jay Teets, Fern

Siochi, Rene Panis, Mike Slade, Mary-Margaret Koball, Lars Wiegmann, and Melanie Hatch, thank you.

I would like to thank Tracy McCoy, Teena Long and Sylvia Seavey for their assistance and expertise. Without them, this dissertation would have never made it to print!

Lastly, I would like to thank my wife, Jill, for her love and understanding during this most important, and most difficult, step in my career. Without her guidance and support for the last five and a half years, this dissertation would not have been possible.

Table of Contents

List of Illustrations

List of Tables

Chapter 1: Introduction

The selection of research and development (R&D) projects is a problem that has garnered considerable attention from researchers for several decades. The literature on R&D selection models is extensive; however much of the research on the project selection process focuses on R&D budgeting and management (Liberatore and Titus 1983). Although numerous models have been presented in the literature to help companies make project selection decisions (Baker and Pound 1964, Baker and Freeland 1974, Baker 1974), researchers have found that in practice very few of these models have gained acceptance (Liberatore and Titus 1983, Danila 1989). Practitioners favor evaluation methods that consider projects one at a time, instead of considering the projects as a group and selecting a portfolio that optimizes an objective or set of objectives relevant to the R&D process.

In the late 1970's and early 1980's a number of new techniques were developed that considered the multiple objectives inherent in the R&D project selection process. Multi-criteria modelling techniques such as goal programming allowed researchers to address project selection from a more realistic perspective. However, one aspect of the project selection process that still has not been addressed by these and other models is project scheduling. Practitioners generally select projects based on all relevant goals except scheduling, and then schedule a project set according to general scheduling models, such as PERT/CPM.

Importance of scheduling in R&D project selection

Scheduling would seem to be an important part of the R&D project selection process that should be included in any project selection model. Scheduling is a realistic constraint on management's decision-making process when selecting R&D projects. The time available to complete a set of projects is normally limited, R&D project completion time is usually subject to important due dates, and research personnel, resources and/or facilities can overlap if projects are not scheduled efficiently.

If a time constraint is placed on the R&D department to start and complete a set of projects within a specified time horizon, management should realistically select projects that meet not only economic goals, but also the scheduling goals. However, a situation might exist where, for example, projects, A, B, and C, would represent the best selection in terms of the economic goals and resource constraints for the R&D selection process, but it would not be possible to complete all three projects within a specified time horizon. This time horizon could be imposed internally by the R&D department, externally by market forces or by other organizational departments, or most likely, by senior management, possibly through the strategic planning process.

Traditionally, this problem has been addressed by first selecting a set of projects that meet economic goals and resource constraints, and then subsequently attempting to schedule this set of projects using scheduling rules or procedures. When a conflict results and it is not possible to schedule the projects within the desired or required time frame projects may be deleted, alternative projects or sets of projects may be considered, resources may be increased, economic goals reduced, or the desired schedule may be relaxed. None of these alternatives represent a logical or structured solution approach that is likely to provide the best, or even a good selection of R&D projects.

R&D projects frequently must be sequenced through a common set of facilities and/or equipment. These facilities may not be able to handle all the selected R&D projects at the same time. There could be many different sequences of projects through the facilities, however if due dates or time requirements are rigid and binding, the particular sequence selected may not be feasible. In fact, a satisfactory sequence for the projects selected by a traditional project selection model may not even exist if scheduling constraints are imposed. Facilities should be treated as a dynamic resource constraint that is a function of the demands of a project and the time during which a project (or projects) makes use of the facilities. Alternatively, traditional selection models, if they include a facilities constraint at all, consider it only for a fixed point in time, or consider facilities scheduling after the scheduling process.

A conflict between scheduling and R&D personnel can also occur. In this case, the research personnel might be assigned to an R&D project, and would not be available to work on any other R&D projects until the R&D project is completed, or their task is completed. This overlap would affect the completion dates and/or the due dates of those R&D project(s) requiring the research personnel. Scarce or shared resources would affect the R&D process in a similar manner.

Thus, scheduling would seem to be an important goal in the R&D project selection process, and project selection models should logically reflect this importance. However, when investigated closer, the reason for the exclusion of scheduling as part of R&D project selection models presented in the literature is clear.

Difficulties presented by project scheduling

R&D project scheduling belongs to a larger family of scheduling and sequencing problems identified in the literature as being "hard" problems to solve. These scheduling problems are classified as NP-Complete problems. NP-Complete problems are problems for which solution algorithms take an extraordinarily long time to complete. Specifically, if the computational complexity of the "fastest known" algorithm increases non-polynomially as the problem size increases, then the problem is considered NP-Complete. Polynomial expressions are those expressions that can be expressed in the form x^n, where n is the problem size, and any other form is considered non-polynomial. A more complete guide to NP-Complete problems is given in Garey and Johnson (1979).

The implication of this difficulty for scheduling problems is straightforward: the problem can be solved optimally or near optimally for only the smallest problems. As the problem size increases, the time required to solve the solution algorithm becomes too long for even the fastest computers available today. The result is that all practical scheduling problems are solved using heuristic algorithms. Unfortunately, the most effective and robust quantitative R&D project selection models, especially multi-criteria models, cannot incorporate a scheduling heuristic within their framework. These models require exact mathematical relationships between the model objectives and constraints in the model formulation. Put directly, the scheduling heuristics are incompatible with methods such as linear, nonlinear, or integer programming.

The elegance of exact methods for project selection, and the comparative simplicity of solutions obtained from these methods, makes them popular among researchers. Because of the difficulty of including scheduling goals in the multi-criteria models developed for R&D project selection problems, scheduling has been relegated to being performed after the selection process. However, scheduling is a realistic and important consideration for R&D managers, and its inclusion in the selection process would logically result in a better, more realistic solution.

The objectives of this dissertation are to: (1) develop a modelling approach that will allow the addition of project scheduling as a goal within a multi-criteria R&D project selection process, (2) determine how close to optimal the model solutions are, and (3) determine if the model can solve realistic problems, in terms of project selection goals, constraints and problem size.

Statement of the problem

The current portfolio of quantitative R&D project selection models is extensive, and includes mathematical programming models, economic models, index models, decision theory models, risk analysis models, frontier models, and scoring models. These models attempt to address the various objectives and constraints that are relevant to the R&D project selection problem. Some of the quantitative models consider one objective at a time, such as expected profit of the selected portfolio of R&D projects, while others are multi-criteria models. These models address multiple goals, such as profit maximization, maximizing the probabilities of success of selected projects, and scheduling goals such as minimizing lateness of selected projects, and meeting target project due dates. Some of these goals, such as profit maximization and probability of individual project success, lend themselves to more exact methods such as math programming, while other goals, such as the scheduling goals, are more difficult to include in exact solution methods. The primary objective of this research is to develop a modelling approach that can incorporate project scheduling as a primary goal in a multi-criteria R&D project selection process.

Scheduling is virtually impossible to include in strict optimization models that have previously been used for R&D project selection and scheduling, such as math programming models. Math programming models have been presented in the literature to solve resource constrained project scheduling problems (Pritsker et al. 1969), but do not consider project selection. However, these math programming models for project scheduling can only handle very small problem sizes, because of the NP-Complete nature of scheduling problems. As a result, exact methods for R&D project scheduling are usually very complex, and after an extensive literature review, no exact mathematical models were found that included both project scheduling and project selection as goals. Some mathematical models have included a scheduling goal by simplifying project scheduling as project completion time, and considering the projects independently (Taylor et al. 1982). If the project completion times are truly independent of each other, and can be easily estimated then this model will work well. However, it is more realistic to consider the scheduling of R&D projects as a inter-dependent process. The selected projects will share R&D facilities, personnel and other resources, and the timing of the use of these resources will affect the individual project completion times. An R&D project selection model that includes scheduling as a goal should reflect this project inter-dependence to provide a more realistic solution.

This dissertation will address this issue of including project scheduling as a primary goal of the R&D project selection process by developing a modelling approach that can accommodate the NP-Complete nature of project scheduling. This model will also be able to handle realistic R&D project selection problems, particularly in terms of problem size.

Purpose and justification

The purpose of this research is to develop a modelling framework that can be used in the selection and scheduling of R&D projects, with the scheduling of the projects as a primary objective, or goal. Project scheduling has not been extensively explored in the research literature on R&D project selection to date, primarily due to the limitations inherent in trying to use exact methods to solve R&D project scheduling problems.

This research has the following objectives: (1) to investigate different methods for solving R&D project selection problems to determine if it is possible to include project scheduling as a goal; (2) to develop a modelling approach that can accommodate the scheduling requirements of the problem, and, is efficient in terms of quality of the solution obtained, and, in the computational time required to arrive at the solution; and, (3) to demonstrate useful and practical implementation of such a modelling approach.

The investigation of different models that have been used previously to solve R&D project selection problems will require an extensive literature search. These methodologies have strengths and limitations regarding the various input and output requirements of the R&D selection problem. Models that include scheduling as part of the selection process are virtually nonexistent in the literature.

Background and significance

An R&D project is a potential product, process or service that has the potential for providing benefits to a company, or organization. Depending on the organization, the benefits from R&D projects can take different forms. Business R&D would primarily focus on increasing future cash flows or profits to the company. However, R&D can also be performed for military organizations, medical organizations, natural and physical sciences, etc. In these cases, the benefits are not necessarily future profits for the organizations, but instead, advances in technology and knowledge. For example, research may be performed to find a cure for a rare disease, in which case the objective is to find the cure as soon as possible, rather than trying to make the cure profitable for the organization. Thus, objectives for business R&D are often different than for other types of organizations, and as a result, many different techniques and models exist for R&D project selection. This research will emphasize business

R&D, and the focus will be on business R&D project selection, subject to goals and constraints appropriate for business organizations (i.e. economic goals and constraints).

Business R&D projects can be separated into several categories: process improvement R&D, product improvement R&D, new process R&D, new product R&D, and new process/new product R&D.

Process improvement R&D focuses on improving existing manufacturing or service processes. The process improvement can be the result of new technologies, materials or equipment, and the goal is to improve the efficiency or profitability of the operations.

Product improvement R&D relates to updating a product's specifications or performance, or perhaps extending a product line, but not enough to consider the resulting product as a new product. The improvements are intended to increase sales, increase market share, and/or increase profits from the product or product line.

New process R&D is aimed at providing an entirely new method for manufacturing a product or service. The new process may be the result of new technologies, materials, equipment, or manufacturing ideas. Like process improvement, the goal of new process R&D is to improve the efficiency or profitability of the manufacturing process.

New product R&D is aimed at generating new products for market. These new product ideas are designed to be produced with existing or conventional manufacturing processes. This type of R&D is inherently risky because the likelihood of commercial success is more difficult to assess than technical success. Also, the profitability of the manufacturing process is more difficult to determine for new products, making it difficult to accurately estimate profits for the new products.

A combination of new process / new product R&D is aimed at producing a new process or processes to produce a new product. This type of R&D is rare, as the risks to the company are greater. In order for the R&D project to be successful, both the new process and the new product must be successful.

An R&D project typically has a budget requirement, which is the maximum amount of cash that the company expects to, or is willing to spend on the project. An R&D project also has an anticipated future cash flow to the company, and this is called the return. The likelihood of the R&D project of being successful is called the probability of success. An R&D project can consume other resources besides cash, such as manpower, materials, time, equipment and computer resources. The limit on the availability of each resource is called a constraint. A company will typically have multiple R&D projects to consider, and usually will not be able to undertake all projects, because of the constraints. The problem becomes one of how to select the best set of R&D projects that will ultimately benefit the company, or organization the most. The selection process should meet the needs of the company, in terms of future benefits, while meeting the system constraints that exist because of limited resources.

The R&D department has the responsibility to develop models for project selection that achieve the company's goals as established by top management. The goals that the R&D department uses for the project selection process and models are designed to be consistent with the established goals and objectives as outlined by top management. The actual goals used depend on a variety of inputs from other levels of management and other departments. The manufacturing, marketing, finance, sales, and accounting departments all provide input to the R&D department concerning the potential R&D projects, helping determine costs, future profit potential, possibilities of both technical and commercial success, and how the R&D project might fit in with other corporate strategies in terms of product mix, etc.

Typical goals used in the R&D selection process include maximization of the expected return from the R&D projects, which is a function of projected cash inflows from the R&D project, and, the probability that the project will be successful. An R&D project can be defined as a technical success if the R&D project is feasible in terms of function and cost, or a commercial success if a demand for the product or service exists and it is profitable to provide it. For business R&D, eventually all R&D projects must result in commercial success. In each case, the R&D project can generate cash flows for the organization.

A related goal used by R&D management for R&D project selection is to maximize the expected profit for the R&D projects selected. This is slightly different from expected return as it takes into account the budgetary cost of the individual projects; the expected profit of the R&D project is the difference between the expected return and the expected cost for the R&D project. In certain cases, top management is interested in generating high cash flows, i.e. sales, and the expected return goal would seek to ensure that R&D projects would be selected that would help meet this overall goal. In this case, the profits generated by the projects selected may not be maximized, since the costs of the projects are not considered in the goal. If top management was interested in maximizing the profits generated by the R&D projects, then maximizing the expected profit would be the appropriate goal to use.

If top management is also interested in maximizing the use of the manufacturing resources and equipment, then maximizing resource utilization would logically be a secondary goal used in the R&D project selection process. This would involve selecting projects with an objective of utilizing the various resources of the R&D department as fully as possible. The R&D department or function might want to include this goal, especially if it has some degree of autonomy within the organization, as a means for self preservation.

Top management might be concerned with generating a stream of successful products or services. If this is the case, then another goal the R&D management could use for R&D project selection would be to maximize the joint probability of success of the projects selected. This would lead to projects being selected that have the best chance of being successful.

The scheduling of the selected projects is also important to R&D management, as the timing of the R&D projects impact the company or organization as a whole. If scheduling of projects is considered during the project selection process, then the R&D department can make estimates of the projects' durations, as well as the time spent in various stages of the R&D process. In addition to the goals outlined above, the inclusion of scheduling in the R&D project selection process can improve the quality of the solutions obtained. The manager can use scheduling goals such as minimum total throughput of the selected projects, or minimize the makespan of the selected projects. If the organization imposes due dates on the R&D projects, the manager could also have as a goal the minimization of the total tardiness, or lateness of the selected projects. Such due dates might result from market requirements, or contract requirements. In any case, by considering these scheduling goals a priori, the company or organization can use the R&D project selection model knowing that the resulting set of selected projects will not only meet the economic and strategic goals, but will also be feasible in terms of scheduling the projects.

R&D project scheduling can also cause a conflict of resources, resulting in unnecessary delays. Consider the case where two or more R&D projects require the same resource. Although there may be enough resource capacity to handle the projects, the resource can only be utilized by one project at a time. The R&D department must now decide the proper sequence of R&D projects for the resource. Sequencing is a typical scheduling problem, and by taking this into account during the project selection process in the form of a goal, conflicts like this can be minimized or avoided.

Previous research on R&D project selection

There presently exists a plethora of models designed to handle R&D project selection. The literature on R&D project selection includes over two hundred quantitative and qualitative models (Danila 1989), each attempting to deal with R&D goals and constraints in order to find the "best" method for selecting a project portfolio. Reviews of the research on R&D project selection can be found in Baker and Pound (1964), Cetron, Martino and Roepcke (1967), Gear, Lockett and Pearson (1971), Baker (1974), Baker and Freeland (1975), Souder (1972), Liberatore and Titus (1983), and Danila (1989).

These reviews of R&D project selection models evaluate nearly every different model type proposed for R&D project selection. This includes scoring models, financial models, portfolio models, math programming models, decision theory models, and consensus models. Each model type addresses different issues and characteristics of the R&D project selection problem. The purpose of this research is to develop a model that incorporates one important issue in R&D project selection that has not been addressed by traditional R&D project selection models: project scheduling.

Some R&D project selection models consider the projects individually during the selection process. These models evaluate each project in terms of the goals defined by management, and then rank these projects. The selection process is completed by selecting the projects in rank order until the budget is exhausted. Scoring models, financial models, checklist models, decision theory models, and consensus models all typically evaluate the projects individually in this manner. This is primarily why project scheduling has not been incorporated into these models. Project scheduling requires consideration of the projects' interaction and/or interdependence. The projects compete for resources and/or facilities during the R&D process, and the sequencing of the projects through these resources and facilities can affect the timing of the project benefits. By considering the projects individually, these models make the assumption that project scheduling and sequencing through the R&D facilities is an insignificant concern for R&D management, and this is not the case.

In order to incorporate scheduling of the R&D projects into these models, all possible project combinations would have to be considered and evaluated. For small problem sizes, this would be possible, although not very practical. As the number of projects increases, the number of different combinations of projects explodes exponentially. The number of different combinations is 2^n, where n is the number of projects under consideration. For realistic problem sizes, it would be difficult to incorporate scheduling into these model types.

Another process is to consider the projects concurrently, rather than independent of each other. By doing this, the projects' utilization of resources is considered with respect to optimizing the R&D project selection goals. Math programming and multi-criteria models use this approach for selecting R&D projects, and generally give superior results to the simpler ranking approach. However, the solution methodology is much more complex as well.

Portfolio selection models, including math programming techniques such as linear programming, dynamic programming, integer programming, and nonlinear programming consider the project selection process as the creation of an optimal portfolio, or group, of R&D projects, subject to certain objectives and constraints in the R&D process. These models require the R&D manager to formulate the R&D objectives and constraints into exact mathematical equations relating the objectives and constraints to each other. For goals such as profit maximization, resource utilization and probabilities of success for the individual projects, efficient algorithms have been developed for solving the R&D project selection problem. However, even some of these methods have their limitations. For instance, linear programming and integer programming problems are solved using algorithms that have been shown to have exponential time complexity (Garey and Johnson 1979). The simplex algorithm is an exponential time algorithm, but in practice has a record of running quickly. Including project scheduling into these models would only compound the problem of complexity, and as a result no formulations for including project scheduling into the R&D project selection problem exist in the literature. Some models in the literature use simplifying assumptions about project scheduling in the project selection model, for example by creating mathematical equations for computing project completion times as a function of the number of researchers assigned to the individual projects (Taylor et al. 1982). This works well if project interdependence is minimal, but if project interdependence is not insignificant the solutions obtained may not be optimal, in terms of scheduling requirements.

In summary, the research on R&D project selection does not realistically consider scheduling during the R&D project selection process. Some models do take into account scheduling in a simplified manner, but in general the effect of project scheduling on project selection is not taken into account. The impact of project scheduling on project selection has not been explored by the models, and these models will be discussed in greater detail in Chapter 2.

Project scheduling

Liberatore and Titus have developed a relatively complete review of the methods used for R&D project scheduling and control. The most popular techniques are Gantt charts, PERT/CPM, and GERT. These techniques are implemented after the R&D projects have been selected, and several companies expressed an interest in "interactive systems for resource allocation and multiproject tracking and control" (Liberatore and Titus 1983).

Multiproject scheduling with limited resources, or resource constrained scheduling has received some attention in the literature. Pritsker, Watters and Wolfe (1969) developed a a zero-one (0-1) linear programming formulation for a multiproject resource constrained scheduling problem, and show that for small cases optimal solutions were obtainable. Because attempts at using integer programming to solve realistic problems of this type were not very successful (Patterson 1984), enumerative approaches were developed to solve different versions of this type of problem. Patterson (1984) compares several enumerative based approaches for solving a multiple constrained resource, project scheduling problem. Each procedure limits the search for optimal solutions by "searching the set of possible solutions in such a way that not all possibilities need be considered individually."

The project scheduling problem, when combined with resource constraints like those in R&D project selection, is very difficult to model, and algorithms that provide optimal solutions are nonexistent.

Significance

Although R&D project selection has been a subject of intense academic research, the research has mostly ignored the impact of project scheduling on project selection. The inclusion of project scheduling represents a more realistic problem, albeit a much more difficult problem to solve.

This research will represent an extension of the existing research on R&D project selection by: (1) developing a methodology for including project scheduling in the project selection process, and, (2) designing a heuristic algorithm that will select and schedule R&D projects within the system constraints and according to goals relevant to the R&D process.

Scope and limitations

This dissertation will incorporate research project scheduling as a goal (or objective) of the R&D project selection process. The scheduling of projects will consist of directing the individual research projects through the various stages of the R&D process. Projects will require different routings through the stages of the R&D process, and projects will likely have to wait for other projects' completion before continuing.

The filtered beam search approach

This research will require the development of a heuristic algorithm to perform the selection and scheduling of R&D projects. A heuristic algorithm is considered the only viable alternative to enumerative methods for scheduling problems, because of the NP-Complete nature of scheduling problems. Exact methods are efficient for very small problem sizes, whereas realistic R&D project selection problems can encompass large problem sets. The methodology selected for this dissertation is a heuristic algorithm based on the "Filtered Beam Search", a search technique developed in research on Artificial Intelligence (AI). The filtered beam search belongs to the class of search techniques called beam search techniques, which have been successfully

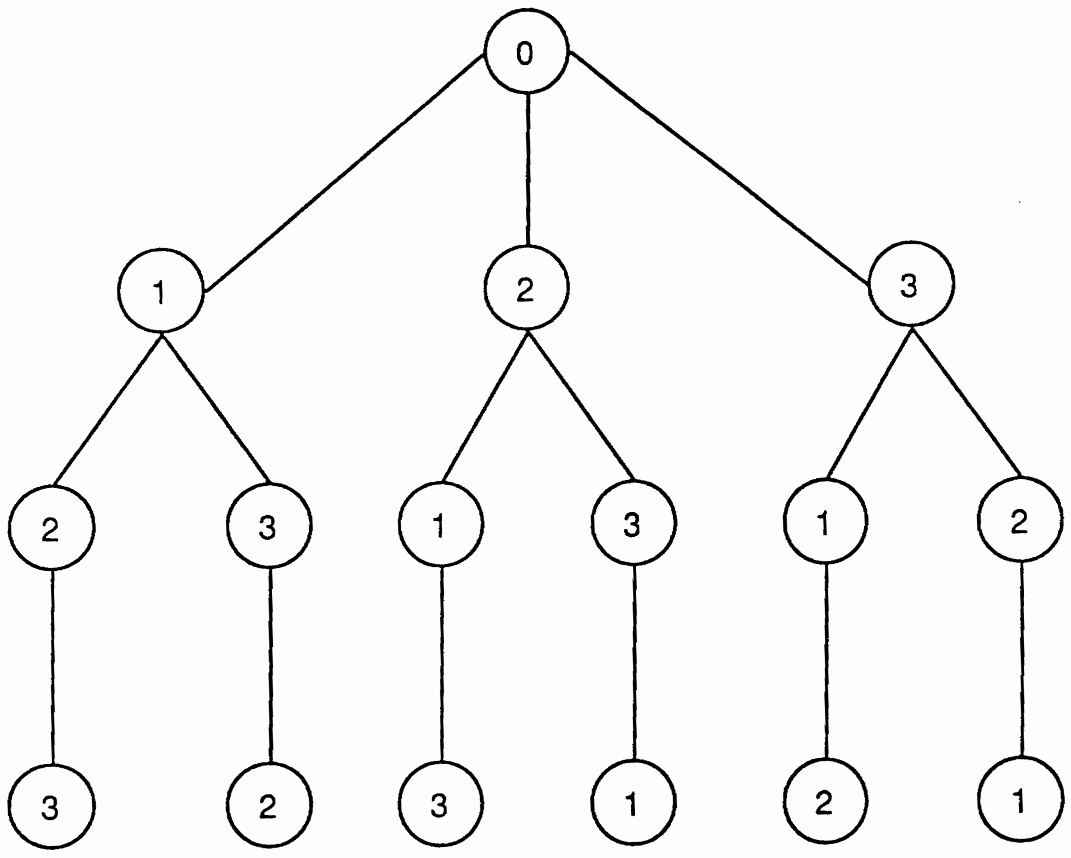

Figure 1.1. ***Example of a search tree***

used in speech understanding and image recognition. Beam search is a heuristic based on exploring a *search tree*. A search tree structures the solution space in the form of a tree, as shown in Figure 1.1. The nodes of the tree represent the individual R&D projects, and the branches "connect" the projects, i.e. the projects are placed together in the solution. The beam search method was first known to have been implemented by Lowerre in HARPY(1976), a speech recognition system. Rubin later used beam search for an image recognition system called ARGOS(1978). Fox used the beam search to find schedules in a complex job shop environment (1983).

Search techniques using search trees have mainly been developed and studied by researchers in Operations Research and Artificial Intelligence (Barr and Feigenbaum 1981, Lawler and Woods 1966, Nillson 1980). Three common search techniques are breadth-first, depth-first and best-first. Breadth first moves down the tree, level by level, evaluating every node at each level, and continues moving down the search tree until the goal node is found, or until all paths downward through the tree are blocked and the search is terminated. The breadth-first search will eventually find the optimal solution, but because it must keep track of all nodes at every level it can be a very inefficient and time consuming approach as the problem size increases.

Depth-first search begins by diving into the search tree on one branch until the goal node is found or the search is halted, i.e., the branch is terminated because of some problem constraint. The depth-first search continues by restarting at the highest unexplored branch, until all branches are searched, or the goal node is found. The depth first search will also eventually find the optimal solution, but can be very time consuming as well.

Best-first search begins by evaluating the nodes at the next level, and searches down through the tree, starting at the best node. The downward search continues through this branch until the goal is found, or until the search halted. Then, the process restarts at the next best node, and the process is repeated until the goal node is found, or all branches of the the search tree have been explored.

The beam search is a variation of best-first search technique, exploring only the "few best" nodes at each level, and never backtracking. This requires evaluating the nodes at each level, and choosing the best nodes according to some evaluation function. Then the search continues to the next level, and the new nodes are evaluated. The few best at this level are selected, and the search process continues until the goal node is found, or all paths terminate. In this case, the "best" node remaining is the solution found by the beam search approach. The beam search is not guaranteed to find the optimal solution, but has been shown to be very effective. The trade-off is the increase in computational efficiency for the beam search versus the exactness of other solution techniques. For the other techniques, computation time can "explode" as the problem size increases, whereas the computational complexity for the beam search is polynomial, making it practical for solving large problems. Thus, the beam search allows an efficient search through a problem domain that may be potentially explosive in size, and still permits close to optimal solutions. The details of the filtered beam search algorithm will be presented in greater detail in chapter 3.

Scheduling techniques

Many techniques have been proposed for R&D project scheduling (Davis 1973), (Dean 1980). Most of these techniques deal with the project network problem of minimizing project duration. The techniques range in complexity from simple Gantt charts to very sophisticated computerized heuristic algorithms, such as Resource Allocation and Multi-Project Scheduling (RAMPS) (Liberatore and Titus 1983). The project scheduling problem itself is extremely complicated when multiple projects and resource limitations are involved.

The structure of the R&D project scheduling problem presented in the literature is very similar to the general job shop scheduling problem. The problem consists of attempting to schedule a number of projects (jobs) through a group of R&D facilities (workcenters), often subject to scarce resource availability. Baker (1974) presents many job shop scheduling heuristics in his text, and which one is appropriate depends on the problem structure and objectives. In project scheduling, the objective can be meeting due dates as closely as possible, minimizing average project completion times, or minimizing overall completion time of the set of projects, etc. Scheduling heuristics and techniques will be discussed further in chapter 2.

The heuristic chosen for this problem is one which minimizes the overall completion time of the set of selected R&D projects. This heuristic is designed explicitly for accomplishing this objective, and this scheduling heuristic will be presented in greater detail in chapter 3. Other heuristics for solving this specific problem could be used in the framework of the solution model presented in this research, but the primary emphasis is to determine the viability of including project scheduling as a primary objective in the project selection process. Once this has been established, determining which scheduling heuristic works best for specific problem types would be an appropriate subject for future research in this area.

The scheduling objective used in the filtered beam search approach is up to the decision maker. It is likely that if different objectives are used, different optimal solutions will result. One advantage of the filtered beam search technique is its modularity, allowing changes to the evaluation process without a massive restructuring of the model. As a result, the impact of different objectives on the solutions obtained could be investigated.

The R&D project selection problem

There will be two different R&D project portfolios investigated in this dissertation. The first will consist of 10 projects, and each will require four resources: (1) budget, (2) researchers, (3) a resource X, and, (4) a resource Y.

Also, each project will be directed through three stages of the R&D process, designated stages A, B, and C. The projects will remain in each stage for a specified duration. This time is deterministic to reduce the computation complexity of the scheduling heuristic used in the solution algorithm.

The projects each have a return (cash flow) and an estimated probability of success. These will be used to evaluate two of the R&D goals, expected return and group probability of success. Minimizing the makespan of the selected projects will be the third goal of the R&D project selection model.

The second R&D project portfolio will consist of 50 projects, in order to test the solution approach with a more realistic problem size. The same goals and the same resource constraints will be used for the larger problem set. However, the amounts of each resource available will be different for the larger problem size.

The two project sets will be used as follows:

- The smaller set (10 projects) will be used to develop and demonstrate the filtered beam search algorithm. The algorithm will be tested and validated using this smaller set of projects.

- The smaller set will be evaluated using a complete enumeration routine to determine the exact optimal solutions. Also, an integer zero one goal program will be used to determine the effect of including project scheduling in the project selection process. The solutions obtained by complete enumeration and by the math programming model will be compared to the solutions obtained by the filtered beam search. The validation procedures will be discussed further in chapter 4.

- Once the algorithm has been validated, the larger project set will be used to demonstrate the algorithm's computational efficiency for a more realistic problem size. The results will be evaluated and several forms of sensitivity analysis will be tested including alternative goal structures.

Model validation and limitations

This research on R&D project selection and scheduling is conducted using a hypothetical R&D project portfolio. The heuristic solution algorithm will not produce optimal results, and therefore must be judged accordingly. However, the algorithm has been shown to be an efficient search technique. The efficiency of the algorithm depends on:

- Problem size. As the problem size increases, the computation time will increase. The increase in computation time for this algorithm is proportional to (wn^2), where w is the beamwidth of the search and n is number of projects considered. The maximum computation time is bounded polynomially, making the algorithm fairly tractable as the problem size is increased to handle realistic problem sizes.

- Beamwidth of the search. The number of paths explored in the search tree will affect the computation time. This relationship is linear, and research has shown that relatively few search paths are needed to approach exact optimal solutions when using the filtered beam search approach (Ow and Morton 1989).

- Filterwidths of the search. The use of filters during the search algorithm trims the search space. These filters are used to incorporate the various sub-goals of the project selection model. Decreasing the size of the filters decreases the computation time required for the algorithm. However, smaller filters increases the likelihood for missing the optimal solutions. The use of filters will be discussed in more detail in chapter 3.

- The scheduling heuristic. The scheduling heuristic used is one presented in the Baker text (1974). The similarities of project scheduling to job shop scheduling are significant (Davis 1973), and allow the use of efficient scheduling heuristics developed for job shop scheduling. This heuristic generates schedules that attempt to delay each project step as little as possible. It is based on an implicit tree search, and can handle reasonably large problem sizes. It is one of many scheduling heuristics available for project scheduling, and was chosen because it was one that was designed to handle an R&D type scheduling problem, i.e., schedule a finite number of projects (jobs) through a finite number of facilities (workcenters). The impact of different scheduling heuristics on the efficiency of solutions obtained is an area for investigation.

Note that a scheduling heuristic is used in the filtered beam search algorithm. The effect of using a heuristic within a heuristic algorithm must also be considered before the validity of this approach can be assessed.

Plan of presentation

Chapter 2, entitled "Review of Related Literature", summarizes the research to date in project selection and project scheduling. The literature of tree search techniques, specifically beam search and filtered beam search techniques will be reviewed. Project scheduling as it relates to R&D project scheduling also will be reviewed.

Chapter 3, entitled "The Filtered Beam Search Approach", describes the filtered beam search model, and the algorithms used to solve the model. The objectives and constraints will be outlined for the 10 project case problem and the 50 project case problem. The project scheduling heuristic used for the model will also be described.

Chapter 4, entitled "Model Solution and Results", describes the solution of the two project cases, the 10 project case and the 50 project case. Also, it provides a sensitivity analysis of the model with respect to different goal structures and scheduling rules.

Chapter 5, entitled "Summary and Conclusions" summarizes the research findings, and suggests avenues for future research in this area.

Chapter 2: Review of Related Literature

R&D project selection and R&D project scheduling have been areas of extensive research over the last three decades. The literature in both areas include a plethora of models and techniques to accomplish project selection or project scheduling, but research addressing both issues concurrently is rare. This chapter will: (1) review individual project selection models, in which project scheduling is not included, (2) present a summary of the comprehensive studies on R&D project selection models, and, (3) review models for and techniques project scheduling.

The general problem that is the focus of much of the research literature in R&D project selection is to decide on a program of research and development that will best meet various objectives in an organization. For instance, R&D projects can be related to the introduction of new product ideas and technologies, or can be related to developing new production methods, or improving existing production methods. Thus, the R&D department is faced with an array of possible R&D projects, each of which has potential benefits to the company. The quantitative R&D project selection models presented in the literature seek to optimize some objective function by choosing R&D projects within a set of constraints (Gillespie and Gear 1973). There have been a large number of models developed in the literature to help R&D managers optimally select a portfolio of R&D projects. In a survey of existing R&D project management models, Danila (1989) found more than two hundred quantitative and qualitative models for selecting R&D projects.

Alternatively, the literature on R&D project scheduling tends to focus on resource constrained project scheduling. In R&D project scheduling, the optimal sequence of the R&D projects through the various R&D facilities and departments is determined with an objective related to the scheduling goals of the R&D department. This objective can be to meet the due dates of the R&D projects, minimize overall time to completion of the selected R&D projects, or to minimize the average completion time of the R&D projects. These goals may be set by the R&D department, by market forces, or by top management through strategic planning. Patterson (1984), and Davis (1973) present excellent reviews of the resource constrained R&D project scheduling literature.

After an extensive literature search, there are only a handful of R&D project selection models that explicitly deal with R&D project selection and scheduling. As such, the R&D project selection literature will be reviewed first, followed by a review of R&D project scheduling literature.

R&D project selection models

In this section, individual model types for project selection will be reviewed. The models are classified according to model type: checklist models, index models, scoring models, financial models, and portfolio models. Representative models of each type will be discussed, along with the limitations of each model and/or model type.

Checklist models

Keifer (1964) presents a checklist model for project evaluation and selection. In the model, fifty-six criteria are proposed for evaluating the individual projects, and these criteria are grouped into five classes: financial, R&D, production, marketing and corporate position. The checklist is used by the decision maker(s) to evaluate each individual project on each of the criteria, and the evaluation choices are: very favorable, average, unfavorable or very unfavorable for each criterion. The projects are then subjectively selected based on the evaluations for each project. Project scheduling can be one of the criterion evaluated, but only a subjective evaluation is made.

Other examples of checklist models are the EIRMA checklist and the BECKER model (Danila 1989). The EIRMA checklist proposed by the European Industrial Research Management Association, uses a scale of 1 to 4 instead of the subjective choices used in the Keifer model. The EIRMA model evaluates factors that influence both technical success and commercial success of candidate R&D projects. The BECKER model uses three checklists, one for the selection of research projects, one for product development projects, and one for process development projects.

More recent checklist techniques such as the DELPHI technique have been applied to the R&D project selection process. In the DELPHI method, R&D experts are asked a series of questions concerning the potential R&D projects, and the results are collected. The group is then presented summarized results from the first round of questions, and are asked to come to a consensus concerning which projects should be selected. This method is limited in the sense that results are dependent on the questions, projects considered and the experts involved. Whenever a qualitative model is developed, the high element of judgement involved in the process creates variation in results and makes standardization difficult to achieve (Souder 1972).

Recently, Analytical Hierarchy Process (AHP) has garnered some attention in the research on project selection. AHP was developed by Saaty (1980), and its main advantage is the process which is used. It is similar to the DELPHI method in that a group of experts is asked to consider the projects, but in AHP the group also considers the weights associated with the various criteria. Thus the group not only evaluates that the individual candidate projects, but also evaluates the R&D decision process by determining the relative importance of the decision criteria. AHP is a consensus type of process, and like the DELPHI approach, it is repeated several times before a final decision is made. The AHP model uses computer software packages to assist in the analysis for the process, calculating the numerous weights and rankings needed for the process. Computer software is necessary for large problems because of the explosion in the required number of comparisons during the ranking process as the problem size increases. The AHP process is somewhat tedious, and, because it is essentially a qualitative approach, it is subject to the same limitations as the DELPHI model.

Checklist models have several limitations that make them impractical for R&D project selection and scheduling. They are subjective evaluation models, which makes it difficult to obtain consistent results. The selection of projects depends partly on the decision makers involved, and changes in the group members can affect the outcome of the process. Also, there is no guarantee that checklist models will provide optimal solutions for the project selection process. Checklist models consider the projects individually, ignoring portfolio interactions, which makes it impractical to include project scheduling. To do so would require the checklist model to evaluate every possible project combination, which would be all but impossible for large problem sizes.

Index models

Index models attempt to measure the attractiveness of an individual R&D project by creating a performance index for each project, typically a ratio between some measurable benefit (savings, earnings, profit, cash inflows, etc.), and costs associated with the R&D project. This is an improvement over more subjective models such as checklist models, and provides a quantitative comparison between the R&D projects. A Cost/Benefit model is an example of a index model, in which an index is computed for every R&D project, dividing the estimated benefits of the R&D project by the overall costs of the project. The projects are ranked in descending order of the index ratio, and projects are selected until the budget is exhausted.

