

Modeling, Analysis and Solution Approaches for Some Optimization
Problems: High Multiplicity Asymmetric Traveling Salesman, Primary
Pharmaceutical Manufacturing Scheduling, and Lot Streaming in an
Assembly System

Liming Yao

Dissertation submitted to the Faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Industrial and Systems Engineering

Subhash C. Sarin, Chair

F. Frank Chen

Barbara M. P. Fraticelli

Robert H. Sturges

May 27, 2008

Blacksburg, Virginia

Keywords: Lot-sizing, scheduling, lot streaming, high multiplicity asymmetric traveling salesman, subtour elimination, primary pharmaceutical manufacturing, sequence-dependent setup, carryover setup, assembly system.

Copyright 2008, Liming Yao

Modeling, Analysis and Solution Approaches for Some Optimization Problems: High Multiplicity Asymmetric Traveling Salesman, Primary Pharmaceutical Manufacturing Scheduling, and Lot Streaming in an Assembly System

Liming Yao

(ABSTRACT)

This dissertation is devoted to the modeling, analysis and development of solution approaches for some optimization-related problems encountered in industrial and manufacturing settings. We begin by introducing a special type of traveling salesman problem called “High Multiplicity Asymmetric Traveling Salesman Problem” (HMATSP). We propose a new formulation for this problem, which embraces a flow-based subtour elimination structure, and establish its validity for this problem. The model is, then, incorporated as a substructure in our formulation for a lot-sizing problem involving parallel machines and sequence-dependent setup costs, also known as the “Chesapeake Problem”. Computational results are presented to demonstrate the efficacy of our modeling approach for both the generic HMATSP and its application within the context of the Chesapeake Problem.

Next, we investigate an integrated lot-sizing and scheduling problem that is encountered in the primary manufacturing facility of pharmaceutical manufacturing. This problem entails determination of production lot sizes of multiple products and sequence in which to process the products on machines, which can process lots (batches) of a fixed size (due to limited

capacity of containers) in the presence of sequence-dependent setup times/costs. We approach this problem via a two-stage optimization procedure. The lot-sizing decision is considered at stage 1 followed by the sequencing of production lots at stage 2. Our aim for the stage 1 problem is to allocate batches of products to time-periods in order to minimize the sum of the inventory and backordering costs subject to the available capacity in each period. The consideration of batches of final products, in addition to those for intermediate products, which comprise a final product, further complicates the lot-sizing problem. The objective for the stage 2 problem is to minimize sequence-dependent setup costs. We present a novel unifying model and a column generation-based optimization approach for this class of lot-sizing and sequencing problems. Computational experience is first provided by using randomly generated data sets to test the performances of several variants of our proposed approach. The efficacy of the best of these variants is further demonstrated by applying it to the real-life data collected with the collaboration of a pharmaceutical manufacturing company.

Then, we address a single-lot, lot streaming problem for a two-stage assembly system. This assembly system is different from the traditional flow shop configuration. It consists of m parallel subassembly machines at stage 1, each of which is devoted to the production of a component. A single assembly machine at stage 2, then, assembles products after components (one each from the subassembly machines at the first stage) have been completed. Lot-detached setups are encountered on the machines at the first and second stages. Given a fixed number of transfer batches (or sublots) from each of the subassembly machines at stage 1 to the assembly machine at stage 2, our problem is to find subplot sizes so as to minimize the makespan. We develop optimality conditions to determine subplot sizes for the general problem, and present polynomial-time algorithms to determine optimal subplot sizes for the assembly system with two and three subassembly machines at stage 1.

Finally, we extend the above single-lot, lot streaming problem for the two-stage assembly system to multiple lots, but still, for the objective of minimizing the makespan. Due to the presence of multiple lots, we need to address the issue of the sequencing of the lots along

with lot-splitting, a fact which adds complexity to the problem. Some results derived for the single-lot version of this problem have successfully been generalized for this case. We develop a branch-and-bound-based methodology for this problem. It relies on effective lower bounds and dominance properties, which are also derived. Finally, we present results of computational experimentation to demonstrate the effectiveness of our branch-and-bound-based methodology. Because of the tightness of our upper and lower bounds, a vast majority of the problems can be solved to optimality at root node itself, while for others, the average gap between the upper and lower bounds computed at node zero is within 0.0001%. For a majority of these problems, our dominance properties, then, effectively truncate the branch-and-bound tree, and obtain optimal solution within 500 seconds.

To my father Shigen Yao, my mother Xiaoping Tang, and my wife Min Ji.

Acknowledgements

This dissertation would not be possible without the exceptionally valuable assistance, guidance and support that I have received from my advisor, Dr. Subhash C. Sarin. He has introduced me to various interesting research areas and project work, and has spent tremendous amount of time and effort on discussing the issues related to problem features, modeling, development of solution methodologies, and on correcting my dissertation writing. He has transformed me from an empty-handed novice to an experienced researcher who is now ready to take challenges. Dr. Sarin has also given me lots of advice in my personal life and career development. I regard him as my teacher and personal friend.

I would like to sincerely thank the other members of my dissertation committee for all the help and suggestions that I have received at various stages of my Ph.D. study. In particular, I would like to thank Dr. F. Frank Chen, who provided me with the opportunity to work on my first project during my Ph.D. study at Virginia Tech. In addition, I learned a great deal from participating his research group meetings, which also helped me in refining my presentation skills. I want to thank Dr. Barbara M.P. Fraticelli for introducing me to the optimization world by teaching a wonderful linear programming class. I felt so lucky to be in her class, and mathematical programming no longer remained a mystery to me. I also thank Dr. Robert H. Sturges, for whom I worked as a teaching assistant in his undergraduate class. His unique teaching style has been inspirational at various stages of my dissertation research.

I am grateful to Dr. Hanif D. Serali for offering so many useful optimization theories and techniques in his classes. Sitting in his class that began at eight clock in the morning

has been my unforgettable and pleasant memory at Virginia Tech. I also thank him for his valuable advice in my dissertation research.

I would also like to thank all the staff in the ISE department. In particular, my special thanks go to Ms. Lovedia Cole for her untiring and timely assistance and support starting from the ISE recruiting weekend back in February, 2003 through my last day in the ISE department, and to Ms. Kim Ooms for her extremely patient assistance in many administrative-related aspects of my study and work in the ISE department.

I would not be able to accomplish such a long journey without the continuous support from all my friends. I would like to thank my roommates Jian Zuo and Ende Pan for making wonderful dinners together and sharing time in our “Blacksburg Kitchen”. I want to thank Yuqiang Wang, Ming Chen, Lixin Wang, Chengbin Zhu, Yong Yang, Xiaomei Zhu, Guorong Huang, Xiangshang Tong, Weiping Chen, Cheng Guo, Ming Cheng, Shiyong Liu, Seon ki Kim and Ying (Ella) Fu for their support and for bringing lots of fun to my life in Blacksburg. I would like to thank Hungda Wan, Jianchen Su, Rami Musa, Leonardo Rivera, Wei Tang and Radu Babiceanu, in FMS group, for their support and constructive feedback during group meetings. Also, special thanks go to Hungda Wan for his sincere collaboration in our joint project work.

My journey would never have started without my parents. I would like to express my earnest thanks to my father, Shigen Yao and my mother Xiaoping Tang, for their unconditional love and support for the entire thirty two years.

Finally, I am so deeply grateful to my wife, Min Ji, for her understanding, inspiration and encouragement during these years, for the sacrifice you have made, and for the hard time we have gone through together. I am so lucky to have you in my life!

Contents

Acknowledgements	vi
List of Tables	xi
List of Figures	xiii
1 Introduction	1
1.1 Motivation and Scope of Research	1
1.2 Organization of Dissertation	3
2 High Multiplicity Asymmetric Traveling Salesman Problem (HMATSP)	5
2.1 Introduction	5
2.2 Existing Formulation for the HMATSP	9
2.2.1 Notation	9
2.2.2 Grigoriev and van de Klundert’s Model	10
2.3 A Polynomial-Length Formulation for the HMATSP	11
2.3.1 Comparison of HMATSP-GK and HMATSP-P	13
2.4 The Chesapeake Problem	13
2.4.1 Model Formulation	13
2.4.2 Computational Results	21
2.5 Concluding Remarks	22
3 Primary Pharmaceutical Manufacturing Scheduling Problem (PPMSP)	23

3.1	Background and Motivation	23
3.2	Literature Review	25
3.3	Problem Description	28
3.4	Basic Mixed Integer Programming Formulation for the PPMSP	32
3.5	Column Generation Heuristic Approach	37
3.5.1	Pattern Definition	37
3.5.2	Column Generation Method for PPMSP1 (PPMSP-CG1)	38
3.5.2.1	Master Problem: MP-SEQUENCE	38
3.5.2.2	Subproblem: SP-LSP	39
3.5.3	Column Generation Method for PPMSP2 (PPMSP-CG2)	40
3.5.3.1	Master Problem: MP-SEQUENCE	40
3.5.3.2	Subproblem: SP-LSP	41
3.5.4	Outline of the Column Generation Approach	41
3.6	Computational Experimentation	42
3.6.1	Comparison of PPMSP1 and PPMSP2	43
3.6.2	Comparison of PPMSP-CG1 and PPMSP-CG2	44
3.7	A Real-life-size Example	49
3.8	Concluding Remarks	53
4	Single-Lot Lot Streaming in a Two-stage Assembly System	55
4.1	Introduction	55
4.2	Problem Description and Formulation	59
4.2.1	Mathematical Formulation	60
4.3	Development of Optimality Conditions	64
4.3.1	Machine Dominance Properties	64
4.3.2	Optimality Conditions	68
4.4	Algorithm for 2+1 Problem	76
4.4.1	Algorithm 2+1	80
4.4.2	Example 1	81

4.5	Algorithm for 3+1 Problem	83
4.5.1	Algorithm 3+1	86
4.5.2	Example 2	87
4.6	Integer-size Sublots	90
4.6.1	Algorithm -Integer Sublot Sizes	91
4.6.2	Example 3	92
4.7	Concluding Remarks	93
5	Multiple-Lot Lot Streaming in a Two-stage Assembly System	94
5.1	Introduction	94
5.2	Problem Description and Basic Properties	98
5.3	A Mixed Integer Programming Formulation	100
5.4	A Branch-and-Bound-based Methodology for the TSMLSP	102
5.4.1	Expression of Makespan	102
5.4.2	Determination of Lower and Upper Bounds	104
5.4.3	Development of Dominance Rules	106
5.4.3.1	Properties for the First and the Last Sublots	107
5.4.3.2	Dominance Rules	111
5.4.4	Branch-and-Bound-based Algorithm	119
5.5	Computational Experimentation	121
5.5.1	Computational Test of TSMLSP-BB	121
5.5.2	Comparison of TSMLSP-BB and TSMLSP-MIP	125
5.5.3	Computational Test of TSMLSP-BB for Large-size Problems	126
5.6	Concluding Remarks	130
6	Summary and Conclusion	131
	Bibliography	134

List of Tables

2.2	Comparison of HMATSP-GK and HMATSP-P	14
2.4	CHES problems	21
2.5	Results for the instances of the CHES problem	22
3.2	Sets of problem instances	42
3.3	Comparison of PPMSP1 and PPMSP2	44
3.4	Comparison of PPMSP-CG1 and PPMSP-CG2	45
3.5	Computational results of PPMSP-CG1	47
3.6	Comparison of PPMSP-CG1 and PPMSP2	48
3.7	Data of product families and inventory levels (kg)	49
3.8	A matrix of material requirement for product family B	49
3.9	Demand data (kg)	50
3.10	Available time for bays in each time-period (hrs)	50
3.11	Processing Data for bays (where q stands for batch size (kg) and pt stands for processing time (hrs))	51
4.1	Data for a 2+1 problem	81
4.2	Data for a 3+1 problem	88
4.3	Iteration 1	92
4.4	Iteration 2	92
5.2	Sets of problem instances	121
5.3	Computational results of TSMLSP-BB	123

5.4	Sets of problem instances	125
5.5	Comparison of TSMLSP-MIP with TSMLSP-BB	126
5.6	Sets of problem instances	127
5.7	Computational experimentation for TSMLSP-BB	128

List of Figures

2.1	An example of connected and disconnected cycles for the HMATSP	6
2.2	A feasible solution to the CHES problem	8
3.1	A primary pharmaceutical manufacturing shop floor configuration	28
3.2	The relationship of final products to their intermediaries	31
3.3	An example of a schedule with two bays, three periods and four product families	31
3.4	Sequence-dependent setup in bay k from time-period t to $t + 1$	32
3.5	The Proposed Schedule of the real-life example	52
3.6	The Current Schedule of the real-life example	53
4.1	An example of geometric subplot sizes	56
4.2	Example of lot streaming in a two-stage assembly system	58
4.3	Network representation of the lot streaming problem	63
4.4	Illustration of completion times on two machines	64
4.5	Illustration of completion times on three machines	67
4.6	Illustration of completion times on m machines	68
4.7	Illustration of optimality conditions	75
4.8	Flow-chart for Algorithm 2 + 1	80
4.9	Illustration of the optimal solution to Example 1	83
4.10	Flow-chart for Algorithm 3 + 1	88
4.11	Illustration of the optimal solution to Example 2	90
5.1	Example depicting streaming of multiple lots in a two-stage assembly system	97

5.2	The branch-and-bound tree for the TSMLSP	103
5.3	Flowchart for the proposed branch-and-bound approach	120
5.4	The ACT values and the ANN explored by TSMLSP-BB	124
5.5	Computational comparison of TSMLSP-BB and TSMLSP-MIP with respect to the ACT values and the ANN explored	127
5.6	The ACT, NSR and NU values for TSMLSP-BB	129

Chapter 1

Introduction

1.1 Motivation and Scope of Research

The classical traveling salesman problem (TSP) is to determine a cycle over a given set of cities, which starts from a base city, visits all the other cities once, and returns to the base city, while minimizing the total distance traveled. If the distance from a city “ a ” to city “ b ” is different from that from city “ b ” to city “ a ”, then this problem is termed as an asymmetric traveling salesman problem. The problem that we address is different from this classical asymmetric traveling salesman problem in that each node is visited multiple times. We call such a variant of the classical traveling salesman problem as “The High Multiplicity Traveling Salesman Problem” (HMATSP). Our main motivation for the study of HMATSP is its occurrence as a substructure in the lot-sizing problem involving parallel machines and sequence-dependent setup costs, also known as the “Chesapeake Problem”. This is a general form of a production scheduling problem that has been addressed in the literature. A critical construct for the HMATSP is the formulation of subtour elimination constraints (SECs), which maintain the connectivity of a tour. Not much work has been reported in this regard in the literature. We develop flow-based, polynomial-length subtour elimination constraints for the HMATSP and prove their validity. Computational results are presented to demonstrate the efficacy of our modeling approach for both the generic HMATSP and its

application within the construct of the Chesapeake Problem.

Secondly, we investigate a primary pharmaceutical manufacturing problem. The operational stage of the pharmaceutical supply chain is one of its crucial components. At this stage, which involves both primary and secondary manufacturing, it is not unusual to have long cycle times (of the orders of hundred of days). This severely impacts responsiveness to changing market trends. More often than not, erratic dynamics of the operational stage (and the supply chain) are introduced by internal business processes than by external demand, and can be eliminated by effectively re-designing the internal processes (see Shah (2004)). Besides, time-to-market is the single most important driver now-a-days in the pharmaceutical industry as significant revenues are reaped in the early life of a drug. Because of an intense competition among the companies, the competition-free period has also decreased from 5 to 1-2 years. All of these facts point to a greater need of improving the performance of the operational stage of the pharmaceutical supply chain. Our proposed problem directly addresses this issue. In particular, our aim is to develop and validate an approach for the optimal assignment and sequencing of pharmaceutical products to various processing bays of a primary manufacturing facility for the objective of meeting customer requirements with minimal production costs, given a set of equipment with specified capacities as well as a product-mix and the demand rate for each product. We present a novel, unifying model (integrated lot-sizing and scheduling) and a column generation-based optimization approach for this class of lot-sizing and sequencing problems. Variants of this approach are tested numerically on randomly generated data to determine the best one. The selected variant, is then, used on an industrial-size data provided by a pharmaceutical manufacturing company to demonstrate the applicability of the proposed methodology in practice.

Thirdly, we address a single-lot lot streaming problem for a two-stage assembly system. The assembly system that we have considered is different from the flow shop configuration that is typically assumed in the lot streaming literature. An example of this two-stage assembly system is a supplier-manufacturing tandem for the assembly of a product. The required components are produced by suppliers (at stage 1), one component by each supplier,

for assembly at stage 2. The real-life instances of such a system include dressing of engines and transmissions by suppliers (stage 1) for their assembly at the assembly plant (stage 2), or preparation of integrated circuits (ICs) and other components by suppliers (stage 1) for their assembly on printed circuit boards (PCBs) (stage 2). The assembly facility needs to coordinate its operations with the availability of components provided by the suppliers. We develop new machine dominance properties and optimality conditions for the determination of subplot sizes, and present polynomial-time algorithms to determine optimal subplot sizes for the cases of two and three subassembly machines (suppliers).

Finally, we extend the single-lot, lot streaming problem for a two-stage assembly system to multiple production lots. The consideration of multiple lots introduces another issue pertaining to the sequencing of the lots. The problem of sequencing the lots for the two-stage assembly system that we consider has been addressed in the literature but without the consideration of lot splitting. Therefore, we present new results for our problem that are derived from the properties of the single-lot, lot streaming problem. We, then, propose a branch-and-bound-based methodology that relies on effective lower bounds and dominance properties. Finally, results of computational experimentation are presented on the use of this branch-and-bound-based methodology for the solution of our problem in order to show its efficacy.

1.2 Organization of Dissertation

The remainder of this dissertation is organized as follows. In Chapter 2, we first introduce the “high multiplicity asymmetric traveling salesman problem (HMATSP)”. Next, we adopt a flow-based subtour elimination structure and establish its validity for the HMATSP. Subsequently, we incorporate this problem as a substructure in the formulation for the lot-sizing problem involving parallel machines and sequence-dependent setup costs, also known as the “Chesapeake Problem”. Computational results are presented to demonstrate the efficacy of our approach for both the generic HMATSP and its application within the construct of

the Chesapeake Problem. In Chapter 3, we introduce and discuss a primary pharmaceutical manufacturing scheduling problem and its operational features. We develop a comprehensive model and optimization approach for this problem, and present a decomposition-based approach for its solution. We also present results on the application of our proposed approach to a real-life problem instance (and data) provided by a pharmaceutical company. In Chapter 4, we address a single-lot, lot streaming problem encountered in a two-stage assembly system. We present and exploit machine dominance properties and optimality conditions for this problem to develop polynomial-time algorithms for instances involving two and three subassembly machines. In Chapter 5, we, then, extend the work presented in Chapter 4 to the case of multiple lots, and develop a branch-and-bound-based methodology. It relies on tight lower and upper bounds and dominance rules for an effective fathoming of the branch-and-bound tree. Results of a detailed experimentation are also presented that clearly depict efficacy of our solution approach for this problem. Finally, in Chapter 6, we conclude this dissertation with a summary of the works accomplished and present directions for future research.