Another example of a ratio model is the Disman model (Baker and Pound 1964). In this model, the discounted net worth of the project is multiplied by the probability of success for the project to create a "maximum expenditure justified" (MEJ) for a candidate project. A ratio of the MEJ to the project cost is calculated for each project as an index, and these indices are used, along with the MEJ itself to select the projects. In the selection process, a high ratio would make the project attractive, as well as a high MEJ. The R&D manager would have to decide on the ranking of the projects according to this criteria. A project with a high ratio and high MEJ would obviously be an attractive project, but a project with a high ratio and a low MEJ might not. On the other hand, a project may have a low ratio but have a high MEJ, making it difficult to decide whether or not a project is a candidate for selection.

Jin and Porter (1987) present an index model that uses a ratio of output to input (O/I) as a means of evaluating projects. The values of output and input are defined as weighted scores, and these scores represent the performance of the different projects according to various decision criteria. The input criteria are defined as project duration, staffing requirements, funding requirements and workspace requirements. The output criteria are invention (counts of patents, copyrights, etc.), published papers, unpublished technical reports, books, secondary effects and training. The different criteria are weighted according to importance and a project score is computed as an index. The scores for each criteria are determined by R&D management in a consensus format, and each project is evaluated with respect to the other candidate projects. A high output to input index would indicate that a particular project is a good candidate for selection. The value of this model is that it takes into account many qualitative criteria in the project selection process, however it also suffers from the limitations of being a qualitative approach.

Other indices that have been used for project selection models are return on investment (ROI) and internal rate of return (IRR) (Danila 1989). These model types evaluate the candidate projects according to the selected index, and the projects are ranked accordingly. The projects are then selected one at a time until the budget is exhausted. Examples of these index models can be found in Ansoff (1961), and Danila (1989).

Index models have several inherent limitations. First, the R&D projects are considered individually, and any independence or interaction between the projects is ignored. This independence can affect the benefits or the cost associated with a project. Project Interactions such as resource utilization, facility requirements and project scheduling are ignored. In order to include these interactions, the R&D manager would have to evaluate every possible project combination, and this would be impractical for all but the smallest problems.

Scoring models

Scoring models attempt to address the multiple criteria of the R&D project selection process by quantitatively evaluating the various characteristics of individual R&D projects. The Mottley-Newton model (Baker and Pound 1964) is an example of a scoring model. In these models, an overall project score is computed based on rating each individual decision criterion on a numerical scale. The criteria include research budget, project risk, and overall program balance. The overall score may be an additive or multiplicative score, depending on the model. Moore and Baker (1969) note that an additive scoring model index consistently provides a higher degree of rank-order consistency than the multiplicative index. Scoring models take into account noneconomic criteria by assigning a quantitative value or score to the criteria, thus attempting to handle multiple criteria that do not have a common measure.

Moore and Baker (1969) present a complex scoring model that provides comparable results to more complex math programming models in terms of rank ordering, but this model is very problem specific, and changes in constraint ranges, profit ranges and probabilities of successes would require a very careful restructuring of the scoring model. This underscores a characteristic limitation of scoring models. The projects are scored, or evaluated, individually without concern for interrelationships between R&D projects. Resource requirements, facility requirements and scheduling are very difficult to include in scoring models.

One possible alternative is to consider each feasible combination of R&D projects and evaluate each combination on the various decision criteria. Of course, this may be possible for very small problem sizes, but as the number of R&D projects increase to any reasonable size, the number of possible combinations of projects explodes exponentially. Scoring models are quite popular in practice because of their relative simplicity, but as with checklist models, they do not guarantee consistent, or optimal results. It would be very difficult, if not impossible, to include scheduling in a scoring model if accurate results were desired by the R&D management.

Financial models

Financial models used for R&D project selection use expected benefits from an R&D project and the timing of the benefits to help determine which R&D projects should be selected. A period payback model is a simple financial model in which the time it takes for the project expenditures to be "paid back" in the form of project benefits. This type of analysis is very popular because of its simplicity (Liberatore and Titus 1983), but doesn't take into account the many other decision criteria normally included in the R&D project selection process.

Net Present Value (NPV) is another financial modelling technique used in R&D project selection. In this model type, an interest rate is used to discount future cash flows back to the current period, and a net worth of the project at the current point in time, a net present value, is computed for each R&D project. The projects are ranked according to net present values and selected, with the project with the highest net present value selected first, and so on until the R&D budget is exhausted. An excellent example of a model based on this technique is by Cramer and Smith (Cetron et al. 1967). In this model, each alternative R&D project is evaluated, with a net present value and probability of occurrence for each project calculated. Utility curves are also obtained for the projects, and the projects are ranked and selected based on expected value, or expected utility.

Again, these models do not explicitly consider resource utilization or facility requirements. Project scheduling is considered only in the sense of the timing of the project benefits, but these estimates are made independent of the other projects being considered. These models are frequently used for the project evaluation process, and provide input for the project selection process.

Portfolio models

Portfolio models attempt to maximize, or optimize, a benefit function in relation to model restrictions or constraints relevant to the R&D problem, by considering the projects in combination. This is an attempt to allow for project interdependence with respect to resource utilization, facility requirements, budget requirements, and scheduling objectives. Models designed for selecting a R&D project portfolio are usually based on mathematical programming,i.e. linear, integer, quadratic, or dynamic programming (Gear et al. 1971). Models using mathematical programming began to appear in the early 1960's. The models by Freeman (1960), Asher (1962), and Dean and Sengupta (1962) use linear programming to maximize expected net value of a project portfolio, subject to constraints such as budget, facilities, personnel, and materials. These models are limited in the sense that partial solutions are possible, making it difficult to decide which projects should be selected. However, one advantage of linear programming formulations is that the output can provide sensitivity analysis, and shadow prices which help the manager in making the project selection decision.

Linear Programming

An early linear programming model presented in the literature is by Bell et al. (1967). This model uses a linear programming approach that is designed to maximize a portfolio benefit function, subject to resource availabilities in each of several future time periods. The resources considered in the model are budget and manpower availability in the future time periods. The model allows for various levels of manpower classifications, increasing the flexibility of the model. For example, the model considers four grades of basic skill levels, G1, G2, G3, and G4. Grade G1 personnel are considered available for work requiring grade G1, as well as work requiring grades G2, G3, and G4. On the other hand, grade G2 would only be available for work requiring grades G2, G3, and G4, but not G1.

The Bell model uses the assumption that once a project is started, it will continue to completion. This limits the model, particularly if the R&D department can handle multiple projects at once. However, this is one of the few portfolio models that attempts to incorporate some form of project scheduling into the project selection process. This model attempts to sequence the projects in such a way as to maximize the expected benefit of the portfolio. Unfortunately, the relaxation of this assumption makes the problem considerably more complex, and using this model, or others like it, would be impractical. Another limitation of the Bell model is the use of continuous variables in the formulation. Fractions of versions of projects can appear in the solution, and it is difficult to decide how to interpret the partial solution, although Gear et al. point out that the major part of the solution is easy to interpret, leaving only a small subset of the solution for further study (Gear et al. 1971).

In summary, linear programming models are limited for R&D project selection and scheduling because of the linear decision variables, and because it is impractical to formulate project scheduling, let alone project selection and scheduling, in a linear programming form.

Integer Programming

Integer programming overcomes the limitation of using continuous decision variables. Typically, the formulations are in the form of zero-one integer programming, where a one represents selecting the project, and a zero represents not selecting the project. One of the earliest zero-one integer programming models for R&D project selection is by Beged-Dov (1965). In this model, the objective is to minimize the overall cost of assigning various research projects to several research teams, subject to budget limitations which place an upper bound on the amount of budget that can be allotted to the various research teams. The solution uses a transportation method algorithm to provide the optimal allocation of projects and budgets to the various research teams that will minimize the overall R&D cost expenditures. In this model, the only resource constraints considered are a budget constraint, and a maximum number of projects the various research teams can handle. Other constraints such as materials, facilities, etc. are not considered.

Other integer programming models are by Watters (1967), Brockhoff (1969), and Dean and Roepcke (1969). Cochran et al. (1971) present an investment model for R&D project selection using zero-one integer programming. This model employs a return on investment approach, using a discounted cash flow technique. The objective is to maximize the expected net value of the project portfolio, subject to budget constraints for various time periods. The model does not consider other resource constraints.

Bernardo (1977) presents an integer programming formulation, based on a graph-theoretic interpretation of the selection process. Early zero-one integer programming models consider only one decision criterion as an objective: a financial goal such as expected net value, or expected profit from the selected portfolio. In this model, Bernardo realizes the limitation of reducing the problem to the optimization of a single objective function. The model defines multiple decision criteria as objectives for project selection, and then the criteria are weighted by a rank-order assessment of the alternatives, as determined by a panel of experts. The decision criteria evaluated in this model are (1) need for application, (2) analytical investigation, and (3) development of new instrumentation and techniques. The objective is to maximize the benefit contributed by the portfolio based on the objectives, subject to a single resource constraint. Bernardo transforms the subjective constraints to quantitative constraints through the use of an inclusion matrix, which helps determine the value of including the project in the portfolio in terms of the ranked objectives. This model is an interesting approach to the multi-criteria nature of the R&D project selection process, but is rather cryptic and difficult to implement because it requires multiple runs of the integer program, and requires a subjective ranking of the projects. This would make it difficult to obtain consistent or optimal results for various problem sets. Project scheduling is not included in the model. including project scheduling would be impractical in the model, because the inclusion matrix entry for project scheduling would merely indicate the value of including a particular project in the portfolio, not the time to completion of the project or set of projects.

Integer programming models have various limitations for R&D project selection and scheduling. The multiple criteria involved in the R&D project selection process often have no common measure, making it difficult to formulate an objective function that is practical or realistic. Also, because project scheduling cannot be formulated in math programming form for any reasonable problem size, it would be nearly impossible to incorporate R&D project scheduling into an integer programming model. Lastly, because integer programming problems are NP-Complete, as the problem size increases, integer programming models are likely to run into computational difficulties. This last limitation has limited the use of integer programming for R&D project selection in practice.

Goal Programming

Goal programming addresses the multi-criteria nature of the R&D project selection problem. Keown, Taylor and Duncan (1979) present a zero-one integer goal programming approach to R&D resource allocation and project selection. In the model, nine goals are considered in the selection process : (1) budget, (2) facility capacity goals, (3) manpower capacity, (4) priority projects, (5) strategic balance, (6) risk, (7) sales goal, (8) market share growth, and (9) net present value. The various goals are subject to constraints, and the objective is to minimize or maximize the deviation from the constraints as much as possible, depending on the goal. The problem consists of twenty R&D projects to choose from, and is solved using the Lee-Morris algorithm. Project scheduling is not included in the model, as the timing of the projects, or the time to completion of the individual projects are not considered during the selection process.

Taylor, Moore and Clayton present a model using integer nonlinear goal programming (1982). This model includes the use of nonlinear relationships in the R&D project selection process. It also addresses project scheduling in the formulation by estimating the individual time to completion for the R&D projects as a nonlinear function of the number of researchers assigned to the R&D project. However, project interdependence is not considered in the scheduling calculation. The objective of the model consists of 9 goals : (1) probability of individual project success, (2) budget, (3) expected return of selected projects, (4) joint probability of success of selected projects, (5) time to project completion, (6) total time allocated to selected projects, (7) computer capacity utilization, (8) mutually exclusive projects, and (9) preferred projects. The model attempts to achieve the goals as fully as possible subject to the various system constraints, and the model is solved using a specially written FORTRAN program that utilizes an implicit enumeration procedure. The model achieved satisfactory results, and also performed sensitivity analysis in terms of achieving the various goals. The assumption of project independence for project scheduling would be difficult to address in this model because of the math programming formulation requirements.

Santhanam, Muralidhar and Schniederjans (1989) present a zero-one integer goal programming approach for information system project selection. In this model, an attempt is made to address the lack of a common measure for the multiple criteria in project selection by using a ordinal ranking scheme in the objective function of the goal programming model. The various goals for the project selection process are identified: tangible corporate goals such as cost and profit goals, and intangible goals such as organizational learning, planning improvements, and decision making effectiveness. In the model, every project is rated for the project selection goals using an interval scale of 1 to 10, where a score of one indicates a high benefit to the organization, and a 10 would indicate low benefit to the organization. The objective function for the goal programming formulation is written to encourage the model to select the projects that will provide the greatest benefit to the organization. The actual math programming formulation includes a budget constraint, as well as other constraints such as CPU time availability, programmer availability and systems analyst availability. This model would be an excellent model in the case where the organization has many qualitative goals, but because the ranking scheme is subjective, it would be difficult to incorporate quantitative project scheduling into the model.

All of the math programming models generally omit the goal of project scheduling. Since scheduling is difficult to include in mathematical programming models for any realistic problem size because of computational difficulties, project scheduling has not been addressed in conjunction with R&D project selection models using any form of math programming models. This, along with the fact that most of the more recent models require complex computer programs that run into computational problems for project sets of any size, has resulted in a need for new approaches for R&D project selection.

Comprehensive studies of R&D project selection models

Baker and Pound

In 1964 Baker and Pound published a comprehensive review of the R&D project selection literature to that point in time. These authors reviewed various R&D project selection models with respect to (1) familiarity of the models to R&D managers in industry, (2) the extent to which the models had been tested, and (3) the extent to which the models had been used in practice. This paper reviewed ten quantitative models, classified into three categories: decision theory models, operations research models, and economic models. The authors review a representative example of each category of R&D project selection model.

The decision theory model, which is an example of a scoring model, is by Mottley-Newton (1959). In this model each project alternative is subjectively rated on a three point scale for several criteria: technical success, time to completion of the project, estimated cost of the project, the strategic marketing need for the research, and the market gain (gross revenues) expected from the R&D project. All criteria are assumed to be independent. The R&D project selection process consists of three steps : (1) an overall project score is computed by multiplying the five subjective rankings for each project together, (2) the projects are ranked in descending order, based on this multiplicative score, and (3) the projects are selected, with the project with the best overall score selected first, the project with the second best score next, and so on until the R&D budget is exhausted.

Project scheduling is only considered in terms of time to completion of the project, on a scale of 1 to 3. A score of 1 would indicate the project is expected to take a "long" time to complete, and a score of 3 would indicate the project is expected to take a "short" time to complete. These rankings could be based on mathematical estimates of project completion times, but most likely would be based on an "educated guess" on the decision maker(s) part. The subjectivity of this evaluation highlights a limitation of scoring models in general, and this will be discussed in more detail in the sections dealing with the different R&D project selection model types.

The economic model reviewed by Baker and Pound is by Disman (1962). This model is an index model, and is representative of many models using discounted cash flows, present worth, and interest rates for R&D project selection. In the model, the discounted net worth of the project multiplied by the probability of success for the project is considered to be the "maximum expenditure justified" (MEJ) for the project. An index is computed by dividing the MEJ by the project cost, and the projects are ranked according to the index. In this model the alternative R&D projects are considered independently when computing the net present values, without regard to the actual scheduling requirements of the projects. This model was discussed earlier in the section on index models.

The operations research type model reviewed by Baker and Pound is by Hess (1962), which is an example of a portfolio model. This model is dynamic programming model, in which Hess proposes that the R&D project selection process is sequential in nature, and, uses a dynamic programming formulation to maximize an expected discounted gross value of the set of projects minus the current period R&D expenditures. Baker and Pound note that the model data requirements and assumptions are restrictive, and the examples provided by Hess are artificial.

The general conclusion reached by Baker and Pound was that operations research models, and economic models, had not been generally accepted in practice. This was primarily due to the fact that R&D project selection methods have not been tested using real data. Baker and Pound also pointed out that the R&D project selection process itself tends to be sketchy and poorly organized in reality, making it difficult to assess the strengths and weaknesses of any particular R&D project selection method.

Cetron, Martino and Roepcke

In a 1967 survey of quantitative models for R&D project selection models, the Baker and Pound paper was extended. Thirty quantitative R&D project selection models were evaluated according to a substantial list of criteria considered important for a R&D project selection model. The thirty models consisted of the ten models presented in the Baker and Pound paper, and twenty more models presented in the literature since the Baker and Pound paper. No recommendations or comparisons of the different models are given, instead, charts reflecting the performance of each model based on the evaluation criteria as outlined by Cetron et al. are presented.

These criteria include measures of utility of an R&D project, probability of success for a project, orthogonality of criteria, sensitivity to changes in inputs, scheduling requirements, strategies of the R&D department, optimization criteria used, and constraints of the R&D problem.

Moore and Baker

Moore and Baker in their 1969 paper quantitatively compared a scoring model for R&D project selection, an economic model, and, a constrained optimization model using a sample R&D problem. The purpose of the comparison was not to develop a new scoring model for R&D project selection, but rather to discuss the model development process itself, and determine if the performance of a scoring model is comparable to other quantitative techniques, such as financial models and math programming models.

In the paper, Moore and Baker investigated the impact of probability of success of the individual projects, the timing of the stream of income payments from the R&D project and the timing of the stream of the cost expenditures for the R&D project on the performance of the three models. The optimization criteria used was expected discounted income from the project set. Moore and Baker concluded that given careful structuring, it is possible to construct a scoring model which is rank order consistent, i.e. the project scores determined by a scoring model lead to the same projects being selected as compared with forms of economic and constrained optimization models. However, in the scoring model presented in the paper, the ranking of criteria is not subjectively evaluated, but is transformed from a normal distribution to a 1 to 9 scale. For example, twenty projects are generated using a Monte Carlo method, and if the probability of success generated for a project is within plus or minus 0.25 standard deviations of the mean, it is given a subjective score of 5, for the purposes of the scoring model. The scores of 1 through 9 correspond to certain intervals around the mean value. In effect, this is a complex hybrid scoring model that attempts to alleviate some of the inherent weaknesses in scoring models.

Gear, Lockett and Pearson

Gear, Lockett and Pearson in their 1971 study presented an analytical review of representative R&D portfolio selection models. Nine models were reviewed, and the models were selected with the intention of providing a good coverage of approaches available for portfolio selection. The models reviewed were by (1) Bell, Chillcott, Read and Solway, (2) Watters, (3) Brandenburgh and Stedry, (4) Brockhoff, (5) Dean and Roepcke, (6) Hess, (7) Rosen and Souder, (8) Dean and and Hauser, and (9) Charnes and Stedry.

The Bell et al. (1967) model, which was discussed earlier, is based on a linear programming approach. In this model, each project, or alternative version of a project is represented by a variable that may take any value, between zero (project not selected) and one (the project is selected completely). The objective of this model is to maximize an economic benefit, in this case cost savings due to increased plant efficiency, subject to budgetary and manpower constraints. Scheduling of the projects is not considered. An interesting characteristic of this model pointed out by Gear et al. is that because a linear model formulation is used and not a strict 0-1 integer model, fractions of versions of projects can appear in the solution. This can lead to difficulty in practice determining how to implement a partial project.

The Watters (1967) model is a zero-one integer programming model that incorporates uncertainty into the R&D project selection process. The model uses an objective function that incorporates expected portfolio return, the variance of the portfolio return and a "coefficient of risk aversion." This objective function is an attempt to reflect the uncertainty of the project returns and the decision maker's level of risk aversion. The model also reflects uncertainty into the constraints of the problem, which are budgetary constraints in each of several future time periods. The probability of exceeding a budgetary limit in each period is developed, and incorporated into the constraints of the model. Watters solves the R&D project selection problem for a range of values for the coefficient of risk aversion, leaving the final selection of projects up to the decision maker, depending on the actual values, or range of values of risk aversion. Watters' model assumes independence of the cost and return within each project. Also, only budgetary constraints are considered, ignoring manpower and facility constraints for R&D project selection. The introduction of uncertainty into the model formulation increases the computational complexity of the model, particularly because it is a zero-one formulation. This would make it impractical for all but the smallest of R&D problems.

The Brandenburgh and Stedry (1966) financial model of the R&D project selection process is based on the "basic horizon model" of capital budgeting. The model uses a linear programming formulation, with the objective being to maximize the net worth of the project portfolio in the final time period. The constraints of the model consider budgetary costs in each time period, and lending and borrowing between the firm and the R&D laboratory from one period to the next. Interest on the money borrowed and loaned is used as a discount rate in the objective function for computing net worth. Gear et al. point out that "although the model includes variables for lending and borrowing, the great majority of R&D laboratories operate on fixed budgets with no lending or borrowing." This model considers only the financial aspects of the R&D project selection problem, ignoring the resource and facility requirements.

The Brockhoff (1969) R&D project selection and scheduling model is based on capital budgeting. Only financial objectives and constraints are used, and a mixed integer programming formulation is developed to select and schedule the R&D projects so as to maximize the expected profits of the project portfolio. The model also uses a discount rate to determine a maximum net present value. Because only financial constraints are used, it has the same limitations as the models developed by Watters and by Brandenburgh, i.e. it does not consider resource and facility limitations as constraints. Also, because the model uses a mixed integer formulation it can run into computational problems as the problem size grows large. However this is one of the few models in the literature to consider the scheduling of R&D projects in the selection process. The model attempts to "schedule" the projects by optimizing the timing of the projects in order to maximize the overall expected return of the portfolio. This scheduling is simply an ordering of the projects in time, and does not take into account resource requirements or facility requirements. The model does not include scheduling goals into the model, only financial goals. Because the model assumes independence of the projects, and only financial constraints are used, the increase in computational complexity for the model is minimized.

The Dean and Roepcke (1969) model is designed to assist resource allocation decisions, using mixed integer programming. In this model, developed for military applications, the purpose is to maximize a relative military value of the R&D portfolio, subject to budgetary constraints. Each project is evaluated in terms of its value in contributing to certain military objectives, and the kind of science or technology required to complete the project. A weighted military value for each project is computed, and the model is designed to maximize the cost effectiveness of the R&D project portfolio. The model does not consider a time factor, thus it would not be possible to consider the scheduling of the projects in this model. Also, Gear et al. point out that because the formulation is a zero-one integer programming formulation, large problem sets may present computational difficulties.

Three of the models reviewed by Gear et al. are based on dynamic programming: (1) Hess (1962), (2) Rosen and Souder (1965), and (3) Dean and Hauser (1967). In each case, only financial criteria are used in the project selection process, and budget allocations are only considered in the first period. The dynamic programming formulations involved are complex, and computational difficulties would arise for large problem sizes. It would be extremely difficult, if not impossible to incorporate project scheduling into these R&D project selection models.

The last model reviewed by Gear et al. is by Charnes and Stedry (1966). In this model, individual project selection is not considered, but instead focuses on optimal funding of research support. The model minimizes expected cost of the R&D portfolio, subject to resource constraints that take uncertainty into account. This is done by formulating the problem as a chance constrained programming problem. The model reviewed gives only mathematical formulations, without giving any numerical examples. Because the model does not consider project selection, it would not be appropriate for R&D project selection and scheduling.

Gear et al. concluded that it is difficult to decide on the appropriateness of various R&D selection models in a given situation, and that results of practical tests of these models using actual field data are needed. The authors also point out that the available models have serious limitations concerning multiple objectives and project interrelationships.

Souder

Souder (1972) presents a comparative analysis of operations research models for R&D project selection from 1955-1970. In this paper, Souder evaluates 41 qualitative R&D project selection models, based on five criterion considered important to R&D decision making:

1. Realism criterion. This includes multiple objectives, multiple constraints, orthogonal (mutually exclusive) variables, market risk, technical risk, manpower constraints, facility constraints, budget constraints, and uncertainty.

2. Flexibility criterion. The model can handle applied R&D projects, basic R&D, priority decisions, initiation decisions, budget allocation applications, and project funding allocations.

3. Capability criterion. The model's capability to perform future time analyses, optimization analyses, simulation analyses, project selection, allocation analyses and scheduling analyses.

4. Use criterion. This relates to the model's ease of use to the R&D decision maker, which includes sensitivity to changes in input, continuous or discrete variables, computer requirements, data obtainability, and parameter estimation.

5. Cost criterion. The costs of using the model, including set-up costs, use costs, personnel costs, computer costs and data costs.

Souder evaluates the 41 models, which are provided in a bibliography in the paper, by separating the models into six categories: linear models, nonlinear models, zero-one models, scoring models, profitability index, and utility function models. The conclusion is that linear, nonlinear and zero-one models have the highest flexibility, realism and capability of the models evaluated, making them the most suitable for R&D project selection. Project scheduling is included as one of the sub-topics of the capability criterion, but Souder does not make mention any of the models' capabilities for project scheduling. The evaluation of the models is done using a additive scoring model, with the criteria scoring estimates made by a panel of R&D administrators and R&D specialists.

Baker

Baker (1974) assessed R&D project selection models up to 1974. In this paper, Baker presents a general overview of the R&D project selection problem, its characteristics, inputs, outputs and selection criteria. He points out "that the R&D project selection decision can be characterized as having multiple criteria with no common, underlying measure." Baker also points out that the problem is often separated into two distinct processes in practice, benefit estimation for the individual projects, and project selection/resource allocation. These two processes are related: the output of R&D benefit estimation is part of the input for the R&D project selection/resource allocation process. The uncertainty inherent in benefit estimation introduces uncertainty into the project selection process.

Baker does not extensively review or present any quantitative models for R&D project selection, but outlines some the limitations of the models in the literature. These limitations include inadequate treatment of project and parameter interrelationships, inadequate treatment of uncertainty as it impacts on benefit measurement and parameter estimation, inadequate treatment of multiple, interrelated decision criteria that have no common underlying measure, inadequate treatment of time variant properties of the parameters and criteria, and recognition of important individual R&D personnel. Estimates of costs, probabilities of success and times to completion of projects tend to change as the projects proceed, and Baker notes that the project selection models in the literature are limited in their capabilities to handle these changes.

Baker pointed out that the trend in R&D project selection was towards models for program or portfolio, rather than individual measures, and techniques such as goal programming are suitable for handling the multiple criteria associated with R&D project selection problems.

Liberatore and Titus

Liberatore and Titus (1983) reviewed the use of management science techniques in R&D project management, particularly in the areas of project selection, project evaluation, budgeting decisions, and project scheduling and control. This paper attempts to determine the usage of management science techniques in the

R&D activities of 29 "Fortune 500" firms by using a nonrandom sample of 40 respondents from these firms. The respondents were questioned about the familiarity of financial methods, risk assessment techniques, formal budgeting systems, scheduling and control charts, math programming models, behavior models, and subjective evaluation techniques. Liberatore and Titus concluded that although techniques such as linear, nonlinear and goal programming are recognized by R&D managers in practice, very few utilize these techniques. They attribute this to the fact that R&D managers feel that the models available to them will not appreciably improve R&D project management.

Danila

Danila (1989) reviewed the literature on project evaluation and selection up to that time. The author categorizes the different models into thirteen different model types: (1) ratio, (2) scoring, (3) math programming, (4) portfolio, (5) matrix, (6) systemic, (7) checklists, (8) relevance trees, (9) tables, (10) multi-criteria, (11) consensus, (12) graphic, and, (13) integrative. Many of these categories overlap, and can be placed into the categories outlined earlier in this dissertation. Models in these categories were subjectively evaluated by R&D managers in France and Japan on the basis of ease of formulation and the degree of implementation. Danila does not make any conclusions about which models were superior in formulation or use, but briefly mentions that some of the methods have enough possibilities to contribute to not only the R&D strategy but also to corporate strategy.

Summary of selection models and reviews.

The R&D project selection problem has multiple criteria with no common underlying measure, and the various project selection models developed in the literature to address multiple criteria in the R&D problem have done so with varying degrees of success. However, project scheduling has not been incorporated in the models for project selection as a criteria, and scheduling can have an impact on many of the parameters and estimates for the project selection problem. The scheduling of projects can impact cost estimates, time to completion estimates and resource requirements, all as a result of portfolio interactions. Models that assume independence between projects, i.e. ignore portfolio interactions, such as checklist models, index models, scoring models, and financial models, are inadequate in their treatment of project scheduling. Portfolio models attempt to incorporate various project interactions, but the models in the literature cannot incorporate project scheduling as an objective.

Project scheduling

Project scheduling and control has received considerable attention in the literature. In contrast to the limited usage of mathematical techniques for R&D project selection, the usage of of mathematical techniques for R&D project scheduling and control is relatively high. An excellent review of project scheduling can be found in Davis (1973), and reviews of R&D project scheduling can be found in Dean and Chaudhuri (1980) and Liberatore and Titus (1983).

Project scheduling, while seemingly one of the important criteria associated with R&D project selection, is not generally included in project selection models.

Including project scheduling in the R&D project selection process will result in a more realistic model, reflecting the objectives of the R&D department and the organization as a whole. The R&D project selection models in the literature neglect scheduling as an objective primarily because of the difficulty of incorporating scheduling into the selection process, and as a result project scheduling is relegated to being performed after the project selection process. If the resulting schedule meets the organization's objectives then the project selection process is sufficient. However, often the schedule is not acceptable and projects must be deleted, alternative projects or sets of projects selected, resources increased, economic goals reduced or the desired schedule relaxed. These alternatives could be avoided if project scheduling is performed during the selection process, and thus the project selection process would reflect a more realistic R&D environment and provide more useful outputs. A review of project scheduling literature will provide some insight into which techniques may be used in the selection process.

The literature on R&D project scheduling can be classified into two categories: single project scheduling and multiple project scheduling. Davis (1973) presents a historical review of procedures for both categories of scheduling. In the R&D project selection problem presented in this dissertation, the objective is to select a portfolio of R&D projects, with the scheduling of the projects as an objective of the selection process. Thus, the appropriate scheduling techniques would be the techniques for multiple project scheduling.

Procedures for multiple project scheduling can be classified into two major categories: heuristic procedures and optimal procedures (Davis 1973). The heuristic procedures use a "rule of thumb" to help determine the best schedule for a particular set of projects. The rule of thumb helps determine the priorities among projects competing for the available, but limited resources. The heuristic procedures are designed to provide very good solutions very quickly and efficiently, but are not guaranteed to provide the exact optimal solution to any given project scheduling problem. The trade-off in exactness is the computational efficiency that is gained by using the heuristic approach.

In contrast, optimal procedures are designed to give the best possible, or optimal solution. Optimal methods for multiple project scheduling are normally based on some type of mathematical programming formulation or are based on an enumerative approach. In either case, these optimal approaches are generally only practical for very small problem sizes because as the problem size increases, the computation time for these procedures increases exponentially. Because of this limitation, optimal procedures have not been utilized very often for multiple project scheduling problems.

Approaches to Project Scheduling

The Gantt chart is used extensively for project scheduling (Liberatore and Titus 1983), but for large complex projects, the Gantt chart is limited because of its inability to reflect the interdependence between project activities and the uncertainties in the project activity durations (Dean and Chaudhuri 1980). Network methods are also used extensively for R&D project scheduling and control, and they overcome some of the limitations of the Gantt chart. Program Evaluation and Review Technique (PERT) was developed in the Polaris missile program, and is used extensively in practice (Liberatore and Titus 1983). The precedence relations between the different project activities, as well as the uncertainty in the project activities is reflected in PERT. The Critical Path Method (CPM) is similar to PERT, and the two methods are often used in conjunction. The idea behind PERT/CPM is to identify the activity, or activities, crucial to completing the project within a specified time. When multiple projects are introduced to PERT/CPM methods, the problem of resource allocation arises. The various projects utilize resources, and this resource demand needs to be checked against the resource capacities. It may be that the total demand exceeds the availability of resources in specific time periods. If the demand is variable within individual projects, the expected demand may be less than the resource capacity but the variance of demand may result in a shortage. Lastly, if the project duration is considered "excessive", increases in resources may be required to shorten the project duration. The different problems in resource usage lead to different techniques for project scheduling (Davis 1973).

PERT/CPM assumes unlimited resource availability, and this limitation has lead to a large amount of research in the area of resource constrained project scheduling, and to a lesser extent, R&D project scheduling under resource constraints.

Heuristics used for project scheduling

Many heuristic procedures exist for project scheduling. Because project scheduling is conceptually similar to job shop scheduling, many of the techniques used for scheduling multiple jobs through a manufacturing facility have been used for project scheduling and control (Davis 1973), (Dean 1980). Though a few exact algorithms have been developed for job shop scheduling, they have been shown to be impractical for all but the smallest problems (Baker 1974), (Davis 1973), (Patterson 1984). As a result, heuristic procedures for job shop scheduling, and project scheduling, are pervasive the literature and in practice.

R&D project scheduling is generally done under resource constraints. R&D facilities, materials and personnel represent realistic constraints on the R&D project scheduling problem. The research on project scheduling has therefore concentrated on resource constrained project scheduling and the development of heuristic procedures to solve this problem. Davis (1973) reviews multi-resource heuristic procedures for project scheduling. One important heuristic, from a historical viewpoint, is the Resource Allocation and Multi-Project Scheduling program (RAMPS), developed by DuPont in 1960. This program minimizes costs on a period to period basis by examining all feasible combinations of competing jobs, or project activities, in each time period. Each combination is scored according to a function that incorporates variables such as total slack, idle resources, and project delays, and the costs associated with each variable. It can handle up to 100 separate projects, each consisting of up to 2000 activities and requiring up to 100 different resource types. The program is still widely used within DuPont.

Davis and Heidorn (1971) present an algorithm for optimal project scheduling under multiple constraints, using a bounded enumeration approach, and techniques originally designed for use in assembly line balancing problems. The problem is fairly small, consisting of seven project activities that compete for three different resource types. Davis points out that the procedure has a major drawback : the number of feasible subsets of solutions increases very quickly as the problem size grows. The technique of bounded enumeration eliminates many feasible, but less desirable, solutions by using a "target" solution, in this case a project duration. If a subset does not "beat" the target, it is eliminated from future consideration. The efficiency of the algorithm depends on the accuracy of the target value used. If the target value is too low, very few subsets will be eliminated. If the target value is too high, no solutions will be found. As a starting point, the authors recommend the use of heuristic designed for the problem to generate a starting solution, and the continue the algorithm from that point.

Sherali and Rios (1984) present a scheduling heuristic for solving an allocation and scheduling problem for the Air Force. In this paper, the authors attempt to maximize the utilization rates of the aircraft, subject to meeting flight requirements and resource limitations. The problem is solved in two steps, the first of which solves the problem using a set of relaxed assumptions, and generates a starting point for the second step. The second step involves tightening the assumptions used in the first step, to reflect the actual problem constraints, and uses a series of simulations to determine a good, feasible schedule. This paper is significant in that simulation is used, along with several scheduling/sequencing heuristics, to solve the multiple task resource constrained scheduling problem. Simulation is an alternative to exact approaches, and has appeared in the literature for R&D project scheduling. Simulation will be discussed in a later section.

Optimal Procedures

Pritsker et al. (1969) present a zero-one programming approach to multiproject scheduling with limited resources. This model is a math programming formulation that will provide an exact, optimal solution to the problem. However, this model cannot handle a realistic problem size. The model presented is a three project, eight job, three resource problem. The zero-one linear programming formulation for this problem size requires 72 variables, and 125 constraints. Because project scheduling is an NP-Complete problem, this type of problem gets very complex as the problem size increases. Baker (1974) notes that in this type of zero-one integer formulation, the number of constraints increase on the order of n^2 where n is the number of projects. The number of variables required for the formulation also increases to the power of two as the problem size increases.

Patterson (1984) presents a comparison of exact approaches for solving the multiple constrained resource, project scheduling problem. Three procedures are compared : a bounded enumeration routine, a branch and bound routine and an implicit enumeration routine. The problems solved by these routines are single project, multiple task problems, with the number of tasks ranging from 7 to 50 tasks. In this particular instance, the branch and bound based procedure was superior to the bounded enumeration and implicit enumeration techniques in terms of computation speed. However, Patterson points out that the branch and bound required a high level of computer storage to run effectively.

Exact methods for resource constrained project scheduling have proved unsatisfactory. Davis (1973) notes that "the major alternative to heuristic procedures were various linear programming formulations; an apparent approach which in general has proved to be computationally unsatisfactory for even very small problems ". As a result, the majority of research on resource constrained project scheduling has focused on the improvement of heuristic procedures, particularly in the areas of implicit enumeration and simulation.

Simulation methods

As computer algorithms for simulation models improve, simulation may become an alternative to heuristics in project selection and scheduling. Simulation software packages such as GERT and Q-GERT have been used for R&D project scheduling in the literature.

Graphical evaluation review technique (GERT) allows for a less rigid, or deterministic, network model of R&D project scheduling. GERT is a stochastic modelling technique that allows for uncertainty in project outcomes, durations and optional paths through a project scheduling problem. GERT allows for different distributions in task durations, which can help the R&D managers determine project durations under different input conditions, such as resource allocations, facility capacities, etc. Moore and Taylor (1977) use GERT for a multi-team, multiproject research problem. The model presented solves a four project, two team problem using the GERT simulation software. The model provides R&D management with project scheduling statistics concerning time and cost. However, Taylor and Moore point out that as the number of R&D projects increase the GERT models become increasingly complex (1980).

Q-GERT is an evolution of GERT in that it has special queue nodes, and other features that make it helpful for R&D project scheduling and control. This helps accommodate larger, more realistic problem sizes. In Q-GERT the entities, in this the R&D projects, have the capability of "carrying" attributes through the project network. In R&D project scheduling, the individual attributes can be the various resources required for task completion, and in Q-GERT if the resources are not currently available, i.e. the resource is in use, the projects "wait" until the resource is available. In this manner, the R&D manager can investigate the impact of different levels of resources, task durations, etc. Using Q-GERT, Taylor and Moore (1980) solve a five stage, five project, two team problem using Q-GERT. The models are solved using the Q-GERT software developed by A.A.B. Pritsker.

Project selection models incorporating scheduling

In general, the R&D project selection problem, and the R&D project scheduling problem have each garnered considerable attention in the literature. However, the combined problem of R&D project selection and scheduling has received scant attention because of the complexity involved in combining the two problem types. Several models have been presented in the literature that attempt to incorporate scheduling into the project selection process. The model by Bell (1967) uses a simplifying assumption about the project schedules to incorporate scheduling into a portfolio model. Several models based on dynamic programming also use a simplifying assumption to incorporate scheduling into a portfolio model. The model by Hess (1962) is an early example of this type of model. Gupta (1992) also presents a model using dynamic programming, and also uses the simplifying assumption that once a project is started, it is completed without interruption, and only one project can be completed at a time. This assumption is restrictive, because R&D departments often work on multiple projects at a time. This model also only has one constraint, a budgetary constraint, where it is not uncommon to have multiple constraints in a realistic R&D problem. The Gupta model also requires numerous dynamic programming formulations. As the problem size increases, this could cause computational difficulties.

Summary

The various model types used for R&D project scheduling were presented, along with a review of representative models for each type. A review of important historical summaries of R&D projects selection was also presented, showing the progression of R&D project selection models over time. R&D project selection models have improved significantly over time, addressing many of the important characteristics of the R&D problem. Unfortunately, one important objective of R&D project selection, project scheduling, has not been realistically incorporated into the selection process. Several models have been presented that incorporate project scheduling using simplifying assumptions, but these models generally do not adequately reflect the R&D project scheduling problem.

R&D project scheduling models were also discussed in this chapter, particularly multiple project scheduling models. Heuristic and optimal procedures were reviewed, along with a fairly recent technique, simulation. R&D project scheduling is similar to job shop scheduling, allowing the use of proven scheduling heuristics in R&D scheduling.

The purpose of this research is to incorporate project scheduling into R&D project selection, and this requires considering multiple project scheduling techniques. Because project scheduling has not been incorporated into project selection at all, it would seem logical to start with a fairly simple project structure. The project structure for this dissertation requires at most 3 tasks per project, and would not require the complex heuristics outlined by Davis (1975), and Patterson (1984). Straightforward job shop scheduling heuristics can be used for R&D project scheduling, and the textbook on scheduling by Baker (1974) outlines numerous job shop scheduling heuristics. The scheduling heuristic in this dissertation used will be described in the following chapter.

Chapter 3: The Filtered Beam Search Approach

In this dissertation, an R&D project selection problem is solved using a heuristic model based on the filtered beam search approach used predominantly in Artificial Intelligence research. This chapter describes the R&D project problem, heuristic search techniques, scheduling heuristics and the filtered beam search approach to R&D project scheduling and selection. The following chapter discusses the solutions and the results obtained for examples using the filtered beam search approach.

The general R&D selection problem with scheduling

An R&D project is a potential product, process or service that has the potential for providing benefits to a company, or organization. These benefits can take different forms, but for business R&D projects, they are primarily future cash flows or profits to the company. This research will focus on a business R&D project selection and scheduling problem, subject to goals and constraints appropriate for a business organization.

The filtered beam search approach will be presented in the context of an example R&D problem, with project scheduling included in the project selection process. The R&D problem investigated in this dissertation will be approached in two stages: the first stage will consist of an R&D project selection problem with ten potential R&D projects, and the second will consist of fifty potential R&D projects. The ten project set will be used to develop, demonstrate, test and validate the filtered beam search algorithm. The smaller problem size makes it feasible to use validation methods based on enumeration, and current R&D project selection models, which can incur computational difficulties for large problems. The fifty project set will be used to demonstrate the computational efficiency of the filtered beam search approach on a more realistic problem size. The same goals and resource constraints will be used for both problems. However, the amounts of each resource available will be greater for the larger problem set.

Three primary goals will be used in the filtered beam search approach: (1) maximizing the expected return of the project portfolio, (2) maximizing the probability of success for the project portfolio, and (3) minimizing the overall time to completion of the project portfolio. The expected return of the project portfolio is computed using the projected profits of the individual projects and the individual probabilities of success for the projects. The expected return of an individual project is the product of the projected profit and the probability of success for the project, and the expected return for the portfolio is the sum of the expected returns for the selected projects in the portfolio. Maximizing the expected return of the project portfolio is the most common objective used in the literature on R&D project selection (Baker 1964, Baker and Pound 1974, Cetron 1967, Danila 1989). The portfolio probability of success will be defined as an index, calculated by summing the individual probabilities of success for selected projects. Moore and Baker (1969) note that an additive index for probability of success provides a better rank ordering of the projects when used as an objective in project selection. Maximizing the group probability of success will encourage the selection of those projects with the best chance of succeeding.

The third, and most important goal for this research, is scheduling the R&D projects as part of the selection process. The actual scheduling goal itself is not as important as including a scheduling goal into the project selection process. Because scheduling has not been included into the selection process previously, no guidelines for scheduling goals have been established. Several goals would seem appropriate for use in R&D project selection: meeting target due dates, minimizing average completion times for the projects and minimizing the overall time to completion of the selected project portfolio. Each of these goals is a realistic goal for the project selection process, and the choice of one goal over another is subjective, and would depend on the organization's objectives. However, the R&D process is inherently uncertain, and setting due dates for R&D projects might be difficult to do. In some situations, market forces could dictate due dates, but generally due dates would be set internally. If this is the case then meeting these due dates would be an appropriate R&D project scheduling goal.

A realistic situation is where management uses a planning horizon for R&D project management. In this case, the R&D manager would like to complete all of the projects selected prior to the horizon date. It would be important to determine if the selection of projects would change if this scheduling objective is imposed on the selection process. The goal of minimizing the overall completion time of the selected projects, or makespan of the project portfolio, would help meet the scheduling objectives of the organization. This is the goal used in the filtered beam search approach presented in this dissertation.

The computation of the makespan for the selected projects is performed using a scheduling heuristic originally designed for use in job shop scheduling. The objective in this problem is to schedule the various projects (jobs) through the required R&D facilities (workcenters). Three R&D facilities are used, facilities A, B, and C. These facilities represent various departments or organizational areas in R&D project management. R&D projects typically have to be engineered, designed, fabricated, tested, reworked, etc. and these facilities could represent one or more of the stages an R&D project must go through. The duration a project spends at each facility is estimated by the R&D department, and for this problem these times will be deterministic. The filtered beam search approach uses a scheduling heuristic to estimate the time it will take for all of the selected projects to complete the R&D process, and uses this as one of the primary objectives in selecting the optimal set of R&D projects that the organization should undertake. The description of the scheduling heuristic, and a numerical example of the heuristic follows.

Scheduling heuristic

The scheduling goal used by the filtered beam search approach is minimization of makespan for the candidate R&D projects. Exact methods for determining makespan are impractical for realistic problem sizes, so a scheduling heuristic must be used. Because R&D project scheduling is very similar to job shop scheduling, heuristics designed for makespan calculations in a job shop setting are appropriate for the R&D problem. Many heuristics exist, and the one used in the filtered beam search approach has been selected from the text by Baker (1974). The actual heuristic used is up to the R&D management, and the heuristic should be appropriate for the particular R&D environment. The primary purpose of this research is to show that scheduling can be incorporated in the project selection process, and a recognized job shop scheduling heuristic is used to provide validity to this approach.

The objective for for project scheduling is to minimize the makespan for the project subset under consideration, i.e. minimize the time to complete all of the projects as a group. Makespan is defined as the elapsed time, starting at time zero, that it takes to complete all of the projects in the subset, and the job shop scheduling heuristic by Baker is specifically designed to minimize the elapsed time. The heuristic works on the premise that the projects should start in the necessary facility as early as possible, and if there is a conflict, i.e. one or more projects can start in a facility a tie-breaker is used to determine which one goes first. The scheduling heuristic used in this research is presented here:

- Let PS_t represent the partial schedule containing 't' operations, where 't' is the current index of the algorithm.

- Let S_t represent the set of schedulable operations at stage t, corresponding to a given PS_t.

- Let σ_j represent the earliest time at which operation j as a subset of S_t could be started.

- Let σ^* represent the minimum σ_j for operation j as a subset of S_t.

The steps of the heuristic are:

Step 1. Let t=0 and begin with PS_t as the null partial schedule. Initially S_t includes all operations with no predecessors.

Step 2. Determine σ^* and the facility m^* on which σ^* could be realized.

Step 3. For each operation j that is in S_t that requires facility m^* and for which σ_j equals σ^*, calculate a priority index according to a specific rule. In the filtered beam search approach, the priority rule used is the shortest processing time rule (SPT). If a tie exists, it is broken arbitrarily. Find the operation with the smallest index and add this operation to PS_t as early as possible, thus creating the next partial schedule, PS_{t+1}, for the next stage.

Step 4. For the new partial schedule created in step 3, update the data as follows: (a) remove operation j from S_t, (2) form S_{t+1} by adding the direct successor of operation j to S_t, and (3) increment t by one.

Step 5. Return to step 2 for the PS_{t+1} created in steps 3 and 4 until a complete schedule is generated.

One advantage of this scheduling heuristic is that it not only determines the overall time of completing all projects, but also generates a partial schedule along the way. When the algorithm finishes, a schedule of operations through the facilities is also generated, which could be given to the R&D manager as a scheduling aid. Thus not only is the makespan computed, but the project sequence that achieves the makespan is also provided.

Example of Scheduling Heuristic

This example will schedule three projects, projects 1, 4 and 9 from the ten project R&D problem, using the scheduling heuristic outlined above.

The project data is as follows:

Project	***Sequence 1***	***Sequence 2***	***Sequence 3***
1	a/6	b/8	c/2
4	b/5	c/4	a/6
9	a/7	c/8	b/4

Note: a/6 means the project is at facility "a" for 6 time units.

Let operation j be represented by "xy", where xy represents project x at facility y, e.g. j=3a would represent the operation where project 3 is serviced at facility a.

σ_j = max {predecessor to j completed, facility needed for j available}; this is the earliest time operation j can be started.

Step 1:

Initialize: t=0, PS_0 = { }

S_0 = {1a,4b,9a}

These are the operations that have no predecessors.

Iteration 1.

Step 2: $\sigma_{1a} = \sigma_{4b} = \sigma_{9a} = 0$

$\sigma^* = \min \sigma_j = 0$

$m^* = a,b$

Step 3:

Since a tie exists, use SPT to break tie and select operation 4b; operation 4b will be complete at time = 0+5 = 5

Step 4:

t=t+1

$PS_1 = \{4b\}$; $S_1 = \{1a,4c,9a\}$

Facility a avail at time=0, b avail at time=5, c avail at time=0.

Iteration 2.

Step 2: $\sigma_{1a} = \max\{0,0\} = 0;$

$\sigma_{9a} = \max\{0,0\} = 0;$

$\sigma_{4c} = \max\{5,0\} = 5;$

$\sigma^* = \min \sigma_j = 0;$

$m^* = a$

Step 3:

Since a tie exists use SPT, select operation 1a, process time = 6

Operation 1a will be completed at time = 0 + 6= 6

Step 4:

t=t+1

$PS_2 = \{4b,1a\}$; $S_2 = \{1b,4c,9a\}$

Facility a avail at time=6, b avail at time=5, c avail at time=0.

Iteration 3.

Step 2: $\sigma_{1b} = \max\{6,5\} = 6;$

$\sigma_{4c} = \max\{5,0\} = 5;$

$\sigma_{9a} = \max\{0,6\} = 6;$

$\sigma^* = \min \sigma_j = 5$

$m^* = c$

Step 3:

Select 4c, process time =4, complete at time=9

Step 4:

t=t+1

$PS_3 = \{4b,1a,4c\}$; $S_3 = \{1b,4a,9a\}$

Facility a avail at time=6, b avail at time=5, c avail at time=9

Iterations 4 - 12.

The process continues in this manner until all operations are scheduled, and the final sequence of operations is:

PS = {4b,1a,4c,9a,1b,4a,9c,1c,9b}

Makespan of this project set = 25 units.

This heuristic is designed to find the sequence of operations that minimizes the overall time to completion of the entire set of projects. Because it is a heuristic, a better sequence of operations may exist for a given set of projects.

Resource constraints in the R&D problem

Several resource constraints will be included in the R&D project problem presented in this example: project budget, research personnel, and two generic resources, resource X and resource Y. The budget constraint is the most common resource constraint used for R&D project selection in the literature (Baker 1964, Baker and Pound 1974, Cetron 1967, Danila 1989). The number of researchers available is an often mentioned, but rarely used resource constraint in the literature. This constraint represents the availability of specially skilled R&D personnel, as opposed to availability of unskilled personnel. The availability of research personnel may be difficult to alter during the planning horizon, because of special training requirements, knowledge requirements, etc. In this model, the special research personnel are treated as a regular constraint, i.e. the number required by the selected projects must be less than or equal to the number available at the beginning of the selection process. This assumption will simplify the scheduling heuristic in the filtered beam search approach. Scheduling heuristics for multiple resource constrained project scheduling exist, but are more complex than the one used in this research. The heuristic utilized in this problem uses the assumption that once a facility is available, all other resources needed are available.

Two generic resources, resource X and resource Y, represent other appropriate resource constraints for a business R&D problem. Examples of other resource constraints might include material resources, other manpower constraints, computer resources, and equipment resources. The choice of two additional constraints is arbitrary, and the addition of more constraints would not adversely affect the filtered beam search approach.

It should be pointed out that no mention has been made about the functional nature of the resource constraints, i.e. linear, nonlinear, exponential, etc. The details will be explained in the section dealing with the filtered beam search algorithm, but it should be pointed out that the filtered beam search algorithm can handle constraints in any mathematical form. This allows the R&D department to define resource utilization formulas in any functional form, and the filtered beam search can handle them without any serious effect on the computational complexity of the algorithm.

The general R&D project selection and scheduling problem can be summarized as follows:

Goals:

Maximize Expected Profit(Portfolio);

Minimize Makespan(Portfolio);

Maximize Probability of Success(Portfolio);

Subject to:

Resource X usage(Portfolio)	< =	Resource X capacity
Resource Y usage(Portfolio)	< =	Resource Y capacity
Number of researchers(Portfolio)	< =	Number of researchers available
Budget usage(Portfolio)	< =	Budget available

For this problem, the objective functions and resource constraints are linear functions.

Heuristic search techniques

Because it is infeasible to use exact methods for project selection and scheduling, a heuristic approach must be used. Search techniques explore alternative solutions, until the optimal solution is obtained. One obvious search technique is complete enumeration, where every alternative is explored and the optimal solution is the alternative that best meets the decision criteria. Complete enumeration is feasible for small problems, but in R&D project selection all combinations of projects must be considered, and the problem size grows quickly as the number of projects considered increases. The number of possible combinations of projects is 2^n, where n is the number of projects. For n equal to 10 projects, the number of possible combinations is 1,024. Increasing n to 50 projects results in 1.125 thousand trillion different combinations! Fifty projects is a realistic number of projects to consider in a business, and the sheer number of possible project combinations makes any type of selection algorithm that is based on complete enumeration impractical.

Researchers in Artificial Intelligence (AI) have developed several search routines to deal with this combinatorial explosion in problem size. The techniques that will be discussed here are routines that search for the goal or objective in the solution space, rather than attempting to find a shortest, or least cost, path through the solution space. In the R&D project selection problem, the optimal project set is not known ahead of time, so some of these techniques are not directly applicable to the problem, but are presented here to provide a background to the filtered beam search approach. The techniques are used to find paths from starting positions to goal positions when the length of the discovered paths is not important (Winston 1984). One interesting characteristic of the R&D project selection problem that helps define the solution space is the zero-one nature of the decision variables. In the R&D project selection, the choice is to either include a project in the portfolio, or not include it. The fact that the decision variables are well defined will help when generating the solution space.

The general problem in R&D project selection is to choose a set of projects that best meets the objectives of the R&D department. The optimal answer obviously is not known ahead of time. The search technique must explore the solution space, evaluate the alternative solutions, and determine which of the alternatives is the best answer. The search techniques presented here use different strategies to explore the solution space and find the optimal solution.

The solution space for the R&D project selection space can be visualized as a *search tree*. The search tree begins at level zero, which represents the beginning of the search process. In the R&D project selection problem, this is the starting point, where no projects have been selected. The next level, one, consists of all of the candidate projects being considered one at a time. Level two would consist of all combinations of two projects together. Figure 3.1 shows an abbreviated search tree for the ten project problem. At level one, all of the projects are shown. Level two is shown only for the combinations of two projects that include projects one and two. Note that there are only nine different combinations of projects that include project one. It would not makes sense to show a branch in the search tree that shows project one combined with itself. Also notice that there are only eight different combinations of two projects shown in the search tree that include project two. It would be redundant to include projects two and one together at this level, since the level already contains projects one and two together, as shown on the branches emanating

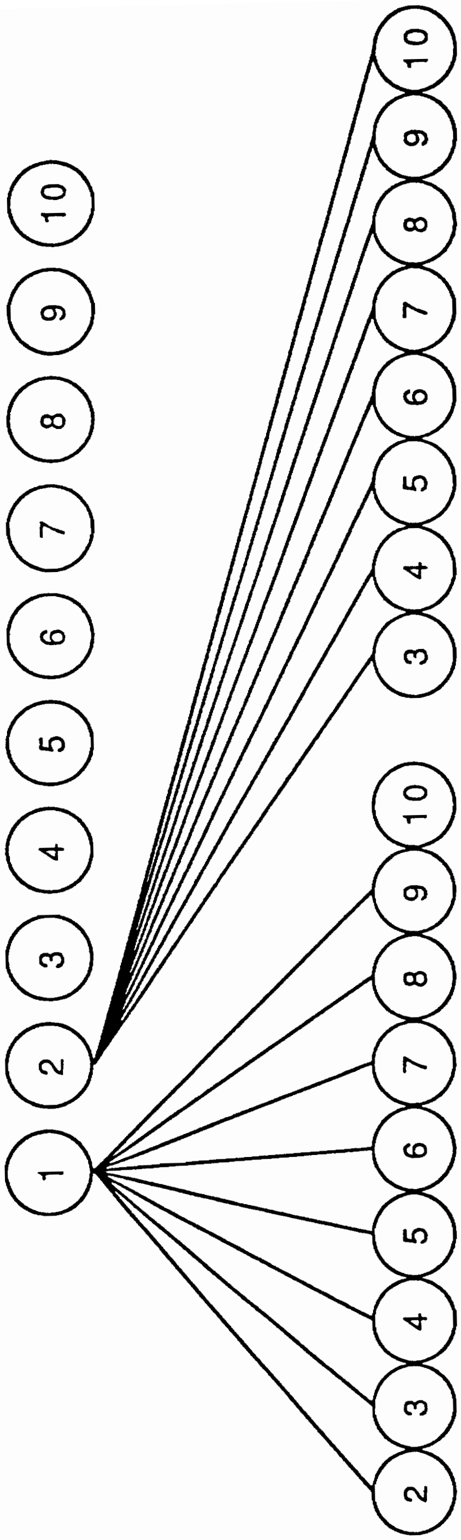

Figure 3.1. Search tree for 10 project problem

from project one at level one. The rest of level two would be constructed similarly, with branches emanating from all ten projects at level one.

The full search tree could be constructed by continuing to create levels downward, with branches emanating from the nodes at the previous level. These branches extending downward provide the paths for the different search techniques to explore, representing a set of projects being considered for selection. Extending the branches is analogous to adding projects to the subset of projects under consideration. For example, level one of the search tree in Figure 3.1 represents considering the projects one at a time. In most cases, more than one R&D project can be selected, and the solution space grows by extending the search tree by one level, i.e. considering the projects two at a time. The search tree can be extended level by level, i.e. more projects being considered together, until the branches cannot be extended due to various restrictions on the problem. The restrictions are normally the resource constraints for the R&D project selection problem.

As the problem size increases, the size and complexity of the search tree will increase dramatically; the size of the search tree grows exponentially with problem size. Thus, it is important for the search technique designed for the R&D project selection problem to be efficient in exploring the search tree, in order to handle realistic problem sizes.

Each search technique presented in this section explores the search tree in a different manner while searching for an optimal path through the search tree. The first three techniques presented, depth-first search, breadth-first search, and the best-first search, assume the goal is known ahead of time, and are designed to find a path to the goal node, but do not attempt to find an optimal path, while the last technique, beam search, is a directed search technique and attempts to find the optimal path through the search space. Because the beam search is a heuristic, it is not guaranteed to find the optimal path.

Depth-first search

The depth first search technique simply selects a starting point, in this case a single project, and dives into the search tree, and continues to add projects to the branch until all projects are added to the branch or until no more projects may be added, because of project constraints. In a realistic R&D problem, project constraints will prevent all projects from being selected, so in the depth first search the branches terminate due to project constraints. Once a branch terminates, the set of projects along the branch are evaluated according to the various objectives for the R&D selection process. The depth first search restarts at the source node, and dives into the search tree along a different branch, and repeats the process of adding projects until the branch terminates. This entire process continues until all branches have been searched. The depth first search is typically used for the case where the goal is known ahead of time, and the process normally terminates once a path to the goal node is found. Obviously, since the optimal answer is not known ahead of time in the R&D problem, the depth first search technique would not be practical.

Breadth-first search

The breadth-first search is another search technique that is used to explore alternatives. The breadth-first technique pushes uniformly down into the search tree, one level at a time, generating alternatives and searching for the optimal solution.

In this technique, the starting points are the individual projects, and the next level would be all possible combinations of two projects. Every feasible branch is evaluated at this level, with infeasible branches being "pruned", or discarded for further exploration. The next level would be all feasible combinations of three projects emanating from the previous level. The search continues to push down into the search tree until no more feasible branches can be generated. Obviously, the breadth-first search technique suffers from similar drawbacks to those of the depth-first search, i.e. because the optimal solution is not known ahead of time the technique would not be practical for R&D project selection.

Best-first search

The best first search technique is similar to the depth-first search, but the order in which the branches are explored is determined by using an evaluation function relating to the objective of the search. In the best-first search, the source nodes are evaluated, and the node that is the best according to the evaluation is searched first, in the same manner as the depth-first search. Once the branch terminates, if the goal node is not found, the search restarts. The search backtracks to the best unexplored node, and resumes by diving down that branch. The search process continues until the goal node is found, or the number of branches to search is exhausted. Again, because the goal for R&D project selection is not known ahead of time, the best-first search would not be appropriate for the R&D problem.

These three techniques are mentioned in order to build the foundation for the technique used in this dissertation, the filtered beam search which is variation of the beam search technique, and is related to the three search techniques mentioned above.

Beam search

The beam search technique is similar to the breadth-first technique, in that it proceeds level by level. Unlike the breadth-first technique, however, the beam search does not blindly continue to press on down into the search tree, searching for feasible branches. The beam search pauses at each level, and uses an evaluation function to decide on which branches to continue the search. Consequently, the beam search keeps the number of branches searched to a manageable number. This is important in R&D project selection because the number of possible branches, i.e. the number of possible combinations of projects, can become extremely large. The basic idea behind the beam search is to use the evaluation function to determine which branches, i.e. project subsets, seem to be doing the best according to the objectives, and only continue to search along those branches, ignoring the rest. Because the search does eliminate branches from consideration the possibility of missing the optimal solution exists. The trade-off is the computational efficiency of the search versus the exactness of the optimal solutions obtained from complete enumeration search techniques.

The beam search consists of the following steps:

Step 1. Begin by generating source nodes for the search tree. In the R&D project selection problem, this would be all feasible projects being considered.

Step 2. Evaluate every node at the current level, using an evaluation function that represents the objective of the search.

Step 3. Select the *w* best nodes according to the evaluation function. Here, *w* represents the beamwidth of the search, i.e. how many branches down through the search tree that will be explored. These are the beams of the search.

Step 4. For each current branch, generate the next level by adding all nodes that are not currently on the branch. Do this for each beam.

Step 5. Check all branches for feasibility according to the problem constraints. Any infeasible branches are eliminated from consideration. If there are no feasible branches left, the search is complete.

Step 6. Return to step 2, and continue the process until the search terminates due to the problem constraints, i.e. no more feasible branches can be added.

When the beam search is completed, there will be *w* solutions, one for each beam in the search. These solutions are evaluated according the evaluation function, and the solution with the highest value according to the evaluation function is selected.

The basic idea in the beam search is to search a limited number of solution paths in parallel, and the intent is to search the solution space quickly. The beam search does not backtrack, and as a result the beam search is not guaranteed to provide optimal solutions since there is no guarantee that the node on the optimal branch will always be selected. However, this possibility is normally very slight, and increasing the number of paths explored, i.e. the beamwidth of the search, improves the quality of solutions obtained using the beam search method. It has been shown in previous research studies (Ow and Morton 1989) that the beam search can find optimal, or near optimal solutions with relatively small beamwidths. An example of the beam search, shown in Figure 3.2, follows.

Beam search example

Figure 3.2 provides a generic example of the beam search, and details concerning evaluation functions, constraints, etc., will be omitted. Five projects are being considered for selection. The process begins at level one, where all five projects are evaluated according to an evaluation function. After the evaluation process, the best two projects are selected as the beams for the next step. In this example, projects 2 and 5 are selected.

Level two is created by combining project 2 with projects 1, 3, 4, and 5. Note that project 2 is not combined with itself. Level two is completed by combining project 5 with projects 1, 3, and 4. Project 2 is not combined with project 5 on this beam since the combination of projects 2 and 5 was already created when project 2 was combined with project 5.

The seven nodes at level two represent the "two-project" combinations being considered for selection. The project combinations are first screened, checking for feasibility according to the problem constraints. In this case, all project combinations at level two are feasible. The best two nodes at this level are selected, using the evaluation function. In Figure 3.2, nodes 4 and 1 are selected at level two. These two nodes represent the project combinations {2,4} and {5,1} respectively.

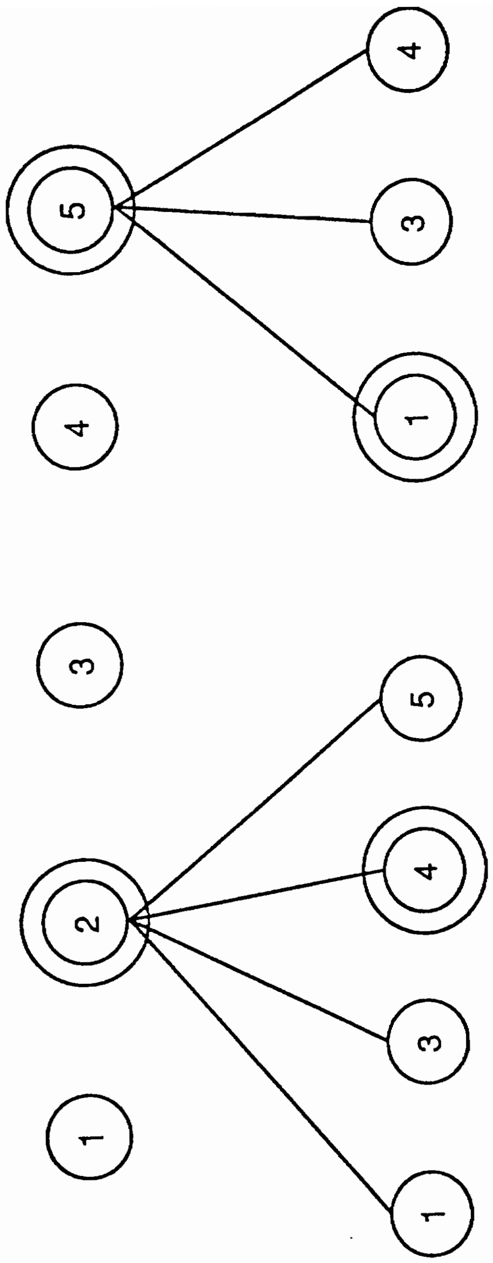

Figure 3.2. Beam search example

The process continues in this manner until the search is terminated when new levels cannot be created, i.e. the problem constraints prevent the addition of any more projects to the beams. When the beam search process is complete, there will be two *beams* or project sets. These are the "best" solutions according to the beam search. To determine the overall best solution, the two project sets can be ranked using the evaluation function.

Filtered beam search

Background

The beam search approach has only been discussed in the literature since 1976, and has been used in the areas of speech recognition (Lowerre 1976), image recognition (Rubin 1978), and job shop scheduling (Fox 1983, Ow and Morton 1989). The beam search is not a specific algorithm, but rather an approach to to searching extremely large solution spaces. There are many variations of the beam search approach, but they all share a common feature: the search method only explores a limited number of partial solutions in parallel at any point in time (Ow and Morton 1989).

There are very few examples of the beam search approach in business applications. Leu et al. (1992) use a variation of the beam search approach for resource allocation in a decentralized organization. In this paper, the authors use the beam search in conjunction with two other AI techniques, using the beam search to select a list of potential projects, and using the other techniques to evaluate the projects.

The beam search works well when there is only one objective for the search problem, or if multiple objectives can be combined into one evaluation function. However, the R&D project selection problem has been described as having multiple criteria with no common measure. This makes it very difficult to create an evaluation function that adequately captures the decision criteria for the R&D problem. Goal programming is often used for R&D project selection because it can handle multiple goals with different measures. In goal programming, the goals are prioritized, with the most critical goal for R&D project selection given a high weight, or priority, and the other goals are weighted accordingly. The filtered beam search uses an approach that is analogous to goal programming.

Description of the filtered beam search process

In the beam search described earlier, at every level the feasible branches, or project subsets, are evaluated according to an evaluation function, the best *w* of those are retained for further exploration, and all other branches are eliminated from the search. In the filtered beam search the evaluation process is modified slightly. Because an evaluation function cannot be written to encompass all the desired goals, the evaluation process is done in stages. The various goals are prioritized, and the most important goal is used for the final evaluation and selection of branches. The other goals are used to filter out branches that perform poorly according to the subgoals of the search. The idea is to utilize these subgoals in the process to eliminate these obviously poorer branches, leaving only the better branches for the final evaluation step.

Note that following this filtering process could result in missing a good solution. If a solution with very poor performance at a lower level goal is filtered early in the filtered beam search, and if that solution in fact has a very high value for the primary goal, then it is possible that the solution would not be available for selection at the lower levels of the filtered beam search. An alternative procedure that could be employed is filtering based on the primary goal or some set of higher level goals, followed by selection of the final solution based on the lower level goals, while insuring that achievement of the higher level is not compromised.

Deciding which goal should be used for the final evaluation process, and which goals to use for the filtering process, is the responsibility of the R&D manager. In this research, the expected profit goal will be used for the final evaluation step. Expected profit is the most popular goal for R&D project selection models in the literature, so it would seem appropriate to select it as the most critical goal. The filtering goals will be project scheduling and probability of portfolio success. In the filtered beam search approach, the projects will be filtered first on the probability of success, and then on the project scheduling goal, makespan of the project subset.

The filtered beam search uses the scheduling heuristic to rank the projects in the second filtering step. The first filter eliminates those projects subsets that perform poorly according to probability of success. The second filter, the project scheduling filter, eliminates those project subsets that perform poorly according to the scheduling objective. After the projects have been through the second filter, the projects are selected according to the final objective, expected profit for the project subset.

The first filter will be a "loose" filter, with a fairly large filterwidth. The second filter will be a little "tighter", with a filterwidth smaller than the first, but still larger than the beamwidth. The idea is to pare down the possible branches step by step, leaving a set of branches for the final evaluation stage. Because the filtered beam search is a relatively new technique, literature providing guidance for filterwidth and beamwidth selection is virtually nonexistent. The selection of beamwidths and filterwidths appear to be very problem specific (Leu 1992). In this dissertation several different combinations of beamwidths and filterwidths will be investigated for both the ten project problem and the fifty project problem.

The filtered beam search for the R&D project selection problem consists of the following steps.

Step 1. Start by generating the source nodes as starting point for the search. In the R&D project selection problem, there are as many source nodes as there are projects under consideration. For example, in the ten project problem there will be ten source nodes, and for the fifty project problem, there will be fifty source nodes at the beginning of the process. These are the current nodes for evaluation in step 2.

Step 2. Evaluate the nodes at the current level using the lowest priority objective for the problem. This is done using an evaluation function for the goal, and the projects are ranked according to this goal, from best to worst. At this point, only the best *f1* nodes are retained, with *f1* representing the filterwidth for the first goal considered. The filterwidths for the less important goals are typically fairly wide in comparison to the beamwidth of the search. The idea at this point is to only eliminate those nodes, or project subsets, that perform very poorly according to the lesser goals. This will allow the lesser goals to have an impact on the process, but not at the expense of the more important goals. In the R&D example presented here, the first filtering goal used is the group probability of success. This filter will eliminate from consideration those project sets that have the lowest probability of success index at this point, but does not eliminate them from the process completely.

Step 3. The next most important goal is used to continue the filtering process. The *f1* best nodes resulting from the first filtering process are now evaluated according to an evaluation function that mathematically represents the next filtering goal goal. The projects are now ranked again according to the new evaluation function, and the best *f2* are kept. The second filterwidth, *f2,* is typically smaller than *f1,* but is larger than the beamwidth of the search. This set of *f2* projects is sent on to the next step, the final evaluation and selection step. The goal used for the second filter is the project scheduling goal, and the evaluation is not performed using a simple function, but instead is the mathematical result of the scheduling heuristic that determines the makespan of the node, or project subset, under consideration at this level.

Step 4. The nodes passing through the filtering steps are now evaluated according to the most important goal, expected profit. The surviving nodes, or project subsets, are evaluated and ranked according to their expected profits, and the best *w* nodes are retained, and the remaining branches are eliminated from future consideration. It should be pointed out that only the branches are eliminated from consideration, not the projects on those branches. The best *w* nodes, or project subsets, are now the active branches for the search.

Step 5. The next level of the search is generated by using the active branches selected in step 4. Each active branch represents a feasible subset of projects under consideration, and the *w* active branches represent the "best so far." Now, the search tree is expanded by taking each active branch, and extending it by adding projects that are not currently in the subset, generating a new node for each new project under consideration. This is done for each active branch. Note that this is how the filtered beam search simplifies the search through the tree. At each level, the maximum number of nodes considered is bounded by *(wn)*, where n is the number of projects, and as the search continues, the number of nodes generated at each level actually decreases because the number of projects remaining to be considered decreases.

Step 6. The nodes generated are now checked for feasibility. The four resource constraints in our model are used to determine if the node violates any one of the constraints. This is done by determining which projects are on the branch being checked, and computing the resource utilizations. In this case, all four resource constraints are linear, so the computation is straightforward. If the project subset violates any constraint it is eliminated from further consideration. This process is repeated for every node generated in step 5. Note that other forms of constraint functions could be substituted very easily at this point. If the constraint can be put in a mathematical form, then the project subset resource utilization can be computed at this point, and compared with the resource capacity. This modularity is one advantage of the filtered beam search, and the use of nonlinear constraints does not adversely affect the algorithm.

Step 7. If any feasible nodes remain, the process returns to step 2 and repeats using these nodes. If no feasible nodes remain, the process is complete. The filtered beam search algorithm will generate *w* different solutions, and these are the "best" solutions remaining after the entire selection process. Determining which one of these is the optimal solution at this point is at the discretion of the R&D manager if there is no clear cut "best" solution according to the goals. Often, one solution will be better according to one goal, but second best according to the another goal. In any case, each of the solutions obtained using the filtered beam search algorithm can be considered as very good, if not optimal. When using multiple goals with no common measure solutions become "satisfactory" rather than optimal.

The filtered beam search approach was written in Turbo Pascal, version 4.0. Two programs were written, one for problems up to ten projects in size, and one for problems up to fifty projects in size. The reason for having two programs is that the program for small problems sizes, ten projects or less, runs much quicker than if the larger program is used for the same problem. The ten project program consists of 1,188 lines of code, and the fifty project program consists of 1,742 lines of code, with the primary difference being the size of the arrays used in each program. The ten project problem can be solved using either program, and the resulting solutions are identical. The programs were run on a model DC-2010 Leading Edge personal computer, and the programs generated solutions in approximately twenty minutes. It is difficult to determine exact CPU times for the software used in this research, so solution times are given as actual elapsed time. The Turbo Pascal code for these two programs and the complete enumeration program is given in appendix 1.

Limited illustration of filtered beam search approach

In this example, two iterations of the filtered beam search approach will be presented, using the ten project R&D problem. The data for the ten project problem is given in Table 3.1. The beamwidth for the example is two, and the two filterwidths

Table 3.1. Project information for 10 project problem

Project	*Budget*	*Profit*	*Probability of success*	*Resource X*	*Resource Y*	*Number of Researchers*
1	10	20	0.8	3	6	1
2	5	8	0.7	5	3	2
3	15	35	0.8	4	7	2
4	14	25	0.7	7	9	1
5	19	30	0.9	2	3	3
6	12	20	0.85	9	2	1
7	9	19	0.8	8	8	4
8	20	35	0.9	12	6	3
9	18	50	0.6	6	9	4
10	15	40	0.7	5	10	3

Project	*Sequence 1*	*Sequence 2*	*Sequence 3*
1	a/6	b/8	c/2
2	a/7	c/3	b/0
3	b/3	a/6	c/0
4	b/5	c/4	a/6
5	c/8	a/2	b/0
6	c/9	b/1	a/0
7	b/3	c/10	a/0
8	b/10	a/5	c/6
9	a/7	c/8	b/4
10	a/12	b/8	c/0

are ten and six respectively. The makespan computation is performed using the scheduling heuristic described earlier.