Chapter 2

High Multiplicity Asymmetric Traveling Salesman Problem (HMATSP)

This chapter presents a new model for a special type of traveling salesman problem called the “High Multiplicity Asymmetric Traveling Salesman Problem (HMATSP)”. The formulation adopts a flow-based subtour elimination structure and establishes its validity for this problem. The model is then incorporated as a substructure in a formulation for the lot-sizing problem involving parallel machines and sequence-dependent setup costs, also known as the “Chesapeake Problem”. Computational results are presented to demonstrate the efficacy of our modeling approach for both the generic HMATSP and its application within the context of the Chesapeake Problem.

2.1 Introduction

The High Multiplicity Traveling Salesman Problem (HMATSP) is a variant of the classical traveling salesman problem in which each node is visited multiple times. We can define this problem as follows:

Given a complete undirected graph with vertex set N , an asymmetric distance matrix $[c_{ij}]$, and positive integers n_i , $i \in N$, find a connected and minimal length (non-simple) cycle that starts from some base node and ends at the same node after having visited node i exactly n_i times, $\forall i \in N$.

Figure 2.1 depicts a problem with eight cities where each city requires a pre-specified number of visits. Figure 2.1(a) illustrates a feasible (connected) solution to the problem, a connected cycle in which each city is visited a specified number of times, while Figure 2.1(b) shows an infeasible (disconnected) cycle with two subtours, although each city is visited the required number of times.

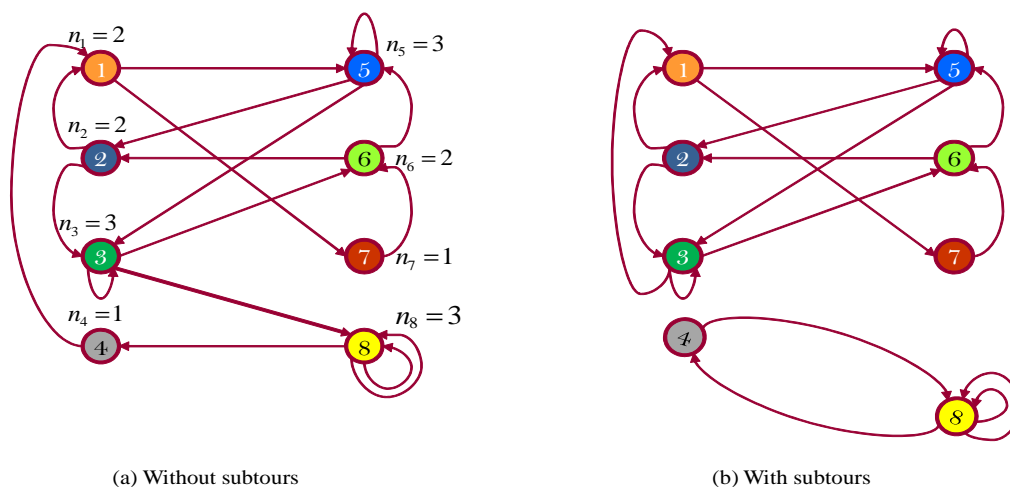


Figure 2.1: An example of connected and disconnected cycles for the HMATSP

Such a problem arises as for example when scheduling the manufacture of some product types on a single machine, where each product type contains several identical items that need to be separately processed. Cosmadakis and Papadimitriou (1984) have developed an algorithm for this problem that requires an effort of $O(e(|N|) \log(\sum\{n_i\}))$, where $e(|N|)$ is an exponential function of $|N|$. Grigoriev and van de Klundert (2006) have investigated the sensitivity of the objective function to a change in the numbers of visits to the nodes. The algorithm proposed by Cosmadakis and Papadimitriou (1984) is effective only when the total

number of cities is small (≤ 10) compared to the number of visits to each city. However, it is not unusual to find a large number of cities (product types) in real-life applications. The formulation due to Grigoriev and van de Klundert (2006) for the HMATSP utilizes an exponential number of subtour elimination constraints, which grow rather quickly even for small-sized problems. Moreover, it is difficult to adapt this formulation to the case of multiple-machines.

In this chapter, we present a compact but tight mathematical formulation that contains a polynomial number of subtour elimination constraints for the HMATSP and that also permits an extension to multiple-machine environments.

In addition, we also apply the proposed HMATSP formulation to a real industrial problem. Our work is motivated by a report from Chesapeake Decision Sciences, Inc. (1989), which provides a set of real-life problem instances. These problems, termed CHES, were submitted by Dupont, BASF, James River, and Champion International, based on existing business practices. The particular CHES problem that is of concern here is to determine, for each of a given set of parallel machines, an allocation of batches of different products and a sequence in which to process these batches that are assigned to it. Since batches of only a given maximum size can be processed on any machine (because of limited machine capacity), a production lot may be produced via several batches over a sequence of time periods. A sequence-dependent setup cost is incurred between the processing of batches on each machine. Therefore, the objective of the CHES problem is to determine an allocation of batches of the products on the machines and a sequence for processing these batches such that the sum of the setup, production, and holding costs, minus the sales revenue, is minimized, while ensuring that, in each period, each product demand is satisfied and that the production on each machine does not exceed its capacity. This problem can also be classified as a capacitated lot-sizing problem (CLSP), albeit, in the presence of sequence-dependent setup costs.

Figure 2.2 depicts a feasible solution to the foregoing CHES problem, where the circles indicate the batches that are processed on two machines over three periods. Note that if production takes place during a period on a machine, then there exists a first batch and a

last batch, and perhaps batches in-between. A carryover setup (see Kang et al. (1999)) is incurred in the example of Figure 2.2 when transitioning from period 1 to period 2, and from period 2 to period 3 on machine 1, while, on machine 2, no production is scheduled in the second period, and hence, the carryover setup occurs from the end of period 1 to the start of period 3.

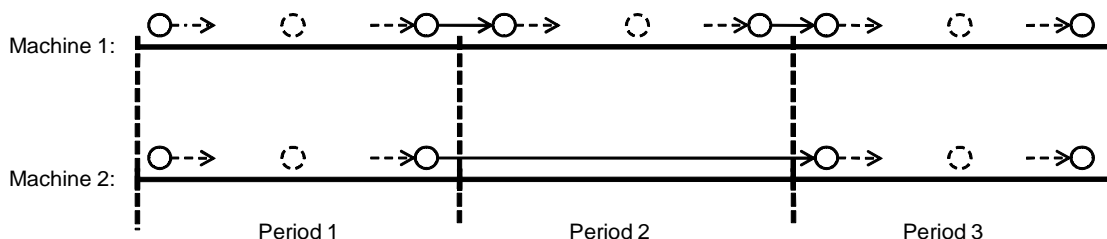


Figure 2.2: A feasible solution to the CHES problem

Furthermore, consecutive production runs of batches belonging to the same product are prohibited on any machine. This feature precludes the possibility of modeling the problem as a classical asymmetric traveling salesman problem. For instance, the sequence of batches $\{1, 2, 2, 3, 3, 4\}$ belonging to various products is not feasible, whereas the sequence $\{1, 2, 3, 2, 3, 4\}$ is feasible.

The CHES problems were first presented in a paper by Baker and Muckstadt (1989) as a collection of practical problems compiled by Chesapeake Decision Sciences. A CHES problem comprises parallel production lines and sequence-dependent setup cost but no setup time. Kang et al. (1999) developed a column generation and branch-and-bound-based scheme for this problem and presented a mixed-integer programming (MIP) formulation without an explicit representation of the subtour elimination constraints. Belvaux and Wolsey (2001) have provided an MIP formulation containing an exponential number of subtour elimination constraints (SECs) for the CHES problem, where their SECs are progressively generated within their solution approach as needed. Meyr (2002) has proposed a procedure that combines a meta-heuristic (threshold accepting or simulated annealing) with dual-optimization for gen-

eral lot-sizing and scheduling problems on parallel machines, and has tested this method on various industrial-sized problems including the CHES problem. Apart from the CHES problem, Dastidar and Nagi (2005) have presented an MIP formulation and have proposed a heuristic to solve the lot-sizing and scheduling problem having sequence-dependent setups, carryover setup, and parallel machines. An application for the single machine, sequence-dependent setup cost and time, and carryover setup has been presented by Hasse and Kimms (2000) and Gupta and Magnusson (2005). The MIP formulation by Hasse and Kimms (2000) considers only ‘efficient’ sequences, while Gupta and Magnusson (2005) propose a heuristic procedure for its solution. Small bucket lot-sizing problem having sequence-dependent setups have been considered by Fleischmann (1994) and have been modeled as a TSP with time-windows. A Lagrangean relaxation approach in combination with a heuristic was used to solve this problem.

The remainder of this chapter is organized as follows. In Section 2.2, we present an existing formulation for the HMATSP due to Grigoriev and van de Klundert (2006), and then, introduce our proposed formulation for the HMATSP in Section 2.3 and establish its validity. Computational results are also presented in this latter section to exhibit the relative efficacy of our formulation. In Section 2.4, we incorporate our HMATSP formulation as a substructure to model the CHES problem, and demonstrate the effectiveness of this model in solving the instances for the CHES problem that are presented in the literature. Finally, concluding remarks are provided in Section 2.5.

2.2 Existing Formulation for the HMATSP

2.2.1 Notation

We use the following notation.

Parameters:

N - Set of cities.

n_i - Number of visits required for city i .

c_{ij} - Distance from city i to j .

Decision Variables:

x_{ij} - Number of times city i directly precedes city j .

2.2.2 Grigoriev and van de Klundert's Model

An integer programming model due to Grigoriev and van de Klundert (2006) for the HMATSP is as follows.

HMATSP-GK:

$$\text{Minimize} \quad \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \quad (2.1a)$$

$$\text{subject to:} \quad \sum_{i \in N} x_{ij} = n_j, \quad \forall j \in N, \quad (2.1b)$$

$$\sum_{i \in N} x_{ji} = n_j, \quad \forall j \in N, \quad (2.1c)$$

$$\sum_{i \in N'} \sum_{j \in N - N'} x_{ij} \geq 1, \quad \forall N' \subset N, N' \neq \emptyset, \quad (2.1d)$$

$$x_{ij} \geq 0 \text{ and integer}, \quad \forall i, j \in N. \quad (2.1e)$$

The constraint sets (2.1b) and (2.1c), respectively, ensure that each node j is visited and exited n_j times. Note that, we do not require $i \neq j$ since self-loops are permitted. The constraint set (2.1d) eliminates subtours, i.e., disconnected cycles in the present context. This formulation contains $O(2^{|N|})$ such subtour elimination constraints. Grigoriev and van de Klundert (2006) show that given a solution $x_{ij}, \forall i, j \in N$, to the above problem, it is possible to then construct the corresponding cycle or sequence of visits in $O(|N|^4)$ time.

Remark 2.1. Note that similar to the ATSP, the restrictions (2.1d) are equivalent to the following DFJ (Dantzig-Fulkerson-Johnson) type of subtour elimination constraints:

$$\sum_{i \in N'} \sum_{j \in N'} x_{ij} \leq \sum_{i \in N'} n_i - 1, \quad \forall N' \subset N, N' \neq \emptyset. \quad (2.2)$$

To see this, note that from (2.1c), we have

$$\sum_{i \in N'} n_i = \sum_{i \in N'} \sum_{j \in N} x_{ij} = \sum_{i \in N'} \sum_{j \in N-N'} x_{ij} + \sum_{i \in N'} \sum_{j \in N'} x_{ij}.$$

Hence, (2.1d) holds true if and only if (2.2) does.

2.3 A Polynomial-Length Formulation for the HMATSP

Next, we present an alternative formulation for the HMATSP that replaces the exponential number of subtour elimination constraints of model HMATSP-GK with a polynomial number of maximum flow constraints of the type used by Wong (1980), Sarin et al. (2005), and Sherali et al. (2006) in their formulations for the ATSP. We denote this formulation as HMATSP-P.

HMATSP-P:

$$\text{Minimize} \quad \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \quad (2.3a)$$

$$\text{subject to} \quad \sum_{i \in N} x_{ij} = n_j, \quad \forall j \in N, \quad (2.3b)$$

$$\sum_{i \in N} x_{ji} = n_j, \quad \forall j \in N, \quad (2.3c)$$

$$0 \leq p_{ij}^u \leq x_{ij}, \quad \forall u, j \in N - \{1\}, \forall i \in N, i \neq j, i \neq u, \quad (2.3d)$$

$$\sum_{j \in N - \{1\}} p_{1j}^u = 1, \quad \forall u \in N - \{1\}, \quad (2.3e)$$

$$\sum_{j \in N - \{1\}, j \neq i} p_{ij}^u - \sum_{j \in N, j \neq i} p_{ji}^u = 0, \quad \forall i, u \in N - \{1\}, i \neq u, \quad (2.3f)$$

$$\sum_{j \in N, j \neq u} p_{ju}^u = 1, \quad \forall u \in N - \{1\}, \quad (2.3g)$$

$$x_{ij} \geq 0 \text{ and integer.} \quad (2.3h)$$

Note that we have defined node 1 to be the base node. The variable x_{ij} is as defined earlier, and $p_{ij}^u = 1$ if commodity u , which is required to traverse from node 1 to node u ,

$\forall u \in N - \{1\}$, flows along the way from node $i \in N$ to $j \in N$, $j \neq i$, and equals 0, otherwise. Constraints (2.3b) and (2.3c) ensure, as before, that each node is visited and exited the required number of times. Constraints (2.3d) assert that if any two nodes are not directly connected ($x_{ij} = 0$), then there is no direct flow transmission between them. Constraints (2.3e) – (2.3g) are the flow conservation constraints that require a unit flow of commodity u to traverse from the base node 1 to node u , $\forall u \in N - \{1\}$. Finally, (2.3h) represents logical restrictions.

Next, we show the validity of formulation HMATSP-P for the HMATSP by showing its equivalence to the formulation HMATSP-GK.

Proposition 2.1. *HMATSP-P is a valid formulation for the HMATSP.*

Proof. Consider a digraph induced by the \mathbf{x} -variables for any feasible solution (\mathbf{x}, \mathbf{p}) to HMATSP-P, where x_{ij} arcs are constructed from node i to node j , $\forall i, j \in N$. By virtue of the flow constraints (2.3d) – (2.3g), there exists a path from node 1 to each node u , $u \in N - \{1\}$. Hence, this digraph is connected and so, has no subtours. Therefore, \mathbf{x} is feasible to HMATSP-GK with the same objective value. Conversely, given any feasible solution \mathbf{x} to HMATSP-GK, there must exist a path from the base node 1 to each node u , $u \in N - \{1\}$, which is induced by the \mathbf{x} -variables. Let T^u denote such a path. We can now consider a flow along this path, which starts at node 1 and terminates at node u , and accordingly set $p_{ij}^u = 1$ for all $(i, j) \in T^u$, whence i.e., $x_{ij} \geq 1$, and let $p_{ij}^u = 0$ otherwise. This yields a flow from the base node 1 to node u that satisfies the flow conservation constraints (2.3d) – (2.3g). Hence, there exists a feasible solution (\mathbf{x}, \mathbf{p}) to HMATSP-P having the same objective value. \square

In addition, let $\text{LP}(\text{HMATSP-GK})$ and $\text{LP}(\text{HMATSP-P})$ represent the respective solution spaces for the LP relaxations of the corresponding formulations. Wong (1980) has shown the equivalence between the LP relaxation of the foregoing type of flow-based formulation for the ATSP and that of the DFJ formulation for ATSP. Using similar arguments and noting

Remark 2.1, it can be shown that:

$$\text{LP(HMATSP-P)} = \text{LP(HMATSP-GK)}.$$

2.3.1 Comparison of HMATSP-GK and HMATSP-P

We tested the relative effectiveness of the HMATSP-GK and HMATSP-P formulations by applying them to several instances of the HMATSP. The data was generated randomly for problems involving 6, 8, 10, 12, 14, 16, 20, 24, 28, and 32 cities. Four instances were randomly generated for each case, by letting the number of visits for each city be obtained from uniform distributions $U(1, 5)$, $U(6, 10)$, $U(11, 15)$, and $U(15, 20)$, respectively. The setup cost matrix was also randomly generated by using a uniform distribution, $U(0, 10)$. All the runs are implemented on an Intel Xeon 3.6 GHz computer. The results obtained for each instance are displayed in Tabel 2.2. The HMATSP-P formulation obtained optimal solutions for all the problem instances tested with a very small cpu time effort, while HMATSP-GK could not solve the problems having 14 or more cities because of excessive memory requirements.

2.4 The Chesapeake Problem

In this section, we incorporate the proposed HMATSP formulation as a substructure for modeling the CHES problem described in Section 2.1. The CHES problem is a generalization of the HMATSP due to the presence of capacitated parallel machines. We need to determine a sequence of batches of different products to process on each of these machines in the presences of sequence-dependent setup costs.

2.4.1 Model Formulation

Our model for the CHES problem is similar to that presented by Kang et al. (1999) and Belvaux and Wolsey (2001) except for the subtour elimination constraints. We use the following notation.

Table 2.2: Comparison of HMATSP-GK and HMATSP-P

Number of cities	Problem	Objective value	cpu time (seconds)	
			HMATSP-GK	HMATSP-P
6 cities	1	75	0.0312	0.0312
	2	82	0.0781	0.0468
	3	209	0.0312	0.0312
	4	162	0.0156	0.0312
8 cities	1	34	0.0468	0.0468
	2	67	0.0468	0.0468
	3	165	0.0312	0.0625
	4	151	0.0468	0.0468
10 cities	1	39	0.0625	0.0468
	2	102	0.0625	0.0625
	3	118	0.0625	0.0468
	4	142	0.0937	0.0781
12 cities	1	49	0.2968	0.0937
	2	106	0.25	0.0781
	3	184	0.25	0.0625
	4	157	0.2656	0.0781
14 cities	1	58	—	0.0937
	2	72	—	0.0937
	3	91	—	0.0937
	4	258	—	0.0781
16 cities	1	37	—	0.125
	2	72	—	0.1875
	3	140	—	0.1406
	4	210	—	0.125
20 cities	1	48	—	0.4218
	2	60	—	0.2343
	3	122	—	0.25
	4	90	—	0.3906
24 cities	1	31	—	0.6562
	2	34	—	1.4062
	3	65	—	0.625
	4	208	—	0.4062
28 cities	1	21	—	2.4375
	2	28	—	3.3125
	3	31	—	1.7656
	4	56	—	2.7344
32 cities	1	24	—	6.1562
	2	26	—	3.8906
	3	33	—	2.9532
	4	78	—	2.9062

Sets:

- J - Set of products, $\{1, 2, \dots, n\}$.
- T - Set of time-periods, $\{1, 2, \dots, \tau\}$.
- M - Set of parallel machines.
- J_k - Set of products that can be produced on machine k .
- J_k^0 - $J_k \cup \{0\}$, where 0 is a dummy product.
- M_i - Set of machines that can process product i .

Parameters:

- d_i^t - Minimal demand of product i to be satisfied in period t .
- h_i^t - Inventory cost for product i in period t .
- pc_i^t - Cost of processing a unit of product i in period t .
- pr_i - Selling price for product i .
- sc_{ijk} - Setup cost incurred when producing product j following product i on machine k .
- pt_{ik} - Processing time of a unit of product i on machine k .
- w_{ik}^1 = 1, if product i is the first product produced on machine k in period 1, and 0, otherwise.
- ec_{ik} - Cost incurred if product i is the last product produced on machine k .
- U_{ik}^t - Maximal lot-size permitted for product i on machine k in period t .
- L_{ik}^t - Minimal lot-size permitted for product i on machine k in period t .
- Q_k - Maximal capacity available on machine k in each period.

Decision Variables:

- I_i^t - Inventory of product i held in period t .
- D_i^t - Number of units of product i sold in period t .
- P_{ik}^t - Number of units of product i produced on machine k in period t .
- Y_{ik}^t - Number of setups of product i on machine k in period t .
- X_{ijk}^t - Number of setups for producing product j following product $i \neq j$ on machine k , in period t .
- $\chi_{ijk}^{tt'}$ = 1, if there is a setup for producing product j in period t' following

- product i in period $t < t'$ on machine k , and
0, otherwise.
- Z_{ik}^t = 1, if product i is the last product produced on machine k in period t , and
0, otherwise.
- Θ_{ik}^t = 1, if product i is produced on machine k in period t , and
0, otherwise.
- $(p_{ijk}^u)^t$ = 1, if commodity u flows from product (node) i to product (node) j
on machine k in period t , and
0, otherwise.

Our formulation for the CHES problem is as follows.

CHESP:

Objective Function:

$$\begin{aligned}
\text{Minimize } & \sum_{t \in T} \sum_{i \in J} h_i^t \cdot I_i^t + \sum_{t \in T} \sum_{k \in M} \sum_{i \in J} \sum_{j \in J} sc_{ijk} \cdot X_{ijk}^t + \sum_{t \in T} \sum_{\substack{t' \in T \\ t' > t}} \sum_{k \in M} \sum_{i \in J} \sum_{j \in J} sc_{ijk} \cdot \chi_{ijk}^{tt'} \\
& + \sum_{k \in M} \sum_{i \in J} ec_{ik} \cdot Z_{ik} + \sum_{t \in T} \sum_{k \in M} \sum_{i \in J} pc_i^t \cdot P_{ik}^t - \sum_{t \in T} \sum_{i \in J} pr_i \cdot D_i^t
\end{aligned} \tag{2.4a}$$

The objective function contains various cost components, due to inventory, setup within periods, carryover setup (from one period to another), end setup, and production. In addition, the sales revenue is subtracted from the cost expression.

Constraints:

1) Flow Balance Constraints:

$$I_i^t + D_i^t - \sum_{k \in M_i} P_{ik}^t = I_i^{t-1}, \quad \forall i \in J, \forall t \in T. \tag{2.4b}$$

This constraint is a typical flow balance constraint, which equates the input (given by the beginning inventory and amount produced) to the output (given by the amount sold and

the end inventory) for each time-period.

2) Minimal and Maximal Lot-Size Constraints:

$$L_{ik}^t \cdot Y_{ik}^t \leq P_{ik}^t \leq U_{ik}^t \cdot Y_{ik}^t, \quad \forall i \in J, \forall k \in M, \forall t \in T. \quad (2.4c)$$

This constraint maintains a sufficient number of batches (setups) of each product so that the permissible batch size limits are not violated.

3) Capacity Constraints:

$$\sum_{i \in J_k} p_{ik}^t \cdot P_{ik}^t \leq Q_k, \quad \forall k \in M, \forall t \in T. \quad (2.4d)$$

This constraint assures that the total production on each machine in each period does not exceed the available capacity.

4) Demand Constraints:

$$D_i^t \geq d_i^t, \quad \forall i \in J, \forall t \in T. \quad (2.4e)$$

This constraint ensures that at least the specified minimal demand for product i , $i \in J$, is satisfied during each period.

5) Transportation Constraints:

$$\sum_{\substack{i \in J_k \\ i \neq j}} X_{ijk}^1 + w_{jk}^1 = Y_{jk}^1, \quad \forall k \in M, \forall j \in J_k, \quad (2.4f)$$

$$\sum_{\substack{i \in J_k \\ i \neq j}} X_{ijk}^t + \sum_{\substack{t' \in T \\ t' < t}} \sum_{\substack{i \in J_k \\ i \neq j}} \chi_{ijk}^{t't} = Y_{jk}^t, \quad \forall k \in M, \forall j \in J_k, \forall t \in T, t \geq 2, \quad (2.4g)$$

$$\sum_{\substack{j \in J_k \\ j \neq i}} X_{ijk}^t + \sum_{\substack{t' \in T \\ t' > t}} \sum_{\substack{j \in J_k \\ j \neq i}} \chi_{ijk}^{tt'} + Z_{ik}^t = Y_{ik}^t, \quad \forall k \in M, \forall i \in J_k, \forall t \in T, t \leq \tau - 1, \quad (2.4h)$$

$$\sum_{\substack{j \in J_k \\ j \neq i}} X_{ijk}^\tau + Z_{ik}^\tau = Y_{ik}^\tau, \quad \forall k \in M, \forall i \in J_k. \quad (2.4i)$$

Constraint (2.4f) captures the fact that, for any given machine k , the number of setups of product j in period 1 is equal to the number of times it is preceded by a different product in period 1 plus an extra setup if it is the first product produced in this period. Constraint (2.4g) is similar to (2.4f), and in particular, accounts for the fact that a batch of product j might be the first one produced in some period t on a given machine k , whence a setup from another batch in an earlier period must be incurred. Constraint (2.4h) represents the out-flow counterpart of Constraint (2.4g) for each period t on any given machine. The constraints (2.4g) and (2.4h) capture the fact that no consecutive production runs of batches belonging to the same product may be scheduled in any given period. Constraint (2.4i) asserts that, on any given machine, the number of setups of product i in the last period equals the number of times another product directly succeeds it plus a setup term to account for the case when it is the last product produced on this machine.

6) Logical Constraints:

$$w_{jk}^1 = X_{0jk}^1, \quad \forall k \in M, \forall j \in J_k, \quad (2.4j)$$

$$Z_{ik}^\tau = X_{i0k}^\tau, \quad \forall k \in M, \forall i \in J_k, \quad (2.4k)$$

$$\sum_{\substack{i \in J_k \\ i \neq j}} \sum_{\substack{t' \in T \\ t' < t}} \chi_{ijk}^{t't} = X_{0jk}^t, \quad \forall k \in M, \forall j \in J_k, \forall t \in T, t \geq 2, \quad (2.4l)$$

$$\sum_{\substack{j \in J_k \\ j \neq i}} \sum_{\substack{t' \in T \\ t' > t}} \chi_{ijk}^{tt'} + Z_{ik}^t = X_{i0k}^t, \quad \forall k \in M, \forall i \in J_k, \forall t \in T, t \leq \tau - 1. \quad (2.4m)$$

By adding a dummy node 0 into the production sequence in each time period, we are able to obtain an Eulerian cycle that both starts from and ends at the dummy node. The above constraints construct such Eulerian cycles. Specifically, by Constraint (2.4j), if product j is the first one produced on machine k in the first time-period, then product j is enforced to immediately follow the dummy product in an Eulerian cycle. Similarly, Constraint (2.4k) indicates that if product i is the last one produced on machine k , then i directly precedes the dummy product in the Eulerian cycle in the last time period. In a period other than the first period, if product j is the first one produced on machine k , which is implied by the carryover

setup performed from a product in an earlier period, then product j should immediately follow the dummy product in the Eulerian cycle, which is enforced by Constraint (2.4l). Similarly, in a period other than the last period, if product i is the last one produced on machine k , including the case when a carryover setup is performed for some other product in a later period, then product i should directly precede the dummy product in the Eulerian cycle.

7) Production Constraints:

$$\Theta_{ik}^t \leq Y_{ik}^t \quad \text{and} \quad \Theta_{ik}^t \leq 1, \quad \forall k \in M, \forall i \in J_k, \forall t \in T, \quad (2.4n)$$

$$\left\lceil \frac{Q_k}{L_{ik}^t p_{tik}^t} \right\rceil \cdot \Theta_{ik}^t \geq Y_{ik}^t, \quad \forall k \in M, \forall i \in J_k, \forall t \in T. \quad (2.4o)$$

The above constraint enforces an appropriate binary value for Θ_{ik}^t in relation to Y_{ik}^t for each product i produced on machine k in period t .

8) Subtour Elimination Constraints:

$$0 \leq (p_{ijk}^u)^t \leq X_{ijk}^t, \quad \forall k \in M, \forall u, j \in J_k, \forall i \in J_k^0, i \neq j, i \neq u, \forall t \in T, \quad (2.4p)$$

$$\sum_{j \in J_k} (p_{0jk}^u)^t = \Theta_{uk}^t, \quad \forall k \in M, \forall u \in J_k, \forall t \in T, \quad (2.4q)$$

$$\sum_{\substack{j \in J_k \\ j \neq i}} (p_{ijk}^u)^t - \sum_{\substack{j \in J_k^0 \\ j \neq i}} (p_{jik}^u)^t = 0, \quad \forall k \in M, \forall u \in J_k, \forall i \in J_k, i \neq u, \forall t \in T, \quad (2.4r)$$

$$\sum_{\substack{j \in J_k^0 \\ j \neq u}} (p_{juk}^u)^t = \Theta_{uk}^t, \quad \forall k \in M, \forall u \in J_k, \forall t \in T. \quad (2.4s)$$

The above subtour elimination constraints are a straightforward extension of the subtour elimination constraints for the HMATSP-P except that the right-hand sides in (2.4q) and (2.4s) are represented by a binary variable instead of by 1 and -1 , respectively. This is due to the existence of multiple time-periods and parallel machines, that is, if there is no production run of a product u on machine k in period t , then, the right-hand sides in (2.4q) and (2.4s) become 0, which obviates the need for ensuring the connectivity of this node within an Eulerian cycle.

9) Other Constraints:

$$X_{ijk}^t \geq 0 \text{ integer}, \quad \forall k \in M, \forall i, j \in J_k^0, i \neq j, \forall t \in T, \quad (2.4t)$$

$$Y_{ik}^t \geq 0 \text{ integer}, \quad \forall k \in M, \forall i \in J_k, \forall t \in T, \quad (2.4u)$$

$$\chi_{ijk}^t \in \{0, 1\}, \quad \forall k \in M, \forall i, j \in J_k, i \neq j, \forall t \in T, \quad (2.4v)$$

$$\Theta_{ik}^t \in \{0, 1\}, \quad \forall k \in M, \forall i \in J_k, \forall t \in T, \quad (2.4w)$$

$$0 \leq Z_{ik}^t \leq 1, \quad \forall k \in M, \forall i \in J_k, \forall t \in T, \quad (2.4x)$$

$$I_i^t, D_i^t \geq 0, \quad \forall i \in J, \forall t \in T, \quad (2.4y)$$

$$P_{ik}^t \geq 0, \quad \forall k \in M, \forall i \in J, \forall t \in T. \quad (2.4z)$$

Remark 2.2. As alluded to earlier, the above formulation is a modified version of the formulation proposed by Kang et al. (1999) with proposed subtour elimination constraints (2.4p) – (2.4s) and additional constraints (2.4j) – (2.4o). Constraints (2.4a) – (2.4i) appear identically in Kang et al. (1999). The formulation by Belvaux and Wolsey (2001) is a restricted version of the formulation by Kang et al. (1999) in that it does not allow a period having no production in it. Although Kang et al. (1999) do not explicitly include subtour elimination constraints in their formulation, the subtour elimination constraints proposed by Belvaux and Wolsey (2001) for the problem are as follows:

$$\sum_{i \in J'_k} \sum_{j \in J'_k} X_{ijk}^t \leq \sum_{i \in J'_k} Y_{ik}^t - \frac{1}{q} Y_{uk}^t, \quad \forall k \in M, \forall J'_k \subset J_k, \forall u \in J'_k, \forall t \in T \quad (2.5)$$

where q is a general upper bound value for the number of setups for each product on each machine in each time-period. Note that there are exponential number of such subtour elimination constraints.

Proposition 2.2. *CHESP is a valid formulation for the CHES problem.*

Proof. The sequencing of batches of various products processed on each machine is captured by Constraints (2.4f) – (2.4i). These constraints also accommodate the fact that production may not be scheduled in some periods (see machine 2 in Figure 2.2), and that there exist carryover setups. A product is produced in batches a number of times as determined by

(2.4c). The elimination of subtours ensures the connectivity of Eulerian cycles, which is afforded by Constraints (2.4q) – (2.4s). Other logical relationships among the problem variables are represented by (2.4j) – (2.4o) and (2.4t) – (2.4z), and flow balance, minimal and maximal lot size, capacity, and demand restriction are modeled by (2.4b), (2.4d), (2.4c) and (2.4e), respectively. \square

2.4.2 Computational Results

In this section, we report computational results to compare the formulations presented by Kang et al. (1999), Belvaux and Wolsey (2001), and our formulation, CHESP. The five CHES problem instances presented in the literature (see Kang et al. (1999) and Belvaux and Wolsey (2001)) are identified in Table 2.4 and are used for this experimentation.

Table 2.4: CHES problems

Instances	$ J $	$ T $	$ M $
CHES1	10	1	10
CHES2	21	1	8
CHES3	11	3	1
CHES4	11	1	2
CHES5	12	3	2

The results obtained are presented in Table 2.5 and give the optimal objective value derived by solving the model CHESP using CPLEX 9.0, along with the corresponding cpu time (in seconds) for each instance, where all the runs were implemented on an Intel Xeon 3.6 GHz computer. Table 2.4 also displays the percentage gap values attained along with the associated cpu times using the formulations due to Belvaux and Wolsey (2001) (BW-GAP) and Kang et al. (1999) (KMT-GAP). The cpu efforts for the BW and KMT formulations are noted from their papers (see Belvaux and Wolsey (2001) and Kang et al. (1999)), where the former runs were implemented on a Pentium 200 MHz under Windows NT system when terminated using time limit of 3600 seconds, while the latter runs were performed with a

limit of 60 iterations of column generation on a Pentium 75 MHz under Windows 95 system. The computational time for CHES1 was not mentioned in Kang et al. (1999), while the results for this instance were not reported in Belvaux and Wolsey (2001).

It is clear from the results presented in Table 2.5 that CHESP is an improved and effective formulation, which yields a 0% GAP for all the problem instances with a very reasonable effort.

Table 2.5: Results for the instances of the CHES problem

Instances	Opt. Value	cpu (sec)	BW-GAP	cpu (sec)	KMT-GAP	cpu (sec)
CHES1	121.84	0.11	-	-	0.00%	-
CHES2	-2889.79	7.20	0.46%	3600	1.23%	587.5
CHES3	-1303791.97	0.95	0.00%	3600	0.33%	7420.8
CHES4	-647403.52	0.16	0.00%	5	0.01%	389.4
CHES5	-7159.9	83.11	4.86%	3600	4.19%	1815.6

2.5 Concluding Remarks

In this chapter, we have studied the high multiplicity asymmetric traveling salesman problem (HMATSP). A compact (polynomial-length) formulation for this problem was developed and validated, and was computationally demonstrated to significantly outperform an alternative formulation for the problem presented in the literature. An application of the HMATSP structure to model a lot-sizing problem on parallel machines in the presence of sequence-dependent setup costs (known as a CHES problem) has also been presented. A compact formulation for this problem was developed that makes use of the flow-based sub-tour elimination constraints developed for the HMATSP. Our computational results with this formulation reveal its significant advantage over the alternative formulations proposed by Kang et al. (1999) and Belvaux and Wolsey (2001).

Chapter 3

Primary Pharmaceutical Manufacturing Scheduling Problem (PPMSP)

A multi-product, lot-sizing and sequencing problem in the face of machine-fixed batch sizes and sequence-dependent setups arises in many industrial environments that use fixed-capacity processors (for instance, containers) for production. A prime example of such an instance is a primary pharmaceutical manufacturing facility, where active pharmaceutical ingredients (APIs) are produced in containers of a fixed size. We address this problem in this chapter. We present a novel model and a column generation-based optimization approach for this class of lot-sizing and sequencing problems. We have applied our methodology to a real-life problem, for which the data are collected from a major pharmaceutical manufacturing company, to demonstrate the applicability of the proposed methodology in practice.

3.1 Background and Motivation

Pharmaceutical manufacturing consists of two functional steps called primary and secondary manufacturing. The primary manufacturing component is responsible for the production of

active pharmaceutical ingredients (APIs) and normally involves several chemical synthesis and separation stages to compose the complex molecules involved. The secondary manufacturing phase, on the other hand, involves the addition of “excipient” inert materials to the APIs obtained from the primary manufacturing facilities, and it further processes and packages the APIs into tablets. The secondary manufacturing is a continuous process that is pulled by customer orders, while the primary manufacturing is characterized by a highly discrete, batch production process. The overall effectiveness of pharmaceutical manufacturing depends upon the responsiveness of the primary function to the requirements generated by the secondary manufacturing. However, this responsiveness of the primary manufacturing facilities is known to be rather poor, and it often results in delays and high operating costs (see Shah (2004)). The operational control of the primary pharmaceutical manufacturing is complex and needs a systematic study to design effective strategies for its execution.

Supply chain optimization and integration have been discussed rather extensively in the literature, and yet, only a very small fraction of this work directly addresses the issues faced in the pharmaceutical sector. The operational stage of the pharmaceutical supply chain is one of its crucial components. At this stage, which involves both primary and secondary manufacturing, it is not unusual to have cycle times of 300 days. This severely impacts responsiveness to changing market trends. More often than not, erratic dynamics of the operational stage (and the supply chain) are introduced by internal business processes than by external demand, and can be eliminated by effectively re-designing the internal processes (see Shah (2004)). Besides, time-to-market is the single most important driver now-a-days in the pharmaceutical industry as significant revenues are reaped in the early life of a drug. Because of an intense competition among the companies, the competition-free period has also decreased (from 5 to 1-2 years). All of these facts point to a greater need of improving the performance of the operational stage of the pharmaceutical supply chain. Our proposed problem directly addresses this issue.