Iteration one:

- Start at by creating level one of the search tree, which consists of all ten R&D projects.

- Perform the first filtering step, which is done based on portfolio probability of success. Since the ten project portfolios consist of only one project each, the portfolio probability of success index is the individual probability of success. The first filterwidth is ten, so no projects will be filtered out at this step.

- The ten projects passed on by the first filter are now evaluated according to the second filter objective, project scheduling. The makespans for the ten candidate projects are shown in Table 3.2. The best six are retained by the filter, and these are projects 2, 3, 4, 5, 6, and 7. These six projects are passed on to the selection step.

- The two best projects are now selected from the six candidates. The two projects with the best expected profit are projects 3 and 5. The expected profits for the six projects are given in Table 3.2. The two projects are the beams for the next iteration in the filtered beam search approach, as shown in Figure 3.3.

Iteration two:

- Level two of the search tree is created by combining project 3 with the other nine projects, and project 5 with the other nine projects as well. The candidate project

Table 3.2. Data for filtered beam search example; iteration one.

Information for the filtered beam search example; Iteration one.

Project	P(success)	Makespan	Expected Profit
1	0.8	16	
2	0.7	10	5.6
3	0.8	9	28
4	0.7	15	17.5
5	0.9	10	27
6	0.85	10	17
7	0.8	13	15.2
8	0.9	21	
9	0.6	19	
10	0.7	20	

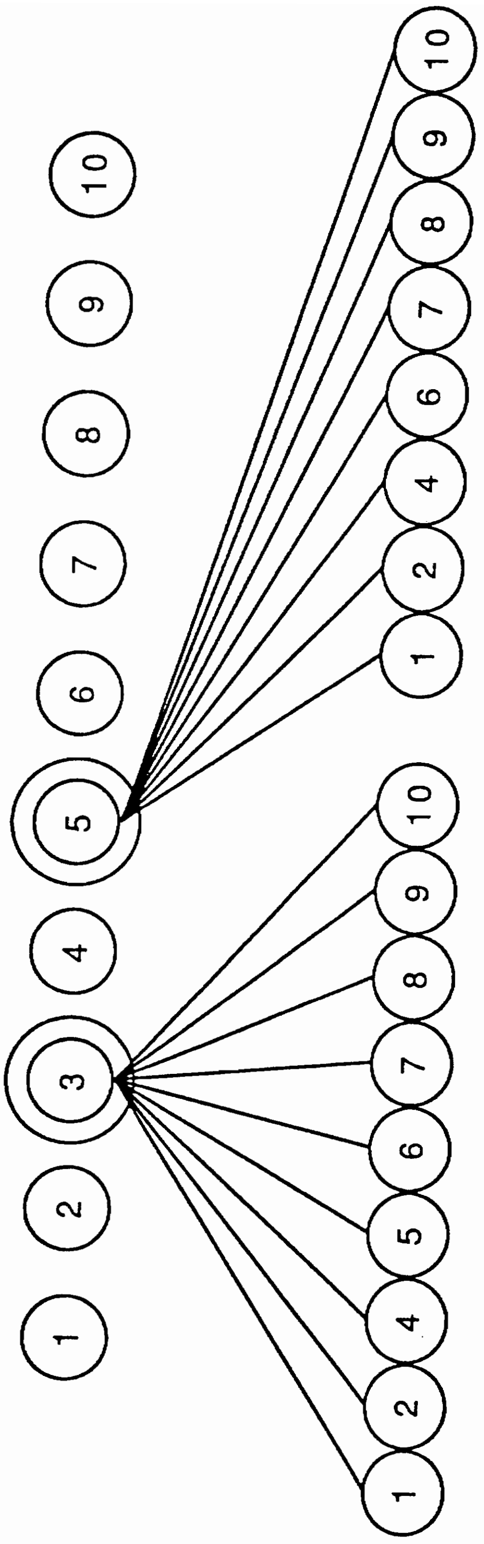

Figure 3.3. Filtered beam search illustration; Iteration one

Table 3.3. Data for filtered beam search example; iteration two.

Information for filtered beam search example; Iteration two.

Project Set	P(success)	Makespan	Expected Profit
{3,1}	1.6	16	44
{3,2}	1.5		
{3,4}	1.5		
{3,5}	1.7	11	55
{3,6}	1.65	10	45
{3,7}	1.6	16	43.2
{3,8}	1.7	24	
{3,9}	1.4		
{3,10}	1.5		
{5,1}	1.7	16	43
{5,2}	1.6	11	32.6
{5,4}	1.6		
{5,6}	1.75	18	
{5,7}	1.7	18	
{5,8}	1.8	21	
{5,9}	1.5		
{5,10}	1.6		

sets are shown in Table 3.3. Note that there is no project set {5,3}. This is because project set {3,5} is the same as project set {5,3} in the filtered beam search approach.

- The seventeen candidate project sets are checked for feasibility. The four resource constraints are checked, and if any project set uses too much of any resource, it is eliminated from consideration at this level. In this case, all project sets are feasible.

- The projects are now passed on to the first filter. The ten best project sets according to the portfolio probability of success index are kept, and passed on to the next filter step: {3,1}, {3,5}, {3,6}, {3,7}, {3,8}, {5,1}, {5,2}, {5,6}, {5,7}, and {5,8}. The data for probability of success is given in Table 3.3. In the event of ties during the filtering steps, the convention used for choosing project sets is that the leftmost project sets in the search tree are selected first.

- The ten project sets are passed on to the next filter step, and the best six are chosen according to makespan. The makespans for the ten project sets are given in Table 3.3. The best six project sets according to makespan are: {3,1}, {3,5}, {3,6}, {3,7}, {5,1}, and {5,2}. These six project sets are now passed on the selection step.

- The two best project sets are selected from the six candidate project sets. The two best are project sets {3,5}, and {3,6}. These two project sets now are the beams for the next iteration, as shown in Figure 3.4.

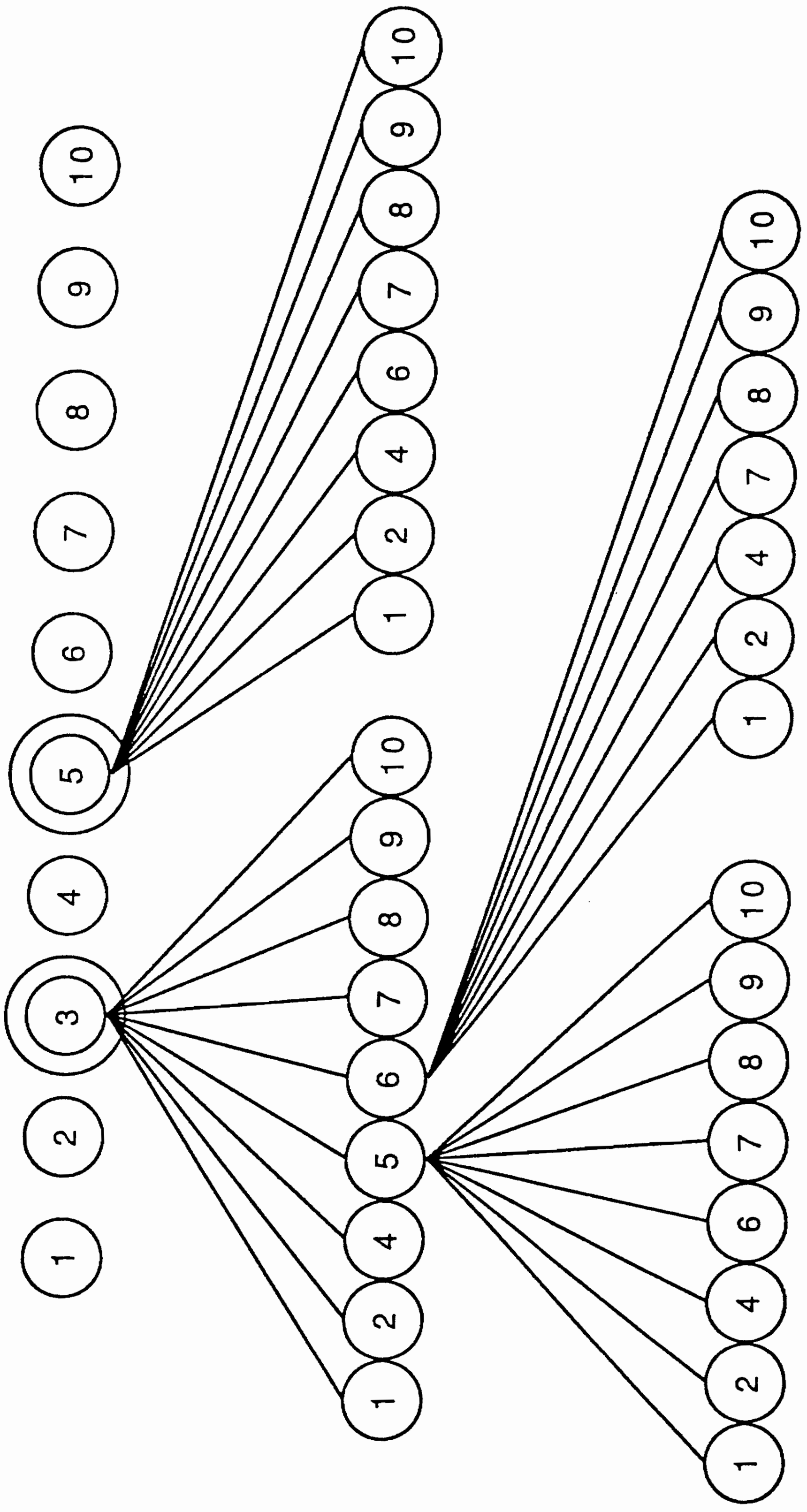

Figure 3.4. Filtered beam search illustration; Iteration two

The process continues, generating new levels of the search tree, checking for feasibility, filtering and selecting project sets. Note that the next level of the search tree emanates only from the beams as determined by the selection step. This feature of the filtered beam search greatly reduces the scope of the search tree, allowing the approach to be used on large project sizes, and completed in reasonable computation time. This example will be completely solved in the following chapter.

Summary

This chapter has outlined the filtered beam search approach, in the context of a sample R&D project selection problem. Project scheduling was included in the project selection process via a filtering process in the filtered beam search approach. The general R&D problem was defined, including the goals (1) expected profit of the project portfolio, (2) the makespan of the portfolio, and , (3) the probability of success for the portfolio. The project constraints were also outlined: (1) budget, (2) number of researchers, (3) resource X, and, (4) resource Y.

Heuristic search techniques were reviewed, and the development of the filtered beam search approach was presented in relation to these heuristic techniques. The steps of the filtered beam search approach for R&D project scheduling were presented, along with an abbreviated example of how the approach works. Project makespan was defined as the project scheduling goal for the R&D problem in this dissertation, and the heuristic used for computing makespan was presented in this chapter, along with a numerical example.

In the following chapter, entitled "Model Solution and Results", the two example R&D project selection problems will be solved using the filtered beam search approach, and the solutions obtained will be compared with solutions obtained from traditional R&D project selection models.

Chapter 4: Model Solution and Results

Results and interpretations

The filtered beam search approach described in chapter 3 is used to solve the R&D project selection and scheduling problem for ten projects and fifty projects. The program provides several candidate solutions, depending on the beamwidth selected for the search, the projects selected for each candidate solution, as well as the expected profit, makespan and the portfolio probability of success.

The filtered beam search approach is programmed in Turbo Pascal 4.0, encompassing the steps outlined in chapter 3. The complete enumeration model is also programmed in Turbo Pascal 4.0, for the ten project problem only. This program used the same procedures as the filtered beam search for calculating expected profit, makespan and portfolio probability of success. This is done so that the comparison of results for the two methods would be valid. The details of the complete enumeration model are given in the section entitled "Complete enumeration model" in this chapter.

The zero-one integer linear programming model is formulated for the ten project problem, and solved using an integer linear programming software package, QSB+. The results for this model are given in the section entitled "Math programming model" in this chapter. All three approaches, the filtered beam search, complete enumeration and zero-one integer linear programming used the same resource constraints and resource capacities.

The fifty project problem is solved using the filtered beam search approach programmed in Turbo Pascal 4.0. The filtered beam search program for the fifty project problem uses a slightly different program than used for the ten project problem; the program for the fifty project filtered beam search can handle any number of projects up to fifty projects, and the ten project problem could be solved using the fifty project Turbo Pascal program. A different program for the smaller problem size is used because it is more efficient to use a program specifically designed for the ten project problem. The programs are identical except for the size of storage arrays used, and the two programs give identical solutions for the ten project problem.

Experimental Results: ten project problem

The ten project problem is solved using the filtered beam search approach algorithm, programmed in Turbo Pascal 4.0. The model uses the three goals, maximize expected profit, minimize makespan, and maximize portfolio probability of success, subject to the following resource constraints: budget constraint = 80 units, number of researchers = 10, resource X constraint = 30 units, and resource Y constraint = 35 units.

The ten project problem is solved using two different beamwidths, a beamwidth of two and a beamwidth of three, a first filterwidth equal to ten, and a second filterwidth equal to six. The selection of beamwidths and filterwidths tends to be very problem specific (Leu 1992). Looking at the limited literature on beam searches, a beamwidth of three appeared to give satisfactory results in job shop scheduling problems using 25 jobs (Ow and Morton 1989), so a beamwidth of three was also selected for this problem.

The choice of filterwidths is more subjective, since literature for filtered beam searches is limited, and no examples were found that used more than one filtering goal at each level of the filtered beam search. Some researchers have found that increasing the filterwidths too much can adversely affect the filtered beam search approach by causing a deterioration in the results (Ow and Morton 1989). The best filterwidth for the research from the literature appears to be in an interval between one-third and two-thirds of the problem size, when one filter is used at each level of the search. The filter used in previous research using the filtered beam search approach is analogous to the second filter used in the filtered beam search approach presented in this research, the project scheduling filter, and using this as a guideline, the second filterwidth is selected at two-thirds of the problem size, or a filterwidth of six.

There is little or no guidance in the literature for the selection of the first filterwidth for the filtered beam search approach when using more than one filter. Logically, it would seem that the first filterwidth needs to be larger than the second, otherwise the second filter will have no effect on the search. The idea of the filter is eliminate to the poorer nodes at each level of the search, and the maximum number of nodes possible at any given level is bounded by the beamwidth times the number of projects. For the ten project problem, and a beamwidth of three, the maximum number of nodes would be thirty. The first filterwidth should be related to the number of nodes generated at the various levels during the filtered beam search approach. If the first filterwidth is too close to the number of nodes generated at the various levels, then the filter goal will have little effect on the solutions. However, if the first filterwidth is small relative to the number of nodes generated, then the filter goal will have a significant effect on the solutions generated by the filtered beam search approach. A first filterwidth of ten is used as a starting point, because it is one third of the maximum number of nodes possible at any level of the search.

Two solutions for the ten project problem using the filtered beam search approach are given in Table 4.1, one for a beamwidth of two and another for a beamwidth of three. The filtered beam search approach will provide an answer along each beam, or branch, with number of solutions equal to the beamwidth of the search. At this point, if a clear cut optimal solution does not exist, a subjective choice needs to be made. In the R&D project selection problem, the R&D management will probably decide which project set best meets the needs of the organization.

In the ten project example problem, there seems to be a clear cut choice, the set of projects {1,3,5,6,8}. This set has the best expected profit, 119.5 units, the lowest makespan, 25 units, and the best portfolio index of success, 4.25, of the solutions provided by the filtered beam search. The other solutions given by the filtered beam search approach would probably be good alternatives, but do not perform quite as well according to the goals. The filtered beam search is not guaranteed to provide the optimal solution, although in R&D project selection the definition of "optimality"

Table 4.1. Results for ten project problem, f1 = 10, f2 = 6, w = 3

Solutions from the filtered beam search approach

Beamwidth = 3	Beamwidth = 2
Project Set 1 = {1,3,5,6,8}	Project Set 1 = {1,3,5,6,8}
Expected Profit = 119.5 units	
Makespan = 25 units	
Portfolio Success = 4.25	
Project Set 2 = {3,4,5,6,10}	Project Set 2 = {3,4,5,6,10}
Expected Profit = 117.5 units	
Makespan = 26 units	
Portfolio Success = 3.95	
Project Set 3 = {1,3,5,6,10}	
Expected Profit = 116 units	
Makespan = 34 units	
Portfolio Success = 4.05	

is subjective since the objectives of the R&D project selection and scheduling process often have no common measure, making it difficult to determine if one solution is better than another.

The filtered beam search solutions from the ten project problem will be compared with the results of the complete enumeration model for the same problem, and with the results obtained from a zero-one integer linear programming model. By comparing results, the effectiveness of the filtered beam search approach can be evaluated, and the effect of including project scheduling in the project selection process can be investigated.

Complete enumeration model

One method of choosing an R&D project portfolio when there are a small number of projects is to look at every combination of R&D projects and evaluate them according to the objectives of the problem. For larger problems complete enumeration would be impossible, however, the ten project example can be solved using complete enumeration, which will enable us to determine if a better solution exists than the solution obtained by the filtered beam search approach. For the ten project problem, to completely enumerate all possible project combinations requires the generation of 2^{10}, or 1,024, project combinations. These 1,024 project combinations represent every possible, though not necessarily feasible, combination of R&D projects.

The complete enumeration model starts by evaluating every project by itself, then all combinations of two projects, all combinations of three projects, etc. up to all ten projects considered together. Each project combination is checked for feasibility by determining the resource requirements of the project combination, and compares it to the resource constraints used in the ten project problem. The budget requirement, number of researchers required, and the amounts of resource X and resource Y required for each project combination are calculated, and if the project combination violates any one of the resource constraints it is eliminated from the search process.

The resource constraints for the complete enumeration model are the same used for the filtered beam search approach. Once all project combinations are checked for feasibility, the feasible project sets are then evaluated according to the expected profit goal, the project scheduling goal, and the portfolio probability of success goal, in the same manner the filtered beam search calculates each of these goals. The fifteen project sets with the highest expected profit are listed in Table 4.2, and are ranked in descending order according to expected profit.

The three project sets selected by the filtered beam search approach are noted with an asterisk in Table 4.2. Note that the project set with the best expected profit, project set {1,3,4,5,8}, was not selected by the filtered beam search as one of the best solutions. This is probably due to the fact that this project set has a lengthy makespan, making it a less attractive project set according to the the project scheduling goal using the filtered beam search approach. The filtered beam search did select the second, third and fifth best project sets, ranked according to the expected profit goal. However, observing the top fifteen project sets generated by the complete enumeration routine, there does not appear to be any single project set that is "better" based on the three goals/objectives for the project selection process than the project sets selected by the filtered beam search. The project set {1,3,5,6,8} is the

Table 4.2. Best 15 solutions using complete enumeration

Project Set	Expected Profit	Makespan	Portfolio Success Index
{1,3,4,5,8}	120	37	4.10
{1,3,5,6,8}	119.5	25	4.25 *
{3,4,5,6,10}	117.5	26	3.95 *
{1,3,4,5,10}	116.5	40	3.90
{1,3,5,6,10}	116	34	4.05 *
{1,2,3,4,5,6}	111.1	30	4.75
{1,3,4,6,9}	108.5	31	3.75
{1,4,5,6,9}	107.5	35	3.85
{3,4,8,9}	107	30	3.00
{1,3,4,6,10}	106.5	32	3.85
{1,4,5,6,10}	105.5	26	3.95
{1,3,8,9}	105.5	31	3.10
{1,3,4,5,6}	105.5	23	4.05
{3,4,8,10}	105	29	3.10
{3,6,8,10}	104.5	29	3.25

second best project set according to profit, and when compared to the project set with the best expected profit, it has a much lower makespan, 25 units versus 37 units, and a higher portfolio probability of success, 4.25 versus 4.10. In fact, the project set {1,3,5,6,8} only sacrifices half of a unit of profit to the project set {1,3,4,5,8}.

The purpose of the complete enumeration is to identify feasible projects sets, and determine if the filtered beam search overlooked any project sets that would provide a better solution to the ten project problem. The filtered beam search, using a beamwidth of three, and filterwidths of ten and six respectively, appears to efficiently generate solutions, and these solutions appear to be of good quality.

Math programming model

In chapter 2 it was pointed out that math programming models cannot handle project scheduling goals because of the complexity of the project scheduling problem, which makes a direct comparison of the math programming approach and the filtered beam search approach impossible. However, the effects of including the project scheduling goal into the project selection process can be investigated by formulating the ten project problem as a zero-one integer goal programming and solving it using the same goals and constraints, minus the scheduling goal, and observing the results. Ideally, the solutions obtained would be different, with the filtered beam search obtaining a "better" solution than the math programming model, because the filtered beam search uses all three goals, and the math programming model uses only the expected profit and portfolio success goals.

The ten project problem was formulated as shown in Table 4.3. The variables X_1, X_2, etc., are zero-one variables that represent the selection of the projects, i.e. a

Table 4.3. Math programming formulation and solution.

Minimize $D_1^- + D_2^-$

Subject to:

(Expected Profit)
$16X_1 + 5.6X_2 + 28X_3 + 17.5X_4 + 27X_5 + 17X_6 + 15.2X_7 + 31.5X_8 + 30X_9 + 28X_{10} + D_1^- - D_1^+ = 500$

(Portfolio Success)
$0.8X_1 + 0.7X_2 + 0.8X_3 + 0.7X_4 + 0.9X_5 + 0.85X_6 + 0.8X_7 + 0.9X_8 + 0.6X_9 + 0.7X_{10} + D_2^- - D_2^+ = 10$

(Resource X)
$3X_1 + 5X_2 + 4X_3 + 7X_4 + 2X_5 + 9X_6 + 8X_7 + 12X_8 + 6X_9 + 5X_{10} \leq 30$

(Resource Y)
$6X_1 + 3X_2 + 7X_3 + 9X_4 + 3X_5 + 2X_6 + 8X_7 + 6X_8 + 9X_9 + 10X_{10} \leq 35$

(Number of Researchers)
$X_1 + 2X_2 + 2X_2 + X_4 + 3X_5 + X_6 + 4X_7 + 3X_8 + 4X_9 + 3X_{10} \leq 10$

(Budget)
$10X_1 + 5X_2 + 15X_3 + 14X_4 + 19X_5 + 12X_6 + 9X_7 + 20X_8 + 18X_9 + 15X_{10} \leq 80$

$X_1, X_2, \ldots, X_{10} = \{0,1\}$

$D_1^-, D_1^+, D_2^-, D_2^+ \geq 0$

Solution = Projects {1,3,4,5,8}

Expected Profit = 120 units

Portfolio Success Index = 4.10

one means the project is selected, and a zero means the project is not selected. The variables D_1^+ and D_1^- represent the deviational variables for the first goal, expected profit. One goal of the R&D project selection process is to maximize expected profit, so a large target value was selected and the model objective is to minimize the negative deviation from this large target value. This will encourage the selection the projects that have the highest expected profit. D_2^+ and D_2^- are the deviational variables for the second goal, portfolio success. The objective is to maximize the portfolio probability of success index, and the model does this by minimizing the negative deviation, encouraging the selection of projects with the highest probabilities of success.

The constraints represent the expected profit goal, the portfolio success goal, resource X, resource Y, the number of researchers, and the budget constraints respectively. The zero-one integer linear programming model uses the same resource constraints as the filtered beam search approach and the complete enumeration model. The target values used in the expected profit and portfolio success constraints were chosen arbitrarily large to ensure that only the deviation variables D_1^- and D_2^- would have solution values.

The problem was solved using the branch and bound based integer programming algorithm from the QSB2+ software. The algorithm required 15 iterations, and 69.277 CPU seconds to solve the ten project problem, and the final solution is given in Table 4.3. This solution for the ten project problem is different than the solution obtained using the filtered beam search approach. The zero-one integer linear goal programming model selects the projects {1,3,4,5,8}, having an expected profit of 120 units, and a portfolio probability of success index of 4.10. (This information can be obtained from the standard output of the model, provided by the QSB2+ software). Note that the solution is the project set with the highest expected profit, as determined by complete enumeration. The math programming model cannot provide the makespan of the project set, since project scheduling is not included in the model formulation. The makespan for the math programming solution is 37 units, calculated using the scheduling heuristic used in the filtered beam search approach.

Summary of ten project problem

The ten project problem was solved using three different modeling techniques: complete enumeration, zero-one integer linear programming, and the filtered beam search approach. The results of the filtered beam search approach highlighted the weaknesses of approaches that do not incorporate project scheduling into the scheduling, or in the case of complete enumeration, how these approaches are impractical for large problem sizes. The solution of the math programming model indicates what can occur when project scheduling is not included in the project selection process. The zero-one integer linear goal programming model selects the project set that best meets the goals of expected profit and portfolio probability of success, at the expense of project scheduling. This technique, commonly used for project selection, selected a project set with the highest expected profit, and a high portfolio probability of success index, however, this solution requires a significantly longer time to complete the project set. The filtered beam search approach identified a solution with a nearly identical expected profit, a higher portfolio probability of success index, and a makespan that is two thirds of the makespan for the project set selected by the math programming model, thus demonstrating the capability of the filtered beam search approach to incorporate a project scheduling goal into the project selection process.

The fifty project example problem

A realistic example of an R&D project selection and scheduling problem would likely contain more than ten projects. The basis for using the filtered beam search technique is to develop an approach that can handle a realistic problem size that incorporates project scheduling into the selection process. For small problem sizes, such as ten projects, a complete enumeration model can handle both project selection and scheduling. However, it would not take a very large increase in the problem size to make a complete enumeration model intractable. As mentioned earlier, the number of possible combinations of projects increases exponentially, with the number of combinations escalating into the billions with a problem size of thirty or more projects. Therefore, it is desirable to develop an approach that can handle thirty, forty, fifty or more projects in a reasonable amount of computation time.

An sample problem of fifty projects was randomly generated using a BASIC program and the uniform random number generator for BASIC. The project budgets, anticipated profits, probabilities of success, the facility sequences and durations were all generated randomly using the BASIC program for fifty projects. The project resource data is given in Table 4.4 and the project sequence and duration data is given in Table 4.5.

The resource constraint levels used for the fifty project problem are slightly different than for the ten project problem. The levels of budget, number of research personnel, resource X, and resource Y were all increased slightly to reflect a larger, more realistic problem. The budget was increased to 150 units, the number of re-

Table 4.4. ***Resource data for fifty project problem***

Project	*Budget*	*Profit*	*Probability of success*	*Resource X*	*Resource Y*	*Number of Researchers*
1	13	31	0.78	12	11	1
2	42	131	0.89	11	2	1
3	29	92	0.59	15	4	3
4	25	55	0.74	11	7	1
5	39	122	0.76	14	9	4
6	18	23	0.56	10	14	3
7	36	113	0.55	6	11	1
8	49	103	0.86	13	10	5
9	34	73	0.95	3	13	4
10	28	61	0.74	10	12	2
11	12	41	0.59	7	15	2
12	17	22	0.73	6	4	5
13	5	20	0.96	10	4	4
14	18	23	0.84	13	13	1
15	34	73	0.66	10	8	3
16	13	18	0.62	10	14	1
17	5	10	0.62	2	13	3
18	2	7	0.95	6	4	4
19	7	12	0.68	15	9	3
20	40	125	0.65	1	6	2
21	21	47	0.81	3	3	4
22	7	19	0.84	1	7	3
23	1	8	0.68	7	3	1
24	23	28	0.87	5	1	5
25	46	97	0.94	3	3	1
26	30	65	0.59	15	10	2
27	13	18	0.73	14	10	3
28	21	47	0.93	9	6	2
29	15	50	0.67	14	11	2
30	9	32	0.86	12	4	4
31	24	29	0.88	7	12	3
32	4	17	0.93	6	12	5
33	1	5	0.68	3	2	4
34	3	11	0.76	5	14	5
35	36	77	0.75	10	4	3
36	40	125	0.84	14	7	3
37	24	53	0.58	2	8	2
38	36	77	0.87	12	10	5
39	8	21	0.80	2	1	1
40	48	149	0.60	10	10	1
41	33	104	0.96	1	10	3
42	8	21	0.85	9	2	4
43	13	44	0.63	7	5	3
44	9	32	0.78	4	6	5
45	28	89	0.53	6	8	3
46	9	23	0.75	4	8	1
47	3	11	0.99	8	11	4
48	40	125	0.91	7	15	4
49	16	37	0.96	1	15	3
50	6	11	0.81	10	2	4

Table 4.5. Project sequences and durations for fifty project problem

Project	*Sequence 1*	*Sequence 2*	*Sequence 3*
1	a/9	b/2	c/10
2	a/8	b/6	c/7
3	b/5	a/6	c/0
4	a/9	b/0	c/0
5	c/3	b/9	a/0
6	b/2	a/8	c/1
7	a/6	c/5	b/1
8	a/3	b/9	c/8
9	c/10	a/5	b/10
10	a/1	c/10	b/10
11	c/2	b/10	a/7
12	a/10	b/3	c/1
13	b/4	c/10	a/0
14	c/6	b/4	a/9
15	a/1	c/9	b/4
16	a/2	b/5	c/3
17	b/8	c/1	a/4
18	a/10	b/4	c/8
19	a/7	c/4	b/6
20	c/2	b/10	a/10
21	a/5	c/2	b/0
22	a/1	b/10	c/4
23	a/6	c/7	b/9
24	a/1	b/1	c/7
25	c/1	a/4	b/9
26	c/10	b/9	a/3
27	b/10	a/8	c/2
28	a/10	b/2	c/7
29	a/2	b/8	c/6
30	b/2	c/7	a/4
31	a/3	b/1	c/5
32	a/3	c/5	b/10
33	a/10	b/9	c/8
34	a/3	b/5	c/7
35	b/5	a/2	c/3
36	c/6	b/9	a/7
37	a/7	b/3	c/6
38	a/9	c/2	b/8
39	a/7	b/2	c/0
40	c/3	a/8	b/4
41	a/3	b/1	c/2
42	a/10	b/10	c/5
43	c/9	a/5	b/6
44	c/8	b/2	a/8
45	b/2	c/3	a/8
46	c/1	a/9	b/9
47	a/10	b/9	c/9
48	a/1	b/1	c/0
49	a/5	b/9	c/10
50	a/2	b/9	c/2

search personnel was increased to 15, and the levels of resource X and Y were increased to 50 and 70 units respectively.

The solutions for the fifty project problem are essentially impossible to verify using complete enumeration and math programming because of the computational difficulties these approaches incur for a problem of this size. As an alternative, several different beamwidths and filterwidths were tested to observe their impact on the solutions obtained from the filtered beamwidth approach.

Filterwidth1=50, Filterwidth2=30

The fifty project problem was first solved using filterwidths that were proportional to the filterwidths used for the ten project problem, which were ten and six for the first and second filterwidths respectively. Increasing these by a factor of five results in filterwidths of fifty and thirty respectively. Using these filterwidths as a starting point, the problem was solved using a beamwidth of three, and then solved again using beamwidths of four and five.

The filtered beam search approach provides three alternate solutions, corresponding to the beamwidth of three as shown in Table 4.6. The three solutions are listed in descending order according to profit, but it would be up to the R&D manager to decide which of these solutions is the "best" solution according to the goals of the problem. In this case, there is no clear cut choice. The first project set, {2,9,23,41,48}, has the highest expected profit, but the second project set, {2,22,28,36,48}, has a better makespan and a higher portfolio probability of success. The third project set, {2,28,32,36,48}, has virtually the same expected profit, and has the same makespan as the second project set, but has a higher portfolio probability

Table 4.6. Solution of fifty project problem, w=3, f1=50, f2=30

Solutions from the filtered beam search approach

Project Set 1 = {2,9,23,41,48}
- Expected Profit = 405.53 units
- Makespan = 36 units
- Portfolio Success = 4.39

Project Set 2 = {2,22,28,36,48}
- Expected Profit = 396.34 units
- Makespan = 35 units
- Portfolio Success = 4.41

Project Set 3 = {2,28,32,36,48}
- Expected Profit = 396.19 units
- Makespan = 35 units
- Portfolio Success = 4.50

of success. A logical argument could be made for for each of these solutions, as all three appear to be of good quality.

The fifty project problem was solved again using the same filterwidths, but the beamwidth was increased to four. The effect of this is to increase the scope of the search that the filtered beam search undertakes when exploring the search tree for the problem. This increases the computational complexity of the approach, but decreases the likelihood of eliminating a good solution during the selection step at each level, since more beams are retained. The solution for this case is given in Table 4.7. Note that two of the project sets have six projects in them, and two have five projects. The filtered beam search managed to find two feasible project sets that consisted of six projects and two that consisted of five. This occurs several times in the results, and can be explained. During the last step of the filtered beam search process, it is possible for several of the beams to extend down one more level in the search tree, while other the other beams cannot be extended, due to the problem constraints. In these instances, if the beams that are not extended perform better than those that are, they will stay in the final solution. Thus it is possible to have solutions with a different number of projects in them.

The change in beamwidth alters the solution obtained earlier by the filtered beam search approach that used a beamwidth of three. Only one of the project sets in the previous solution set appears now, {2,9,23,41,48}. The project set with the highest expected profit is {2,23,28,41,43,48}. This project set seems to be the clear cut "best" solution between the project sets that contain six projects, but this set has a higher makespan than the two solutions that have only five projects. The decision

Table 4.7. Solution of fifty project problem, w=4, f1=50, f2=30

Solutions from the filtered beam search approach

Project Set 1 = {2,23,28,30,41,48}
Expected Profit = 407.41 units
Makespan = 41 units
Portfolio Success = 5.23

Project Set 2 = {2,23,28,41,43,48}
Expected Profit = 407.53 units
Makespan = 42 units
Portfolio Success = 5.00

Project Set 3 = {2,9,23,41,48}
Expected Profit = 405.53 units
Makespan = 36 units
Portfolio Success = 4.39

Project Set 4 = {2,30,36,48,49}
Expected Profit = 399.53 units
Makespan = 34 units
Portfolio Success = 4.45

as to which of these four solutions would be most attractive is up to the R&D manager.

The fifty project problem was then solved using the same filterwidths as before, and a beamwidth of five. Again, this will increase the computational complexity of the filtered beam search approach, but it will decrease the likelihood of missing a "good" solution. When the problem was solved using a beamwidth of five, four new project sets appeared in the solution, with only project set {2,23,28,30,41,48} remaining from the solution using a beamwidth of four. Two project sets appeared with a higher expected profit and a lower makespan than this solution, although a clear cut choice still did not emerge. The first project set, {2,13,28,41,46,48} has the highest expected profit, but the project set {2,13,28,39,41,48} has a slightly lower expected profit, and a makespan that is two time units less. The project set {2,23,28,30,41,48} from the previous solution appears to be the "third-best" choice of the solutions.

In summary, it appears that as the beamwidth is increased, the primary goal, expected profit, has a greater effect. The project sets in the solution for the filtered beam search approach using a beamwidth of five have higher expected profits than the solutions obtained using a beamwidth of three or four.

Filterwidth1 = 30, Filterwidth2 = 20

The filterwidths in the filtered beam search approach are the mechanism for incorporating the various goals of the R&D problem into the project selection process. In the filtered beam search approach, the nodes of the search tree are evaluated and filtered at every level. The "tighter" the filterwidth, the more nodes filtered out at that level. This seems to imply that the tighter the filterwidth, the more important

Table 4.8. Solution of fifty project problem, w = 5, f1 = 50, f2 = 30

Solutions from the filtered beam search approach

Project Set 1 = {2,13,28,41,46,48}
Expected Profit = 410.85 units
Makespan = 40 units
Portfolio Success = 5.40

Project Set 2 = {2,13,28,39,41,48}
Expected Profit = 410.47 units
Makespan = 38 units
Portfolio Success = 5.45

Project Set 3 = {2,23,28,30,41,48}
Expected Profit = 407.41 units
Makespan = 41 units
Portfolio Success = 5.23

Project Set 4 = {2,23,29,41,48,49}
Expected Profit = 405.15 units
Makespan = 41 units
Portfolio Success = 5.07

Project Set 5 = {2,23,41,43,48,49}
Expected Profit = 399.24
Makespan = 41 units
Portfolio Success = 5.03

the goal associated with that filter becomes. For example, at the second filter (for project scheduling), if only a few project sets are kept then many candidate sets will be eliminated from consideration. In other words, the project sets that perform poorly according to their project schedules will not be eligible for selection, no matter how good they perform according to expected profit.

The fifty project problem is solved again, using filterwidths that are slightly "tighter" than before. The problem is solved using filterwidths of thirty and twenty for the first and second filterwidths respectively. The purpose is to determine if tightening the filterwidths will increase the portfolio index of success and decrease the makespans of the project sets in the solution.

The problem is first solved using a beamwidth of three. The solution is given in Table 4.9. The three project sets in this solution are also in the solution for the previous case, which used a beamwidth of five, and filterwidths of fifty and thirty. Comparing the solution for the narrower filterwidths with the solution obtained previously, a noticeable difference occurs. The three project sets selected using the tighter filterwidths all have higher expected profit, but also higher makespans. The portfolio index of success is also higher, which is understandable since the relative importance of that goal was increased by narrowing the filterwidth for that goal.