In particular, our aim is to develop and validate an approach for the optimal assignment and sequencing of pharmaceutical products to various processing bays of a primary manufac-

turing facility for the objective of meeting customer requirements with minimal production costs, given a set of equipment with specified capacities as well as a product-mix and the demand rate for each product. We designate this problem, which is germane to primary pharmaceutical manufacturing, as the *Primary Pharmaceutical Manufacturing Scheduling Problem (PPMSP)*.

3.2 Literature Review

The areas that are related to the PPMSP include: scheduling in the presence of sequence-dependent setups (SDS scheduling), scheduling of batch processes and the multi-item capacitated lot-sizing. Even though there exists extensive literature pertaining to each of these areas, yet, there has not been much reported about a problem like PPMSP that lies at the interface of these three areas. A brief review of the reported work in these and related areas is presented next.

The multi-item capacitated lot-sizing problem (CLSP): The CLSP generally involves discrete periods, finite horizons, resources with finite capacities, and multiple products with known demands. We focus our review on the mathematical programming-based methodologies proposed for this problem. Mixed integer programming formulations for the CLSP have been presented by Barany et al. (1984), Leung et al. (1989), Eppen and Martin (1987) and Belvaux and Wolsey (2001). Barany et al. (1984) and Leung et al. (1989) have reformulated multi-item capacitated lot-sizing problem using a class of valid inequalities, which are facets for the single-item uncapacitated problem. Eppen and Martin (1987) have used a variable re-definition technique to provide a tighter linear relaxation of the CLSP. Belvaux and Wolsey (2001) provide a framework for modeling various aspects of the lot-sizing problem encountered in practice. Heuristic methods based on Lagrangian relaxation have been proposed by Billington et al. (1983), Thizy and van Wassenhove (1985), Trigeiro (1987), and Diaby et al. (1992) where the original CLSP is decomposed into N single item uncapaci-

tated lot-sizing subproblems by relaxing capacity constraints. Heuristics methods based on branch-and-bound have been proposed by Diaby et al. (1992), Hindi (1995) and Amentano et al. (1999). Methods based on set partitioning and column generation have been proposed by Manne (1960), Cattrysse et al. (1990), and Chen and Thizy (1990).

Scheduling with sequence-dependent setups (SDS scheduling): The SDS scheduling problems have been shown to be strongly NP-hard since the simplest form of the problem, the single machine SDS scheduling problem for the makespan objective function, is equivalent to the Traveling Salesman Problem (TSP) (see Pinedo (2002)). We focus our review on the exact solution methodology for scheduling in the presence of sequence-dependent setups for the single machine and parallel machine configurations. For the single machine scheduling problem, branch-and-bound algorithms have been proposed by Barnes and Vanston (1981), Rabadi et al. (2004), and Asano and Ohta (1996, 1999). Their branch-and-bound algorithms vary due to the different problem parameters and objective functions used. SDS scheduling in the parallel machine configuration can be found in Balakrishnan et al. (1998), Bitran and Gilbert (1990), Yalaoui and Chu (2003), and Sarin et al. (2008), Balakrishnan et al. (1998) have proposed a Benders' decomposition approach to solve the problem of minimizing weighted earliness and tardiness problem, where the master problem yields assignments to machines and the sequence at each machine, and the subproblem prescribes the completion time of each job. Bitran and Gilbert (1990) have proposed a hybrid method to solve sequencing problem among product families using Branch-and-Bound and address problem within family using heuristics. Yalaoui and Chu (2003) have presented a problem with job splitting and proposed a decomposition method in which they first solve TSPs using Branch-and-Bound and then improve the solution by splitting jobs. Sarin et al. (2008) have considered a hot strip rolling scheduling problem with precedence relationships. They provided a tight ATSP formulation by applying the Reformulation-Linearization Technique (RLT) of Sherali and Adams (1990, 1994). A comprehensive review on scheduling with sequence-dependent setups can be found in a paper by Zhu and Wilhelm (2006).

Simultaneous lot-sizing and scheduling problem with sequence-dependent setup:

The earliest mention of such a problem in the literature is due to Baker and Muckstadt (1989) who presented the CHES problems, a collection of practical problems that have been gathered by Chesapeake Decision Sciences. The CHES problems consist of parallel production lines and sequence-dependent setup cost (but no setup time). Kang et al. (1999) have developed a column generation and branch-and-bound scheme based on a sequence splitting model for the CHES problems and presented an incomplete MIP formulation in the absence of the subtour elimination constraints. Belvaux and Wolsey (2001) have provided an MIP formulation for the CHES problems with an exponential number of subtour elimination constraints. Their solution approach proceeds without all SECs but appends cuts when a violation is found. Meyr (2002) has proposed a heuristic which combines meta-heuristics (threshold accepting or simulated annealing) with dual-optimization for general lot-sizing and scheduling problem for parallel machines. The proposed methodology is tested on various industrial-size problems including the CHES problems. Dastidar and Nagi (2005) have presented an MIP for a lot-sizing and scheduling problems involving parallel machines with sequence-dependent and carryover setups, and they have proposed a heuristic for its solution. An application of the single machine problem involving sequence-dependent setup cost (and time) and carryover setup has been presented by Hasse and Kimms (2000) and Gupta and Magnusson (2005). Hasse and Kimms (2000) formulated an MIP that considers only ‘efficient’ sequences. Gupta and Magnusson (2005) formulated an MIP and proposed a heuristic procedure for its solution. Small bucket lot-sizing models with SDS have been proposed by Fleischmann (1994) that rely on the TSP with time windows, and they have employed lagrangean relaxation in combination with a heuristic to determine lower bounds.

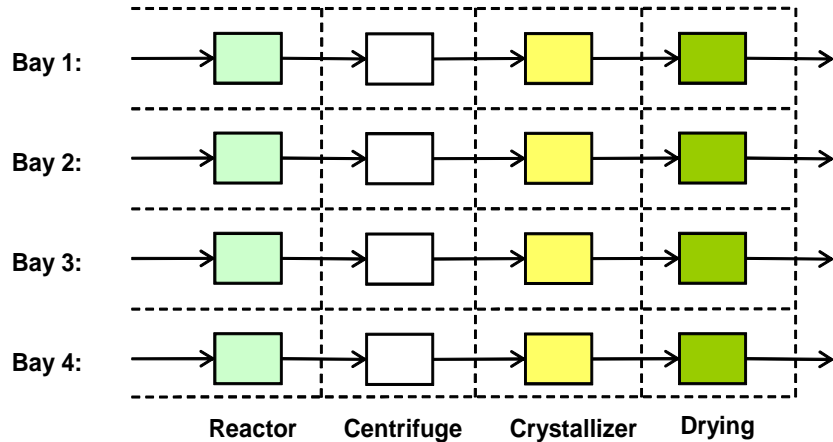


Figure 3.1: A primary pharmaceutical manufacturing shop floor configuration

3.3 Problem Description

Typically, a primary manufacturing facility consists of several processing bays in parallel. There are four serial processing stages in each bay, namely, reactor, centrifuge, crystallizer, and dryer. An instance of the shop floor configuration for primary manufacturing is depicted in Figure 3.1. The processors at a stage are containers that require different processing and setup times. There is no external storage for holding the products between any two consecutive steps of a bay. Therefore, these equipment not only function as processing equipment, but also, are used to retain a product before transferring it to the next step. The production of both a final product and its intermediate products are performed in batches in a bay. The intermediate products can be stored in a separate storage area after having finished their processing at the last step (i.e., dryer).

The production of a particular product type (including its intermediaries and final products) at a stage, in between two changeovers, is called a campaign. While the total amount of a product to be produced in the scheduling horizon is known in advance, we need to split this total amount into campaigns of requisite lengths (lot-sizing). A changeover from one product type to another is very time consuming. Therefore, the number of batches (length of a campaign) of a product that is scheduled for production before the equipment switches

to another product type must be carefully determined. If all the batches of a product type are run continually, then benefits will accrue because of savings in the changeover cost, and also, because of minimal cross-contamination. However, by doing so, there is a possibility of accumulating excess inventory for some products, while delaying others, which would lead to a low level of responsiveness. Therefore, it is essential to determine optimal production batches of different products (or types), the bay in which to process a product, and the sequence in which to process the products assigned to a bay so as to minimize production-related costs and meet customer requirements while staying within the production capacity of each bay.

We have a set of products N (including both intermediaries and final products), and each product $i \in N$ belongs to a product family f , for $f \in F$, where F is a set of product families (types). Products are to be scheduled in bays in a time horizon of multiple periods, denoted by set T . In each period, there is a demand d_i for product i . A certain amount of an intermediate product is required to complete its corresponding subsequent product. There are M processing bays (in parallel) available to process the batches of products in each period. The capacity of bay k to process product i is q_i^k . Because of the limited capacity of each bay, we need to determine the number of batches of each product (both intermediate and final) for processing in a bay. A significant sequence-dependent setup is incurred from the processing of one batch to another in a bay if the two batches belong to different products. Consequently, a cost-effective sequence in which to process the batches of the products in a bay must also be determined. In addition, there are other requirements that are peculiar to the PPMSP, as follows.

- 1) Since the bays produce both intermediate and final products, the necessary material requirement and precedence relationships among the intermediate and final products must be maintained.
- 2) Only certain bays are qualified to produce a product and its intermediaries, that is, not all the products can be produced in all the bays.

- 3) Some products are not allowed to immediately follow some other products in a bay.
- 4) The total production time in a bay cannot exceed the total time available in each time-period.
- 5) The last setup state of a bay in a period can be carried over to the next period only if the same product is produced at the start of the next period.

The PPMSP can now be formally stated as follows.

Given a number of bays with limited production capacities and a set of customer demands (of finished products), select a bay in which to process each product and its intermediaries, determine the campaign size for each product and the number of batches in which to process it, and sequence the products assigned to a bay subject to the requirements listed above in (1), (2), (3), (4), and (5) so as to minimize the sequence-dependent setup plus the inventory and backorder costs.

There are four key issues that distinguish the PPMSP from the related problems discussed in the literature. They are as follows:

(i) **Processing units (bays) have fixed processing capacity, and hence, the processing batch sizes of the products assigned to a bay are fixed.** This feature is practiced in primary manufacturing facilities mainly due to two reasons. First, it maximizes utilization of each processing unit. Second, the processing recipe for each run is also fixed, and hence, it affords an ease for executing quality control and management functions. From an operational control perspective, the limited processing capacity forces a production lot to be produced in several (smaller) batches, thereby requiring each product to be processed by a bay several times.

(ii) **Consideration of the intermediaries of a product, and also, the given immediate precedence relations among them.** For illustration, consider two product families 1 and 2, where final product $F1$ and its intermediaries $A1$, $B1$ and $F1$ belong to family 1, and final product $F2$ and its intermediaries $A2$, $B2$ belong to family 2. These are shown in Figure 3.2. In Figure 3.2(a), the processing of a unit of final product $F1$ requires two

units of its intermediate product $B1$, and each unit of $B1$ requires a unit of product $A1$, while in Figure 3.2(b), the processing of a unit of final product $F2$ requires one unit of its intermediate product $B2$, and each unit of $B2$ requires one unit of product $A2$.

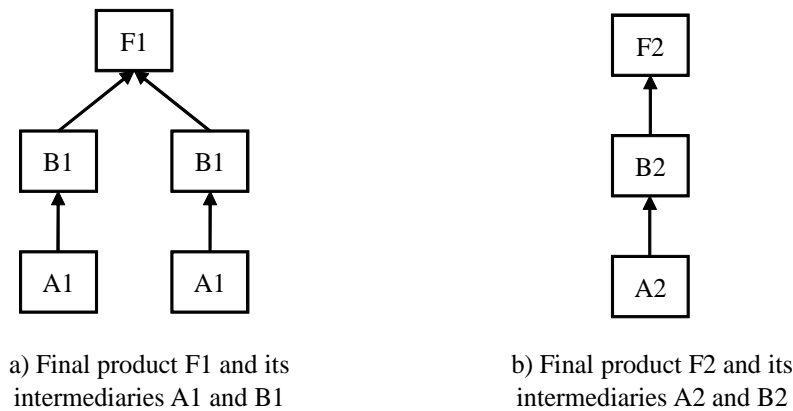


Figure 3.2: The relationship of final products to their intermediaries

(iii) **Products belonging to the same family, when scheduled for production in a bay during a time-period (i.e., a month), should be produced together, and any given bay can only process products of a single family during a time-period.** This is illustrated in Figure 3.3. This feature arises due to two reasons: (a) it minimizes cross contamination between different product types; (b) only specific bays are qualified to produce particular products due to stringent FDA requirements.

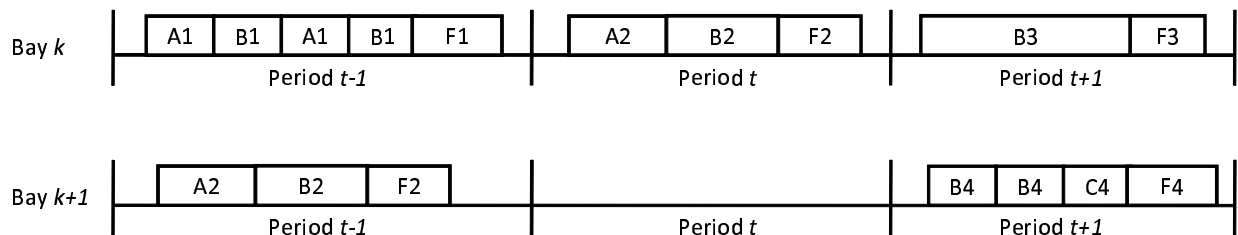


Figure 3.3: An example of a schedule with two bays, three periods and four product families

(iv) **Presence of sequence-dependent carryover setup from one period to the next period.** A significant amount (i.e., 50 hours - 200 hours) of sequence-dependent setup is incurred in between the processing of two different product families for requisite cleanup and preparation. Because of an excessive length of this setup, it may be started in one period, and then, continued in the next period. We designate this as carryover setup, and it is depicted in Figure 3.4.



Figure 3.4: Sequence-dependent setup in bay k from time-period t to $t + 1$

3.4 Basic Mixed Integer Programming Formulation for the PPMSP

In this section, we first introduce the notation that we use. This is followed by the formulation of a model for the PPMSP. Subsequently, we also present some valid inequalities, which further tighten our formulation.

Notation

Sets:

- F - Set of all product types (families) including the dummy family, denoted by $\{0\}$.
- N - Set of all products.
- N_f - Set of all products belonging to product family f , $N_f \subset N$.
- M - Set of all bays, $\{1, \dots, m\}$.
- M_i - Set of bays that are qualified to process product i .
- N^k - Set of all final products that can be processed by bay k .
- T - Set of time-periods in the planning horizon, $\{1, \dots, \tau\}$

Parameters:

- d_{fi}^t - Demand of product i of family f in period t .
- q_{fi}^k - Batch size of bay k when processing product i of family f .
- h_{fi} - Inventory cost for a unit of product i of family f .
- b_{fi} - Backorder cost for a unit of product i of family f .
- st_{fg}^k - Setup time from product family f to g in bay k .
- pt_{fi}^k - Processing time per batch of product i of family f in bay k .
- Q^{kt} - Available time capacity of bay k in period t .
- ρ_{fij} - BOM ratio defined as units of product i required to produce a unit of product j of the same family f in bay k , for $i, j \in N_f$.
- I_{fi}^0 - Initial inventory at the beginning of the first period.
- I_{fi}^τ - End inventory at the end of planning horizon.

Decision Variables:

- I_{fi}^t - Inventory of product i of product family f held in period t .
- B_{fi}^t - Amount of product i of product family f backlogged in period t .
- W_{fi}^{kt} - Number of batches of product i of family f produced in bay k during period t .
- S_{fi}^t - Units of product i of family f consumed as an intermediary during period t .
- $Z_f^{kt} = 1$, if products of family f are processed in bay k during period t , and
0, otherwise.
- $X_{fg}^{kt} = 1$, if there is a setup for producing family g in period $t + 1$ following
product f in period t in bay k , and
- α_{fg}^{kt} - proportion of a setup incurred at the end of period t from family f to family g .
- β_{fg}^{kt} - proportion of a setup incurred at the beginning of period t from family f to family g

Note that dummy family 0 is used to represent the cleanup state of a bay in a period. If a bay is idle, or equivalently, processing dummy family 0 in some period, then we assume that the bay is being cleaned up. For instance, if a bay is idle in current period, then, a carryover setup must be incurred in the form of cleanup from the previous period to the current period. Also, we assume that all the machines have been cleaned up at the beginning of the first

time-period, and are required to be cleaned up at the end of the last period.

In the next section, we provide a mixed integer programming formulation for the PPMSP. We denote the MIP formulation for the PPMSP by **PPMSP1**.

PPMSP1:

Objective function:

$$\text{Minimize } \sum_{t \in T} \sum_{f \in F} \sum_{i \in N} h_{fi} I_{fi}^t + \sum_{t \in T} \sum_{f \in F} \sum_{i \in N} b_{fi} B_{fi}^t + f \cdot \sum_{t \in T \cup \{0\}} \sum_{k \in M} \sum_{f \in F} \sum_{g \in F} s_{tfg}^k X_{fg}^{kt} \quad (3.1a)$$

The objective function contains cost components due to inventory, backorders and carryover setups from one period to the next.

Constraints:

1) Flow balance constraint

$$I_{fi}^{t-1} + B_{fi}^t + \sum_{k \in M_i} q_{fi}^k W_{fi}^{kt} = I_{fi}^t + B_{fi}^{t-1} + d_{fi}^t + S_{fi}^t, \quad \forall t \in T, \forall f \in F, \forall i \in N_f \quad (3.1b)$$

This constraint set represents typical flow balance constraints, which enforce the input (given by the beginning inventory and amount produced) to equal the output (given by the demand, inventory, and the amount consumed to produce a next-stage product) for each time period. Note that the backorders incurred in period $(t - 1)$ and t are also properly accounted for in (3.1b).

2) Material requirement constraint

$$S_{fi}^t = \sum_{k \in M_j} \rho_{fij}^k q_{fj}^k W_{fj}^{kt}, \quad \forall t \in T, \forall f \in F, \forall i, j \in N_f, \rho_{fij} > 0 \quad (3.1c)$$

This constraint ensures that a sufficient amount of an intermediate product, i , is consumed to produce its corresponding next-stage product, j for $i, j \in N_f$.

3) Capacity constraint

$$\sum_{f \in F} \sum_{i \in N_f \cap N^k} p_{fi}^{kt} W_{fi}^{kt} + \sum_{f \in F} \sum_{g \in F} s_{fg}^k \left(\alpha_{fg}^{kt} + \beta_{fg}^{kt} \right) \leq cap^{kt}, \quad \forall t \in T, \forall k \in M \quad (3.1d)$$

This constraint set maintains the total production and setup time consumed in each bay within its available capacity during each time period. The relevant values of α_{fg}^{kt} and β_{fg}^{kt} , for a given t , are determined by Constraints (3.1i) to (3.1k) below.

4) Assignment constraint

$$\sum_{f \in F} Z_f^{kt} = 1, \quad \forall t \in T, \forall k \in M \quad (3.1e)$$

This constraint set ensures that products of at most one product family can be processed in a bay during any time-period. Note that when dummy family 0 is processed, it is assumed that the bay is idle and is in the cleanup state.

5) Sequencing constraint

$$Z_f^{k1} \leq X_{0f}^{k0}, \quad \forall k \in M, \forall f \in F \quad (3.1f)$$

$$Z_f^{k\tau} \leq X_{f0}^{k\tau}, \quad \forall k \in M, \forall f \in F \quad (3.1g)$$

$$Z_f^{kt} + Z_g^{k,t+1} - 1 \leq X_{fg}^{kt}, \quad \forall t \in T - \{\tau\}, \forall k \in M, \forall f, g \in F \quad (3.1h)$$

These constraints assert that a carryover setup is incurred only when two product families have been assigned to the same bay during two consecutive periods. Note that, in case a dummy family is processed in a bay, the carryover setup is incurred in the form of cleanup. Also, the initial and final states of a bay are assumed to be cleanup. Constraint set (3.1f) captures the fact that there exists a setup from dummy family 0 to a product family at the beginning of the first period. Similarly, constraint set (3.1g) ensures that a setup from a product family to the dummy family 0 is incurred at the end of the last period τ .

6) Carryover setup constraint

$$X_{fg}^{k0} = \beta_{fg}^{k1}, \quad \forall k \in M, \forall f, g \in F \quad (3.1i)$$

$$X_{fg}^{k\tau} = \alpha_{fg}^{k\tau}, \quad \forall k \in M, \forall f, g \in F \quad (3.1j)$$

$$X_{fg}^{kt} = \alpha_{fg}^{kt} + \beta_{fg}^{k,t+1}, \quad \forall t \in T - \{\tau\}, \forall k \in M, \forall f, g \in F \quad (3.1k)$$

Constraint set (3.1i) ensures that the first setup occurs in the first time-period. Similarly, constraint set (3.1j) ensures that the last setup occurs in the last time-period. Constraint set (3.1k) captures the fact that a carryover setup can be split into two portions with one portion performed in the current period and the other in the subsequent period.

7) Production batch Constraints:

$$Z_f^{kt} \leq \sum_{i \in N_f \cap N^k} W_{fi}^{kt} \leq \left\lfloor \frac{cap^{kt}}{\min_{i \in N_f \cap N^k} pt_{fi}^k} \right\rfloor Z_f^{kt}, \quad \forall k \in M, \forall t \in T, \forall f \in F \quad (3.1l)$$

The above constraint set captures the relationships among W and Z variables.

8) Others:

$$Z_f^{kt} \in \{0, 1\}, \quad \forall k \in M, \forall t \in T, \forall f \in F \quad (3.1m)$$

$$X_{fg}^{kt} \in \{0, 1\}, \quad \forall k \in M, \forall t \in T, \forall f, g \in F \quad (3.1n)$$

$$\alpha_{fg}^{kt}, \beta_{fg}^{kt} \geq 0, \quad \forall k \in M, \forall t \in T, \forall f, g \in F \quad (3.1o)$$

$$W_{fi}^{kt} \geq 0 \text{ integer}, \quad \forall k \in M, \forall t \in T, \forall f \in F, \forall i \in N_f \cap N^k \quad (3.1p)$$

$$I_{fi}^t \geq 0, \quad \forall t \in T, \forall f \in F, \forall i \in N_f, \quad (3.1q)$$

These constraints represent nonnegative and integer constraints.

Observe that the following valid equalities can also be used in the formulation **PPMSP1**.

$$\sum_{g \in F} X_{gf}^{kt} = Z_f^{k,t+1}, \quad \forall t \in T + \{0\} - \{\tau\}, \forall k \in M, \forall f \in F \quad (3.2a)$$

$$\sum_{g \in F} X_{fg}^{kt} = Z_f^{kt}, \quad \forall t \in T - \{\tau\}, \forall k \in M, \forall f \in F \quad (3.2b)$$

$$\sum_{g \in F} X_{g0}^{k\tau} = 1, \quad \forall k \in M \quad (3.2c)$$

$$\sum_{g \in F} X_{0g}^{k0} = 1, \quad \forall k \in M \quad (3.2d)$$

Constraint set (3.2a) captures the fact that, for any given bay k , if a family f is produced in period $t + 1$, then a setup must be incurred at the end of the previous period, from the same or different product family. Constraint set (3.2b) is similar to (3.2a), and in particular, accounts for the fact that, for any given bay k , if a family f is produced in period t , a setup must exist from product family f to another product family (may be the same family) at the end of period t . We shall refer to the model with the foregoing valid equalities as **PPMSP2**.

3.5 Column Generation Heuristic Approach

In this section, we propose a column generation scheme for the solution of the PPMSP that comprises two stages. Stage I (or master problem) selects a processing pattern in accordance with the given constraints, while the stage II problem (or subproblem) utilizes the dual variables obtained by solving the master problem, and generates a column (pattern) for insertion in the master problem. The master problem, and the subproblem are, thus, solved iteratively until a specified termination criterion is met.

3.5.1 Pattern Definition

Define P_s , for $s \in S$, where S is a set of all generated patterns, as follows:

$$P_s = \left\{ Z_{fs}^{kt}, \quad \forall t \in T, \forall k \in M, \forall f \in F \right\} \quad (3.3)$$

Note that each pattern, s , is a three dimensional matrix, where its element Z_{fs}^{kt} indicates whether the batches of product family f are processed in bay k during time-period t . Also, for columns, let c_s be the total inventory cost required if the s^{th} pattern is selected, and p_s^{kt} be the total processing time required in bay k during time-period t . In the next two sections, we present column generation-based methodologies for PPMSP-CG1 and PPMSP-CG2.

3.5.2 Column Generation Method for PPMSP1 (PPMSP-CG1)

3.5.2.1 Master Problem: MP-SEQUENCE

Let

$$\theta_s = \begin{cases} 1, & \text{if pattern } s \text{ is selected, } \forall s \in S \\ 0, & \text{otherwise.} \end{cases}$$

MP-SEQUENCE:

Minimize

$$\sum_{s \in S} c_s \theta_s + \sum_{t \in T \cup \{0\}} \sum_{k \in M} \sum_{f \in F} \sum_{g \in F} st_{fg}^k X_{fg}^{kt} \quad (3.4a)$$

subject to:

$$\sum_{s \in S} \theta_s = 1, \quad (3.4b)$$

$$\sum_{s \in S} p_s^{kt} \theta_s + \sum_{f \in F} \sum_{g \in F} st_{fg}^k \left(\alpha_{fg}^{kt} + \beta_{fg}^{kt} \right) \leq Q^{kt}, \quad \forall t \in T, \forall k \in M, \quad (3.4c)$$

$$\sum_{s \in S} Z_{fs}^{k1} \theta_s \leq X_{0f}^{k0}, \quad \forall k \in M, \forall f \in F \quad (3.4d)$$

$$\sum_{s \in S} Z_{fs}^{k\tau} \theta_s \leq X_{f0}^{k\tau}, \quad \forall k \in M, \forall f \in F \quad (3.4e)$$

$$\sum_{s \in S} \left(Z_{fs}^{kt} + Z_{gs}^{k,t+1} \right) \theta_s - 1 \leq X_{fg}^{kt}, \quad \forall t \in T - \{\tau\}, \forall k \in M, \forall f, g \in F \quad (3.4f)$$

$$X_{fg}^{k0} = \beta_{fg}^{k1}, \quad \forall k \in M, \forall f, g \in F \quad (3.4g)$$

$$X_{fg}^{k\tau} = \alpha_{fg}^{k\tau}, \quad \forall k \in M, \forall f, g \in F \quad (3.4h)$$

$$X_{fg}^{kt} = \alpha_{fg}^{kt} + \beta_{fg}^{k,t+1}, \quad \forall t \in T - \{\tau\}, \forall k \in M, \forall f, g \in F \quad (3.4i)$$

$$\theta_s \in \{0, 1\}, \quad \forall s \in S \quad (3.4j)$$

$$X_{fg}^{kt} \in \{0, 1\}, \quad \forall k \in M, \forall t \in T, \forall f, g \in F \quad (3.4k)$$

$$\alpha_{fg}^{kt}, \beta_{fg}^{kt} \geq 0, \quad \forall k \in M, \forall t \in T, \forall f, g \in F \quad (3.4l)$$

We propose to solve the LP-relaxation of MP-SEQUENCE using column generation with an artificial basis at start. Once a feasible solution is obtained, we price the θ -variables using

the resulting dual variables to possibly introduce one or more new columns into the master problem.

3.5.2.2 Subproblem: SP-LSP

Let π , λ_{kt} , ϕ_f^{kt} , φ_f^{kt} and ψ_{fg}^{kt} be the dual variables associated with Constraints (3.4b), (3.4c), (3.4d), (3.4e) and (3.4f). By pricing a θ variable, we have the following reduced cost.

$$c - \sum_{t \in T} \sum_{k \in M} \lambda^{kt} p^{kt} - \sum_{k \in M} \sum_{f \in F} \phi_f^{kt} Z_f^{k1} - \sum_{k \in M} \sum_{f \in F} \varphi_f^{kt} Z_f^{k\tau} - \sum_{\substack{t \in T \\ t < \tau}} \sum_{k \in M} \sum_{f \in F} \sum_{g \in F} \psi_{fg}^{kt} \left(Z_f^{kt} + Z_g^{k,t+1} \right) - \pi, \quad (3.5)$$

where c and p^{kt} are given by as follows:

$$c = \sum_{t \in T} \sum_{f \in F} \sum_{i \in N} h_{fi} I_{fi}^t + \sum_{t \in T} \sum_{f \in F} \sum_{i \in N} b_{fi} B_{fi}^t, \quad (3.6)$$

$$p^{kt} = \sum_{f \in F} \sum_{i \in N_f \cap N^k} p_{fi}^{kt} W_{fi}^{kt}. \quad (3.7)$$

Then, the SP-LSP can be modeled as the following multiple-item uncapacitated lot-sizing problem with material requirement.

SP-LSP:

Minimize

$$\begin{aligned} & \sum_{t \in T} \sum_{f \in F} \sum_{i \in N_f} h_{fi} I_{fi}^t + \sum_{t \in T} \sum_{f \in F} \sum_{i \in N} b_{fi} B_{fi}^t - \sum_{t \in T} \sum_{k \in M} \lambda^{kt} \sum_{f \in F} \sum_{i \in N_f \cap N^k} p_{fi}^{kt} W_{fi}^{kt} \\ & - \sum_{k \in M} \sum_{f \in F} \phi_f^{kt} Z_f^{k1} - \sum_{k \in M} \sum_{f \in F} \varphi_f^{kt} Z_f^{k\tau} - \sum_{\substack{t \in T \\ t < \tau}} \sum_{k \in M} \sum_{f \in F} \sum_{g \in F} \psi_{fg}^{kt} \left(Z_f^{kt} + Z_g^{k,t+1} \right) - \pi \end{aligned} \quad (3.8a)$$

subject to:

$$I_{fi}^{t-1} + \sum_{k \in M_i} q_{fi}^k W_{fi}^{kt} = I_{fi}^t + d_{fi}^t + S_{fi}^t, \quad \forall t \in T, \forall f \in F, \forall i \in N_f \quad (3.8b)$$

$$S_{fi}^t = \sum_{k \in M_j} \rho_{fij}^k q_{fj}^k W_{fj}^{kt}, \quad \forall t \in T, \forall f \in F, \forall i, j \in N_f, \rho_{fij} > 0 \quad (3.8c)$$

$$\sum_{f \in F} Z_f^{kt} = 1, \quad \forall t \in T, \forall k \in M \quad (3.8d)$$

$$Z_f^{kt} \leq \sum_{i \in N_f \cap N^k} W_{fi}^{kt} \leq \left\lfloor \frac{Q^{kt}}{\min_{i \in N_f \cap N^k} p_{fi}^k} \right\rfloor Z_f^{kt}, \quad \forall k \in M, \forall t \in T, \forall f \in F \quad (3.8e)$$

$$Z_f^{kt} \in \{0, 1\}, \quad \forall k \in M, \forall t \in T, \forall f \in F \quad (3.8f)$$

$$W_{fi}^{kt} \geq 0 \text{ integer}, \quad \forall k \in M, \forall t \in T, \forall f \in F, \forall i \in N_f \cap N^k \quad (3.8g)$$

$$I_{fi}^t \geq 0, \quad \forall t \in T, \forall f \in F, \forall i \in N_f, \quad (3.8h)$$

If the reduced cost obtained is negative, then we add the above column P_s along with c_s and p_s^{kt} , for $k \in M, t \in T$ to the master problem, **MP-SEQUENCE**, which is re-solved. This is continued until the minimum reduced cost obtained is non-negative (or exceeds some small negative tolerance $-\varepsilon$).

3.5.3 Column Generation Method for PPMSP2 (PPMSP-CG2)

In this section, we propose another column generation model, where the constraints (3.4d) - (3.4f) in the above master problem **MP-SEQUENCE** are replaced by the (3.2a) - (3.2d). This model is denoted by **PPMSP-CG2**.

3.5.3.1 Master Problem: MP-SEQUENCE

MP-SEQUENCE:

Minimize

$$\sum_{s \in S} c_s \theta_s + \sum_{t \in T \cup \{0\}} \sum_{k \in M} \sum_{f \in F} \sum_{g \in F} st_{fg}^k X_{fg}^{kt} \quad (3.9a)$$

subject to:

$$(3.4b), (3.4c), (3.4g), (3.4h), (3.4i), (3.4j), (3.4k), (3.4l),$$

$$\sum_{g \in F} X_{gf}^{kt} = \sum_{s \in S} Z_{fs}^{k,t+1} \theta_s, \quad \forall t \in T + \{0\} - \{\tau\}, \forall k \in M, \forall f \in F \quad (3.9b)$$

$$\sum_{g \in F} X_{fg}^{kt} = \sum_{s \in S} Z_{fs}^{kt} \theta_s, \quad \forall t \in T, \forall k \in M, \forall f \in F \quad (3.9c)$$

$$\sum_{g \in F} X_{g0}^{k\tau} = 1, \quad \forall k \in M \quad (3.9d)$$

$$\sum_{g \in F} X_{0g}^{k0} = 1, \quad \forall k \in M \quad (3.9e)$$

Let ϕ_f^k and φ_f^k be the dual variables associated with constraints (3.9b) and (3.9c). We have the following subproblem.

3.5.3.2 Subproblem: SP-LSP

SP-LSP:

Minimize

$$\begin{aligned} & \sum_{t \in T} \sum_{f \in F} \sum_{i \in N_f} h_{fi} I_{fi}^t - \sum_{t \in T} \sum_{k \in M} \lambda^{kt} \sum_{f \in F} \sum_{i \in N_f \cap N^k} p_{fi}^{kt} W_{fi}^{kt} \\ & - \sum_{\substack{t \in T \\ t < \tau}} \sum_{k \in M} \sum_{f \in F} \phi_f^{kt} Z_f^{k,t+1} - \sum_{t \in T} \sum_{k \in M} \sum_{f \in F} \varphi_f^{kt} Z_f^{k\tau} - \pi \end{aligned} \quad (3.10)$$

subject to: $\left\{ (3.8b), (3.8c), (3.8d), (3.8e), (3.8f), (3.8g), (3.8h) \right\}$

3.5.4 Outline of the Column Generation Approach

Initialization. Generate initial columns comprising plans for products of all product families over the given planning horizon. Let S be the set of initial columns.

Step 1. Solve LP relaxation of the master problem (MP-SEQUENCE), and obtain the values of dual variables.

Step 2. Solve subproblem SP-LSP, and obtain optimal reduced cost. If the reduced cost is non-negative, go to Step 3; else, append the column generated from SP-LSP to the master problem, and go to Step 1.

Step 3. Solve the MP-SEQUENCE as an integer program to optimality, and terminate.

3.6 Computational Experimentation

In this section, we conduct numerical experimentation to study the computational effectiveness of the proposed formulations and solutions methodologies. The data used for this experimentation is shown in Figure 3.2, which depicts the values that are used for the number of periods (horizon length), number of bays, number of product families and total number of products. Four data sets are created as shown. The setup cost per setup hour is fixed at 16 for all data sets. We also fix the values of inventory and backorder costs for each set once they are generated using $U(0.01, 0.1)$ and $U(1, 10)$, respectively. The values of batch processing time and batch size are fixed once generated using $U(100, 200)$ and $U(200, 500)$, respectively.

Table 3.2: Sets of problem instances

Problem set	Instance	No. of periods	No. of bays	No. of families	No. of products
Set 1	1	4	2	6	8
	2	4	2	6	8
	3	4	2	6	8
	4	4	2	6	12
	5	4	2	6	12
	6	4	2	6	12
Set 2	1	4	4	6	8
	2	4	4	6	8
	3	4	4	6	8
	4	4	4	6	12
	5	4	4	6	12
	6	4	4	6	12
Set 3	1	6	4	8	16
	2	6	4	8	16
	3	6	4	8	16
	4	6	4	8	24
	5	6	4	8	24
	6	6	4	8	24
Set 4	1	8	4	10	20
	2	8	4	10	20
	3	8	4	10	20
	4	8	4	10	30
	5	8	4	10	30
	6	8	4	10	30

In addition, we also generate three replications per problem size, that result in six instances for each problem set obtained by varying other data, namely, demand and setup time. Demand data is generated using uniform distribution $U(0, 30000)$, and, the setup time matrix ($|F| \times |F|$) for each bay consists of randomly generated numbers from $U(150, 500)$. All the runs were implemented on an Intel Xeon 3.6 GHz computer, and we used AMPL (version 8.1) along with CPLEX MIP Solver (version 10.1).

3.6.1 Comparison of PPMSP1 and PPMSP2

In this section, we present the computational results for the formulations PPMSP1 and PPMSP2 and compare the two formulations with respect to their effectiveness in solving both LP relaxation and IP versions of the problem instances. Data sets 1 and 2 are used in the experimentation, and a total number of 12 runs are performed. Table 3.3 presents the lower bound values derived by solving the LP relaxation for each formulation, as well as the optimal integer solution values. The inventory cost, backorder cost and setup cost obtained for each case are also depicted for the sake of comparison. The percentage gap for the LP-based lower bound ($GAP = \frac{IP-LP}{LP}$), number of branch-and-bound nodes used for both formulations, and the cpu time (in seconds) required are also displayed.