The problem is solved again using a beamwidth of four. The only change in the solution is the addition of one more solution set. The solutions are shown in Table 4.10. The top three project sets remained the same, and the new solution set is the fourth project set listed in the solution.

Table 4.9. Solution of fifty project problem, w=3, f1=30, f2=20

Solutions from the filtered beam search approach

Project Set 1 = {2,13,28,41,46,48}
Expected Profit = 410.85 units
Makespan = 40 units
Portfolio Success = 5.40

Project Set 2 = {2,13,28,30,41,48}
Expected Profit = 410.47 units
Makespan = 38 units
Portfolio Success = 5.45

Project Set 3 = {2,23,28,30,41,48}
Expected Profit = 407.41 units
Makespan = 41 units
Portfolio Success = 5.23

Table 4.10. Solution of fifty project problem, w=4, f1=30, f2=20

Solutions from the filtered beam search approach

Project Set 1 = {2,13,28,41,46,48}
Expected Profit = 410.85 units
Makespan = 40 units
Portfolio Success = 5.40

Project Set 2 = {2,13,28,39,41,48}
Expected Profit = 410.47 units
Makespan = 38 units
Portfolio Success = 5.45

Project Set 3 = {2,23,28,30,41,48}
Expected Profit = 407.41 units
Makespan = 41 units
Portfolio Success = 5.23

Project Set 4 = {2,13,23,28,41,48}
Expected Profit = 399.13 units
Makespan = 37 units
Portfolio Success = 5.33

When the beamwidth is increased to five, the same thing happens and the solution is the same with one more project set added to the solution obtained for a beamwidth of four, as shown in Table 4.11.

Using filterwidths of thirty and twenty, there is still no clear cut choice for the "best" portfolio. This might be expected, given the enormous number of project combinations possible. The first project set, projects {2,13,28,41,46,48} has the best expected profit, but the second project set, {2,13,28,39,41,48} has virtually the same expected profit, but a better makespan and portfolio index of success than the first project set. The other project sets in the solution have smaller expected profits, but in some cases a better makespan.

Filterwidth1=25, Filterwidth2=10

In order to further investigate the effect of the filterwidths on the solutions obtained by the filtered beam search approach, the filterwidths are tightened even more for the fifty project problem. The filterwidths are reduced to twenty-five and ten, respectively. The expected outcome is to generate solutions that have better makespans and portfolio indices of success than the solutions obtained with wider filterwidths.

The problem is solved for beamwidths of three, four and five. The solution for a beamwidth of three is given in Table 4.12. Note that in general, when these results are compared with the results earlier for a beamwidth of three, the expected profits are lower, and the makespans are better. This might be expected when the filterwidths are decreased. By tightening the filterwidths, the filtered beam search approach makes the goals associated with the filters slightly more important. The

Table 4.11. Solution of fifty project problem, w = 5, f1 = 30, f2 = 20

Solutions from the filtered beam search approach

Project Set 1 = {2,13,28,41,46,48}
Expected Profit = 410.85 units
Makespan = 40 units
Portfolio Success = 5.40

Project Set 2 = {2,13,28,39,41,48}
Expected Profit = 410.47 units
Makespan = 38 units
Portfolio Success = 5.45

Project Set 3 = {2,23,28,30,41,48}
Expected Profit = 407.41 units
Makespan = 41 units
Portfolio Success = 5.23

Project Set 4 = {2,13,23,28,41,48}
Expected Profit = 399.13 units
Makespan = 37 units
Portfolio Success = 5.33

Project Set 5 = {2,22,23,28,41,48}
Expected Profit = 395.93
Makespan = 38 units
Portfolio Success = 5.21

Table 4.12. Solution of fifty project problem, w=3, f1=25, f2=10

Solutions from the filtered beam search approach

Project Set 1 = {2,14,23,41,48,49}
Expected Profit = 391.09 units
Makespan = 37 units
Portfolio Success = 5.24

Project Set 2 = {2,30,41,48,49}
Expected Profit = 393.67 units
Makespan = 34 units
Portfolio Success = 4.58

Project Set 3 = {2,28,30,36,41}
Expected Profit = 393.33 units
Makespan = 32 units
Portfolio Success = 4.48

final selection is still based on expected profit, but the selection occurs only from the project sets that perform very well according to probability of success and makespan. Of course, the branches extended down the search tree are still very good choices, since the beams are selected based on expected profit.

As the beamwidth is increased, expanding the search process, other project sets appear in the solution. When the beamwidth is increased to four, the solutions completely change. The solutions for a beamwidth of four are given in Table 4.13. Notice that none of the solutions appeared from the previous solution. The expected profits increased, and the makespans remain about the same. The beamwidth of three appeared to have limited the search, and some of these new solutions were "missed" during the search process. Also, because the filterwidths are so narrow, many projects were filtered out at one level, only to reappear at a later level. Because of project interactions, these projects were able to enter the solution again, facilitated by the additional beam used in the search.

The problem is solved again using a beamwidth of five, and the solution changed dramatically. Four new project sets appear, with improvements in expected profit. The results are shown in Table 4.14. Several of these project sets were in the earlier solutions obtained using wider filterwidths, namely project sets {2,13,28,39,41,48} and {2,13,28,41,46,48}. These are the only two project sets to appear in all three solutions using a beamwidth of five. There is no clear cut best solution in the five project sets.

Table 4.13. Solution of fifty project problem, w=4, f1=25, f2=10

Solutions from the filtered beam search approach

Project Set 1 = {2,4,13,23,41,48}
Expected Profit = 396.22 units
Makespan = 31 units
Portfolio Success = 5.11

Project Set 2 = {2,10,13,23,41,48}
Expected Profit = 400.79 units
Makespan = 36 units
Portfolio Success = 5.14

Project Set 3 = {2,13,41,48,49}
Expected Profit = 385.38 units
Makespan = 30 units
Portfolio Success = 4.65

Project Set 4 = {2,13,31,41,48}
Expected Profit = 375.53 units
Makespan = 28 units
Portfolio Success = 4.57

Table 4.14. Solution of fifty project problem, w=5, f1=25, f2=10

Solutions from the filtered beam search approach

Project Set 1 = {2,13,23,28,41,48}
Expected Profit = 399.13 units
Makespan = 37 units
Portfolio Success = 5.33

Project Set 2 = {2,14,23,41,48,49}
Expected Profit = 391.09 units
Makespan = 37 units
Portfolio Success = 5.24

Project Set 3 = {2,13,28,39,41,48}
Expected Profit = 410.47 units
Makespan = 38 units
Portfolio Success = 5.45

Project Set 4 = {2,13,28,41,46,48}
Expected Profit = 410.85 units
Makespan = 40 units
Portfolio Success = 5.40

Project Set 5 = {2,13,41,48,49}
Expected Profit = 385.38
Makespan = 30 units
Portfolio Success = 4.57

Summary of results for the fifty project problem

The filtered beam search appears to be sensitive to choice of beamwidths, and especially filterwidths. Ow and Morton note this in their paper (1989), and this was apparent in the R&D example when the filterwidths were decreased dramatically. The first solution presented was for fairly wide filterwidths, of fifty and thirty for the first and second filterwidths, respectively. The solutions from this set of filterwidths seemed to be satisfactory, although they changed slightly as the beamwidth was increased, expanding the search space for the filtered beams search algorithm. As the search space was expanded, the solutions obtained by the approach improved according to expected profit, which was the primary goal. This is what might be expected of the filtered beam search approach.

The second set of filterwidths, a first filterwidth of thirty and a second filterwidth of twenty, appeared to stabilize the filtered beam search. The solutions did not change as the beamwidth increased, but rather a new project set was added each time the beamwidth was increased. The solutions obtained were satisfactory when compared with the solutions obtained using different filterwidths.

A third solution set was generated using very narrow filter widths, which should have increased the importance of the goals associated with the filters. The solutions generated did perform better according to the filter goals than the solutions obtained from the two other sets of filterwidths. Also, as the beamwidth was increased, the solutions improved according to expected profit. However, the narrow filterwidths caused many different project sets to appear in the solutions as the beamwidth was increased. In the previous examples, the expected profit of the final solutions increased as the beamwidth increased. In the case of narrow filterwidths, which caused more importance to be placed on the filter goals, there seemed to be a transition from emphasizing project scheduling and portfolio probability of success to emphasizing expected profit as the beamwidth was increased. This would account for the drastic change in the solutions as the beamwidth changed.

It is important to note that when using a beamwidth of five, many of the same project sets appeared in the solution, no matter what filterwidths were used. The project sets {2,13,28,41,46,48} and {2,13,28,39,41,48} appeared in all three solutions, and the project sets {2,13,23,28,41,48} and {2,23,28,30,41,48} appeared in two of the three solutions. These project sets tended to be selected no matter what filterwidths were used, indicating that these project sets perform best according to the three R&D goals. Also, many of the same projects continue to appear in the solutions. Projects 2, 41, and 48 are in nearly every solution produced by the filtered beam search. This would indicate to the R&D manager that these projects should probably be selected for implementation.

The filtered beam search gives the R&D management flexibility in selecting R&D projects while still incorporating project scheduling as an objective of the selection process. Previous R&D project selection methods were unable to incorporate project scheduling at all, but the filtered beam search allows the R&D manager to change filterwidths and beamwidths and see the effect of changing the relative importance of the various goals. This can be done in minutes, as the filtered beam search approach is computationally efficient. Each of the solutions in this chapter were generated on a PC in several minutes, using the Turbo Pascal software. The filtered beam search was able to reduce a computationally explosive search space, and generate excellent solutions. The choice of beamwidths also provides the R&D manager with alternative solutions to the selection process, with an insignificant increase in computation time. This would permit greater flexibility in the decision process.

Summary

This chapter presented solutions to the R&D project selection problem with project scheduling as a goal. The filtered beam search model was used to solve the ten project problem, and the solutions were compared with the solutions obtained from a complete enumeration model and a portfolio model based on zero-one integer goal programming. The results of the filtered beam search compared very favorably with these others approaches, finding very good, if not optimal, solutions to the ten project problem. The effect of not including the project scheduling goal in the selection process was demonstrated using the math programming model and comparing the results to those obtained by the filtered beam search approach.

A fifty project R&D problem was solved to demonstrate the filtered beam search approach for a realistic problem size. Because of the problem size, it was impractical to compare the solutions obtained from the filtered beam search with a complete enumeration model, or a math programming model. The effect of different beamwidths and filterwidths was investigated for the fifty project problem.

The following chapter, "Summary and Conclusions", concludes this study by summarizing the research performed, presenting guidelines for incorporating the filtered beam search in R&D project selection problems, identifying limitations of the filtered beam search approach and outlining avenues for future research in R&D project selection incorporating project scheduling.

Chapter 5: Summary and Conclusions

This chapter concludes this study by presenting a summary of the research performed, providing several guidelines for using the filtered beam search approach for R&D project selection problems, outlining the limitations of the filtered beam search approach, and, suggesting avenues for future research in R&D project selection and scheduling using the filtered beam search approach.

Summary of research

This dissertation develops a model that incorporates project scheduling as a goal in the project selection process. The nature of project scheduling is such that it is difficult to include it in existing project selection models, such as linear programming, integer programming, and, goal programming. The purpose of developing the model is to reflect a more realistic R&D selection problem, in which project scheduling is considered jointly with other goals, rather than after the selection process as is typically done. This research provides several important observations concerning existing R&D project selection models and project scheduling:

- It is essentially impossible to incorporate project scheduling into the project selection process using exact methods, without using simplifying assumptions about the R&D environment. Several exact models incorporating project scheduling into the selection process exist in the literature, but include simplifying assumptions about project scheduling, such as requiring that the R&D projects be done one at a time, and that once started they must be completed without interruption (Gupta 1992, Bell 1967, Hess 1962). If these assumptions are relaxed, then it is not practical to use an exact method for project selection which incorporates project scheduling as an objective. The result is that a heuristic for project selection is necessary in order to incorporate project scheduling as a goal.

- Problem size is another limiting factor for current R&D project selection models. Exact methods used for project selection tend to run into computational difficulty when the problem size reaches a realistic size. For larger problems, an efficient model that provides optimal, or very good, solutions would be valuable to an R&D manager.

- The practice of first selecting projects and then scheduling them can lead to poor solutions, even if an excellent project selection model is used. Ignoring project scheduling during the project selection process can result in projects being selected that require an extraordinary long time to complete, or may be infeasible to schedule at all if a time constraint is placed on the R&D department. Project scheduling is a realistic goal for the R&D manager, and should be considered *during* the selection process.

Because of the difficulty in using exact methods for project scheduling, a special heuristic technique is used for the R&D project selection model developed in this research. The *filtered beam search approach* incorporates several important goals for R&D project selection, including project scheduling, into the selection process. This approach is shown to be a viable alternative, particularly when the problem size is large.

The filtered beam search approach is used to solve two example R&D problems, one containing ten projects and the other containing fifty projects. The problems have resource constraints which are representative of a realistic R&D environment, including budget availability, the availability of special research personnel, and two generic resource constraints. These generic constraints are included to illustrate the ease of including other relevant constraints into the filtered beam search approach.

The model incorporates three goals in the example problems: (1) expected profit from the project portfolio, (2) a probability of success index for the portfolio, and, (3) a scheduling goal, makespan of the portfolio. The model developed in this research selects a portfolio of projects, subject to the resource constraints, that will perform very well according to the goals. The notion of optimality in an R&D project selection problem having multiple goals with no common measure is also addressed in this research.

The effect of including project scheduling in the project selection process is demonstrated through the use of an example R&D problem. The results of the filtered beam search approach for a ten project example problem are compared with the results of a complete enumeration model, which included project scheduling as a goal, and a zero-one integer goal programming model, which did not include project scheduling as a goal. The comparison of results establish that the filtered beam search approach is able to select excellent R&D portfolios without "missing" better solutions, as determined by the complete enumeration of all possible R&D portfolios for the ten project problem. When the solutions obtained by the filtered beam search are compared with the solution generated by a math programming model, the effect of excluding project scheduling is readily apparent. The zero-one integer goal programming model selects a portfolio that is essentially equal to the solution obtained by the filtered beam search approach in terms of expected profit and portfolio index of success, but has a makespan that is significantly longer.

An example R&D problem containing fifty projects is solved using the filtered beam search approach. The fifty project example is large enough to cause computational difficulties for typical R&D project selection models, such as math programming. The filtered beam search model incorporates project scheduling into the selection process, and is able to handle a problem of this size with relative ease, generating solutions in minutes on a model DC-2010 Leading Edge personal computer.

Alternative goal structures are investigated in the fifty project problem by altering the beamwidths and filterwidths of the filtered beam search approach, which changes the emphasis placed on the various goals used in the R&D project selection problem.

The filtered beam search approach used in this research is programmed in Turbo Pascal 4.0, which provides the flexibility necessary to handle the multiple goals and resource constraints of the R&D project selection problem. The modular nature of Turbo Pascal permits changes in the goal structure, resource constraints, and the scheduling heuristic used in the problem.

Practical guidelines

An examination of the results provided in this research suggests several guidelines for using the filtered beam search approach:

(1) Choice of beamwidth. The choice of beamwidth can affect the solutions obtained from the filtered beam search approach. Using a beamwidth that is too narrow can lead to "missing" an excellent solution. Ow and Morton (1989) note that too narrow a beamwidth can cause a significant deviation from the optimal solution. However, since the filtered beam search approach is an efficient method for searching a large solution space, increasing the beamwidth does not seriously affect the computation time required. The computational complexity of the filtered beam search is on the order of (wn^2) where w is the beamwidth of the search, and n is the number of projects. Increasing the beamwidth will increase the computation time in a linear fashion (Leu et al. 1992), but will decrease the probability of "missing" a optimal solution. The benefits of using a larger beamwidth generally outweigh the costs. However, the literature on the filtered beam search suggests that the beamwidth required to generate excellent solutions tends to be small (Ow and Morton 1989), and increasing it beyond a certain beamwidth does not improve the quality of the solution, but only causes the computation time to increase. Thus, the best choice of beamwidth for a particular R&D problem is problem specific, and requires some experimentation. In this research, a beamwidth of five was sufficient, generating excellent solutions for the ten project and fifty project problems.

(2) Choice of filterwidths. There is little or no guidance in the literature about choosing filterwidths, particularly when more than one filter is used at each level. This research several guidelines for selecting filterwidths for the filtered beam search approach. The choice of filterwidths determines the relative importance of the goal associated with the filter. A narrow filterwidth increases the importance of the goal, while a wide filterwidth decreases it. Because very little research has been done concerning multiple filters in a filtered beam search, a little experimentation may be required to determine appropriate filterwidths for the technique.

The filterwidths also can affect the convergence of the filtered beam search approach. This research suggests that when using narrow filterwidths, a larger beamwidth is necessary to obtain consistent results. However, the converse does not appear to be true, i.e. wide filterwidths do not allow the use of a small beamwidth, as shown by the results obtained for the fifty project problem.

(3) Choice of scheduling heuristic. This research uses a scheduling heuristic that represents a specific objective and problem structure, namely minimizing makespan of the R&D portfolio through three research facilities. The example problems solved in this research allow the use of job shop scheduling heuristics in the filtered beam search approach. However, the choice of scheduling heuristic is problem dependent, and caution should be taken to ensure that the scheduling heuristic reflects the actual R&D problem. The filtered beam search is robust in the sense that it can accommodate a variety of scheduling objectives, allowing the R&D manager to use a scheduling heuristic that best represents the actual R&D environment. The only change required in the filtered beam search approach is in the filtering step where the potential project sets are evaluated at each level. The appropriate scheduling heuristic is evaluated at this step in the approach, and the filtered beam search is otherwise unchanged.

Limitations of the filtered beam search approach

The filtered beam search approach outlined in this research has several limitations when applied to the R&D project selection problem:

(1) Because the filtered beam search is a heuristic approach to solving the R&D project selection and scheduling problem, it does not guarantee an optimal solution. However, the R&D selection problem typically contains multiple objectives with no common measure, and this makes it difficult to decide on an "optimal" solution to the problem. A comparison of several possible solutions might reveal that one performs best according to one objective, while other solutions perform better for a different goal. If it is impossible, or impractical, to mathematically equate the goals then the notion of "optimality" becomes more complex. Thus, the fact that the filtered beam search is a heuristic may not limit its effectiveness as a means of generating solutions to the R&D project selection problem when scheduling is included in the selection process.

(2) Another limitation of the model is using deterministic activity durations while evaluating the scheduling goal. The purpose of using deterministic activity durations is to allow the use of a recognized job shop scheduling heuristic, and thereby illustrate the feasibility of incorporating project scheduling into the search process. Care should be taken when using the filtered beam search approach as outlined if the activity durations are not deterministic. If an adequate scheduling heuristic using non-deterministic activity durations can be found, then it can be incorporated into the filtered beam search approach. This would represent a more realistic R&D environment.

(3) The allocation of special research personnel in the filtered beam search approach is another limitation. In the model, the special research personnel are treated as a regular constraint, i.e. the number of research personnel required by the project portfolio must be less than or equal to the number available at the beginning of the R&D process. In reality, this type of research personnel "floats" between projects, i.e. after a project is completed, the personnel are not used up, but instead can be transferred to another project. This type of situation is an example of multiple resource constrained scheduling. It would be possible to incorporate multiple resource constrained project scheduling into the filtered beam search approach by identifying an appropriate scheduling heuristic, and using it at the project scheduling filter step.

(4) The filtered beam search approach is not a specific model for R&D project selection, but instead is a general technique for solving this type of problem. Thus, a model, and an appropriate computer program, must be developed for each individual R&D project selection problem. This makes it difficult to develop a "canned" al-

gorithm for R&D project selection incorporating scheduling. However, this can be considered a strength of the filtered beam search approach. Each individual R&D problem is not "fit" to a specific algorithm, which often requires limiting assumptions concerning problem objectives and constraints, but instead the solution technique is developed to "fit" the problem. The modularity of the filtered beam search approach allows many different goal structures and resource constraints to be defined. The goals and constraints are not limited to specific functional types, and this flexibility of the filtered beam search allows it to realistically reflect the actual R&D problem.

Future research

This research has developed an effective and efficient model for incorporating project scheduling into the R&D project selection problem. However, the model developed has several limitations, suggesting avenues for further research in this area. One area for further research is investigating the impact of different scheduling goals on project selection. The goal used in the model presented here, project makespan, is one of several plausible project scheduling goals. In order to incorporate a different project scheduling goal in the filtered beam search approach, the computing procedure that evaluates the scheduling goal would have to be rewritten. This would be straightforward due to the modular nature of the Turbo Pascal programming language. No other portions of the computer program would need to be significantly altered.

The impact of other scheduling heuristics on project selection could also be investigated. The job shop scheduling heuristic used in this research is one of many available. It would be straightforward to investigate different heuristics, requiring a change in the Turbo Pascal procedure that does the scheduling calculations. The other portions of the program would remained unchanged.

A major topic for future research would be the incorporation of a multiple resource constrained project scheduling heuristic into the model. This would require a major rewrite of the Turbo Pascal scheduling procedure, but would be possible since a number of multiple resource constraint heuristics exist in the literature. The heuristic would have to be "translated" into Turbo Pascal, or whatever program language that is used, in order to incorporate it into the filtered beam search approach. This would represent a more realistic R&D environment, and as long as the scheduling heuristic is fairly efficient according to computation time, it would not affect the computational efficiency of the filtered beam search approach.

The index for portfolio success might be another area for future research. Multiplicative, additive, and other types of indices have been investigated in the literature, but not in the context of the filtered beam search. The impact of different indices of success on project selection could be investigated. However, the primary thrust of this research is the impact of project scheduling on project selection, so the index of portfolio success should probably be investigated in the context of project scheduling.

Bibliography

Ansoff, H.I., "Evaluation of applied research in a business firm," *Technological Planning on the Corporate Level*, J. R. Bright, Ed., Proc. Conference at Harvard Business School, Harvard University, Cambridge,Mass., 1962, pp.209-224.

Asher, D.I., "A linear programming Model for the allocation of R&D efforts," *IRE Trans. on Engineering Management*, vol.EM-9, Dec.1962, pp.154-157.

Baker, Norman R., "R&D Project Selection Models: An Assessment," *IEEE Transactions on Engineering Management*, vol.EM-21, no.4, November 1974, pp.165-171.

Baker, K.R., *Introduction to Sequencing and Scheduling*, Wiley, New York, 1974.

Baker, Norman, and Freeland, James, "Recent Advances in R&D Benefit Measurement and Project Selection Methods," *Management Science*, vol.21, no.10, June 1975, pp.1164-1175.

Baker, N. R., and Pound, W. H., "R and D Project Selection: Where We Stand," *IEEE Transactions on Engineering Management*, vol.EM-11, December 1964, pp.124-134.

Bard, Jonathan F., "Parallel Funding of R & D Tasks with Probabilistic Outcomes," *Management Science*, vol.31, no.7, July 1985.

Barr, A. and Feigenbaum, E.A., *The Handbook of Artificial Intelligence*, vol.1., Heuris Tech Press, Stanford, Ca., 1981.

Beged-Dov, A. G., "Optimal Assignment of Research and Development Projects in a Large Company Using an Integer Programming Model," *IEEE Transactions on Engineering Management*, vol.EM-12, no.4, December 1965, pp.138-143.

Bell, D.C., Chilcott, J.E., Read, A.W., and Solway, R.A., "Application of a research project selection method in the northern region scientific services department," R&D Dept., Central Electricity Generating Board (U.K.), RD/H/R2, 1967.

Bernardo, John J., "An Assignment Approach to Choosing R&D Experiments," *Decision Sciences*, vol.8, 1977, pp.489-501.

Brandenburgh, R.G. and Stedry, A., "Planning and budgeting in a multiphase R&D process," Management Science Research Group, Grad. Sch. Ind. Admin., Carnegie Inst. Technol., Res. Rep. 87, Pittsburgh, Pa. 1966.

Brockhoff, K., "Some problems and solutions in the selection of an R&D portfolio," presented at 1969 Conf. Int. Fed. of Operations Research Society, Venice, Italy.

Cetron, Marvin J., Martino, Joseph, and Roepcke, Lewis, "The Selection of R&D Program Content-Survey of Quantitative Methods," *IEEE Transactions on Engineering Management*, vol.EM-14, no.1, March 1967, pp.4-13.

Charnes, A. and Stedry, A.C., "A chance-constrained model for real time control in research and development management," *Management Science*, vol.12, 1966, pp.B353-362.

Cochran, Michael A., Pyle, Edmund B., Greene, Leon C., Clymer, Harold A., and Bender, A. Douglas, "Investment Model for R&D Project Evaluation and Scheduling," *IEEE Transactions on Engineering Management*, vol.EM-18, no.3, August 1971, pp.89-100.

Cooley,Stephen C., Hehmeyer,Jan, and Sweeney,Patrick J., "Modelling R & D Resource Allocation," *R & D Management*, January- February 1986, pp.40-45.

Danila, Nicolas, "Strategic Evaluation and Selection of R&D Projects," *R&D Management*, vol.19, no.1, pp.47-62.

Davis, Edward W., "Project Scheduling under Resource Constraints- Historical Review and Categorization of Procedures," *AIIE Transactions*, vol.5, no.4, December 1973, pp.297-313.

Davis, Edward W., and Heidorn, George E., "An Algorithm for Optimal Project Scheduling Under Multiple Resource Constraints," *Management Science*, vol.17, no.12, August 1971, pp.B803-B816.

Davis, Edward W., and Patterson, James H., "A Comparison of Heuristic and Optimum Solutions in Resource Constrained Project Scheduling," *Management Science*, vol.21, no.8, April 1975, pp.944-955.

Dean, Burton V., and Chaudhuri, Asok K., "Project Scheduling: A Critical Review," *Management of Research and Innovation*, North Holland Publishing, Amsterdam, B.V. Dean and J.L. Goldhar Eds., 1980, pp.215-233.

Dean, B.V., and Roepcke, L.A., "Cost effectiveness in R&D organizational resource allocation," *IEEE Trans. Eng. Manag.*, vol.EM-16, Nov. 1969, pp.222-242.

Dean, B.V., and Sengupta, S.S., "Research Budgeting and Project Selection", *IRE Trans. on Engineering Management*, vol. EM-9, Dec.1962, pp.158-169.

Disman, S., "Selecting R&D projects for profit," *Chem. Engrg.*, vol.69, Dec. 1962, pp.87-90.

Freeman, R. J., "A stochastic model for determining the selection and allocation of the research budget," *IRE Trans. on Engineering Mgt.*, vol.EM-7, Mar.1960, pp.2-7.

Fox, M.S., "Constraint Directed Search: A Case Study of Job-Shop Scheduling," Ph.D. Dissertation, Carnegie-Mellon University, 1983.

Garey, Michael R., and Johnson, David S., *Computers and Intractability : A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., San Francisco, 1979.

Gear, A. E., Lockett, A. G., and Pearson, A. W., "Analysis of Some Portfolio Models for R&D," *IEEE Transactions on Engineering Management*, vol.EM-18, no.2, May 1971, pp.66-75.

Gillespie, J.S., and Gear, A. E., "Suitability of Management Science Models," *IEEE Transactions on Engineering Management*, vol.EM-20, no.4, November 1973, pp.121-129.

Granot, Daniel, and Zuckerman, Dror, "Optimal Sequencing and Resource Allocation in Research and Development Projects," *Management Science*, vol.37, no.2, February 1991.

Gupta, S.K., Kyparisis, J., and Ip, C., "Project Selection and Sequencing to Maximize Net Present Value of the Total Return," *Management Science*, vol.38, no.5, May 1992, pp.751-752.

Hess, S.W., "A dynamic programming approach to R&D budgeting and project selection," *IRE Trans. on Engineering Management*, vol.EM-9, Dec.1962, pp.170-179.

Higgins, J. C., and Watts, K. M., "Some perspectives on the use of management science techniques in R&D management," *R&D Management*, vol.16, no.4, 1986, pp.291-296.

Jin, Xiao-Yin, and Porter, Alan L., "R&D Project Selection and Evaluation: A Microcomputer Based Approach," *R&D Management*, vol.17, no.4, 1987, pp.277-288.

Keefer, Donald L., "Allocation Planning for R & D with Uncertainty and Multiple Objectives," *IEEE Transactions on Engineering Management*, vol.EM-25, no.1, February 1978.

Keefer, Donald L., and Pollock, Stephen M., "Approximations and Sensitivity in Multiobjective Resource Allocation," *Operations Research*, vol.28, no.1, January-February 1980.

Keifer, D.M., "Winds of change in industrial Chemical Research," *Chemical and Engineering News*, vol.42, pp.88-109.

Keown, Arthur J., Taylor, Bernard W. III, and Duncan, Calvin P., "Allocation of Research and Development Funds: A Zero-One Goal Programming Approach," *The International Journal of Management Science*, vol.7, no.4, pp.345-351.

Lawler, E.L., and Woods, D.E., "Branch and Bound Methods: A Survey," *Oper. Res.*, vol.14, no.4, July-August 1966, pp.699-719.

Leu, Y., Rakes, T.R., Rees, L.P., and Cecucci, W.A., "Modelling Resource Allocation in a Decentralized Organization with an AI-Based, Goal Directive Model," working paper, 1992.

Liberatore, Matthew J., "An Extension of the Analytical Hierarchy Process for Industrial R & D Project Selection and Resource Allocation," *IEEE Transactions on Engineering Management*, vol.EM-34, no.1, February 1987.

Liberatore, Matthew J., and Titus, George J., "The Practice of Management Science in R&D Project Management," *Management Science*, vol.29, no.8, August 1983, pp.962-974.

Lockett, A. Geoffrey, and Gear, Anthony E., "Programme Selection in Research and Development," *Management Science*, vol.18, no.10, June 1972, pp.b575-b589.

Lowerre, B.T., "The HARPY Speech Recognition System," Ph.D. Dissertation, Carnegie-Mellon University, April, 1976.

Moore, John R., and Baker, Norman R., "Computational Analysis of Scoring Models for R&D Project Selection," *Management Science*, vol.16, no.4, December 1969, pp.b212-b232.

Moore, Laurence J., and Taylor, Bernard W. III, "Multiteam, Multiproject Research and Development Planning with GERT," *Management Science*, vol.24, no.4, December 1977, pp.401-410.

Mottley, C.M., and Newton, R.D., "The selection of projects for industrial research," *Operations Res.*, vol.7, Nov-Dec 1959, pp.740-751.

Nillson, Nils J., *The Principles of Artificial Intelligence*, Tioga Publishing, Palo Alto, Ca., 1980.

Ow,Peng Si, and Morton,Thomas E., "The Single Machine Early/Tardy Problem," *Management Science*, vol.35, no.2, February 1989, pp.177-191.

Patterson, James H., "A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem," *Management Science*, vol.30, no.7, July 1984, pp.854-867.

Pritsker,A. Alan B., Watters, Lawrence J., and Wolfe, Philip M., "Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach," *Management Science*, vol.16, no.1, September 1969.

Rubin,S., "The ARGOS Image Understanding System," Ph.D. Dissertation, Carnegie-Mellon University, April, 1978.

Saaty, T., *The Analytical Hierarchy Process*, New York: McGraw-Hill, 1980.

Santhanam, R., Muralidhar, K., and Schniederjans, M., "A Zero-One Goal Programming Approach for Information System Project Selection," *Omega*, vol.17, no.6, pp.583-593

Sherali, H.D. and Rios, M., "An Air Force Crew Allocation and Scheduling Problem," *Journal of the Operation Research Society*, vol.35, no.2, pp.91-103.

Souder, William E., "Comparative Analysis of R&D Investment Models," *AIIE Transactions*, vol.4, no.1, March 1972, pp.57-64.

Souder, William E., "Analytical Effectiveness of Mathematical Models for R&D Project Selection," *Management Science*, vol.19, no.8, April 1973, pp.907-923.

Souder, William E., "Utility and Perceived Acceptability of R&D Project Selection Models," *Management Science*, vol.19, no.12, August 1973, pp.1384-1394.

Taylor, Bernard W. III, and Moore, Laurence J., "R&D Project Planning with Q-GERT Network Modeling and Simulation," *Management Science*, vol.26, no.1, 1980, pp.44-59.

Taylor, Bernard W. III, Moore, Laurence J., and Clayton, Edward R., "R&D Project Selection and Manpower Allocation with Nonlinear Goal Programming," *Management Science*, vol.28, no.10, October 1982, pp.1149-1158.

Watters, L.D., "Research and development project selection: interdependence and multi-period probabilistic budget constraints", Ph.D. dissertation, Arizona State University, Tempe, Az.,1967.

Winofsky,E.P., Baker,N.R., and Sweeney,D.J., "A Decision Process Model of R & D Resource Allocation in Hierarchical Organizations," *Management Science*, vol.27, no.3, March 1981.

Winston, Patrick Henry, *Artificial Intelligence*, Addison-Wesley, Reading, Massachusetts, 1984.