The results reveal the following facts: (i) for the comparison of LP relaxations of two formulations, the values of inventory, backorder and setup costs, generated by PPMSP2 are consistently tighter than those generated by PPMSP1, which results in smaller GAP values for PPMSP2. Among the inventory, backorder and setup costs, the differences in setup costs between PPMSP1 and PPMSP2 are most significant; (ii) solving the LP relaxation of PPMSP2 is also faster than that to PPMSP1; (iii) the tightness of the LP lower bound for PPMSP2 positively contributes to fewer branch-and-bound nodes, which directly saves the computational time when solving the integer version of the problem instances. In particular, for larger problem instances in data set 2, the resulting savings in the computational time are particularly significant (see instance 2.4, 2.5 and 2.6).

Table 3.3: Comparison of PPMSP1 and PPMSP2

Instance	Model	Inventory cost		Backorder cost		Setup cost		GAP	B&B nodes	cpu time (seconds)	
		LP	IP	LP	IP	LP	IP			LP	IP
1.1	PPMSP1	19156	24156	104575	114046	505	20544	27.78%	201	0.0625	0.3906
	PPMSP2	19199		104576		899		27.33%	25	0.0625	0.2344
1.2	PPMSP1	12512	16753	157876	167650	535	30416	25.68%	116	0.0469	0.3594
	PPMSP2	12518		157878		941		25.38%	82	0.0781	0.3281
1.3	PPMSP1	20784	23090	144307	148933	480	32464	23.50%	58	0.0625	0.3812
	PPMSP2	20908		144309		799		23.17%	44	0.0312	0.3281
1.4	PPMSP1	7026	11783	124028	131277	1391	24231	26.31%	941	0.0625	1.1875
	PPMSP2	7186		124096		2120		25.40%	191	0.0625	0.5625
1.5	PPMSP1	11706	13342	100509	114342	605	20800	27.99%	1215	0.0781	2.1563
	PPMSP2	11706		100509		847		27.97%	697	0.0625	1.2969
1.6	PPMSP1	15293	20244	159957	193634	1210	28800	37.53%	969	0.0781	1.875
	PPMSP2	15444		159962		1617		37.08%	742	0.0469	1.4844
2.1	PPMSP1	36861	44048	22217	30617	1010	48058	104%	695	85.99	3.2656
	PPMSP2	36868		22219		1625		102%	453	83.96	2.5937
2.2	PPMSP1	27776	30855	70666	83997	953	33904	49.65%	117206	0.0937	104
	PPMSP2	27781		70667		1484		48.85%	98912	0.0781	91.25
2.3	PPMSP1	21793	24275	12395	15567	845	35104	113%	589	0.0625	1.6718
	PPMSP2	21854		12395		1320		110%	384	0.0781	1.2656
2.4	PPMSP1	22117	27159	167885	211994	974	40000	46.17%	363330	0.1094	398.20
	PPMSP2	22192		167888		1314		45.83%	213162	0.0938	170.09
2.5	PPMSP1	23391	26336	197409	231084	1039	41680	34.82%	1984104	0.125	1973.16
	PPMSP2	23399		197411		1459		34.57%	295520	0.0781	257.047
2.6	PPMSP1	29061	31547	156982	196074	1019	39792	42.95%	2039731	0.0938	3537.81
	PPMSP2	29063		156986		1208		42.81%	789723	0.0781	646.453

3.6.2 Comparison of PPMSP-CG1 and PPMSP-CG2

In this section, we compare the computational effort required for implementing PPMSP-CG1 and PPMSP-CG2 using the problem instances in set 1 and set 2. We use two stopping criteria: reduced cost greater than $-\epsilon (= -0.0001)$ and a number of iterations greater than 100 iterations, whichever is arrived at first, for both formulations. The computational results are shown in Figure 3.4. The optimal objective values mentioned in the second column are obtained from the previous runs of PPMSP1 and PPMSP2 (see Table 3.3). The GAP1,

GAP2 and GAP3 values are calculated as follows.

$$GAP1 = \frac{IP(CG) - LP(CG)}{LP(CG)},$$

$$GAP2 = \frac{IP(OPT) - LP(CG)}{LP(CG)},$$

$$GAP3 = \frac{IP(CG) - IP(OPT)}{IP(OPT)}.$$

Table 3.4: Comparison of PPMSP-CG1 and PPMSP-CG2

Instance	Optimal obj. value	Model	Best obj. value		GAP1	GAP2	GAP3	cpu time (seconds)	Iter.
			LP	IP					
1.1	158746	PPMSP-CG1	151452	159780	5.49%	4.82%	0.65%	5.34	14
		PPMSP-CG2	168764	168764	/	/	6.31%	9.5	100
1.2	214819	PPMSP-CG1	202310	214819	6.18%	6.18%	0%	1.1719	8
		PPMSP-CG2	232305	232305	/	/	8.14%	10.25	100
1.3	204487	PPMSP-CG1	192794	204487	6.07%	6.07%	0%	1.1875	10
		PPMSP-CG2	213137	213137	/	/	4.23%	8.9844	100
1.4	167291	PPMSP-CG1	159867	169246	5.87%	4.64%	1.17%	4.8906	12
		PPMSP-CG2	176835	176835	/	/	5.71%	14.6094	100
1.5	148536	PPMSP-CG1	143984	148536	3.16%	3.16%	0%	25.0469	14
		PPMSP-CG2	148536	148536	/	/	0%	136.547	100
1.6	242678	PPMSP-CG1	226848	242678	6.98%	6.98%	0%	7.0781	12
		PPMSP-CG2	245987	245987	/	/	1.36%	19.3438	100
2.1	122724	PPMSP-CG1	107932	126225	16.95%	13.70%	2.85%	90.8594	32
		PPMSP-CG2	144875	144875	/	/	18.04%	24.4375	100
2.2	148756	PPMSP-CG1	130811	157658	20.52%	13.71%	5.98%	228.766	42
		PPMSP-CG2	174100	174100	/	/	17.04%	11.6562	100
2.3	74946	PPMSP-CG1	67731	75030	10.78%	10.65%	0.11%	36.0156	34
		PPMSP-CG2	127487	127487	/	/	70.11%	11.5781	100
2.4	279153	PPMSP-CG1	262033	292124	11.48%	6.53%	4.65%	2371.03	22
		PPMSP-CG2	300527	300527	/	/	7.66%	85.5625	100
2.5	299100	PPMSP-CG1	286512	322117	12.45%	4.39%	7.69%	3772.04	22
		PPMSP-CG2	312712	321105	/	/	7.36%	638	100
2.6	297413	PPMSP-CG1	281653	301368	6.99%	5.59%	1.33%	3342.28	42
		PPMSP-CG2	302412	302412	/	/	1.68%	197.5	100

The results reveal the following facts: (i) for all tested instances, PPMSP-CG2 do not converge within 100 iterations. It is observed that in most of computational runs, the objective function value drops within the first several iterations, and then, does not improve.

The cpu times required are those for 100 iterations. On the other hand, PPMSP-CG1 consistently converges within 100 iterations, and invariably generates better GAP values; (ii) PPMSP-CG1 generates tighter lower-bound values in comparison with those generated by PPMSP1 and PPMSP2, which can be seen by comparing the GAP1 values in Table 3.4 and GAP values in Table 3.3; (iii) for larger problem instances (see instance 2.1 - 2.6), it takes relatively large computational efforts by PPMSP-CG1.

It is important to note that the computational times required by both PPMSP-CG1 and PPMSP-CG2 (shown in Table 3.4) are larger than those required by PPMSP1 and PPMSP2 shown in Table 3.3), which is mainly due to a significant computational time required for solving the subproblem to optimality in each iteration. Note that the subproblem in the above proposed column generation approach is a lot-sizing problem in the presence of material requirement restrictions between the final and intermediate products. The solution of this subproblem is expected to be time consuming when the number of product families and the number of intermediate products are large (see the computational times of instances 2.1, 2.2, 2.3 and 2.4, 2.5, 2.6 in Table 3.4). Note that, the purpose of solving the subproblem is to generate ‘optimal’ patterns progressively for the master problem. However, as long as we add a column that is good enough to improve the object value of the master problem, the algorithm still progresses towards an optimal solution, albeit in smaller steps. In other words, each subproblem does not need to be solved to optimality, which would save computational time. We implement a computational strategy for PPMSP-CG1 in which the computational effort for ‘solving’ the lot-sizing subproblem is limited in order to improve the overall computational performance. To that end, we use an upper bound on the number of branch-and-bound nodes that are explored for solving each subproblem. When this upper-bound is reached, the best incumbent feasible solution on-hand is added as a new column. The results are depicted in Table 3.5, which shows the problem instances of set 1 and set 2 that have been solved using PPMSP-CG1 under different upper-bounds of 10,000, 20,000, 40,000 and 80,000 number of nodes for the solution of the subproblems. These are represented by PPMSP-CG1¹, PPMSP-CG1², PPMSP-CG1³ and PPMSP-CG1⁴,

respectively.

Table 3.5 reveals the following facts: (i) use of small upper-bound on number of branch-and-bound nodes significantly reduces the computational time in solving the subproblems, which lead to a reduction in solving the overall problem; (ii) it seems that the use of a smaller upper-bound does not always increase the number of iterations or deteriorate the objective function value.

Table 3.5: Computational results of PPMSP-CG1

Instance	Model	Best obj. value	cpu time (seconds)	Iter.	Instance	Model	Best obj. value	cpu time (seconds)	Iter.
3.1	PPMSP-CG1 ¹	1876305	454.23	29	4.1	PPMSP-CG1 ¹	8397363	80.391	11
	PPMSP-CG1 ²	1880626	500.34	16		PPMSP-CG1 ²	8397354	199.81	14
	PPMSP-CG1 ³	1873637	1469.1	24		PPMSP-CG1 ³	8388773	234.67	10
	PPMSP-CG1 ⁴	1885024	1509.5	15		PPMSP-CG1 ⁴	8397357	584.44	12
3.2	PPMSP-CG1 ¹	2149202	32.953	5	4.2	PPMSP-CG1 ¹	6487631	182.22	8
	PPMSP-CG1 ²	2146424	96.406	8		PPMSP-CG1 ²	6446766	789.71	16
	PPMSP-CG1 ³	2146678	220.68	10		PPMSP-CG1 ³	6440206	726.13	9
	PPMSP-CG1 ⁴	2146400	379.81	9		PPMSP-CG1 ⁴	6432495	1446.27	10
3.3	PPMSP-CG1 ¹	3057582	20.2812	7	4.3	PPMSP-CG1 ¹	5899432	498	26
	PPMSP-CG1 ²	3057582	21.8281	7		PPMSP-CG1 ²	5945077	1154.34	27
	PPMSP-CG1 ³	3057582	20.9844	7		PPMSP-CG1 ³	5894362	2680.05	42
	PPMSP-CG1 ⁴	3057582	21.0156	7		PPMSP-CG1 ⁴	5887952	5272.83	44
3.4	PPMSP-CG1 ¹	4354620	198.33	11	4.4	PPMSP-CG1 ¹	11251579	96.265	6
	PPMSP-CG1 ²	4374345	227.78	7		PPMSP-CG1 ²	11244219	348.04	10
	PPMSP-CG1 ³	4374078	428.28	8		PPMSP-CG1 ³	11238076	322.37	5
	PPMSP-CG1 ⁴	4371530	532.16	5		PPMSP-CG1 ⁴	11238076	395.83	3
3.5	PPMSP-CG1 ¹	6412462	35.922	5	4.5	PPMSP-CG1 ¹	15369476	28.906	2
	PPMSP-CG1 ²	6406156	97.172	6		PPMSP-CG1 ²	15369476	66.891	2
	PPMSP-CG1 ³	6444632	159.86	6		PPMSP-CG1 ³	15369476	118.859	2
	PPMSP-CG1 ⁴	6403843	400.02	7		PPMSP-CG1 ⁴	15369476	242.17	2
3.6	PPMSP-CG1 ¹	4763760	37.344	4	4.6	PPMSP-CG1 ¹	11743920	262.27	15
	PPMSP-CG1 ²	4764472	87.53	4		PPMSP-CG1 ²	11729795	387.38	9
	PPMSP-CG1 ³	4759605	111.16	3		PPMSP-CG1 ³	11720427	584.59	8
	PPMSP-CG1 ⁴	4759414	391.94	5		PPMSP-CG1 ⁴	11711770	1119.61	7

To further demonstrate the efficacy of this computational strategy, we compare the cpu time required by the formulations PPMSP2 and PPMSP-CG1² for solving various instances of all problem sets. The results are shown in Table 3.6. Note that the savings achieved due to the use of an upper-bound on the number of nodes for the solution of each subproblem are quite significant, particularly for large-size problem instances (see instance 2.5 and higher).

Note that for problem instances from 1.1 to 2.4, the cpu time values for PPMSP-CG1 are larger than those for PPMSP2. This is due to the upper bound on the number of nodes (20,000) used in our experimentation. For those problems, we can use a lower value of the upper bound without compromising on the quality of the solution obtained, but at the same time, significantly lowering the cpu time.

Table 3.6: Comparison of PPMSP-CG1 and PPMSP2

Instance	Model	Best obj. value	cpu time (seconds)	Instance	Model	Best obj. value	cpu time (seconds)
1.1	PPMSP2	158746	0.2344	3.1	PPMSP2	1862850	7200
	PPMSP-CG1 ²	159780	5.4521		PPMSP-CG1 ²	1880626	500.344
1.2	PPMSP2	214819	0.3281	3.2	PPMSP2	2146038	7200
	PPMSP-CG1 ²	214819	0.9218		PPMSP-CG1 ²	2146424	96.406
1.3	PPMSP2	204487	0.3281	3.3	PPMSP2	3056121	1180.7
	PPMSP-CG1 ²	204487	1.0781		PPMSP-CG1 ²	3057582	21.8281
1.4	PPMSP2	167291	0.5625	3.4	PPMSP2	4333879	7200
	PPMSP-CG1 ²	169246	5.6981		PPMSP-CG1 ²	4374345	227.781
1.5	PPMSP2	148536	1.2969	3.5	PPMSP2	6487137	6563.45
	PPMSP-CG1 ²	148536	22.8125		PPMSP-CG1 ²	6406156	97.1719
1.6	PPMSP2	242678	1.4844	3.6	PPMSP2	4728820	7200
	PPMSP-CG1 ²	242678	8.9091		PPMSP-CG1 ²	4764472	87.5312
2.1	PPMSP2	122724	2.5937	4.1	PPMSP2	8392933	1062.44
	PPMSP-CG1 ²	126225	17.9375		PPMSP-CG1 ²	8397354	199.812
2.2	PPMSP2	148756	91.25	4.2	PPMSP2	6416542	7200
	PPMSP-CG1 ²	157658	113.49		PPMSP-CG1 ²	6446766	789.703
2.3	PPMSP2	74946	1.6718	4.3	PPMSP2	5871680	7200
	PPMSP-CG1 ²	75030	41.2656		PPMSP-CG1 ²	5945077	1154.34
2.4	PPMSP2	279153	170.09	4.4	PPMSP2	11195894	7200
	PPMSP-CG1 ²	292124	230.062		PPMSP-CG1 ²	11244219	348.047
2.5	PPMSP2	299100	257.047	4.5	PPMSP2	15332615	6201.81
	PPMSP-CG1 ²	322117	172.041		PPMSP-CG1 ²	15369476	66.8906
2.6	PPMSP2	297413	646.453	4.6	PPMSP2	11638454	7200
	PPMSP-CG1 ²	301368	342.28		PPMSP-CG1 ²	11729795	387.38

3.7 A Real-life-size Example

In this section, we present a real-life example where the problem data was collected with the collaboration of a pharmaceutical primary manufacturing plant. Table 3.7 displays 12 product families and their products, in conjunction with their initial inventory levels.

Table 3.7: Data of product families and inventory levels (kg)

Family	Final Products	Intermediate Products
B	FB1(1675)	IB1(0), IB2(2350), IB3(0), IB4(0), IB5(0), IB6(1381)
C	FC1(25788)	IC1(4881)
D	FD1(132.73)	ID1(22), ID2(2), ID3(0), ID4(6)
E	FE1(126124)	/
G	FG1(43.8)	IG1(241.5), IG2(114.7), IG3(14)
H	FH1(1825)	/
I	FI1(58528)	II1(877)
J	FJ1(12)	IJ1(0), IJ2(0), IJ3(0.3)
K	FK1(6227)	IK1(818), IK2(0)
L	FL1(2111)	IL1(267), IL2(587), IL3(252)
M	FM1(1392), FM2(9830)	IM1(15132), IM2(16298), IM3(23419), IM4(7421), IM5(0), IM6(0)
N	FN1(28479)	IN1(1814), IN2(16596), IN3(31131)

Table 3.8 provides a matrix of material requirement relationship for products constituting product family B. Each element in the matrix represents a BOM ratio. For instance, it requires 1 kg of intermediate product IB6 to produce 1 kg of final product FB1.

Table 3.8: A matrix of material requirement for product family B

	FB1	IB1	IB2	IB3	IB4	IB5	IB6
FB1	-	-	-	-	-	-	-
IB1	-	-	1.2	-	-	-	-
IB2	-	-	-	1.24	-	-	-
IB3	-	-	-	-	1.54	-	-
IB4	-	-	-	-	-	1.16	-
IB5	-	-	-	-	-	-	1.43
IB6	1	-	-	-	-	-	-

The demand data for all final products over a 4-period time horizon is given in Table 3.9.

Table 3.9: Demand data (kg)

Product	Period											
	1	2	3	4	5	6	7	8	9	10	11	12
FB1	0.2	1.1	0.425	0.1	1.1	0.4	0.25	1.1	0.1	0.1	0	1
FC1	12320	9660	9660	9660	9660	9660	0	0	0	0	0	0
FD1	11	20	35	65	23	10	8	80	15	7	12	0
FE1	45000	45600	45000	45000	47400	45000	46600	45000	45000	45000	45000	45000
FG1	6	6	6	6	6	6	6	6	6	6	6	6
FH1	0	0	0	3000	0	0	0	0	0	0	0	0
FI1	0	2200	0	0	0	2200	2200	0	2200	0	0	0
FJ1	0	0	5	0	0	0	0	0	0	0	5	0
FK1	4550	2000	1500	2000	1500	2000	0	1500	1500	1500	1000	0
FL1	480	480	480	730	480	480	480	480	480	0	200	480
FM1	1640	1370	1590	3773	1070	2110	990	3210	1970	1290	2080	1230
FM2	50	0	0	50	0	100	50	0	0	100	120	0
FN1	5917	5917	5917	5917	5917	5917	5917	5917	5917	5917	5917	5917

There are 15 parallel bays, and each bay is qualified to process several products. The availability time in hours for each bay in each time-period is given in Table 3.10.

Table 3.10: Available time for bays in each time-period (hrs)

Bay	Period											
	1	2	3	4	5	6	7	8	9	10	11	12
b1	720	720	720	728	728	728	736	736	736	736	736	736
b2	720	720	720	728	728	728	736	736	736	736	736	736
b3	720	720	720	728	728	728	736	736	736	736	736	736
b4	720	720	720	728	728	728	736	736	736	736	736	736
b5	720	720	720	728	728	728	736	736	736	736	736	736
b6	720	720	720	728	728	728	736	736	736	736	736	736
b7	720	720	720	728	728	728	736	736	736	736	736	736
b8	720	720	720	728	728	728	736	736	736	736	736	736
b9	720	720	720	728	728	728	736	736	736	736	736	736
b10	720	720	720	728	728	728	736	736	736	736	736	736
b11	720	720	720	728	728	728	736	736	736	736	736	736
b12	720	720	720	728	728	728	736	736	736	736	736	736
b13	720	720	720	728	728	728	736	736	736	736	736	736
b14	720	720	720	728	728	728	736	736	736	736	736	736
b15	720	720	720	728	728	728	736	736	736	736	736	736

The batch size and batch processing time for each bay are shown in Table 3.11.

Table 3.11: Processing Data for bays (where q stands for batch size (kg) and pt stands for processing time (hrs))

Bay	Products									
b1 (q) (pt)	IL1 1131 50	IJ1 367 36	IM4 479 45							
b2 (q) (pt)	FE1 2400 25.5									
b3 (q) (pt)	IK1 716 48	IK2 840 30	IM1 480 105	IM2 649 32	IM3 1258 41	FH1 805 57				
b4 (q) (pt)	IK1 343 60	IK2 420 32	FA2 376 160	ID1 264 72	ID2 184 120	ID3 372 40	IN1 463 32	IN2 321 27	IN3 684 44	FH1
b5 (q) (pt)	IC1 165 8.5									
b6 (q) (pt)	IM1 571 80	IM4 268 48	IM5 200 24	IK2 210 30	ID1 220 97	ID2 180 120	IN1 781 33	IN2 463 72	IG1 162 101	IG2 399 72
b7 (q) (pt)	IG1 36 57	IG2 133 75	IG3 41 132	FG1 36 48	ID1 60 75	ID2 50 120	ID3 39 133	ID4 90 24	FD1 65 48	
b8 (q) (pt)	IM1 281 110									
b9 (q) (pt)	IB5 22 167	IB6 18 53	FB1 14 44							
b10 (q) (pt)	IB1 6 52	IB2 4 40	IB3 4 120	IB4 8 82	IJ1 21 18	IJ2 9 61	IJ3 65 31	FJ1 15 40		
b11 (q) (pt)	IL2 307 58	IL3 252 43	FL1 240 20							
b12 (q) (pt)	IK1 226 44	FK1 326 15	FN1 433 29							
b13 (q) (pt)	IM4 360 42	IM5 577 41	IM6 577 68	IK1 320 48	FK1 425 24	IA1 950 54	FN1 417 24			
b14 (q) (pt)	FC1 1500 54									
b15 (q) (pt)	FM1 565 24	FM2 565 24	FI1 1130 12							

The data on inventory and backorder costs was not available. However, approximate values of these were given, and consequently, we randomly generated unit inventory and backorder cost using uniform distributions $U(0.1, 1)$ and $U(5, 10)$, respectively. Similarly, the setup time matrix for each bay is generated from $U(100, 200)$, and the setup cost per

setup hour is fixed at value 8. The example was, then, solved using PPMSP-CG1, where a value of 20,000 was set as an upper-bound for the number of branch-and-bound nodes for solving the subproblems. We used two stopping criteria: reduced cost, $-\epsilon(=-0.0001)$ and number of iterations greater than 100, whichever is encountered first. The objective function value obtained is \$627,383 (inventory cost: \$540,011, backorder cost: \$2,349 and setup cost: \$85,024), after 1,154.58 cpu seconds and at iteration number 25. The production schedule for each bay over the time horizon of four periods is given in Figure 3.5, where different colors represent different product families, the product type in each time-period is represented by rectangles, each with product name, and its number of batches in parenthesis.

	1	2	3	4	5	6	7	8	9	10	11	12	
Bay 1	IM4(2)	IL1(2)	IL1(1)		IM4(1)	IL1(4)	IL1(3)	IL1(2)		IM4(1)	IM4(3)	IM4(2)	
Bay 2		FE1(4)		FE1(19)	FE1(20)	FE1(20)	FE1(19)	FE1(18)	FE1(18)	FE1(19)	FE1(19)	FE1(19)	
Bay 3	IN2(2)		IK1(4)		IK1(2)	IK1(3)		IK1(1)	IM3(1)	IK1(3)	IM3(5)	IM3(2)	
Bay 4	IK2(1)	IK2(1)	IK1(3)	FM1(2)	IK1(1)		IN3(1)	IN1(1)	IN3(9)	IN3(27)	IK2(1)	IN3(20)	
Bay 5	IC1(1)	IC1(21)	IC1(6)		IC1(62)	IC1(83)				IC1(9)	IC1(10)		
Bay 6	FM4(14)	IM5(12)	IK2(16)	IM5(17)	IM5(15)	IK2(10)	IM5(5)	IM5(26)	IK2(8)	IM4(1)	IM4(6)		
Bay 7	ID1(2)	ID3(2)	ID4(1)	FG1(1)	IG3(1)	FD1(1)	ID2(1)	ID1(1)	ID3(1)	ID2(2)	ID3(2)	FD1(2)	ID4(1)
Bay 8				IG2(2)		ID3(1)	ID2(2)	ID3(2)	ID4(2)	ID3(1)			
Bay 9	IM4(14)	IB5(9)	IB5(7)	IB5(7)	IB5(7)	IB5(7)	IB5(7)	IB5(7)	IB5(8)			FB1(1)	
Bay 10	IB3(8)	IB3(7)	IB3(8)	IB3(8)	IB3(7)	IB3(8)	IB3(9)		IB3(8)				
Bay 11	IL3(1)	IL2(4)	IL2(2)	FL1(1)	FL1(9)	FL1(3)	IL3(6)	FL1(2)	IL2(5)	FL1(3)	IL2(1)	FL1(2)	
Bay 12	FN1(1)	FK1(2)	FK1(10)	FN1(3)	FK1(2)	FN1(25)		FK1(4)	FK1(5)	FN1(14)	FN1(16)	FK1(3)	
Bay 13	IM4(16)	IK1(1)	IM4(1)	IM5(1)	FK1(2)	FK1(5)	FK1(1)	FN1(3)	FN1(12)	FK1(3)	FK1(3)	FN1(12)	
Bay 14	FC1(3)	FC1(2)	FC1(6)		FC1(6)	FC1(7)	FC1(1)			FC1(1)	FC1(1)		
Bay 15	FM1(1)	FM1(2)	FM1(3)	FM1(7)	FM1(5)		FM1(2)	FM1(9)		FM1(3)	FM1(5)		

Figure 3.5: The Proposed Schedule of the real-life example

For comparison, a schedule generated manually by the primary pharmaceutical manufacturing company is depicted in Figure 3.6. Note that the total cost for this schedule is

\$1,850,228 (inventory cost: \$1,821,460, backorder cost: 0 and setup cost: \$28,768), almost three times of the cost incurred by the schedule shown in Figure 3.5.

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
Bay 1					LI1(11)	LI1(5)						
Bay 2	FE1(20)	FE1(19)	FE1(16)	FE1(25)	FE1(25)	FE1(24)	FE1(25)	FE1(25)	FE1(20)	FE1(25)	FE1(25)	FE1(25)
Bay 3	IK1(13)	IK1(12)	IK1(7)	IK3(10)				IK3(12)	IK3(15)	IK14(7)		IK1(3)
Bay 4	IN3(13)		IN2(19) IN3(18)	IN2(18) IN3(22)	IN2(19) IN3(23)	IN2(10) IN3(22)	IN2(23)	IN2(19) IN3(23)	IN2(19) IN3(22)	IN2(19) IN3(23)	IN2(19) IN3(23)	IN2(19) IN3(22)
Bay 5	IC1(72)	IC1(68)	IC1(72)	IC1(70)								
Bay 6	ID1(1) ID2(2)	IN2(7)	IN1(19)	IN1(18)	IN1(19)	IN1(10)		IN1(19)	IN1(19)	IN1(19)	IN1(19)	IN1(19)
Bay 7		ID3(4)	ID4(3)	FD1(4)	IG2(5) IG3(0.5)	IG1(4) IG3(1.5)	FG1(2)					
Bay 8												
Bay 9				IB5(4) IB6(4)	FB1(4)							
Bay 10	IB1(7)	IB1(4) IB2(3)	IB3(6) IB4(4)			JJ1(2) JJ2(3) JJ3(1) JJ1(1)						
Bay 11	IL2(8)	IL2(8)	IL3(12)	IL2(2) IL3(1) FL1(8)	IL2(9) IL3(1)	IL3(0)	IL3(5) FL1(11)				IL2(3)	IL3(4) FL1(2)
Bay 12	FK1(25)	FK1(21)	FK1(6) FN1(3)	FN1(22)	FN1(16)		FN1(21)	FN1(21)	FN1(21)	FN1(21)		FK1(5)
Bay 13	IM4(15)	IM4(4) IM5(6)	IM4(2) IM5(12)	IM4(14)	IM4(4) IM5(8)	IM4(7) IM5(4)	IM4(8) IM6(1)	IM4(6) IM5(6)	IM4(7) IM5(4)	IM4(8) IM5(2)	IM4(14)	IM4(11) IM5(14)
Bay 14	FC1(7)	FC1(7)	FC1(7)	FC1(7)								
Bay 15		FM1(4)	FM1(13)		FM1(8)	FM1(4)		FM1(8)	FM1(3)	FM1(4)		FM1(14)

Figure 3.6: The Current Schedule of the real-life example

It is worth noting that by slightly increasing setup cost, a significant reduction can be achieved in inventory cost, thereby leading to a significant improvement in the total cost incurred.

3.8 Concluding Remarks

In this chapter, we have studied the primary pharmaceutical manufacturing scheduling problem. Our work has been motivated by a real-life instance of this problem. We have proposed two model formulations for this problem, namely, PPMSP1 and PPMSP2. Accordingly,

two column generation approaches, PPMSP-CG1 and PPMSP-CG2, are developed for these formulations. We have evaluated PPMSP1 and PPMSP2 with respect to three criteria: LP relaxation bounds, number of branch-and-bound nodes, and cpu time required. We have shown that PPMSP2 outperforms PPMSP1 on all of these aspects. For PPMSP-CG1 and PPMSP-CG2, we have compared them from the view-point of optimality GAP and speed of convergence. Our computational experimentation has shown that PPMSP-CG1 outperforms PPMSP-CG2 with respect to both of these criteria. We have implemented a computational strategy for PPMSP-CG1 in which the computational time for ‘solving’ the lot-sizing subproblem is limited by an upper-bound on the number of nodes so that the overall computational performance is improved. The computational results show that the use of such an upper-bound on the number of branch-and-bound nodes significantly reduces the computational time required in solving the subproblems, which lead to a reduction in the cpu time required to solve the overall problem. Finally, we have applied the proposed approach, PPMSP-CG1, to a real-life-size problem, and have demonstrated the superiority of the production schedule generated by our approach over the schedule currently in use.

Chapter 4

Single-Lot Lot Streaming in a Two-stage Assembly System

In this chapter, we address a single-lot lot streaming problem for a two-stage assembly system. The assembly system that we consider is different from the flow shop configuration typically assumed in the lot streaming literature. In our two-stage assembly system, the first stage consists of m parallel subassembly machines, each of which is devoted to the production of a component. A single assembly machine at the second stage, then, assembles a product after all m components (one each from the subassembly machines at the first stage) have finished processing. Lot-detached setups are encountered on the machines at the first and second stages. Given a fixed number of transfer batches (or sublots) between the two stages, the problem is to find subplot sizes so as to minimize the makespan. We develop optimality conditions for the determination of subplot sizes, and present polynomial-time algorithms to determine optimal subplot sizes when $m = 2$ and 3 .

4.1 Introduction

Lot streaming is the process of using transfer batches to move completed portions of a production lot to downstream machines so that their operations can be undertaken in an

overlapping fashion. Kalir and Sarin (2000) have shown the potential benefits of lot streaming with respect to three commonly used performance measures, namely, makespan, mean flow time and average work-in-process. The work that is reported in the flow shop lot streaming area can be classified into three categories, depending upon the number of machines considered. This includes flow shops containing two machines, three machines and m machines. The problems that are addressed differ due to the consideration of consistent or variable subplot sizes, lot-attached/detached or subplot-attached/detached setups, removal or transfer times, no-wait flow shop, and objective function, which, typically pertains to makespan, total completion time or total weighted completion time. In the following, we give a review of the lot streaming research on how to determine optimal subplot size in view of the processing of a lot in a two machine flow shop. Let the production lot consists of U items and per item processing time on machines 1 and 2 are p_1 and p_2 , respectively. The objective is to find subplot sizes s_1, \dots, s_n in order to minimize a given criterion. Trietsch (1987) and Potts and Baker (1989) have addressed the continuous lot streaming problem to minimize the makespan. They have proved that there is no inserted idle time between the processing of sublots on the second machine, and have also shown that the optimal subplot sizes are geometric in nature with a ratio of p_2/p_1 (see Figure 4.1). Since there is no inserted

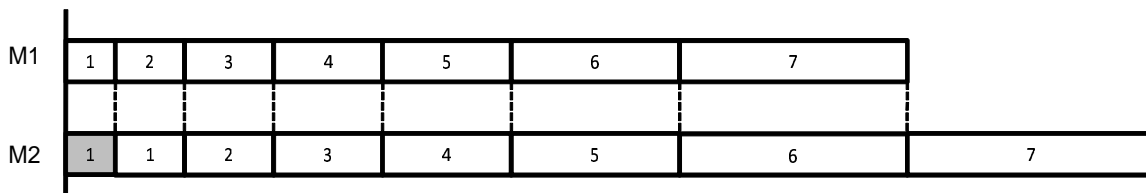


Figure 4.1: An example of geometric subplot sizes

idle time among the processing of the sublots, we have

$$s_i = qs_{i-1}, \quad (4.1)$$

where $q = p_2/p_1$. Equivalently, we have

$$s_i = q^{i-1} s_1. \quad (4.2)$$

Combined with $\sum_{i=1}^n s_i = U$, we can obtain

$$s_1 = \begin{cases} \frac{1-q}{1-(q)^n}U, & \text{if } q \neq 1, \\ \frac{U}{n}, & \text{otherwise.} \end{cases} \quad (4.3)$$

Substituting the value of s_1 from (4.3), the optimal subplot sizes can be given by

$$s_i = \begin{cases} \frac{(q)^{i-1} - (q)^i}{1 - (q)^n}U, & \forall i = 1, \dots, n & \text{if } q \neq 1, \\ \frac{U}{n}, & & \text{otherwise.} \end{cases} \quad (4.4)$$

For the integer subplot size case, Trietsch and Baker (1993) have also presented a polynomial-time procedure to obtain integer subplot sizes, which utilizes the continuous subplot size solution as a starting point. Sriskandarajah and Wagneur (1999) have shown that geometric subplot sizes are also optimal for continuous lot streaming in a no-wait flow shop with lot-detached setups, and they have developed heuristics algorithms for the discrete version of the problem. Chen and Steiner (1999) have proposed approximation algorithms, which generate a makespan value that is within $\min\{p_1, p_2\}$ of the integer optimal makespan for the above problem. Sen et al. (1998) have considered the lot streaming problem for the objective of minimizing the weighted completion time. They have developed a solution procedure to obtain optimal subplot sizes by using the property that an optimal solution consists of equal sublots on both the machines if $p_1 \geq p_2$, and geometric sublots on the first machine and equal sublots on the second machine if $p_1 < p_2$. Bukchin et al. (2002) have addressed a problem for the objective of minimizing the weighted completion time in the presence of subplot-attached setup time. They have developed a heuristic procedure that is based on the Single Machine Bottleneck (SMB) property. For a comprehensive treatment of the work reported in the flow shop lot streaming problem, please see Sarin and Jaiprakash (2007).

Our work in this chapter is different from that presented in the literature in that we consider a lot streaming problem for processing a single lot in a two-stage assembly system. The configuration of the assembly system that we consider is illustrated in Figure 4.2. The first stage consists of multiple, parallel machines where subassemblies are prepared with one

subassembly-type on each machine. These subassemblies, are then, assembled to form a final product at the second stage. For the instance shown in Figure 4.2, the first stage consists of three parallel machines, and the lot consists of 20 units. Lot-detached setup is incurred on every machine (of both the stages). For the example, these values are assumed to be 26, 30, 16 for the three subassembly machines, and 43 for the assembly machine. Let the unit processing times be 2, 3, 4 on the subassembly machines, and 3 on the assembly machine. The processing of the lot without lot streaming is shown in Figure 4.2(a). Figure 4.2(b) depicts its processing in the presence of lot streaming, for which the lot is split into three sublots of sizes 7, 7 and 6. Note that because of lot streaming, the makespan reduces from 156 to 114. However, the solution could be further improved by appropriately determining sublot sizes. This is the focus of our work in this chapter.