Appendix A. Pascal Program Listings

This is the program listing for the filtered beam search program. It filters on project success, then filters on schedlue times. Lastly, the program selects based on expected profit. It runs for the 50 project set.

```
program temporary_filtered_beam_50_projects;
{
This program will select and schedule r&d projects
for the dissertation topic of Mark A. Coffin
using a filtered beam search}

uses crt;
{$M 32768,0,655360}
 type
    operations = (a1,b1,c1,a2,b2,c2,a3,b3,c3,a4,b4,c4,a5,b5,c5,
                  a6,b6,c6,a7,b7,c7,a8,b8,c8,a9,b9,c9,a10,b10,c10,
                  a11,b11,c11,a12,b12,c12,a13,b13,c13,a14,b14,c14,
                  a15,b15,c15,a16,b16,c16,a17,b17,c17,a18,b18,c18,
                  a19,b19,c19,a20,b20,c20,a21,b21,c21,a22,b22,c22,
                  a23,b23,c23,a24,b24,c24,a25,b25,c25,a26,b26,c26,
                  a27,b27,c27,a28,b28,c28,a29,b29,c29,a30,b30,c30,
                  a31,b31,c31,a32,b32,c32,a33,b33,c33,a34,b34,c34,
                  a35,b35,c35,a36,b36,c36,a37,b37,c37,a38,b38,c38,
                  a39,b39,c39,a40,b40,c40,a41,b41,c41,a42,b42,c42,
                  a43,b43,c43,a44,b44,c44,a45,b45,c45,a46,b46,c46,
                  a47,b47,c47,a48,b48,c48,a49,b49,c49,a50,b50,c50,none);

     projdata = record
                number,budget,profit_pot  :  real;
                p_success                 :  real;
                num_research,a_const,b_const :  real;
                seq_1,seq_2,seq_3            : operations;
                tseq_1,tseq_2,tseq_3         : real;
            end;{ of record type}

     limits = record
                 budget_lim,nres_lim,a_const_lim : real;
                 b_const_lim                     : real;
                 end;{ of record limits}

        set_of_projects = set of 1..50;
        proj_block=array[1..50] of projdata;
        choices = array[1..10] of set_of_projects;
        potential_choices = array [1..500] of set_of_projects;
        filter_choices = array[1..50] of set_of_projects;
        start_set = array[1..50] of 1..10;
        array_block_1 = array [1..500] of real;
        array_block_2 = array [1..100] of real;
        array_block_3 = array [1..500] of integer;

var
     projects : text;
     {information about the projects are stored in records of
```

```
      projdata }
      proj      : projdata;(this variable is a holdong variable while
                  reading and assigning values to projinfo}
      projinfo  : proj_block;
      potential_projects     : potential_choices;
      projects_selected      : choices;
      filterlist1,filterlist2           :filter_choices;
      i,j,k,beamwidth,numprojects,filterwidth1,filterwidth2 :integer;
      constraints          : limits;
      pot_expected_profit,potential_project_time : array_block_1;
      startprofit,starting_times : array_block_2;
      starting_projects   : start_set;
      project_time,expected_profit        : array_block_2;
      start,finished : boolean;


 procedure initialize(var dummyselect:choices;
                      var dummypotential: potential_choices;
                      var pep:array_block_1;
                      var ppt:array_block_1;var ep:array_block_2;
                      var pt:array_block_2;var flist:filter_choices);
 var i,j                : integer;

  begin
   for i:=1 to 10 do
    dummyselect[i]:=[];
   for i:=1 to 10 do
    begin
    ep[i]:=0;
    pt[i]:=0;
    end;{for i}
   for i:=1 to 20 do
      flist[i]:=[];
   for j:=1 to 500 do
      begin
      dummypotential[j]:=[];
      pep[j]:=0;
      ppt[j]:=0;
      end;{for j..}
    end;{procedure initialize}



 procedure project_input(var dummy:proj_block);
 {this procedure will input the project attributes
 for the selection and scheduling routines}

   var
      i,j,k1                              :integer;
      bud,profit,nresearch,a,b            : real;
      ts1,ts2,ts3,num                     : real;
      psuccess                            : real;
      s1,s2,s3                            : string[3];
      projects                            : text;

 begin
```

```
k1:=1;
i:=1;
assign(projects,'data50.dat');
reset(projects);
   while not eof(projects) do
   begin
         read (projects,num);
         readln(projects,bud,profit,psuccess,a,b,nresearch);
         readln (projects,s1);
         readln (projects,s2);
         readln (projects,s3);
         readln (projects,ts1,ts2,ts3);
         with dummy[i] do
              begin
              number:=num;
              budget:=bud;
              profit_pot:=profit;
              p_success:=psuccess;
              num_research:=nresearch;
              a_const:=a;
              b_const:=b;
              if s1 = 'a1' then seq_1:=a1;
              if s1 = 'a2' then seq_1:=a2;
              if s1 = 'a3' then seq_1:=a3;
              if s1 = 'a4' then seq_1:=a4;
              if s1 = 'a5' then seq_1:=a5;
              if s1 = 'a6' then seq_1:=a6;
              if s1 = 'a7' then seq_1:=a7;
              if s1 = 'a8' then seq_1:=a8;
              if s1 = 'a9' then seq_1:=a9;
              if s1 = 'a10' then seq_1:=a10;
              if s1 = 'a11' then seq_1:=a11;
              if s1 = 'a12' then seq_1:=a12;
              if s1 = 'a13' then seq_1:=a13;
              if s1 = 'a14' then seq_1:=a14;
              if s1 = 'a15' then seq_1:=a15;
              if s1 = 'a16' then seq_1:=a16;
              if s1 = 'a17' then seq_1:=a17;
              if s1 = 'a18' then seq_1:=a18;
              if s1 = 'a19' then seq_1:=a19;
              if s1 = 'a20' then seq_1:=a20;
              if s1 = 'a21' then seq_1:=a21;
              if s1 = 'a22' then seq_1:=a22;
              if s1 = 'a23' then seq_1:=a23;
              if s1 = 'a24' then seq_1:=a24;
              if s1 = 'a25' then seq_1:=a25;
              if s1 = 'a26' then seq_1:=a26;
              if s1 = 'a27' then seq_1:=a27;
              if s1 = 'a28' then seq_1:=a28;
              if s1 = 'a29' then seq_1:=a29;
              if s1 = 'a30' then seq_1:=a30;
              if s1 = 'a31' then seq_1:=a31;
              if s1 = 'a32' then seq_1:=a32;
              if s1 = 'a33' then seq_1:=a33;
              if s1 = 'a34' then seq_1:=a34;
              if s1 = 'a35' then seq_1:=a35;
              if s1 = 'a36' then seq_1:=a36;
if s1 = 'a37' then seq_1:=a37;
if s1 = 'a38' then seq_1:=a38;
if s1 = 'a39' then seq_1:=a39;
if s1 = 'a40' then seq_1:=a40;
if s1 = 'a41' then seq_1:=a41;
if s1 = 'a42' then seq_1:=a42;
if s1 = 'a43' then seq_1:=a43;
if s1 = 'a44' then seq_1:=a44;
if s1 = 'a45' then seq_1:=a45;
if s1 = 'a46' then seq_1:=a46;
if s1 = 'a47' then seq_1:=a47;
if s1 = 'a48' then seq_1:=a48;
if s1 = 'a49' then seq_1:=a49;
if s1 = 'a50' then seq_1:=a50;
if s1 = 'b1' then seq_1:=b1;
if s1 = 'b2' then seq_1:=b2;
if s1 = 'b3' then seq_1:=b3;
if s1 = 'b4' then seq_1:=b4;
if s1 = 'b5' then seq_1:=b5;
if s1 = 'b6' then seq_1:=b6;
if s1 = 'b7' then seq_1:=b7;
if s1 = 'b8' then seq_1:=b8;
if s1 = 'b9' then seq_1:=b9;
if s1 = 'b10' then seq_1:=b10;
if s1 = 'b11' then seq_1:=b11;
if s1 = 'b12' then seq_1:=b12;
if s1 = 'b13' then seq_1:=b13;
if s1 = 'b14' then seq_1:=b14;
if s1 = 'b15' then seq_1:=b15;
if s1 = 'b16' then seq_1:=b16;
if s1 = 'b17' then seq_1:=b17;
if s1 = 'b18' then seq_1:=b18;
if s1 = 'b19' then seq_1:=b19;
if s1 = 'b20' then seq_1:=b20;
if s1 = 'b21' then seq_1:=b21;
if s1 = 'b22' then seq_1:=b22;
if s1 = 'b23' then seq_1:=b23;
if s1 = 'b24' then seq_1:=b24;
if s1 = 'b25' then seq_1:=b25;
if s1 = 'b26' then seq_1:=b26;
if s1 = 'b27' then seq_1:=b27;
if s1 = 'b28' then seq_1:=b28;
if s1 = 'b29' then seq_1:=b29;
if s1 = 'b30' then seq_1:=b30;
if s1 = 'b31' then seq_1:=b31;
if s1 = 'b32' then seq_1:=b32;
if s1 = 'b33' then seq_1:=b33;
if s1 = 'b34' then seq_1:=b34;
if s1 = 'b35' then seq_1:=b35;
if s1 = 'b36' then seq_1:=b36;
if s1 = 'b37' then seq_1:=b37;
if s1 = 'b38' then seq_1:=b38;
if s1 = 'b39' then seq_1:=b39;
if s1 = 'b40' then seq_1:=b40;
if s1 = 'b41' then seq_1:=b41;
if s1 = 'b42' then seq_1:=b42;
if s1 = 'b43' then seq_1:=b43;
if s1 = 'b44' then seq_1:=b44;
if s1 = 'b45' then seq_1:=b45;
if s1 = 'b46' then seq_1:=b46;
if s1 = 'b47' then seq_1:=b47;
if s1 = 'b48' then seq_1:=b48;
if s1 = 'b49' then seq_1:=b49;
if s1 = 'b50' then seq_1:=b50;
if s1 = 'c1' then seq_1:=c1;
if s1 = 'c2' then seq_1:=c2;
if s1 = 'c3' then seq_1:=c3;
if s1 = 'c4' then seq_1:=c4;
if s1 = 'c5' then seq_1:=c5;
if s1 = 'c6' then seq_1:=c6;
if s1 = 'c7' then seq_1:=c7;
if s1 = 'c8' then seq_1:=c8;
if s1 = 'c9' then seq_1:=c9;
if s1 = 'c10' then seq_1:=c10;
if s1 = 'c11' then seq_1:=c11;
if s1 = 'c12' then seq_1:=c12;
if s1 = 'c13' then seq_1:=c13;
if s1 = 'c14' then seq_1:=c14;
if s1 = 'c15' then seq_1:=c15;
if s1 = 'c16' then seq_1:=c16;
if s1 = 'c17' then seq_1:=c17;
if s1 = 'c18' then seq_1:=c18;
if s1 = 'c19' then seq_1:=c19;
if s1 = 'c20' then seq_1:=c20;
if s1 = 'c21' then seq_1:=c21;
if s1 = 'c22' then seq_1:=c22;
if s1 = 'c23' then seq_1:=c23;
if s1 = 'c24' then seq_1:=c24;
if s1 = 'c25' then seq_1:=c25;
if s1 = 'c26' then seq_1:=c26;
if s1 = 'c27' then seq_1:=c27;
if s1 = 'c28' then seq_1:=c28;
if s1 = 'c29' then seq_1:=c29;
if s1 = 'c30' then seq_1:=c30;
if s1 = 'c31' then seq_1:=c31;
if s1 = 'c32' then seq_1:=c32;
if s1 = 'c33' then seq_1:=c33;
if s1 = 'c34' then seq_1:=c34;
if s1 = 'c35' then seq_1:=c35;
if s1 = 'c36' then seq_1:=c36;
if s1 = 'c37' then seq_1:=c37;
if s1 = 'c38' then seq_1:=c38;
if s1 = 'c39' then seq_1:=c39;
if s1 = 'c40' then seq_1:=c40;
if s1 = 'c41' then seq_1:=c41;
if s1 = 'c42' then seq_1:=c42;
if s1 = 'c43' then seq_1:=c43;
if s1 = 'c44' then seq_1:=c44;
if s1 = 'c45' then seq_1:=c45;
if s1 = 'c46' then seq_1:=c46;
if s1 = 'c47' then seq_1:=c47;
if s1 = 'c48' then seq_1:=c48;
if s1 = 'c49' then seq_1:=c49;
if s1 = 'c50' then seq_1:=c50;
if s2 = 'a1' then seq_2:=a1;
if s2 = 'a2' then seq_2:=a2;
if s2 = 'a3' then seq_2:=a3;
if s2 = 'a4' then seq_2:=a4;
if s2 = 'a5' then seq_2:=a5;
if s2 = 'a6' then seq_2:=a6;
if s2 = 'a7' then seq_2:=a7;
if s2 = 'a8' then seq_2:=a8;
if s2 = 'a9' then seq_2:=a9;
if s2 = 'a10' then seq_2:=a10;
if s2 = 'a11' then seq_2:=a11;
if s2 = 'a12' then seq_2:=a12;
if s2 = 'a13' then seq_2:=a13;
if s2 = 'a14' then seq_2:=a14;
if s2 = 'a15' then seq_2:=a15;
if s2 = 'a16' then seq_2:=a16;
if s2 = 'a17' then seq_2:=a17;
if s2 = 'a18' then seq_2:=a18;
if s2 = 'a19' then seq_2:=a19;
if s2 = 'a20' then seq_2:=a20;
if s2 = 'a21' then seq_2:=a21;
if s2 = 'a22' then seq_2:=a22;
if s2 = 'a23' then seq_2:=a23;
if s2 = 'a24' then seq_2:=a24;
if s2 = 'a25' then seq_2:=a25;
if s2 = 'a26' then seq_2:=a26;
if s2 = 'a27' then seq_2:=a27;
if s2 = 'a28' then seq_2:=a28;
if s2 = 'a29' then seq_2:=a29;
if s2 = 'a30' then seq_2:=a30;
if s2 = 'a31' then seq_2:=a31;
if s2 = 'a32' then seq_2:=a32;
if s2 = 'a33' then seq_2:=a33;
if s2 = 'a34' then seq_2:=a34;
if s2 = 'a35' then seq_2:=a35;
if s2 = 'a36' then seq_2:=a36;
if s2 = 'a37' then seq_2:=a37;
if s2 = 'a38' then seq_2:=a38;
if s2 = 'a39' then seq_2:=a39;
if s2 = 'a40' then seq_2:=a40;
if s2 = 'a41' then seq_2:=a41;
if s2 = 'a42' then seq_2:=a42;
if s2 = 'a43' then seq_2:=a43;
if s2 = 'a44' then seq_2:=a44;
if s2 = 'a45' then seq_2:=a45;
if s2 = 'a46' then seq_2:=a46;
if s2 = 'a47' then seq_2:=a47;
if s2 = 'a48' then seq_2:=a48;
if s2 = 'a49' then seq_2:=a49;
if s2 = 'a50' then seq_2:=a50;

if s2 = 'b1' then seq_2:=b1;
if s2 = 'b2' then seq_2:=b2;
if s2 = 'b3' then seq_2:=b3;
if s2 = 'b4' then seq_2:=b4;
if s2 = 'b5' then seq_2:=b5;
if s2 = 'b6' then seq_2:=b6;
if s2 = 'b7' then seq_2:=b7;
if s2 = 'b8' then seq_2:=b8;
if s2 = 'b9' then seq_2:=b9;
if s2 = 'b10' then seq_2:=b10;
if s2 = 'b11' then seq_2:=b11;
if s2 = 'b12' then seq_2:=b12;
if s2 = 'b13' then seq_2:=b13;
if s2 = 'b14' then seq_2:=b14;
if s2 = 'b15' then seq_2:=b15;
if s2 = 'b16' then seq_2:=b16;
if s2 = 'b17' then seq_2:=b17;
if s2 = 'b18' then seq_2:=b18;
if s2 = 'b19' then seq_2:=b19;
if s2 = 'b20' then seq_2:=b20;
if s2 = 'b21' then seq_2:=b21;
if s2 = 'b22' then seq_2:=b22;
if s2 = 'b23' then seq_2:=b23;
if s2 = 'b24' then seq_2:=b24;
if s2 = 'b25' then seq_2:=b25;
if s2 = 'b26' then seq_2:=b26;
if s2 = 'b27' then seq_2:=b27;
if s2 = 'b28' then seq_2:=b28;
if s2 = 'b29' then seq_2:=b29;
if s2 = 'b30' then seq_2:=b30;
if s2 = 'b31' then seq_2:=b31;
if s2 = 'b32' then seq_2:=b32;
if s2 = 'b33' then seq_2:=b33;
if s2 = 'b34' then seq_2:=b34;
if s2 = 'b35' then seq_2:=b35;
if s2 = 'b36' then seq_2:=b36;
if s2 = 'b37' then seq_2:=b37;
if s2 = 'b38' then seq_2:=b38;
if s2 = 'b39' then seq_2:=b39;
if s2 = 'b40' then seq_2:=b40;
if s2 = 'b41' then seq_2:=b41;
if s2 = 'b42' then seq_2:=b42;
if s2 = 'b43' then seq_2:=b43;
if s2 = 'b44' then seq_2:=b44;
if s2 = 'b45' then seq_2:=b45;
if s2 = 'b46' then seq_2:=b46;
if s2 = 'b47' then seq_2:=b47;
if s2 = 'b48' then seq_2:=b48;
if s2 = 'b49' then seq_2:=b49;
if s2 = 'b50' then seq_2:=b50;

if s2 = 'c1' then seq_2:=c1;
if s2 = 'c2' then seq_2:=c2;
if s2 = 'c3' then seq_2:=c3;
if s2 = 'c4' then seq_2:=c4;
if s2 = 'c5' then seq_2:=c5;
if s2 = 'c6' then seq_2:=c6;
if s2 = 'c7' then seq_2:=c7;
if s2 = 'c8' then seq_2:=c8;
if s2 = 'c9' then seq_2:=c9;
if s2 = 'c10' then seq_2:=c10;
if s2 = 'c11' then seq_2:=c11;
if s2 = 'c12' then seq_2:=c12;
if s2 = 'c13' then seq_2:=c13;
if s2 = 'c14' then seq_2:=c14;
if s2 = 'c15' then seq_2:=c15;
if s2 = 'c16' then seq_2:=c16;
if s2 = 'c17' then seq_2:=c17;
if s2 = 'c18' then seq_2:=c18;
if s2 = 'c19' then seq_2:=c19;
if s2 = 'c20' then seq_2:=c20;
if s2 = 'c21' then seq_2:=c21;
if s2 = 'c22' then seq_2:=c22;
if s2 = 'c23' then seq_2:=c23;
if s2 = 'c24' then seq_2:=c24;
if s2 = 'c25' then seq_2:=c25;
if s2 = 'c26' then seq_2:=c26;
if s2 = 'c27' then seq_2:=c27;
if s2 = 'c28' then seq_2:=c28;
if s2 = 'c29' then seq_2:=c29;
if s2 = 'c30' then seq_2:=c30;
if s2 = 'c31' then seq_2:=c31;
if s2 = 'c32' then seq_2:=c32;
if s2 = 'c33' then seq_2:=c33;
if s2 = 'c34' then seq_2:=c34;
if s2 = 'c35' then seq_2:=c35;
if s2 = 'c36' then seq_2:=c36;
if s2 = 'c37' then seq_2:=c37;
if s2 = 'c38' then seq_2:=c38;
if s2 = 'c39' then seq_2:=c39;
if s2 = 'c40' then seq_2:=c40;
if s2 = 'c41' then seq_2:=c41;
if s2 = 'c42' then seq_2:=c42;
if s2 = 'c43' then seq_2:=c43;
if s2 = 'c44' then seq_2:=c44;
if s2 = 'c45' then seq_2:=c45;
if s2 = 'c46' then seq_2:=c46;
if s2 = 'c47' then seq_2:=c47;
if s2 = 'c48' then seq_2:=c48;
if s2 = 'c49' then seq_2:=c49;
if s2 = 'c50' then seq_2:=c50;

if s3 = 'a1' then seq_3:=a1;
if s3 = 'a2' then seq_3:=a2;
if s3 = 'a3' then seq_3:=a3;
if s3 = 'a4' then seq_3:=a4;
if s3 = 'a5' then seq_3:=a5;
if s3 = 'a6' then seq_3:=a6;
if s3 = 'a7' then seq_3:=a7;
if s3 = 'a8' then seq_3:=a8;
if s3 = 'a9' then seq_3:=a9;
if s3 = 'a10' then seq_3:=a10;
if s3 = 'a11' then seq_3:=a11;
if s3 = 'a12' then seq_3:=a12;
if s3 = 'a13' then seq_3:=a13;
if s3 = 'a14' then seq_3:=a14;
if s3 = 'a15' then seq_3:=a15;
if s3 = 'a16' then seq_3:=a16;
if s3 = 'a17' then seq_3:=a17;
if s3 = 'a18' then seq_3:=a18;
if s3 = 'a19' then seq_3:=a19;
if s3 = 'a20' then seq_3:=a20;
if s3 = 'a21' then seq_3:=a21;
if s3 = 'a22' then seq_3:=a22;
if s3 = 'a23' then seq_3:=a23;
if s3 = 'a24' then seq_3:=a24;
if s3 = 'a25' then seq_3:=a25;
if s3 = 'a26' then seq_3:=a26;
if s3 = 'a27' then seq_3:=a27;
if s3 = 'a28' then seq_3:=a28;
if s3 = 'a29' then seq_3:=a29;
if s3 = 'a30' then seq_3:=a30;
if s3 = 'a31' then seq_3:=a31;
if s3 = 'a32' then seq_3:=a32;
if s3 = 'a33' then seq_3:=a33;
if s3 = 'a34' then seq_3:=a34;
if s3 = 'a35' then seq_3:=a35;
if s3 = 'a36' then seq_3:=a36;
if s3 = 'a37' then seq_3:=a37;
if s3 = 'a38' then seq_3:=a38;
if s3 = 'a39' then seq_3:=a39;
if s3 = 'a40' then seq_3:=a40;
if s3 = 'a41' then seq_3:=a41;
if s3 = 'a42' then seq_3:=a42;
if s3 = 'a43' then seq_3:=a43;
if s3 = 'a44' then seq_3:=a44;
if s3 = 'a45' then seq_3:=a45;
if s3 = 'a46' then seq_3:=a46;
if s3 = 'a47' then seq_3:=a47;
if s3 = 'a48' then seq_3:=a48;
if s3 = 'a49' then seq_3:=a49;
if s3 = 'a50' then seq_3:=a50;

if s3 = 'b1' then seq_3:=b1;
if s3 = 'b2' then seq_3:=b2;
if s3 = 'b3' then seq_3:=b3;
if s3 = 'b4' then seq_3:=b4;
if s3 = 'b5' then seq_3:=b5;
if s3 = 'b6' then seq_3:=b6;
if s3 = 'b7' then seq_3:=b7;
if s3 = 'b8' then seq_3:=b8;
if s3 = 'b9' then seq_3:=b9;
if s3 = 'b10' then seq_3:=b10;
if s3 = 'b11' then seq_3:=b11;
if s3 = 'b12' then seq_3:=b12;
if s3 = 'b13' then seq_3:=b13;
if s3 = 'b14' then seq_3:=b14;
if s3 = 'b15' then seq_3:=b15;
if s3 = 'b16' then seq_3:=b16;
if s3 = 'b17' then seq_3:=b17;
if s3 = 'b18' then seq_3:=b18;
if s3 = 'b19' then seq_3:=b19;
if s3 = 'b20' then seq_3:=b20;
if s3 = 'b21' then seq_3:=b21;
if s3 = 'b22' then seq_3:=b22;
if s3 = 'b23' then seq_3:=b23;
if s3 = 'b24' then seq_3:=b24;
if s3 = 'b25' then seq_3:=b25;
if s3 = 'b26' then seq_3:=b26;
if s3 = 'b27' then seq_3:=b27;
if s3 = 'b28' then seq_3:=b28;
if s3 = 'b29' then seq_3:=b29;
if s3 = 'b30' then seq_3:=b30;
if s3 = 'b31' then seq_3:=b31;
if s3 = 'b32' then seq_3:=b32;
if s3 = 'b33' then seq_3:=b33;
if s3 = 'b34' then seq_3:=b34;
if s3 = 'b35' then seq_3:=b35;
if s3 = 'b36' then seq_3:=b36;
if s3 = 'b37' then seq_3:=b37;
if s3 = 'b38' then seq_3:=b38;
if s3 = 'b39' then seq_3:=b39;
if s3 = 'b40' then seq_3:=b40;
if s3 = 'b41' then seq_3:=b41;
if s3 = 'b42' then seq_3:=b42;
if s3 = 'b43' then seq_3:=b43;
if s3 = 'b44' then seq_3:=b44;
if s3 = 'b45' then seq_3:=b45;
if s3 = 'b46' then seq_3:=b46;
if s3 = 'b47' then seq_3:=b47;
if s3 = 'b48' then seq_3:=b48;
if s3 = 'b49' then seq_3:=b49;
if s3 = 'b50' then seq_3:=b50;

if s3 = 'c1' then seq_3:=c1;
if s3 = 'c2' then seq_3:=c2;
if s3 = 'c3' then seq_3:=c3;
if s3 = 'c4' then seq_3:=c4;
if s3 = 'c5' then seq_3:=c5;
if s3 = 'c6' then seq_3:=c6;
if s3 = 'c7' then seq_3:=c7;
if s3 = 'c8' then seq_3:=c8;
if s3 = 'c9' then seq_3:=c9;
if s3 = 'c10' then seq_3:=c10;
if s3 = 'c11' then seq_3:=c11;
if s3 = 'c12' then seq_3:=c12;
if s3 = 'c13' then seq_3:=c13;
if s3 = 'c14' then seq_3:=c14;
if s3 = 'c15' then seq_3:=c15;
if s3 = 'c16' then seq_3:=c16;
if s3 = 'c17' then seq_3:=c17;
if s3 = 'c18' then seq_3:=c18;
if s3 = 'c19' then seq_3:=c19;
if s3 = 'c20' then seq_3:=c20;
if s3 = 'c21' then seq_3:=c21;
if s3 = 'c22' then seq_3:=c22;
if s3 = 'c23' then seq_3:=c23;
if s3 = 'c24' then seq_3:=c24;
if s3 = 'c25' then seq_3:=c25;
if s3 = 'c26' then seq_3:=c26;
if s3 = 'c27' then seq_3:=c27;
if s3 = 'c28' then seq_3:=c28;
if s3 = 'c29' then seq_3:=c29;
```

```
                    if s3 = 'c30' then seq_3:=c30;
                    if s3 = 'c31' then seq_3:=c31;
                    if s3 = 'c32' then seq_3:=c32;
                    if s3 = 'c33' then seq_3:=c33;
                    if s3 = 'c34' then seq_3:=c34;
                    if s3 = 'c35' then seq_3:=c35;
                    if s3 = 'c36' then seq_3:=c36;
                    if s3 = 'c37' then seq_3:=c37;
                    if s3 = 'c38' then seq_3:=c38;
                    if s3 = 'c39' then seq_3:=c39;
                    if s3 = 'c40' then seq_3:=c40;
                    if s3 = 'c41' then seq_3:=c41;
                    if s3 = 'c42' then seq_3:=c42;
                    if s3 = 'c43' then seq_3:=c43;
                    if s3 = 'c44' then seq_3:=c44;
                    if s3 = 'c45' then seq_3:=c45;
                    if s3 = 'c46' then seq_3:=c46;
                    if s3 = 'c47' then seq_3:=c47;
                    if s3 = 'c48' then seq_3:=c48;
                    if s3 = 'c49' then seq_3:=c49;
                    if s3 = 'c50' then seq_3:=c50;

                    tseq_1:=ts1;
                    tseq_2:=ts2;
                    tseq_3:=ts3;
                    end; {with}
                    i:=i+1;
           end;{while eof statement}
           close(projects);
end;{of procedure project_input}

procedure setlimits(var dummy : limits);
  var
     budlim,researchlim,alim,blim          : real;
     working                               :text;

begin {procedure}
      assign(working,'limits50.dat');
      reset(working);
      readln(working,budlim,researchlim,alim,blim);
      with dummy do
        begin
        budget_lim:=budlim;
        nres_lim:=researchlim;
        a_const_lim:=alim;
        b_const_lim:=blim;
        end;{with}
       close(working);
end;{procedure setlimits}


procedure search_input(var bw,fw1,fw2,np:integer);
{this procedure will input the search attributes
such as beamwidth,number of projects,filterwidth,and others}
     begin
      writeln('input the beamwidth of the search less than 5');
      readln(bw);
      writeln(' input the filterwidthl of the search <= 50');
      readln(fwl);
      writeln('input the second filterwidth <= fwl');
      readln(fw2);
      writeln(' input the number of projects ');
      readln(np);

     end;

procedure generate_next_level(var projects_selected :choices;
                              var potential_projects:potential_choices;
                              var starting_projects:start_set;
                                  beamwidth,numprojects:integer;
                                  constraints : limits;
                                  projinfo : proj_block;
                                  var k : integer;start : boolean );
{this procedure will generate the next level of tree
to search}

procedure check_and_see ( dps:set_of_projects;pp:potential_choices;
                          k:integer; var inset:boolean);

var m,n : integer;

begin
  inset:=false;
  for n:=1 to k do
        if dps=pp[n] then
          inset:=true;

end; {procedure check and see}

procedure check_feasibility (dps:set_of_projects; cons:limits;
                             numprojects:integer; pinfo:proj_block;
                             var feas:boolean);

var i:integer;
    nrtot,atot,btot,budtot : real;

begin
 feas:=true;
 nrtot:=0;
 atot:=0;
 btot:=0;
 budtot:=0;

for i:=1 to numprojects do
  begin
  if i in dps then
    begin
    nrtot:=nrtot+pinfo[i].num_research;
    atot:=atot+pinfo[i].a_const;
    btot:=btot+pinfo[i].b_const;
    budtot:=budtot+pinfo[i].budget;
    end;
  end;{for loop}
```

```
if nrtot > constraints.nres_lim then feas:=false;
if atot > constraints.a_const_lim then feas:=false;
if btot > constraints.b_const_lim then feas:=false;
if budtot > constraints.budget_lim then feas:=false;

end; (procedure check feasibility)




var i,j,l     : integer;
    inset,feasible : boolean;

    dummy_project_set   : set_of_projects;

begin
  k:=0;(this is the global counter I use to keep track of the number of
        potential nodes at each level.  It is only modified in the procedure
        and is used in others)

  if start then (no projects have been selected yet)
    begin
    for i:=1 to numprojects do
      starting_projects[i]:= i;
    end (if start)

   else ( do the rest of the procedure)
     begin
     writeln(' im in the generate next level procedure and k = ',k);
     for i:=1 to beamwidth do
      begin
      for j:=1 to numprojects do
       begin
       if (j in projects_selected[i]) = false then
        begin
        dummy_project_set:=projects_selected[i] + [j];
        check_and_see(dummy_project_set,potential_projects,k,inset);
        if inset = false then
          begin
          check_feasibility(dummy_project_set,constraints,numprojects,projinfo,
                            feasible);
          if feasible = true then
           begin
           k:=k+1;(only time k is ever altered)
           potential_projects[k]:=dummy_project_set;
           end;( if feasible)
          end;( if inset)
        end;( if j in projselected)
      end;(for j)
      end;(for i)
    end;(else)
  end; (procedure generate next level)


procedure compute_expected_profit(var dummye_profit : array_block_1;
                                  var startprofit   : array_block_2;
```

```
                                          dummyinfo : proj_block;
                                          dummypotential : potential_choices;
                                          beamwidth,numprojects : integer;
                                          projects_selected : choices;
                                          k:integer; start : boolean);

var i,j  : integer;
    totalprofit : real;

begin {procedure}
  if start then
     for i:=1 to numprojects do
          startprofit[i]:=dummyinfo[i].profit_pot * dummyinfo[i].p_success;


  if start=false then  {else go to end of procedure}
    begin
    for i:=1 to k do
      begin
      totalprofit:=0;
      for j:=1 to numprojects do
            begin
            if j in dummypotential[i] then
               totalprofit:=totalprofit+(dummyinfo[j].profit_pot*dummyinfo[j].p_
            end;{for j loop}
      dummye_profit[i]:=totalprofit;
     end; {for i loop}
  end; {if start=false}
end; {procedure expected profit}

{ now there is an expected profit computed for each potential choice
 except those that have been set to 0 because they are empty sets.}


procedure scheduler( var starting_times:array_block_2; var potential_project_tim
                         array_block_1;dummypotential:potential_choices;
                         dummyselect:choices; beamwidth:
                         integer; numprojects : integer; projinfo : proj_block;
                         k:integer);





{this procedure will generate schedules according
to a scheduling heuristic and determine time to completion
of all projects.  This procedure can be modified later to
solve for other scheduling requirements.}

type
     steptype = (one,two,three);
     operations_array = array[operations] of real;
     AA = array[1..50] of real;
     XX = set of operations;
     YY = array[1..100] of operations;{this is the schedule, in order, for each
     ZZ = array[1..50] of steptype;{tells me what step each project is on}
     holdingset = set of 1..50;


var  psindex,i,j,l : integer;
     s_sub_t : XX;
     sigma_j : operations_array;
     totaltime,sigma_star,t,fa_avail,fb_avail,fc_avail   :  real;
     nextoperation,counter : operations;
     ps_sub_t : YY;
     holding : holdingset;
     predfinish : AA;
     steps : ZZ;  {tells me which step the project is in at any time}
     done : boolean;

  procedure find_sigma_j (var sigma_j : operations_array;
                              s_sub_t : XX;
                              predfinish : AA;
                              fa_avail,fb_avail,fc_avail : real);

     var
     counter2 : operations;
     pf,favail : real;

       begin {subprocedure find_sigma_j}
         for counter2:=a1 to c50 do
           begin
           if counter2 in s_sub_t then
             begin
             {now find the project # and use info about the step its on}
             case counter2 of
              a1,b1,c1 : pf:=predfinish[1];{these are updated when ps is updated
              a2,b2,c2 : pf:=predfinish[2];
              a3,b3,c3 : pf:=predfinish[3];
              a4,b4,c4 : pf:=predfinish[4];
              a5,b5,c5 : pf:=predfinish[5];
              a6,b6,c6 : pf:=predfinish[6];
              a7,b7,c7 : pf:=predfinish[7];
              a8,b8,c8 : pf:=predfinish[8];
              a9,b9,c9 : pf:=predfinish[9];
              a10,b10,c10 : pf:=predfinish[10];
              a11,b11,c11 : pf:=predfinish[11];{these are updated when ps is upd
              a12,b12,c12 : pf:=predfinish[12];
              a13,b13,c13 : pf:=predfinish[13];
              a14,b14,c14 : pf:=predfinish[14];
              a15,b15,c15 : pf:=predfinish[15];
              a16,b16,c16 : pf:=predfinish[16];
              a17,b17,c17 : pf:=predfinish[17];
              a18,b18,c18 : pf:=predfinish[18];
              a19,b19,c19 : pf:=predfinish[19];
              a20,b20,c20 : pf:=predfinish[20];
              a21,b21,c21 : pf:=predfinish[21];{these are updated when ps is upd
              a22,b22,c22 : pf:=predfinish[22];
              a23,b23,c23 : pf:=predfinish[23];
              a24,b24,c24 : pf:=predfinish[24];
              a25,b25,c25 : pf:=predfinish[25];
              a26,b26,c26 : pf:=predfinish[26];
              a27,b27,c27 : pf:=predfinish[27];
              a28,b28,c28 : pf:=predfinish[28];
              a29,b29,c29 : pf:=predfinish[29];
              a30,b30,c30 : pf:=predfinish[30];
              a31,b31,c31 : pf:=predfinish[31];{these are updated when ps is upd
              a32,b32,c32 : pf:=predfinish[32];
              a33,b33,c33 : pf:=predfinish[33];
              a34,b34,c34 : pf:=predfinish[34];
              a35,b35,c35 : pf:=predfinish[35];
              a36,b36,c36 : pf:=predfinish[36];
              a37,b37,c37 : pf:=predfinish[37];
              a38,b38,c38 : pf:=predfinish[38];
              a39,b39,c39 : pf:=predfinish[39];
              a40,b40,c40 : pf:=predfinish[40];
              a41,b41,c41 : pf:=predfinish[41];{these are updated when ps is upd
              a42,b42,c42 : pf:=predfinish[42];
              a43,b43,c43 : pf:=predfinish[43];
              a44,b44,c44 : pf:=predfinish[44];
              a45,b45,c45 : pf:=predfinish[45];
              a46,b46,c46 : pf:=predfinish[46];
              a47,b47,c47 : pf:=predfinish[47];
              a48,b48,c48 : pf:=predfinish[48];
              a49,b49,c49 : pf:=predfinish[49];
              a50,b50,c50 : pf:=predfinish[50];
              end;{case}

             case counter2 of
             a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,
             a11,a12,a13,a14,a15,a16,a17,a18,a19,a20,
             a21,a22,a23,a24,a25,a26,a27,a28,a29,a30,
             a31,a32,a33,a34,a35,a36,a37,a38,a39,a40,
             a41,a42,a43,a44,a45,a46,a47,a48,a49,a50  :  favail:=fa_avail;

             b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,
             b11,b12,b13,b14,b15,b16,b17,b18,b19,b20,
             b21,b22,b23,b24,b25,b26,b27,b28,b29,b30,
             b31,b32,b33,b34,b35,b36,b37,b38,b39,b40,
             b41,b42,b43,b44,b45,b46,b47,b48,b49,b50 : favail:=fb_avail;

             c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,
             c11,c12,c13,c14,c15,c16,c17,c18,c19,c20,
             c21,c22,c23,c24,c25,c26,c27,c28,c29,c30,
             c31,c32,c33,c34,c35,c36,c37,c38,c39,c40,
             c41,c42,c43,c44,c45,c46,c47,c48,c49,c50 : favail:=fc_avail;

             end;{case}

            if pf>= favail then
               sigma_j[counter2] := pf;

            if pf< favail then
               sigma_j[counter2] := favail;
          {writeln('sigma j is ',sigma_j[counter2]:5:2);}
        end;{if counter2 ...}
      end; {for counter2...}
    end; {subprocedure find sigma j}
```

```
procedure find_sigma_star (var sigma_star : real;
                               sigma_j : operations_array;
                               s_sub_t : XX);

   var
   counter3 : operations;
   smallest : real;

  begin {subprocedure find sigma star}
     smallest := 1000;
     for counter3 := a1 to c50 do
     begin
      if counter3 in s_sub_t then
        begin
        if sigma_j[counter3] <= smallest then
          smallest := sigma_j[counter3]
        end;{if counter3 ...}
     end;{for counter3...}
     sigma_star:=smallest;
  end;{subprocedure find sigma star}

procedure find_next_operation(sigma_star:real; sigma_j : operations_array;
                              steps : ZZ; s_sub_t : XX; projinfo : proj_block;
                              var nextoperation : operations);


var
smallest,spt : real;
counter4 : operations;
spt_operation : operations;
L : integer;

begin {procedure find next operation}

  smallest:=1000;
  for counter4:= a1 to c50 do
   begin
   if counter4 in s_sub_t then
    begin
    if sigma_j[counter4] = sigma_star then
      begin
       case counter4 of
       a1,b1,c1 : L:=1;
       a2,b2,c2 : L:=2;
       a3,b3,c3 : L:=3;
       a4,b4,c4 : L:=4;
       a5,b5,c5 : L:=5;
       a6,b6,c6 : L:=6;
       a7,b7,c7 : L:=7;
       a8,b8,c8 : L:=8;
       a9,b9,c9 : L:=9;
       a10,b10,c10 : L:=10;
       a11,b11,c11 : L:=11;
       a12,b12,c12 : L:=12;
       a13,b13,c13 : L:=13;
       a14,b14,c14 : L:=14;
       a15,b15,c15 : L:=15;
```

```
        a16,b16,c16 : L:=16;
        a17,b17,c17 : L:=17;
        a18,b18,c18 : L:=18;
        a19,b19,c19 : L:=19;
        a20,b20,c20 : L:=20;
        a21,b21,c21 : L:=21;
        a22,b22,c22 : L:=22;
        a23,b23,c23 : L:=23;
        a24,b24,c24 : L:=24;
        a25,b25,c25 : L:=25;
        a26,b26,c26 : L:=26;
        a27,b27,c27 : L:=27;
        a28,b28,c28 : L:=28;
        a29,b29,c29 : L:=29;
        a30,b30,c30 : L:=30;
        a31,b31,c31 : L:=31;
        a32,b32,c32 : L:=32;
        a33,b33,c33 : L:=33;
        a34,b34,c34 : L:=34;
        a35,b35,c35 : L:=35;
        a36,b36,c36 : L:=36;
        a37,b37,c37 : L:=37;
        a38,b38,c38 : L:=38;
        a39,b39,c39 : L:=39;
        a40,b40,c40 : L:=40;
        a41,b41,c41 : L:=41;
        a42,b42,c42 : L:=42;
        a43,b43,c43 : L:=43;
        a44,b44,c44 : L:=44;
        a45,b45,c45 : L:=45;
        a46,b46,c46 : L:=46;
        a47,b47,c47 : L:=47;
        a48,b48,c48 : L:=48;
        a49,b49,c49 : L:=49;
        a50,b50,c50 : L:=50;

        end; {case}

      case steps[L] of
        one : spt :=projinfo[L].tseq_1;
        two : spt :=projinfo[L].tseq_2;
        three : spt :=projinfo[L].tseq_3;
        end; {case}

      if spt< smallest then
       begin
        smallest:=spt;
        spt_operation:=counter4;
       end;{if spt}
     end; {if sigma_j}
    end; {if counter4...}
   end;{for counter4...}
   nextoperation:=spt_operation;
   end; {procedure find next operation}

Procedure update_ps_sub_t (var ps_sub_t : YY; nextoperation : operations;
                           var psindex : integer; var predfinish : AA;
```

```
                                    var fa_avail:real; var fb_avail : real;
                                    var fc_avail:real; steps : ZZ;
                                    projinfo : proj_block);




var
   PN : integer;
   projstep : steptype;
   duration : real;

  begin {procedure update ps sub t}

     duration :=0;
     ps_sub_t [psindex] := nextoperation;
     psindex := psindex+1;
       case nextoperation of
          a1,b1,c1 : PN:=1;
          a2,b2,c2 : PN:=2;
          a3,b3,c3 : PN:=3;
          a4,b4,c4 : PN:=4;
          a5,b5,c5 : PN:=5;
          a6,b6,c6 : PN:=6;
          a7,b7,c7 : PN:=7;
          a8,b8,c8 : PN:=8;
          a9,b9,c9 : PN:=9;
          a10,b10,c10 : PN:=10;
          a11,b11,c11 : PN:=11;
          a12,b12,c12 : PN:=12;
          a13,b13,c13 : PN:=13;
          a14,b14,c14 : PN:=14;
          a15,b15,c15 : PN:=15;
          a16,b16,c16 : PN:=16;
          a17,b17,c17 : PN:=17;
          a18,b18,c18 : PN:=18;
          a19,b19,c19 : PN:=19;
          a20,b20,c20 : PN:=20;
          a21,b21,c21 : PN:=21;
          a22,b22,c22 : PN:=22;
          a23,b23,c23 : PN:=23;
          a24,b24,c24 : PN:=24;
          a25,b25,c25 : PN:=25;
          a26,b26,c26 : PN:=26;
          a27,b27,c27 : PN:=27;
          a28,b28,c28 : PN:=28;
          a29,b29,c29 : PN:=29;
          a30,b30,c30 : PN:=30;
          a31,b31,c31 : PN:=31;
          a32,b32,c32 : PN:=32;
          a33,b33,c33 : PN:=33;
          a34,b34,c34 : PN:=34;
          a35,b35,c35 : PN:=35;
          a36,b36,c36 : PN:=36;
          a37,b37,c37 : PN:=37;
          a38,b38,c38 : PN:=38;
  a39,b39,c39 : PN:=39;
  a40,b40,c40 : PN:=40;
  a41,b41,c41 : PN:=41;
  a42,b42,c42 : PN:=42;
  a43,b43,c43 : PN:=43;
  a44,b44,c44 : PN:=44;
  a45,b45,c45 : PN:=45;
  a46,b46,c46 : PN:=46;
  a47,b47,c47 : PN:=47;
  a48,b48,c48 : PN:=48;
  a49,b49,c49 : PN:=49;
  a50,b50,c50 : PN:=50;

  end; {case}

projstep := steps[PN];
if projstep = one then
  begin
  duration := projinfo[PN].tseq_1;
  predfinish[PN] := sigma_j[nextoperation] + duration;
  {writeln(' project  ',PN,'  just assigned step one and will finish at
            predfinish[PN]:5:2);
  writeln(' the duration of this step is  ',duration:5:2);
  }
end;{if projstep}

if projstep = two then
  begin
  duration := projinfo[PN].tseq_2;
  predfinish[PN] := sigma_j[nextoperation] + duration;
 { writeln('project ',PN,'  just assigned step two and will finish at ti
           predfinish[PN]:5:2);
  writeln('the duration of thjis step is  ',duration:5:2);
  }
  end;

if projstep = three then
  begin
  duration := projinfo[PN].tseq_3;
  predfinish[PN] := sigma_j[nextoperation] + duration;
  end; {if}

case nextoperation of
  a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,
  a11,a12,a13,a14,a15,a16,a17,a18,a19,a20,
  a21,a22,a23,a24,a25,a26,a27,a28,a29,a30,
  a31,a32,a33,a34,a35,a36,a37,a38,a39,a40,
  a41,a42,a43,a44,a45,a46,a47,a48,a49,a50 : fa_avail:=fa_avail + duratio

  b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,
  b11,b12,b13,b14,b15,b16,b17,b18,b19,b20,
  b21,b22,b23,b24,b25,b26,b27,b28,b29,b30,
  b31,b32,b33,b34,b35,b36,b37,b38,b39,b40,
  b41,b42,b43,b44,b45,b46,b47,b48,b49,b50 : fb_avail:=fb_avail + duratio

  c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,
  c11,c12,c13,c14,c15,c16,c17,c18,c19,c20,
          c21,c22,c23,c24,c25,c26,c27,c28,c29,c30,
          c31,c32,c33,c34,c35,c36,c37,c38,c39,c40,
          c41,c42,c43,c44,c45,c46,c47,c48,c49,c50 : fc_avail:=fc_avail + duratio

          end; (case)

   end; (procedure update ps sub t)

procedure update_s_sub_t( var s_sub_t : XX; nextoperation : operations;
                          var steps : ZZ; projinfo : proj_block);

var
PN,z : integer;
projstep : steptype;

begin { procedure update s sub t }
  case nextoperation of
    a1,b1,c1 : PN:=1;
    a2,b2,c2 : PN:=2;
    a3,b3,c3 : PN:=3;
    a4,b4,c4 : PN:=4;
    a5,b5,c5 : PN:=5;
    a6,b6,c6 : PN:=6;
    a7,b7,c7 : PN:=7;
    a8,b8,c8 : PN:=8;
    a9,b9,c9 : PN:=9;
    a10,b10,c10 : PN:=10;
    a11,b11,c11 : PN:=11;
    a12,b12,c12 : PN:=12;
    a13,b13,c13 : PN:=13;
    a14,b14,c14 : PN:=14;
    a15,b15,c15 : PN:=15;
    a16,b16,c16 : PN:=16;
    a17,b17,c17 : PN:=17;
    a18,b18,c18 : PN:=18;
    a19,b19,c19 : PN:=19;
    a20,b20,c20 : PN:=20;
    a21,b21,c21 : PN:=21;
    a22,b22,c22 : PN:=22;
    a23,b23,c23 : PN:=23;

    a24,b24,c24 : PN:=24;
    a25,b25,c25 : PN:=25;
    a26,b26,c26 : PN:=26;
    a27,b27,c27 : PN:=27;
    a28,b28,c28 : PN:=28;
    a29,b29,c29 : PN:=29;
    a30,b30,c30 : PN:=30;
    a31,b31,c31 : PN:=31;
    a32,b32,c32 : PN:=32;
    a33,b33,c33 : PN:=33;
    a34,b34,c34 : PN:=34;
    a35,b35,c35 : PN:=35;
    a36,b36,c36 : PN:=36;
    a37,b37,c37 : PN:=37;
    a38,b38,c38 : PN:=38;
    a39,b39,c39 : PN:=39;
    a40,b40,c40 : PN:=40;
    a41,b41,c41 : PN:=41;
    a42,b42,c42 : PN:=42;
    a43,b43,c43 : PN:=43;
    a44,b44,c44 : PN:=44;
    a45,b45,c45 : PN:=45;
    a46,b46,c46 : PN:=46;
    a47,b47,c47 : PN:=47;
    a48,b48,c48 : PN:=48;
    a49,b49,c49 : PN:=49;
    a50,b50,c50 : PN:=50;

  end; {case}

  projstep:=steps[PN];
  if projstep = one then
    begin
    steps[PN]:=two;
    {writeln(' project  ',PN,' is now in step two');
    writeln('type 1 to continue');
    readln(z);}
    s_sub_t := s_sub_t + [projinfo[PN].seq_2];
    end;

   if projstep = two then
     begin
     steps[PN] :=three;
     {writeln(' project  ',PN,' is now in step three');
     writeln('type 1 to continue');
     readln(z);}
     s_sub_t := s_sub_t + [projinfo[PN].seq_3];
     end;

   s_sub_t := s_sub_t - [nextoperation];{this takes the next operation out of s

end; {procedure update s sub t }

procedure find_total_time(var totaltime : real; numprojects : integer;
                          predfinish : AA; projinfo : proj_block;
                          holding : holdingset);


var

PN : integer;
largest : real;

begin {procedure find total time}

  largest := 0;
  for PN:=1 to numprojects do
    begin
    if PN in holding then
     begin
      if predfinish[PN] > largest then
         largest := predfinish[PN];
     end; { if PN}
    end;(for PN)
    totaltime:=largest;
   end; (procedure find total time)

begin(procedure scheduler)
     done := false;
     totaltime:=0;
     if start then
       begin
       for i:=1 to numprojects do
         begin
         with projinfo[i] do
         starting_times[i] := tseq_1+tseq_2+tseq_3;
         end;(for i loop)
       end;(if start)

     if start=false then
       begin
         writeln(' im in the scheduler procedure and k =    ',k);
         for i:=1 to k do
           begin
               holding:=[];
               psindex:=1;
               for l:=1 to 50 do
                  begin
                  predfinish[l]:=0;
                  steps[l]:=one;
                  end;(for l)
               s_sub_t:=[];
               for l:=1 to 100 do
                    ps_sub_t[l]:=none;
               sigma_star:=0;
               fa_avail:=0;
               fb_avail:=0;
               fc_avail:=0;
               for counter:=a1 to c50 do
                  sigma_j[counter]:=0;
               t:=0;

               for l:=1 to numprojects do
                  begin
                  if l in dummypotential[i] then
                    begin
                    steps[l] := one;
                    holding:=holding + [l];
                    s_sub_t := s_sub_t + [projinfo[l].seq_1];
                    end; ( if l in ...)
               end; (for l ...)
               ( now s sub t has the first schedulable operations in the
                 potential set i.  holding also has the projects in
                 potential set i)

           while done = false do

              begin
              find_sigma_j (sigma_j,s_sub_t,predfinish,fa_avail,fb_avail,
                            fc_avail);
```

```
                find_sigma_star(sigma_star,sigma_j,s_sub_t);
                find_next_operation(sigma_star,sigma_j,steps,
                                    s_sub_t,projinfo,nextoperation);
                update_ps_sub_t(ps_sub_t,nextoperation,psindex,predfinish,
                                fa_avail,fb_avail,fc_avail,steps,projinfo);
                update_s_sub_t(s_sub_t,nextoperation,steps,projinfo);
                if s_sub_t = [] then
                   done := true;
                end; {while loop}

                find_total_time(totaltime,numprojects,predfinish,projinfo,
                                holding);
                potential_project_time[i]:=totaltime;
                done:=false;
              end ;  {for i statement}
        end;{if start...}
    end; {procedure find total time}

  procedure filter2(fwl,fw2:integer;pp:filter_choices;var ppt:array_block_1;
                    var pep: array_block_1; var flist:filter_choices;k:integer;
                    start:boolean;starting_times,startprofit : array_block_2;
                    numprojects:integer);

  var i,kndx   : integer;
      klist : array_block_3;
      tppt,tpep:array_block_1;

    procedure sort_fw (var list1 : array_block_1; var list2 : array_block_3;
                                listmax : integer);

        var indx,jndx,swap2,val2   : integer;
            val1,swap1,largest,d   : real;

       begin
       for indx := 1 to listmax do
         list2[indx] := indx;
       for indx := 2 to listmax do
        if list1[indx] < list1[1] then {this will put the SMALLEST value}
         begin                        {of the list in slot number 1    }
         swap1:=list1[1];
         swap2:=list2[1];
         list1[1]:=list1[indx];
         list2[1]:=list2[indx];
         list1[indx]:=swap1;
         list2[indx]:=swap2;
         end;{if list1...}

     for indx:=2 to listmax do
       begin
       val1:=list1[indx];
       val2:=list2[indx];
       jndx:=indx;
       while list1[jndx-1] > val1 do
         begin
         list1[jndx]:=list1[jndx-1];
         list2[jndx]:=list2[jndx-1];
         jndx:=jndx-1;
        end;{while}

      list1[jndx]:=val1;
      list2[jndx]:=val2;
      end;{for indx...}

   end;{procedure sortfw}

 begin {procedure filter2}
 {this procedure will only do anything if k>filterwidth2}
 {otherwise we go straight to addnodes}
 writeln(' im in the procedure filter2');

 if start then
    begin

    sort_fw(ppt,klist,fw1);

    for i:=1 to fw2 do
      begin
      kndx:=klist[i];
      flist[i]:=pp[kndx];
      tpep[i]:=pep[kndx];
      end;{for i}

    for i:=1 to fw2 do
      begin
      pep[i]:=tpep[i];
      end;{for i}

    end;{if start}

 if start=false then
   begin
   if k < fw2 then
     begin
     sort_fw(ppt,klist,k);
     for i:=1 to k do
       begin
       kndx:=klist[i];
       flist[i]:=pp[kndx];
       tpep[i]:=pep[kndx];
       end;{for i}
     for i:=1 to k do
       pep[i]:=tpep[i];
     end;{for k<fw}

   if k>=fw2 then {sort pp on ppt and keep the fw2 best}
     begin
     sort_fw(ppt,klist,fw1);
      for i:=1 to fw2 do
       begin
       kndx:=klist[i];
       flist[i]:=pp[kndx];
       tpep[i]:=pep[kndx];
      end;{for i}
      for i:=1 to fw2 do
      pep[i]:=tpep[i];
    end;{if k >=fw2}
  end; {if start = false}
end;{procedure filter}

procedure insertsort ( var list1 : array_block_1; var list2 : array_block_3;
                             listmax : integer);

   var indx,jndx,swap2,val2   : integer;
       val1,swap1,largest,d   : real;

   begin
   for indx := 1 to listmax do
     list2[indx] := indx;
   largest := 0;
   for indx := 2 to listmax do
     if list1[indx] > list1[1] then {this will put the largest value}
       begin                         {of the list in slot number 1   }
       swap1:=list1[1];
       swap2:=list2[1];
       list1[1]:=list1[indx];
       list2[1]:=list2[indx];
       list1[indx]:=swap1;
       list2[indx]:=swap2;
       end;{if list1...}

   for indx:=2 to listmax do
     begin
     val1:=list1[indx];
     val2:=list2[indx];
     jndx:=indx;
     while list1[jndx-1] < val1 do
       begin
       list1[jndx]:=list1[jndx-1];
       list2[jndx]:=list2[jndx-1];
       jndx:=jndx-1;
       end;{while}

     list1[jndx]:=val1;
     list2[jndx]:=val2;
     end;{for indx...}

end;{procedure insertsort}


procedure addnodes(var projects_selected:choices;flist:filter_choices;
                   k,fw,numprojects,beamwidth:integer;
                   var start:boolean; var finished:boolean;
                   var ep:array_block_2; pep:array_block_1;
                   var pt:array_block_2; ppt:array_block_1;
                   starttimes : array_block_2);

{this procedure will use the expected profit and add/select the best nodes
to the current active branches.  If none are possible, then the
program is done and output should be generated.}
```

```
procedure k_lessthan_m(var ps:choices;beamwidth,k:integer;
                          flist:filter_choices;var ep:array_block_2;
                          pep:array_block_1;var pt:array_block_2;
                          ppt:array_block_1);

var psset  : set of 1..10;
    tps    : choices;
    i,j    : integer;
    tep,tpt: array_block_2;
    subset : boolean;

begin{procedure k less than m}
  psset:=[];
  for i:=1 to beamwidth do
    psset:=psset+[i];
  for i:=1 to k do
    begin
    for j:=1 to beamwidth do
    begin
    writeln(' im in k less than m and looking at project set  ',j);
    subset:=(ps[j]<=flist[i]);
    writeln(' subset equals   ',subset);
     if subset then {true if every member of psj is in flisti i.e. psj is a tr
      begin
      tps[i]:=flist[i];
      tep[i]:=pep[i];
      tpt[i]:=ppt[i];
      psset:=psset-[j];
      end;{if psj...}
    end;{for j...}
    end; {for i ...}

  {now there will be k entries in tps, and m-k entries is psset}

  i:=k+1;
  for j:=1 to beamwidth do
    if j in psset then
      begin
      tps[i]:=ps[j];
      tep[i]:=ep[j];
      tpt[i]:=pt[j];
      i:=i+1;
      end;{if j...}
  {now tps contains m entries for the next level of ps}

  for i:=1 to beamwidth do
    begin
    ps[i]:=tps[i];
    ep[i]:=tep[i];
    pt[i]:=tpt[i];
    end;{for i...}
end; {procedure k less than m}

var klist : array_block_3;
    i,kndx : integer;
```

```
begin {procedure addnodes}

  writeln(' im in the procedure addnodes...');
  if start then
    begin
     insertsort(pep,klist,fw);
    for i:=1 to beamwidth do
      projects_selected[i]:=[klist[i]];
    start:=false;
    end{if start = true}
  else
    if k>0 then
      begin
      if k< beamwidth then
        begin
        k_lessthan_m(projects_selected,beamwidth,k,flist,ep,pep,pt,ppt);
        end;{if k<...}

      if k>= fw then
        begin
        insertsort(pep,klist,fw);

        for i:=1 to beamwidth do
          begin
          kndx:=klist[i];
          projects_selected[i]:=flist[kndx];
          ep[i]:=pep[i];
          writeln(' the sorted eps k>fw are ',ep[i]:5:1);
          pt[i]:=ppt[kndx];
          end;{for i...}
       end;{if k>=fw}

      if ((k<fw) and (k>beamwidth)) then
        begin
        insertsort(pep,klist,k);
        for i:=1 to beamwidth do
          begin
          kndx:=klist[i];
          projects_selected[i]:=flist[kndx];
          ep[i]:=pep[i];
          pt[i]:=ppt[kndx];
          end;{for i};
        end;{if bw<k<fw}

       end{if k > 0}
      else { if k=0}
        finished:=true;


end;{procedure addnodes}

procedure filter1(fw1:integer;pp:potential_choices;var ppt,pep:array_block_1;
                  var flist:filter_choices; k:integer; start:boolean;
                  starting_times,startprofit:array_block_2;
                  projinfo:proj_block; numprojects:integer);
```

```
var i,kndx,q :integer;
    klist : array_block_3;
    tppt,tpep,p_success_list:array_block_1;


begin {procedure filter1}
{this procedure will only do anything if k>filterwidth1}
{otherwise we go straight to filter2}

writeln(' im in the procedure filter1 ');
if start then
 begin

   for i:=1 to numprojects do
     p_success_list[i]:=projinfo[i].p_success;

   insertsort(p_success_list,klist,numprojects);

   for i:=1 to fw1 do
     begin
     kndx:=klist[i];
     flist[i]:=pp[kndx];
     tpep[i]:=startprofit[kndx];
     tppt[i]:=starting_times[kndx];
     end;{for i}

   for i:=1 to fw1 do
     begin
     pep[i]:=tpep[i];
     ppt[i]:=tppt[i];
     end;{for i}

   end;{if start}

if start=false then
  begin

  for i:=1 to k do {this will generate the list of psuccess}
    begin           {for each potential project}
    p_success_list[i]:=0;
      for kndx:=1 to numprojects do
        if kndx in pp[i] then
          p_success_list[i]:=p_success_list[i]+projinfo[kndx].p_success;
   end;{for i:=1 to k}

  if k < fw1 then
    begin
    insertsort(p_success_list,klist,k);
    for i:=1 to k do
      begin
      kndx:=klist[i];
      flist[i]:=pp[kndx];
      tpep[i]:=pep[kndx];
      tppt[i]:=ppt[kndx];
      end;{for i}

    for i:=1 to fw1 do
```

```
      begin
      pep[i]:=tpep[i];    {this now makes a list of the fwl best }
      ppt[i]:=tppt[i];    {projects based on the p successes}
      end;{for i}      {and keeps track of the ppts and peps }

    end;{if k<fwl}

   if k>=fwl then {sort on the psuccesses and keep the fwl best}
     begin
     insertsort(p_success_list,klist,k);
      for i:=1 to fwl do
       begin
       kndx:=klist[i];
       flist[i]:=pp[kndx];
       tpep[i]:=pep[kndx];
       tppt[i]:=ppt[kndx];
      end;{for i}

      for i:=1 to fwl do
       begin
       pep[i]:=tpep[i];
       ppt[i]:=tppt[i];
       end;{for i}

     end;{if k >=fwl}

   end; {if start = false}
 delay(3000);
 end;{procedure filter2}


 procedure generate_project_list(ps : choices; beamwidth,numprojects:integer;
                                   ep,pt:array_block_2);

 var indx,jndx : integer;

 begin{procedure generate project list}
 clrscr;
   for indx := 1 to beamwidth do
     begin
     for jndx := 1 to numprojects do
       begin
       if jndx in ps[indx] then
         writeln('Project  ',jndx,' is in project set number  ',indx);
       end;{for jndx...}
     writeln;
     writeln('The expected profit for project set  ',indx,' is  ',ep[indx]:5:2);
     writeln(' The time to complete the project set ',indx,' is  ',pt[indx]:5:2);
     delay(3000);
     end;{for indx...}

 writeln('This is the end of the procedure that generate the project lists');
 end;{procedure generate project lists}

 begin{main program}
   project_input(projinfo);
```

```
  setlimits(constraints);
  search_input(beamwidth,filterwidth1,filterwidth2,numprojects);
  initialize(projects_selected,potential_projects,pot_expected_profit,
             potential_project_time,expected_profit,project_time,
             filterlist1);
  start:=true;
  finished:=false;

  while finished = false do
    begin

    generate_next_level(projects_selected,potential_projects,starting_projects,
                        beamwidth,numprojects,constraints,projinfo,k,start);

    compute_expected_profit(pot_expected_profit,startprofit,projinfo,
                            potential_projects,beamwidth,numprojects,
                            projects_selected,k,start);

    scheduler(starting_times,potential_project_time,potential_projects,
              projects_selected,beamwidth,numprojects,projinfo,k);

    filter1(filterwidth1,potential_projects,potential_project_time,
            pot_expected_profit,filterlist1,k,start,starting_times,
            startprofit,projinfo,numprojects);

    filter2(filterwidth1,filterwidth2,filterlist1,potential_project_time,
           pot_expected_profit,filterlist2,k,start,
           starting_times,startprofit,numprojects);

    addnodes(projects_selected,filterlist2,k,filterwidth2,numprojects,
             beamwidth,start,finished,expected_profit,pot_expected_profit,
             project_time,potential_project_time,starting_times);
    end;{while}

  generate_project_list(projects_selected,beamwidth,numprojects,
                        expected_profit,project_time);

end.
```

```
program double_filter;
{
This program will select and schedule r&d projects
for the dissertation topic of Mark A. Coffin
using a filtered beam search}

uses crt;

 type
    operations = (a1,b1,c1,a2,b2,c2,a3,b3,c3,a4,b4,c4,a5,b5,c5,
                 a6,b6,c6,a7,b7,c7,a8,b8,c8,a9,b9,c9,a10,b10,c10,none);

     projdata = record
                number,budget,profit_pot  :  real;
                p_success                 :  real;
                num_research,a_const,b_const :  real;
                seq_1,seq_2,seq_3           : operations;
                tseq_1,tseq_2,tseq_3        : real;
            end;{ of record type}

     limits = record
                 budget_lim,nres_lim,a_const_lim : real;
                 b_const_lim                     : real;
                 end;{ of record limits}

        set_of_projects = set of 1..10;
        proj_block=array[1..10] of projdata;
        choices = array[1..5] of set_of_projects;
        potential_choices = array [1..100] of set_of_projects;
        filter_choices = array[1..10] of set_of_projects;
        start_set = array[1..10] of 1..10;
        array_block_1 = array [1..100] of real;
        array_block_2 = array [1..10] of real;
        array_block_3 = array [1..100] of integer;

var
     projects : text;
     {information about the projects are stored in records of
     projdata }
     proj       : projdata;{this variable is a holdong variable while
                 reading and assigning values to projinfo}
     projinfo   : proj_block;
     potential_projects     : potential_choices;
     projects_selected      : choices;
     filterlist1,filterlist2            :filter_choices;
     i,j,k,beamwidth,numprojects,filterwidth1,filterwidth2 :integer;
     constraints        : limits;
     pot_expected_profit,potential_project_time : array_block_1;
     startprofit,starting_times : array_block_2;
     starting_projects   : start_set;
     project_time,expected_profit       : array_block_2;
     start,finished : boolean;


procedure initialize(var dummyselect:choices;
                     var dummypotential: potential_choices;
                     var pep:array_block_1;
                     var ppt:array_block_1;var ep:array_block_2;
                     var pt:array_block_2;var flist:filter_choices);
var i,j                : integer;
```

```
 begin
  for i:=1 to 5 do
   dummyselect[i]:=[ ];
  for i:=1 to 10 do
   begin
   ep[i]:=0;
   pt[i]:=0;
   flist[i]:=[ ];
   end;{for i}
  for j:=1 to 100 do
     begin
     dummypotential[j]:=[ ];
     pep[j]:=0;
     ppt[j]:=0;
     end;{for j..}
   end;{procedure initialize}



procedure project_input(var dummy:proj_block);
{this procedure will input the project attributes
for the selection and scheduling routines}

  var
     i,j                                :integer;
     bud,profit,nresearch,a,b           : real;
     ts1,ts2,ts3,num                    : real;
     psuccess                           : real;
     s1,s2,s3                           : string[3];
     projects                           : text;

begin
     i:=1;
     assign(projects,'data10.dat');
     reset(projects);
        while not eof(projects) do
        begin
              read (projects,num);
              readln(projects,bud,profit,psuccess,a,b,nresearch);
              readln (projects,s1);
              readln (projects,s2);
              readln (projects,s3);
              readln (projects,ts1,ts2,ts3);
              with dummy[i] do
                   begin
                   number:=num;
                   budget:=bud;
                   profit_pot:=profit;
                   p_success:=psuccess;
                   num_research:=nresearch;
                   a_const:=a;
                   b_const:=b;
                   if s1 = 'a1' then seq_1:=a1;
                   if s1 = 'a2' then seq_1:=a2;
                   if s1 = 'a3' then seq_1:=a3;
                   if s1 = 'a4' then seq_1:=a4;
                   if s1 = 'a5' then seq_1:=a5;
                   if s1 = 'a6' then seq_1:=a6;
if s1 = 'a7' then seq_1:=a7;
if s1 = 'a8' then seq_1:=a8;
if s1 = 'a9' then seq_1:=a9;
if s1 = 'a10' then seq_1:=a10;
if s1 = 'b1' then seq_1:=b1;
if s1 = 'b2' then seq_1:=b2;
if s1 = 'b3' then seq_1:=b3;
if s1 = 'b4' then seq_1:=b4;
if s1 = 'b5' then seq_1:=b5;
if s1 = 'b6' then seq_1:=b6;
if s1 = 'b7' then seq_1:=b7;
if s1 = 'b8' then seq_1:=b8;
if s1 = 'b9' then seq_1:=b9;
if s1 = 'b10' then seq_1:=b10;
if s1 = 'c1' then seq_1:=c1;
if s1 = 'c2' then seq_1:=c2;
if s1 = 'c3' then seq_1:=c3;
if s1 = 'c4' then seq_1:=c4;
if s1 = 'c5' then seq_1:=c5;
if s1 = 'c6' then seq_1:=c6;
if s1 = 'c7' then seq_1:=c7;
if s1 = 'c8' then seq_1:=c8;
if s1 = 'c9' then seq_1:=c9;
if s1 = 'c10' then seq_1:=c10;
if s2 = 'a1' then seq_2:=a1;
if s2 = 'a2' then seq_2:=a2;
if s2 = 'a3' then seq_2:=a3;
if s2 = 'a4' then seq_2:=a4;
if s2 = 'a5' then seq_2:=a5;
if s2 = 'a6' then seq_2:=a6;
if s2 = 'a7' then seq_2:=a7;
if s2 = 'a8' then seq_2:=a8;
if s2 = 'a9' then seq_2:=a9;
if s2 = 'a10' then seq_2:=a10;
if s2 = 'b1' then seq_2:=b1;
if s2 = 'b2' then seq_2:=b2;
if s2 = 'b3' then seq_2:=b3;
if s2 = 'b4' then seq_2:=b4;
if s2 = 'b5' then seq_2:=b5;
if s2 = 'b6' then seq_2:=b6;
if s2 = 'b7' then seq_2:=b7;
if s2 = 'b8' then seq_2:=b8;
if s2 = 'b9' then seq_2:=b9;
if s2 = 'b10' then seq_2:=b10;
if s2 = 'c1' then seq_2:=c1;
if s2 = 'c2' then seq_2:=c2;
if s2 = 'c3' then seq_2:=c3;
if s2 = 'c4' then seq_2:=c4;
if s2 = 'c5' then seq_2:=c5;
if s2 = 'c6' then seq_2:=c6;
if s2 = 'c7' then seq_2:=c7;
if s2 = 'c8' then seq_2:=c8;
if s2 = 'c9' then seq_2:=c9;
if s2 = 'c10' then seq_2:=c10;
if s3 = 'a1' then seq_3:=a1;
if s3 = 'a2' then seq_3:=a2;
if s3 = 'a3' then seq_3:=a3;
if s3 = 'a4' then seq_3:=a4;
if s3 = 'a5' then seq_3:=a5;
if s3 = 'a6' then seq_3:=a6;
```

```
                    if s3 = 'a7' then seq_3:=a7;
                    if s3 = 'a8' then seq_3:=a8;
                    if s3 = 'a9' then seq_3:=a9;
                    if s3 = 'a10' then seq_3:=a10;
                    if s3 = 'b1' then seq_3:=b1;
                    if s3 = 'b2' then seq_3:=b2;
                    if s3 = 'b3' then seq_3:=b3;
                    if s3 = 'b4' then seq_3:=b4;
                    if s3 = 'b5' then seq_3:=b5;
                    if s3 = 'b6' then seq_3:=b6;
                    if s3 = 'b7' then seq_3:=b7;
                    if s3 = 'b8' then seq_3:=b8;
                    if s3 = 'b9' then seq_3:=b9;
                    if s3 = 'b10' then seq_3:=b10;
                    if s3 = 'c1' then seq_3:=c1;
                    if s3 = 'c2' then seq_3:=c2;
                    if s3 = 'c3' then seq_3:=c3;
                    if s3 = 'c4' then seq_3:=c4;
                    if s3 = 'c5' then seq_3:=c5;
                    if s3 = 'c6' then seq_3:=c6;
                    if s3 = 'c7' then seq_3:=c7;
                    if s3 = 'c8' then seq_3:=c8;
                    if s3 = 'c9' then seq_3:=c9;
                    if s3 = 'c10' then seq_3:=c10;
                    tseq_1:=ts1;
                    tseq_2:=ts2;
                    tseq_3:=ts3;
                    end; {with}
                    i:=i+1;
          end;{while eof statement}
          close(projects);
end;{of procedure project_input}

procedure setlimits(var dummy : limits);
  var
     budlim,researchlim,alim,blim       : real;
     working                            :text;

begin {procedure}
      assign(working,'limits.dat');
      reset(working);
      readln(working,budlim,researchlim,alim,blim);
      with dummy do
        begin
        budget_lim:=budlim;
        nres_lim:=researchlim;
        a_const_lim:=alim;
        b_const_lim:=blim;
        end;{with}
       close(working);
       writeln('the num res limit is stored as  ',dummy.nres_lim:8:2);
       writeln(' the budget limit is stored as  ',dummy.budget_lim:8:2);
       writeln(' the a limit is stored as  ',dummy.a_const_lim:8:2);
       writeln(' the b limit is stored as  ',dummy.b_const_lim:8:2);
end;{procedure setlimits}


procedure search_input(var bw,fw1,fw2,np:integer);
{this procedure will input the search attributes
such as beamwidth,number of projects,filterwidth,and others}
```

```
     begin
      writeln('input the beamwidth of the search less than 5');
      readln(bw);
      writeln(' input the first filterwidth of the search of 10 or less');
      readln(fw1);
      writeln(' input the second filterwidth,<= first filterwidth ');
      readln(fw2);
      writeln(' input the number of projects ');
      readln(np);

     end;

procedure generate_next_level(    projects_selected :choices;
                              var potential_projects:potential_choices;
                              var starting_projects:start_set;
                                  beamwidth,numprojects:integer;
                                  constraints : limits;
                                  projinfo : proj_block;
                                  var k : integer;start : boolean );
{this procedure will generate the next level of tree
to search}

procedure check_and_see ( dps:set_of_projects;pp:potential_choices;
                          k:integer; var inset:boolean);

var m,n : integer;

begin
  inset:=false;
  for n:=1 to k do
        if dps=pp[n] then
          inset:=true;

end; {procedure check and see}

procedure check_feasibility (dps:set_of_projects; cons:limits;
                             numprojects:integer; pinfo:proj_block;
                             var feas:boolean);

var i:integer;
    nrtot,atot,btot,budtot : real;

begin
 feas:=true;
 nrtot:=0;
 atot:=0;
 btot:=0;
 budtot:=0;

for i:=1 to numprojects do
  begin
  if i in dps then
    begin
    nrtot:=nrtot+pinfo[i].num_research;
    atot:=atot+pinfo[i].a_const;
    btot:=btot+pinfo[i].b_const;
    budtot:=budtot+pinfo[i].budget;
    end;
  end;{for loop}
```

```
if nrtot > constraints.nres_lim then
        begin
        feas:=false;
        end;
if atot > constraints.a_const_lim then
        begin
        feas:=false;
        end;
if btot > constraints.b_const_lim then
        begin
        feas:=false;
        end;

if budtot > constraints.budget_lim then
        begin
         feas:=false;
         end;
end; {procedure check feasibility}




var i,j,l,m      : integer;
    inset,feasible : boolean;

    dummy_project_set   : set_of_projects;

begin
  k:=0;{this is the global counter I use to keep track of the number of
        potential nodes at each level.  It is only modified in the procedure
        and is used in others}

  if start then {no projects have been selected yet}
    begin
    for i:=1 to numprojects do
      starting_projects[i]:= i;
      writeln(' all are feasible and are added');
    end {if start}

   else { do the rest of the procedure}
     begin
     writeln(' im in the generate next level procedure and k = ',k);
     for i:=1 to beamwidth do
      begin
      for j:=1 to numprojects do
       begin

       if (j in projects_selected[i]) = false then
        begin
        dummy_project_set:=projects_selected[i] + [j];
        check_and_see(dummy_project_set,potential_projects,k,inset);
        if inset = false then
          begin
          check_feasibility(dummy_project_set,constraints,numprojects,projinfo,
                            feasible);
          if feasible = true then
           begin
           k:=k+1;{only time k is ever altered}
           potential_projects[k]:=dummy_project_set;
          end;{ if feasible}
         end;{ if inset}
       end;{ if j in projselected}
     end;{for j}
     end;{for i}
   end;{else}
   writeln(' im leaving generate next level and k is ',k);
 end; {procedure generate next level}


procedure compute_expected_profit(var dummye_profit : array_block_1;
                                  var startprofit   : array_block_2;
                                  dummyinfo : proj_block;
                                  dummypotential : potential_choices;
                                  beamwidth,numprojects : integer;
                                  projects_selected : choices;
                                  k:integer; start : boolean);

var i,j  : integer;
    totalprofit : real;

begin {procedure}
  if start then
     for i:=1 to numprojects do
          startprofit[i]:=dummyinfo[i].profit_pot * dummyinfo[i].p_success;


  if start=false then  {else go to end of procedure}
    begin
    for i:=1 to k do
      begin
      totalprofit:=0;
      for j:=1 to numprojects do
            begin
            if j in dummypotential[i] then
               totalprofit:=totalprofit+(dummyinfo[j].profit_pot*dummyinfo[j].p_
            end;{for j loop}
      dummye_profit[i]:=totalprofit;
     end; {for i loop}
  end; {if start=false}
end; {procedure expected profit}




procedure scheduler( var starting_times:array_block_2; var potential_project_tim
                         array_block_1;dummypotential:potential_choices;
                         dummyselect:choices; beamwidth:
                         integer; numprojects : integer; projinfo : proj_block;
                         k:integer;start : boolean);




{this procedure will generate schedules according
to a scheduling heuristic and determine time to completion
```

```
of all projects.  This procedure can be modified later to
solve for other scheduling requirements.}

type
     steptype = (one,two,three);
     operations_array = array[operations] of real;
     AA = array[1..10] of real;
     XX = set of operations;
     YY = array[1..30] of operations;{this is the schedule, in order, for each p
     ZZ = array[1..10] of steptype;{tells me what step each project is on}
     holdingset = set of 1..10;


var  psindex,i,j,l : integer;
     s_sub_t : XX;
     sigma_j : operations_array;
     totaltime,sigma_star,t,fa_avail,fb_avail,fc_avail   :  real;
     nextoperation,counter : operations;
     ps_sub_t : YY;
     holding : holdingset;
     predfinish : AA;
     steps : ZZ;  {tells me which step the project is in at any time}
     done : boolean;

  procedure find_sigma_j (var sigma_j : operations_array;
                              s_sub_t : XX;
                              predfinish : AA;
                              fa_avail,fb_avail,fc_avail : real);

     var
     counter2 : operations;
     pf,favail : real;

       begin {subprocedure find_sigma_j}
         for counter2:=a1 to c10 do
           begin
           if counter2 in s_sub_t then
             begin
             {now find the project # and use info about the step its on}
             case counter2 of
              a1,b1,c1 : pf:=predfinish[1];{these are updated when ps is updated
              a2,b2,c2 : pf:=predfinish[2];
              a3,b3,c3 : pf:=predfinish[3];
              a4,b4,c4 : pf:=predfinish[4];
              a5,b5,c5 : pf:=predfinish[5];
              a6,b6,c6 : pf:=predfinish[6];
              a7,b7,c7 : pf:=predfinish[7];
              a8,b8,c8 : pf:=predfinish[8];
              a9,b9,c9 : pf:=predfinish[9];
              a10,b10,c10 : pf:=predfinish[10];
              end;{case}

             case counter2 of
             a1,a2,a3,a4,a5,a6,a7,a8,a9,a10 : favail:=fa_avail;
             b1,b2,b3,b4,b5,b6,b7,b8,b9,b10 : favail:=fb_avail;
             c1,c2,c3,c4,c5,c6,c7,c8,c9,c10 : favail:=fc_avail;
             end;{case}

            if pf>= favail then
               sigma_j[counter2] := pf;
            if pf< favail then
               sigma_j[counter2] := favail;
           {writeln('sigma j is ',sigma_j[counter2]:5:2);}
         end;{if counter2 ...}
       end; {for counter2...}
     end; {subprocedure find sigma j}

procedure find_sigma_star (var sigma_star : real;
                              sigma_j : operations_array;
                              s_sub_t : XX);

   var
   counter3 : operations;
   smallest : real;

  begin {subprocedure find sigma star}
     smallest := 1000;
     for counter3 := a1 to c10 do
     begin
      if counter3 in s_sub_t then
        begin
        if sigma_j[counter3] <= smallest then
          smallest := sigma_j[counter3]
        end;{if counter3 ...}
     end;{for counter3...}
     sigma_star:=smallest;
  end;{subprocedure find sigma star}

procedure find_next_operation(sigma_star:real; sigma_j : operations_array;
                              steps : ZZ; s_sub_t : XX; projinfo : proj_block;
                              var nextoperation : operations);


var
smallest,spt : real;
counter4 : operations;
spt_operation : operations;
L : integer;

begin {procedure find next operation}

  smallest:=1000;
  for counter4:= a1 to c10 do
   begin
   if counter4 in s_sub_t then
    begin
    if sigma_j[counter4] = sigma_star then
      begin
       case counter4 of
       a1,b1,c1 : L:=1;
       a2,b2,c2 : L:=2;
       a3,b3,c3 : L:=3;
       a4,b4,c4 : L:=4;
       a5,b5,c5 : L:=5;
       a6,b6,c6 : L:=6;
       a7,b7,c7 : L:=7;
       a8,b8,c8 : L:=8;
       a9,b9,c9 : L:=9;
       a10,b10,c10 : L:=10;
```

```
        end; {case}

      case steps[L] of
        one : spt :=projinfo[L].tseq_1;
        two : spt :=projinfo[L].tseq_2;
        three : spt :=projinfo[L].tseq_3;
        end; {case}

      if spt< smallest then
       begin
        smallest:=spt;
        spt_operation:=counter4;
       end;{if spt}
     end; {if sigma_j}
    end; {if counter4...}
   end;{for counter4...}
   nextoperation:=spt_operation;
   end; {procedure find next operation}

 Procedure update_ps_sub_t (var ps_sub_t : YY; nextoperation : operations;
                         var psindex : integer; var predfinish : AA;
                         var fa_avail:real; var fb_avail : real;
                         var fc_avail:real; steps : ZZ;
                         projinfo : proj_block);




 var
    PN : integer;
    projstep : steptype;
    duration : real;

   begin {procedure update ps sub t}

      duration :=0;
      ps_sub_t [psindex] := nextoperation;
      psindex := psindex+1;
        case nextoperation of
           a1,b1,c1 : PN:=1;
           a2,b2,c2 : PN:=2;
           a3,b3,c3 : PN:=3;
           a4,b4,c4 : PN:=4;
           a5,b5,c5 : PN:=5;
           a6,b6,c6 : PN:=6;
           a7,b7,c7 : PN:=7;
           a8,b8,c8 : PN:=8;
           a9,b9,c9 : PN:=9;
           a10,b10,c10 : PN:=10;
           end; {case}

         projstep := steps[PN];
         if projstep = one then
           begin
           duration := projinfo[PN].tseq_1;
           predfinish[PN] := sigma_j[nextoperation] + duration;
           {writeln(' project  ',PN,'  just assigned step one and will finish at
                     predfinish[PN]:5:2);
           writeln(' the duration of this step is  ',duration:5:2);
```

```
          }
        end;{if projstep}

        if projstep = two then
          begin
          duration := projinfo[PN].tseq_2;
          predfinish[PN] := sigma_j[nextoperation] + duration;
         { writeln('project ',PN,'  just assigned step two and will finish at ti
                  predfinish[PN]:5:2);
          writeln('the duration of thjis step is  ',duration:5:2);
          }
          end;

        if projstep = three then
          begin
          duration := projinfo[PN].tseq_3;
          predfinish[PN] := sigma_j[nextoperation] + duration;
          end; {if}

        case nextoperation of
          a1,a2,a3,a4,a5,a6,a7,a8,a9,a10 : fa_avail:=fa_avail + duration;
          b1,b2,b3,b4,b5,b6,b7,b8,b9,b10 : fb_avail:=fb_avail + duration;
          c1,c2,c3,c4,c5,c6,c7,c8,c9,c10 : fc_avail:=fc_avail + duration;
          end; {case}

    end; {procedure update ps sub t}

procedure update_s_sub_t( var s_sub_t : XX; nextoperation : operations;
                          var steps : ZZ; projinfo : proj_block);

var
PN,z : integer;
projstep : steptype;

begin { procedure update s sub t }
  case nextoperation of
    a1,b1,c1 : PN:=1;
    a2,b2,c2 : PN:=2;
    a3,b3,c3 : PN:=3;
    a4,b4,c4 : PN:=4;
    a5,b5,c5 : PN:=5;
    a6,b6,c6 : PN:=6;
    a7,b7,c7 : PN:=7;
    a8,b8,c8 : PN:=8;
    a9,b9,c9 : PN:=9;
    a10,b10,c10 : PN:=10;
  end; {case}

  projstep:=steps[PN];
  if projstep = one then
    begin
    steps[PN]:=two;
    {writeln(' project  ',PN,' is now in step two');
    writeln('type 1 to continue');
    readln(z);}
    s_sub_t := s_sub_t + [projinfo[PN].seq_2];
    end;

   if projstep = two then
     begin
     steps[PN] :=three;
     (writeln(' project  ',PN,' is now in step three');
     writeln('type 1 to continue');
     readln(z);)
     s_sub_t := s_sub_t + [projinfo[PN].seq_3];
     end;

   s_sub_t := s_sub_t - [nextoperation];(this takes the next operation out of s

end; (procedure update s sub t )

procedure find_total_time(var totaltime : real; numprojects : integer;
                          predfinish : AA; projinfo : proj_block;
                          holding : holdingset);


var

PN : integer;
largest : real;

begin (procedure find total time)

  largest := 0;
  for PN:=1 to numprojects do
    begin
    if PN in holding then
     begin
      if predfinish[PN] > largest then
         largest := predfinish[PN];
     end; ( if PN)
    end;(for PN)
    totaltime:=largest;
   end; (procedure find total time)

begin(procedure scheduler)
     done := false;
     totaltime:=0;
     if start then
       begin
       for i:=1 to numprojects do
         begin
         with projinfo[i] do
         starting_times[i] := tseq_1+tseq_2+tseq_3;
         end;(for i loop)
       end;(if start)

     if start=false then
       begin
         writeln(' im in the scheduler procedure and k =   ',k);
         for i:=1 to k do
           begin
               holding:=[];
               psindex:=1;
               for l:=1 to 10 do
                  begin
                  predfinish[l]:=0;
                  steps[l]:=one;
                  end;(for l)
               s_sub_t:=[];
      begin
      for indx := 1 to listmax do
        list2[indx] := indx;
      for indx := 2 to listmax do
       if list1[indx] < list1[1] then {this will put the SMALLEST value}
        begin                        {of the list in slot number 1    }
        swap1:=list1[1];
        swap2:=list2[1];
        list1[1]:=list1[indx];
        list2[1]:=list2[indx];
        list1[indx]:=swap1;
        list2[indx]:=swap2;
        end;{if list1...}

    for indx:=2 to listmax do
      begin
      val1:=list1[indx];
      val2:=list2[indx];
      jndx:=indx;
      while list1[jndx-1] > Val1 do
        begin
        list1[jndx]:=list1[jndx-1];
        list2[jndx]:=list2[jndx-1];
        jndx:=jndx-1;
        end;{while}

      list1[jndx]:=val1;
      list2[jndx]:=val2;
      end;{for indx...}

   end;{procedure sortfw}

 begin {procedure filter}
 {this procedure will only do anything if k>filterwidth}
 {otherwise we go straight to addnodes}
 if start then
    begin

    sort_fw(ppt,klist,fw1);{only look at the ppt's from first sort}

    for i:=1 to fw2 do
          begin
          kndx:=klist[i];
          flist[i]:=pp[kndx];
          tpep[i]:=pep[kndx];
          end;{for i}

     for i:=1 to fw2 do
        begin
        pep[i]:=tpep[i];
        end;{for i}

    end;{if start}
 if start=false then
   begin
   if k<fw2 then
     begin
     sort_fw(ppt,klist,k);
     for i:=1 to k do
```

```
        begin
        kndx:=klist[i];
        flist[i]:=pp[kndx];
        tpep[i]:=pep[kndx];
        end;{for i}
    for i:=1 to k do
            pep[i]:=tpep[i];
    end;{for k<fw}


  if k>=fw2 then {sort pp on ppt and keep the fw2 best}
   begin
   sort_fw(ppt,klist,fw1);
    for i:=1 to fw2 do
        begin
        kndx:=klist[i];
        flist[i]:=pp[kndx];
        tpep[i]:=pep[kndx];
        end;{for i}
   for i:=1 to fw2 do
        begin
        pep[i]:=tpep[i];
        end;{for i}

    end;{if k>fw2}
  end;{if start=false}
end;{procedure filter2}

procedure insertsort ( var list1 : array_block_1; var list2 : array_block_3;
                                    listmax : integer);

   var indx,jndx,swap2,val2   : integer;
       val1,swap1,largest,d   : real;

   begin
   for indx := 1 to listmax do
     list2[indx] := indx;
   largest := 0;
   for indx := 2 to listmax do
     if list1[indx] > list1[1] then {this will put the largest value}
       begin                          {of the list in slot number 1    }
       swap1:=list1[1];
       swap2:=list2[1];
       list1[1]:=list1[indx];
       list2[1]:=list2[indx];
       list1[indx]:=swap1;
       list2[indx]:=swap2;
       end;{if list1...}

   for indx:=2 to listmax do
     begin
     val1:=list1[indx];
     val2:=list2[indx];
     jndx:=indx;
     while list1[jndx-1] < val1 do
       begin
       list1[jndx]:=list1[jndx-1];
       list2[jndx]:=list2[jndx-1];
       jndx:=jndx-1;
       end;{while}
     list1[jndx]:=val1;
     list2[jndx]:=val2;
     end;{for indx...}

  end;{procedure insertsort}


procedure addnodes(var projects_selected:choices;flist:filter_choices;
                   k,fw,numprojects,beamwidth:integer;
                   var start:boolean; var finished:boolean;
                   var ep:array_block_2; pep:array_block_1;
                   var pt:array_block_2; ppt:array_block_1;
                   starttimes : array_block_2);

{this procedure will use the expected profit and add/select the best nodes
to the current active branches.  If none are possible, then the
program is done and output should be generated.}



  procedure k_lessthan_m(var ps:choices;beamwidth,k:integer;
                             flist:filter_choices;var ep:array_block_2;
                             pep:array_block_1;var pt:array_block_2;
                             ppt:array_block_1);

  var psset  : set of 1..10;
      tps    : choices;
      i,j    : integer;
      tep,tpt: array_block_2;
      subset : boolean;

  begin{procedure k less than m}
    psset:=[];
    for i:=1 to beamwidth do
      psset:=psset+[i];
    for i:=1 to k do
      begin
      for j:=1 to beamwidth do
      begin
      writeln(' im in k less than m and looking at project set  ',j);
      subset:=(ps[j]<=flist[i]);
      writeln(' subset equals   ',subset);
       if subset then {true if every member of psj is in flisti i.e. psj is a tr
        begin
        tps[i]:=flist[i];
        tep[i]:=pep[i];
        tpt[i]:=ppt[i];
        psset:=psset-[j];
        end;{if psj...}
      end;{for j...}
      end; {for i ...}

    {now there will be k entries in tps, and m-k entries is psset}

    i:=k+1;
    for j:=1 to beamwidth do
      if j in psset then
        begin
        tps[i]:=ps[j];
```

```
          tep[i]:=ep[j];
          tpt[i]:=pt[j];
          i:=i+1;
          end;{if j...}
      {now tps contains m entries for the next level of ps}

      for i:=1 to beamwidth do
        begin
        ps[i]:=tps[i];
        ep[i]:=tep[i];
        pt[i]:=tpt[i];
        end;{for i...}
    end; {procedure k less than m}

  var klist : array_block_3;
      i,kndx : integer;


  begin {procedure addnodes}
    if start then
      begin
      insertsort(pep,klist,fw);
      for i:=1 to beamwidth do
        projects_selected[i]:=[klist[i]];
      start:=false;
      end{if start = true}
    else
      if k>0 then
        begin
        if k< beamwidth then
          begin
          k_lessthan_m(projects_selected,beamwidth,k,flist,ep,pep,pt,ppt);
          end;{if k<...}

        if k>=fw then
          begin
          insertsort(pep,klist,fw);

          for i:=1 to beamwidth do
            begin
            kndx:=klist[i];
            projects_selected[i]:=flist[kndx];
            ep[i]:=pep[i];
            pt[i]:=ppt[kndx];
            end;{for i...}
          end;{if k>=fw}

        if ((k<fw) and (k>=beamwidth)) then
           begin
           insertsort(pep,klist,k);
           for i:=1 to beamwidth do
            begin
            kndx:=klist[i];
            projects_selected[i]:=flist[kndx];
            ep[i]:=pep[i];
            pt[i]:=ppt[kndx];
            end;{for i...}
          end;{if k>=fw}

         end{if k > 0}
      else { if k=0}
        finished:=true;

end;{procedure addnodes}

procedure filter1(fw1:integer;pp:potential_choices;var ppt,pep:array_block_1;
                  var flist:filter_choices; k:integer; start:boolean;
                  starting_times,startprofit: array_block_2;
                  projinfo:proj_block; numprojects:integer);

var i,kndx,q   : integer;
    klist : array_block_3;
    tppt,tpep,p_success_list:array_block_1;


begin {procedure filter1}
{this procedure will only do anything if k>filterwidth1}
{otherwise we go straight to filter2}

if start then
   begin
   for i:=1 to numprojects do
       p_success_list[i]:=projinfo[i].p_success;

   insertsort(p_success_list,klist,numprojects);
   for i:=1 to fw1 do
         begin
         kndx:=klist[i];
         flist[i]:=pp[kndx];
         tpep[i]:=startprofit[kndx];
         tppt[i]:=starting_times[kndx];
         end;{for i}

    for i:=1 to fw1 do
       begin
       pep[i]:=tpep[i];
       ppt[i]:=tppt[i];
       end;{for i}

   end;{if start}
if start=false then
  begin

  for i:=1 to k do  {this will generate the list of psuccess for}
    begin           {each potential project}
    p_success_list[i]:=0;
    for kndx:=1 to numprojects do
      if kndx in pp[i] then
        p_success_list[i]:=p_success_list[i]+projinfo[kndx].p_success;
    end;{for i:=1 to k}


  if k<fw1 then
    begin
    insertsort(p_success_list,klist,k);
    for i:=1 to k do
         begin
         kndx:=klist[i];
         flist[i]:=pp[kndx];
         tpep[i]:=pep[kndx];
```

```
          tppt[i]:=ppt[kndx];
          end;{for i}
      for i:=1 to fwl do
           begin
           pep[i]:=tpep[i];
           ppt[i]:=tppt[i];
           end;{for i...}
      end;{if k<fwl}


   if k>=fwl then {sort on the psuccesses and keep the fwl best}
     begin
     insertsort(p_success_list,klist,k);
     for i:=1 to fwl do
          begin
          kndx:=klist[i];
          flist[i]:=pp[kndx];
          tpep[i]:=pep[kndx];
          tppt[i]:=ppt[kndx];
          end;{for i}

    delay(500);

    for i:=1 to fwl do
          begin
          pep[i]:=tpep[i];
          ppt[i]:=tppt[i];
          end;{for i}

      end;{if k>fw}
   end;{if start=false}
end;{procedure filter2}



procedure generate_project_list(ps : choices; beamwidth,numprojects:integer;
                                 ep,pt:array_block_2;pinfo:proj_block);

var indx,jndx : integer;
    sumpsuccess : real;
begin{procedure generate project list}
  for indx := 1 to beamwidth do
    begin
     sumpsuccess:=0;
     for jndx := 1 to numprojects do
      begin
      if jndx in ps[indx] then
        begin
        writeln('Project  ',jndx,' is in project set number  ',indx);
        sumpsuccess:=sumpsuccess+pinfo[jndx].p_success;
        writeln('psuccess of project ',jndx,'is ',pinfo[jndx].p_success:8:3);
        end;{if jndx...}
      end;{for jndx...}
    writeln;
    writeln('The expected profit for project set  ',indx,' is  ',ep[indx]:5:2);
    writeln(' The time to complete the project set ',indx,' is  ',pt[indx]:5:2);
    writeln(' the sum of psuccess for the projects is ',sumpsuccess:8:3);
    delay(3000);
   end;{for indx...}
```

```
writeln('This is the end of the procedure that generate the project lists');
end;{procedure generate project lists}

begin{main program}
  project_input(projinfo);
  setlimits(constraints);
  search_input(beamwidth,filterwidth1,filterwidth2,numprojects);
  initialize(projects_selected,potential_projects,pot_expected_profit,
             potential_project_time,expected_profit,project_time,
             filterlist1);

  start:=true;
  finished:=false;

  while finished = false do
    begin

    generate_next_level(projects_selected,potential_projects,starting_projects,
                        beamwidth,numprojects,constraints,projinfo,k,start);

    compute_expected_profit(pot_expected_profit,startprofit,projinfo,
                            potential_projects,beamwidth,numprojects,
                            projects_selected,k,start);

    scheduler(starting_times,potential_project_time,potential_projects,
              projects_selected,beamwidth,numprojects,projinfo,k,start);

    filter1(filterwidth1,potential_projects,potential_project_time,
            pot_expected_profit,filterlist1,k,start,starting_times,
            startprofit,projinfo,numprojects);

    filter2(filterwidth1,filterwidth2,filterlist1,potential_project_time,
            pot_expected_profit,
           filterlist2,k,start,starting_times,startprofit,numprojects);

    addnodes(projects_selected,filterlist2,k,filterwidth2,numprojects,
             beamwidth,start,finished,expected_profit,pot_expected_profit,
             project_time,potential_project_time,starting_times);


    end;{while}

  generate_project_list(projects_selected,beamwidth,numprojects,
                        expected_profit,project_time,projinfo);

end.
```

```
This is the hard copy of the complete enumeration
routine for checking 10 projects subject
to all the constraints.

program complete_enumeration;
{MAKE SURE THE DATA IS IN THE PROPER FORMAT FOR THE PROGRAM}
 type
    operations = (a1,b1,c1,a2,b2,c2,a3,b3,c3,a4,b4,c4,a5,b5,c5,
                  a6,b6,c6,a7,b7,c7,a8,b8,c8,a9,b9,c9,a10,b10,c10,none);

     projdata = record
                number,budget,profit_pot  :  real;
                p_success                 :  real;
                num_research,a_const,b_const :  real;
                seq_1,seq_2,seq_3            : operations;
                tseq_1,tseq_2,tseq_3         : real;
            end;{ of record type}

     limits = record
                 budget_lim,nres_lim,a_const_lim : real;
                 b_const_lim                     : real;
                 end;{ of record limits}

        set_of_projects = set of 1..10;
        proj_block=array[1..10] of projdata;

        array_block_1 = array [1..100] of real;
        array_block_2 = array [1..10] of real;
        array_block_3 = array [1..100] of integer;

var
     projects,best_output : text;
     {information about the projects are stored in records of
     projdata }
     proj      :projdata;{this variable is a holdong variable while
                  reading and assigning values to projinfo}
     projinfo  : proj_block;
     pset : set_of_projects;
     i,numprojects :integer;
     constraints         : limits;
     p1,p2,p3,p4,p5,p6,p7,p8,p9,p10 : 0..1;
     schedule_time,expected_profit      : real;
     feasible: boolean;


procedure project_input(var dummy:proj_block);
{this procedure will input the project attributes
for the selection and scheduling routines}

  var
     i,j                                :integer;
     bud,profit,nresearch,a,b           : real;
     ts1,ts2,ts3,num                    : real;
```

```
if s2 = 'a3' then seq_2:=a3;
if s2 = 'a4' then seq_2:=a4;
if s2 = 'a5' then seq_2:=a5;
if s2 = 'a6' then seq_2:=a6;
if s2 = 'a7' then seq_2:=a7;
if s2 = 'a8' then seq_2:=a8;
if s2 = 'a9' then seq_2:=a9;
if s2 = 'a10' then seq_2:=a10;
if s2 = 'b1' then seq_2:=b1;
if s2 = 'b2' then seq_2:=b2;
if s2 = 'b3' then seq_2:=b3;
if s2 = 'b4' then seq_2:=b4;
if s2 = 'b5' then seq_2:=b5;
if s2 = 'b6' then seq_2:=b6;
if s2 = 'b7' then seq_2:=b7;
if s2 = 'b8' then seq_2:=b8;
if s2 = 'b9' then seq_2:=b9;
if s2 = 'b10' then seq_2:=b10;
if s2 = 'c1' then seq_2:=c1;
if s2 = 'c2' then seq_2:=c2;
if s2 = 'c3' then seq_2:=c3;
if s2 = 'c4' then seq_2:=c4;
if s2 = 'c5' then seq_2:=c5;
if s2 = 'c6' then seq_2:=c6;
if s2 = 'c7' then seq_2:=c7;
if s2 = 'c8' then seq_2:=c8;
if s2 = 'c9' then seq_2:=c9;
if s2 = 'c10' then seq_2:=c10;
if s3 = 'a1' then seq_3:=a1;
if s3 = 'a2' then seq_3:=a2;
if s3 = 'a3' then seq_3:=a3;
if s3 = 'a4' then seq_3:=a4;
if s3 = 'a5' then seq_3:=a5;
if s3 = 'a6' then seq_3:=a6;
if s3 = 'a7' then seq_3:=a7;
if s3 = 'a8' then seq_3:=a8;
if s3 = 'a9' then seq_3:=a9;
if s3 = 'a10' then seq_3:=a10;
if s3 = 'b1' then seq_3:=b1;
if s3 = 'b2' then seq_3:=b2;
if s3 = 'b3' then seq_3:=b3;
if s3 = 'b4' then seq_3:=b4;
if s3 = 'b5' then seq_3:=b5;
if s3 = 'b6' then seq_3:=b6;
if s3 = 'b7' then seq_3:=b7;
if s3 = 'b8' then seq_3:=b8;
if s3 = 'b9' then seq_3:=b9;
if s3 = 'b10' then seq_3:=b10;
if s3 = 'c1' then seq_3:=c1;
if s3 = 'c2' then seq_3:=c2;
if s3 = 'c3' then seq_3:=c3;
if s3 = 'c4' then seq_3:=c4;
if s3 = 'c5' then seq_3:=c5;
if s3 = 'c6' then seq_3:=c6;
if s3 = 'c7' then seq_3:=c7;
if s3 = 'c8' then seq_3:=c8;
if s3 = 'c9' then seq_3:=c9;
```

```
                    if s3 = 'c10' then seq_3:=c10;
                    tseq_1:=ts1;
                    tseq_2:=ts2;
                    tseq_3:=ts3;
                    end; {with}
                    i:=i+1;
          end;{while eof statement}
          close(projects);
end;{of procedure project_input}

procedure setlimits(var dummy : limits);
  var
     budlim,researchlim,alim,blim        : real;
     working                             :text;

begin {procedure}
      assign(working,'limits.dat');
      reset(working);
      readln(working,budlim,researchlim,alim,blim);
      with dummy do
        begin
        budget_lim:=budlim;
        nres_lim:=researchlim;
        a_const_lim:=alim;
        b_const_lim:=blim;
        end;{with}
       close(working);
end;{procedure setlimits}


procedure search_input(var np:integer);

     begin
     writeln(' input the number of projects ');
      readln(np);

     end;


procedure check_feasibility (dps:set_of_projects;cons:limits;
                             numprojects:integer; pinfo:proj_block;
                             var feas:boolean);

var i:integer;
    nrtot,atot,btot,budtot : real;

begin
 feas:=true;
 nrtot:=0;
 atot:=0;
 btot:=0;
 budtot:=0;

for i:=1 to numprojects do
  begin
  if i in dps then
    begin
```

```
     XX = set of operations;
     YY = array[1..30] of operations;{this is the schedule, in order, for each p
     ZZ = array[1..10] of steptype;{tells me what step each project is on}
     holdingset = set of 1..10;


var  psindex,i,j,l : integer;
     s_sub_t : XX;
     sigma_j : operations_array;
     totaltime,sigma_star,t,fa_avail,fb_avail,fc_avail   :  real;
     nextoperation,counter : operations;
     ps_sub_t : YY;
     holding : holdingset;
     predfinish : AA;
     steps : ZZ;  {tells me which step the project is in at any time}
     done : boolean;

  procedure find_sigma_j (var sigma_j : operations_array;
                              s_sub_t : XX;
                              predfinish : AA;
                              fa_avail,fb_avail,fc_avail : real);

     var
     counter2 : operations;
     pf,favail : real;

       begin {subprocedure find_sigma_j}
         for counter2:=a1 to c10 do
           begin
           if counter2 in s_sub_t then
             begin
             {now find the project # and use info about the step its on}
             case counter2 of
              a1,b1,c1 : pf:=predfinish[1];{these are updated when ps is updated
              a2,b2,c2 : pf:=predfinish[2];
              a3,b3,c3 : pf:=predfinish[3];
              a4,b4,c4 : pf:=predfinish[4];
              a5,b5,c5 : pf:=predfinish[5];
              a6,b6,c6 : pf:=predfinish[6];
              a7,b7,c7 : pf:=predfinish[7];
              a8,b8,c8 : pf:=predfinish[8];
              a9,b9,c9 : pf:=predfinish[9];
              a10,b10,c10 : pf:=predfinish[10];
              end;{case}

             case counter2 of
             a1,a2,a3,a4,a5,a6,a7,a8,a9,a10 : favail:=fa_avail;
             b1,b2,b3,b4,b5,b6,b7,b8,b9,b10 : favail:=fb_avail;
             c1,c2,c3,c4,c5,c6,c7,c8,c9,c10 : favail:=fc_avail;
             end;{case}

            if pf>= favail then
               sigma_j[counter2] := pf;

            if pf< favail then
               sigma_j[counter2] := favail;
          {writeln('sigma j is ',sigma_j[counter2]:5:2);}
          end;{if counter2 ...}
        end; {for counter2...}
      end; {subprocedure find sigma j}

procedure find_sigma_star (var sigma_star : real;
                               sigma_j : operations_array;
                               s_sub_t : XX);

   var
   counter3 : operations;
   smallest : real;

  begin {subprocedure find sigma star}
     smallest := 1000;
     for counter3 := a1 to c10 do
     begin
      if counter3 in s_sub_t then
        begin
        if sigma_j[counter3] <= smallest then
          smallest := sigma_j[counter3]
        end;{if counter3 ...}
     end;{for counter3...}
     sigma_star:=smallest;
  end;{subprocedure find sigma star}

procedure find_next_operation(sigma_star:real; sigma_j : operations_array;
                              steps : ZZ; s_sub_t : XX; projinfo : proj_block;
                              var nextoperation : operations);


var
smallest,spt : real;
counter4 : operations;
spt_operation : operations;
L : integer;

begin {procedure find next operation}

  smallest:=1000;
  for counter4:= a1 to c10 do
   begin
   if counter4 in s_sub_t then
    begin
    if sigma_j[counter4] = sigma_star then
      begin
       case counter4 of
       a1,b1,c1 : L:=1;
       a2,b2,c2 : L:=2;
       a3,b3,c3 : L:=3;
       a4,b4,c4 : L:=4;
       a5,b5,c5 : L:=5;
       a6,b6,c6 : L:=6;
       a7,b7,c7 : L:=7;
       a8,b8,c8 : L:=8;
       a9,b9,c9 : L:=9;
       a10,b10,c10 : L:=10;
       end; {case}
     case steps[L] of
       one : spt :=projinfo[L].tseq_1;
       two : spt :=projinfo[L].tseq_2;
       three : spt :=projinfo[L].tseq_3;
       end; {case}

     if spt< smallest then
      begin
       smallest:=spt;
       spt_operation:=counter4;
      end;{if spt}
    end; {if sigma_j}
   end; {if counter4...}
  end;{for counter4...}
  nextoperation:=spt_operation;
  end; {procedure find next operation}

Procedure update_ps_sub_t (var ps_sub_t : YY; nextoperation : operations;
                          var psindex : integer; var predfinish : AA;
                          var fa_avail:real; var fb_avail : real;
                          var fc_avail:real; steps : ZZ;
                          projinfo : proj_block);




var
   PN : integer;
   projstep : steptype;
   duration : real;

  begin {procedure update ps sub t}

     duration :=0;
     ps_sub_t [psindex] := nextoperation;
     psindex := psindex+1;
       case nextoperation of
          a1,b1,c1 : PN:=1;
          a2,b2,c2 : PN:=2;
          a3,b3,c3 : PN:=3;
          a4,b4,c4 : PN:=4;
          a5,b5,c5 : PN:=5;
          a6,b6,c6 : PN:=6;
          a7,b7,c7 : PN:=7;
          a8,b8,c8 : PN:=8;
          a9,b9,c9 : PN:=9;
          a10,b10,c10 : PN:=10;
          end; {case}

        projstep := steps[PN];
        if projstep = one then
          begin
          duration := projinfo[PN].tseq_1;
          predfinish[PN] := sigma_j[nextoperation] + duration;
          {writeln(' project  ',PN,'  just assigned step one and will finish at
                    predfinish[PN]:5:2);
          writeln(' the duration of this step is  ',duration:5:2);
          }
        end;{if projstep}

        if projstep = two then
          begin
          duration := projinfo[PN].tseq_2;
          predfinish[PN] := sigma_j[nextoperation] + duration;
         { writeln('project ',PN,'  just assigned step two and will finish at ti
                  predfinish[PN]:5:2);
          writeln('the duration of thjis step is  ',duration:5:2);
          }
          end;

        if projstep = three then
          begin
          duration := projinfo[PN].tseq_3;
          predfinish[PN] := sigma_j[nextoperation] + duration;
          end; {if}

        case nextoperation of
          a1,a2,a3,a4,a5,a6,a7,a8,a9,a10 : fa_avail:=fa_avail + duration;
          b1,b2,b3,b4,b5,b6,b7,b8,b9,b10 : fb_avail:=fb_avail + duration;
          c1,c2,c3,c4,c5,c6,c7,c8,c9,c10 : fc_avail:=fc_avail + duration;
          end; {case}

     end; {procedure update ps sub t}

  procedure update_s_sub_t( var s_sub_t : XX; nextoperation : operations;
                            var steps : ZZ; projinfo : proj_block);

  var
  PN,z : integer;
  projstep : steptype;

  begin { procedure update s sub t }
    case nextoperation of
      a1,b1,c1 : PN:=1;
      a2,b2,c2 : PN:=2;
      a3,b3,c3 : PN:=3;
      a4,b4,c4 : PN:=4;
      a5,b5,c5 : PN:=5;
      a6,b6,c6 : PN:=6;
      a7,b7,c7 : PN:=7;
      a8,b8,c8 : PN:=8;
      a9,b9,c9 : PN:=9;
      a10,b10,c10 : PN:=10;
    end; {case}

    projstep:=steps[PN];
    if projstep = one then
      begin
      steps[PN]:=two;
      {writeln(' project  ',PN,' is now in step two');
      writeln('type 1 to continue');
      readln(z);}
     s_sub_t := s_sub_t + [projinfo[PN].seq_2];
     end;

    if projstep = two then
      begin
      steps[PN] :=three;
      {writeln(' project  ',PN,' is now in step three');
      writeln('type 1 to continue');
      readln(z);}
      s_sub_t := s_sub_t + [projinfo[PN].seq_3];
      end;

    s_sub_t := s_sub_t - [nextoperation];{this takes the next operation out of s

end; {procedure update s sub t }

procedure find_total_time(var totaltime : real; numprojects : integer;
                          predfinish : AA; projinfo : proj_block;
                          holding : holdingset);


var

PN : integer;
largest : real;

begin {procedure find total time}

  largest := 0;
  for PN:=1 to numprojects do
    begin
    if PN in holding then
     begin
      if predfinish[PN] > largest then
         largest := predfinish[PN];
     end; { if PN}
    end;{for PN}
    totaltime:=largest;
   end; {procedure find total time}

begin{procedure scheduler}
     done := false;
     totaltime:=0;

               holding:=[];
               psindex:=1;
               for l:=1 to 10 do
                  begin
                  predfinish[l]:=0;
                  steps[l]:=one;
                  end;{for l}
               s_sub_t:=[];
               for l:=1 to 30 do
                    ps_sub_t[l]:=none;
               sigma_star:=0;
               fa_avail:=0;
               fb_avail:=0;
```

```
                fc_avail:=0;
                for counter:=a1 to c10 do
                   sigma_j[counter]:=0;
                t:=0;

                for l:=1 to numprojects do
                   begin
                   if l in pset then
                     begin
                     steps[l] := one;
                     holding:=holding + [l];
                     s_sub_t := s_sub_t + [projinfo[l].seq_l];
                     end; { if l in ...}
                end; {for l ...}
                { now s sub t has the first schedulable operations in the
                  project set}

           while done = false do

               begin
               find_sigma_j (sigma_j,s_sub_t,predfinish,fa_avail,fb_avail,
                             fc_avail);
               find_sigma_star(sigma_star,sigma_j,s_sub_t);
               find_next_operation(sigma_star,sigma_j,steps,
                                   s_sub_t,projinfo,nextoperation);
               update_ps_sub_t(ps_sub_t,nextoperation,psindex,predfinish,
                               fa_avail,fb_avail,fc_avail,steps,projinfo);
               update_s_sub_t(s_sub_t,nextoperation,steps,projinfo);
               if s_sub_t = [] then
                  done := true;
               end; {while loop}

               find_total_time(totaltime,numprojects,predfinish,projinfo,
                               holding);
               schedtime:=totaltime;
end; {procedure scheduler}



begin{main program}
  project_input(projinfo);
  setlimits(constraints);
  search_input(numprojects);
  assign(best_output,'bestout.dat');
  rewrite(best_output);
  for p1:=0 to 1 do
   for p2:=0 to 1 do
    for p3:=0 to 1 do
     for p4:=0 to 1 do
      for p5:=0 to 1 do
       for p6:=0 to 1 do
        for p7:=0 to 1 do
         for p8:=0 to 1 do
          for p9:=0 to 1 do
           for p10:=0 to 1 do
           begin
        schedule_time:=0;
        pset:=[];
        if p1 = 1 then pset:=pset+[1];
        if p2 = 1 then pset:=pset+[2];
        if p3 = 1 then pset:=pset+[3];
        if p4 = 1 then pset:=pset+[4];
        if p5 = 1 then pset:=pset+[5];
        if p6 = 1 then pset:=pset+[6];
        if p7 = 1 then pset:=pset+[7];
        if p8 = 1 then pset:=pset+[8];
        if p9 = 1 then pset:=pset+[9];
        if p10 = 1 then pset:=pset+[10];

        check_feasibility(pset,constraints,numprojects,projinfo,feasible);

        if feasible then
        begin
        compute_expected_profit(expected_profit,pset,projinfo,
                                numprojects);
        scheduler(schedule_time,pset,projinfo,numprojects);

  {write out expected_profit}
  {write out schedule_time}
         if expected_profit >= 100.0 then
             begin
             append(best_output);
             writeln(best_output,'The following projects are in this set:');
             for i:=1 to 10 do
             begin
             if i in pset then writeln(best_output,i);
             end;
               writeln(best_output,'the expected profit is ',expected_profit:
               writeln(best_output,'the schedule time is ',schedule_time:7:1)
               writeln;
             close(best_output);
             end; {if expected..}
        end;{if feasible}
       end;{for p10}

end.
```

Vita

Mark A. Coffin was born in Northampton, Massachusetts on June 14, 1962. He received a Bachelor of Mechanical Engineering degree, with highest honors, from the Georgia Institute of Technology in September, 1983, and a Masters in Business Administration from the Virginia Polytechnic Institute and State University in February 1988. He worked as a graduate teaching assistant and as an instructor of quantitative methods and operations management during his four years in the doctoral program.

Mr. Coffin's industrial experience includes work at General Electric Corporation as a industrial sales engineer trainee, and as a sales engineer for York International in Orlando, Florida.

Mr. Coffin is a member of Phi Kappa Phi honor society, Tau Beta Pi engineering honor society, and Phi Eta Sigma honor society. He is a member of The Institute of Management Sciences, The Decision Sciences Institute, and the Southeast Chapter of The Institute of Management Sciences.

Mark A. Coffin