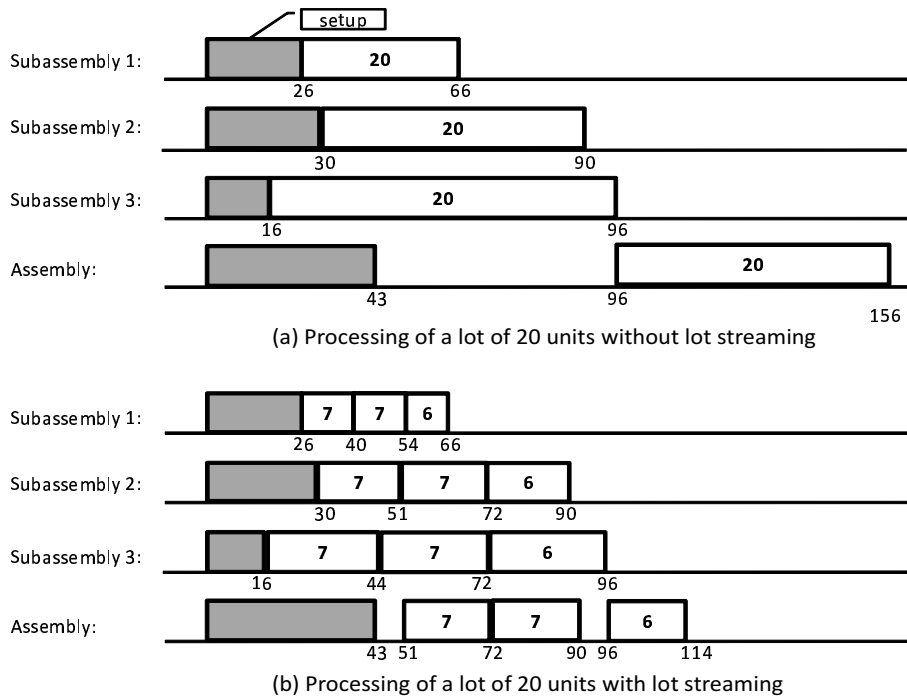


Figure 4.2: Example of lot streaming in a two-stage assembly system

An example of the two-stage assembly system that we consider in this chapter is a supplier-manufacturer tandem for the assembly of a part. The components of an assem-

bly are produced by suppliers (at stage 1) (one component by each supplier), which are then assembled at stage 2. The real-life instances of such a system include dressing of engines and transmissions by suppliers (at stage 1) for their assembly at the assembly plant (at stage 2), and preparation of integrated circuits (ICs) and power supplies among other components, by suppliers (at stage 1) for their assembly on printed circuit boards (PCBs) (at stage 2). The assembly facility needs to coordinate its operations with the availability of components provided by the suppliers. We address this issue in this chapter.

The chapter is organized as follows. In Section 4.2, we present a mathematical model for the problem on hand. In Section 4.3, some machine dominance properties are developed, and we also present a procedure to implement the dominance properties. This is followed by development of necessary optimality conditions for the m -supplier and single-assembly system, designated as $m + 1$ system. In Sections 4.4 and 4.5, we develop sufficient optimality conditions for the $2 + 1$ problem, and present polynomial-time algorithms for finding the optimal solutions for both the $2 + 1$ and $3 + 1$ problems. In Section 4.6, a polynomial-time algorithm is presented to determine optimal, integer subplot sizes for these problems as well. The algorithms are illustrated using numerical examples. Finally, concluding remarks are made in Section 4.7.

4.2 Problem Description and Formulation

The problem that we consider in this chapter can be described as follows. A single lot consisting of U units is to be processed in a two-stage assembly system. There is a set $\Omega = \{1, \dots, m\}$ of m subassembly machines at the first stage and one assembly machine at the second stage. The processing time per item on a machine at the first stage may be different for different machines, and is designated by p_k , for machine k , $\forall k = 1, \dots, m$, and each of these may be different from the time required for assembling m components into a finished product on the assembly machine, which is denoted by p_A . A lot-detached setup time is incurred on each of the subassembly machines at stage 1 and on the assembly machine

at stage 2. These are denoted by $t_k, \forall k = 1, \dots, m$, and t_A for the subassembly machines and the assembly machine, respectively. We assume that machines at both the stages use the same number of sublots, n . Our objective is to determine sizes of the sublots for processing on the subassembly machines and the assembly machine so as to minimize the makespan. We designate the size of sublot $i, \forall i = 1, \dots, n$, processed on subassembly machine $k, \forall k = 1, \dots, m$, by s_{ik} , the completion time of sublot $i, \forall i = 1, \dots, n$, on subassembly machine $k, \forall k = 1, \dots, m$, by C_{ik} , the completion time of sublot $i, \forall i = 1, \dots, n$, on the assembly machine A , by C_{iA} , and the cumulative sublot sizes up to sublot $i, i = 1, \dots, n$, on a subassembly machine k and the assembly machine A , by $S_{ik} = \sum_{u=1}^i s_{uk}$ and $S_{iA} = \sum_{u=1}^i s_{uA}$, respectively.

We assume that: (i) all the machines are available at time zero; (ii) sublot sizes are continuous (we address the integer sublot size case later); (iii) the processing of a sublot i on machine A can begin only after a sufficient number of its components have finished processing at the first stage. Note that, the first assumption is made for the sake of simplicity, since, without loss of generality, the ready time of a machine can be included in its lot-detached setup time. We call this problem as a two-stage lot streaming problem, designed by TSLSP.

4.2.1 Mathematical Formulation

Before introducing our mathematical formulation for the TSLSP, we first show the following result, which helps in curtailing the type of sublots that we need to consider.

Proposition 4.1. *There exists an optimal solution in which the sublot sizes used for processing the lot on subassembly and assembly machines are consistent.*

Proof. We prove this by construction. Let Q be a solution in which $s_{1k}, s_{2k}, \dots, s_{nk}$ and $s_{1A}, s_{2A}, \dots, s_{nA}$ are the sizes of sublots processed on the subassembly machine $k, \forall k = 1, \dots, m$, and the assembly machine A , respectively. Suppose these sublot sizes are not consistent. Let $v(i, k)$ denote the index of the last among the sublots on subassembly machine k that contain items constituting sublot i on assembly machine A . By definition, $S_{iA} \leq S_{v(i,k),k}$.

For the first subplot on assembly machine A , we have

$$C_{1A} = \max\{t_A, \max_{1 \leq k \leq m} \{(t_k + p_k S_{v(1,k),k})\}\} + p_A s_{1A}.$$

For subplot $2 \leq i \leq n$ on machine A , we have

$$C_{iA} = \max\{\max_{1 \leq k \leq m} (t_k + p_k S_{v(i,k),k}), C_{(i-1),A}\} + p_A s_{iA}.$$

Consider another solution Q' as follows. Set $s'_{iA} = s_{iA}$ and $s'_{ik} = s_{iA}$, for all $1 \leq i \leq n$ and $1 \leq k \leq m$. For subplot 1 on machine A , we have

$$C'_{1A} = \max\{t_A, \max_{1 \leq k \leq m} \{(t_k + p_k S'_{1k})\}\} + p_A s'_{1A}.$$

Since in Q' , $S'_{1k} = S'_{1A} = S_{1A} \leq S_{v(1,k),k}$, we have, $C'_{1A} \leq C_{1A}$. Similarly, for subplot 2 in Q' , we have

$$C'_{2A} = \max\{\max_{1 \leq k \leq m} (t_k + p_k S'_{2k}), C'_{1A}\} + p_A s'_{2A}.$$

And, since $S'_{2k} = S'_{2A} = S_{2A} \leq S_{v(2,k),k}$, we have,

$$C'_{2A} \leq C_{2A}.$$

Let $C'_{iA} \leq C_{iA}$, for $1 \leq i \leq q$. We want to show that it holds for $i = q + 1$. Since $S'_{q+1,k} = S'_{q+1,A} = S_{q+1,A} \leq S_{v(q+1,k),k}$, we have

$$C'_{q+1,A} = \max\{\max_{1 \leq k \leq m} \{t_k + p_k S'_{q+1,k}\}, C'_{q,A}\} + p_A s'_{q+1,A}.$$

Consequently, $C'_{q+1,A} \leq C_{q+1,A}$. Thus, a solution with consistent subplot sizes is at least as good as any other solution. \square

As a consequence of the above result, we can drop the subscript “ k ” and “ A ” from the notation for subplot sizes and cumulative subplot sizes. Our model for the TSLSP is as follows.
Minimize

$$C_{nA} \tag{4.5a}$$

subject to:

$$C_{1k} \geq t_k + p_k s_1, \quad \forall k = 1, \dots, m \quad (4.5b)$$

$$C_{1A} \geq t_A + p_A s_1, \quad (4.5c)$$

$$C_{ik} \geq C_{i-1,k} + p_k s_i, \quad \forall i = 2, \dots, n, k = 1, \dots, m \quad (4.5d)$$

$$C_{iA} \geq C_{i-1,A} + p_A s_i, \quad \forall i = 2, \dots, n \quad (4.5e)$$

$$C_{iA} \geq C_{ik} + p_A s_i, \quad \forall i = 1, \dots, n, k = 1, \dots, m \quad (4.5f)$$

$$\sum_{i=1}^n s_i = U, \quad \forall k = 1, \dots, m \quad (4.5g)$$

$$s_i \geq 0, \quad \forall i = 1, \dots, n \quad (4.5h)$$

In the above model, constraints (4.5b) and (4.5c) ensure that the first subplot can start its processing on machine k , $\forall k = 1, \dots, m$, only after the detached setup has been finished on the respective machines. Constraints (4.5d) and (4.5e) ensure that a subplot i , $\forall i = 2, \dots, n$, can start processing on machines k and A only after subplot $(i - 1)$ has finished its processing on the respective machines k and A . Constraints (4.5f) assert that a subplot i , $\forall i = 1, \dots, n$, can start processing on assembly machine A only after it has finished its processing on all subassembly machines. Constraints (4.5g) ensure that the total number of items assigned to sublots is equal to the total number of items available.

A network representation of the TSLSP, given in Figure 4.3, is used to illustrate the process of lot streaming in our two-stage assembly system. A node (k, i) corresponds to the i^{th} subplot on machine k , and it carries a weight of $p_k s_i$ that represents the processing time of the i^{th} subplot on the subassembly machine k . A directed arc from node (k, i) to $(k, i + 1)$ indicates that machine k cannot start processing subplot $(i + 1)$ until it has finished processing subplot i . A directed arc from (k, i) to (A, i) captures the requirement that subplot i cannot begin processing on the assembly machine A unless it has finished processing on the subassembly machine k . The length of a path is the sum of the weights of the nodes that lie on that path. The TSLSP is, thus, to allocate items of the lot to sublots (that is, to determine the weight of each node) in order to minimize the length of the critical path of

the network.

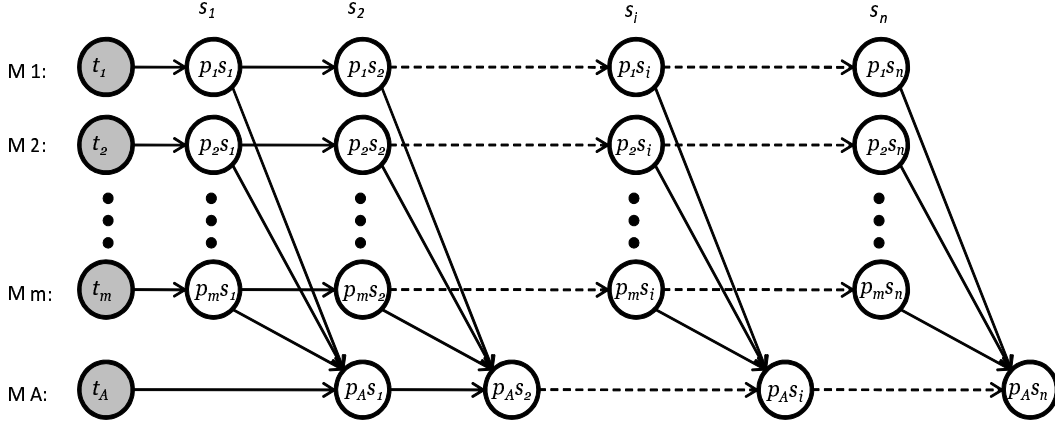


Figure 4.3: Network representation of the lot streaming problem

The makespan resulting from the processing of the sublots of a lot on the subassembly machines $1, \dots, m$ and the assembly machine A , is given by

$$M = \max\left\{ \max_{1 \leq k \leq m} \left\{ t_k + \max_{1 \leq e \leq n} \left\{ p_k \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i \right\} \right\}, t_A + p_A U \right\}, \quad (4.6)$$

where $t_A + p_A U$ is a lower bound of the makespan value. Hence, in order to find the optimal makespan, we need to evaluate the first expression in (4.6). Denote this expression by

$$M_s = \max_{1 \leq k \leq m} \left\{ t_k + \max_{1 \leq e \leq n} \left\{ p_k \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i \right\} \right\}, \quad (4.7)$$

which can be written as

$$M_s \geq t_k + p_k \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i, \quad \forall k = 1, \dots, m, \forall e = 1, \dots, n. \quad (4.8)$$

We call a subplot e to be critical with respect to a subassembly machine k for which the expression (4.8) holds as equality. If after having computed optimal M_s , we have $M_s \geq t_A + p_A U$, then we have an optimal solution of the given problem. In case, $M_s < t_A + p_A U$, then the optimal makespan value is $t_A + p_A U$, while the subplot sizes obtained for M_s are still optimal.

Next, we develop some structural properties of the problem of minimizing M_s that afford an efficient algorithm for its solution.

4.3 Development of Optimality Conditions

In this section, the machine dominance properties and procedures to implement these dominance properties are presented. This is followed by development of necessary optimality condition for the $m + 1$ system.

4.3.1 Machine Dominance Properties

Note that the completion time of subplot e , $e = 1, \dots, n$, on machine k can be represented by a straight line, $C_{ek} = t_k + p_k \cdot S_e$, for $k = 1, \dots, m$, where $S_e = \sum_{i=1}^e s_i$. For this straight line, the lot-detached setup time, t_k , and unit processing time, p_k , can be regarded as its intercept and slope, respectively. For the case with two machines, u and v , if $p_u = p_v$, then the lines C_{eu} and C_{ev} are parallel, otherwise, we are able to find their intersection point, which is given by $\frac{t_u - t_v}{p_v - p_u}$. This is illustrated in Figure 4.4.

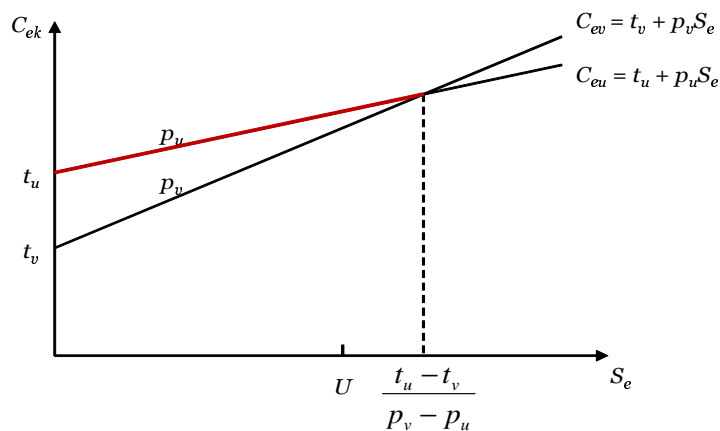


Figure 4.4: Illustration of completion times on two machines

We can reduce the size of the problem on hand due to the following dominance among the machines.

Proposition 4.2. *For any two subassembly machines u and v , if $t_u \geq t_v$ and $t_u + p_u U \geq t_v + p_v U$, then subassembly machine v can be eliminated from the problem.*

Proof. In case $p_u = p_v$, line C_{eu} either overlaps with or lies above line C_{ev} depending on $t_u = t_v$ or $t_u > t_v$, which indicates that machine v is dominated. Otherwise, we have $\frac{t_u - t_v}{p_v - p_u} \geq U$ since $t_u + p_u \cdot U \geq t_v + p_v \cdot U$, and line C_{eu} will lie above line C_{ev} for $0 \leq S_e \leq U$. For a subplot e , $e = 1, \dots, n$, we have $t_u + p_u \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i \geq t_v + p_v \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i$, which indicates that machine v is dominated by machine u . \square

Remark 4.1. The following polynomially-time-bounded procedure can be used to implement the result of Proposition 4.2.

Initialization: Rank and index all the subassembly machines in set Ω based on the non-increasing order of their lot-detached setup times. Define a set $\widehat{\Omega}$, and let $\widehat{\Omega} = \emptyset$ at start. Let $u = 1$ and $v = 2$.

Step 1: If $t_u + p_u U < t_v + p_v U$, then go to Step 2; otherwise, add machine v to $\widehat{\Omega}$, then go to Step 2.

Step 2: If $u = |\Omega| - 1$ and $v = |\Omega|$, stop. Otherwise, we have two cases: case (i): $u < |\Omega| - 1$ and $v = |\Omega|$, then go to Step 3; case (ii): $u < |\Omega| - 1$ and $v < |\Omega|$, then go to Step 4.

Step 3 Let $\Omega = \Omega - \widehat{\Omega}$, re-index all the subassembly machines in set Ω and let $u = u + 1$ and $v = u + 1$, go to Step 1;

Step 4 Let $v = v + 1$, and go to Step 1.

After applying Proposition 4.2, we denote the remaining set of subassembly machines by Ω' . The following conditions exist among the jobs with respect to any two subassembly machines u and $v \in \Omega'$,

$$t_u > t_v, \quad p_u < p_v, \quad t_u + p_u \cdot U < t_v + p_v \cdot U, \quad \text{for } 1 \leq u < v < w \leq m. \quad (4.9)$$

Note that machine 1 has the shortest processing time and the longest lot-detached setup time, while machine m has the longest processing time and the shortest lot-detached setup time.

Next, we develop a dominance property among the machines in Ω' .

Proposition 4.3. *For any three machines u , v and $w \in \Omega'$, where $t_u > t_v > t_w$ and $p_u < p_v < p_w$, subassembly machine v is dominated by either machine u or w , and hence, does not dictate the makespan if the following condition holds*

$$\frac{t_u - t_v}{p_v - p_u} \geq \frac{t_v - t_w}{p_w - p_v}. \quad (4.10)$$

Proof. Consider a subplot e , $e = 1, \dots, n$ in a solution. There are the following three possibilities for the cumulative size up to subplot e : (i) $\sum_{i=1}^e s_i \leq \frac{t_v - t_w}{p_w - p_v}$, (ii) $\sum_{i=1}^e s_i \geq \frac{t_u - t_v}{p_v - p_u}$, or (iii) $\frac{t_v - t_w}{p_w - p_v} < \sum_{i=1}^e s_i < \frac{t_u - t_v}{p_v - p_u}$. In case (i), according to the condition $\frac{t_u - t_v}{p_v - p_u} \geq \frac{t_v - t_w}{p_w - p_v}$ in (4.10), we have $\sum_{i=1}^e s_i \leq \frac{t_u - t_v}{p_v - p_u}$, which leads to $t_u + p_u \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i \geq t_v + p_v \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i$. In case (ii), again according to the condition $\frac{t_u - t_v}{p_v - p_u} \geq \frac{t_v - t_w}{p_w - p_v}$ in (4.10), we have $\sum_{i=1}^e s_i \geq \frac{t_v - t_w}{p_w - p_v}$, which leads to $t_w + p_w \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i \geq t_v + p_v \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i$. In case (iii), we have both $t_u + p_u \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i > t_v + p_v \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i$ and $t_w + p_w \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i > t_v + p_v \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i$ satisfied. For all the above three cases, machine v is dominated by either machine u , w or both. \square

The result of Proposition 4.3 is depicted in Figure 4.5. Note that, in the case of $t_u > t_v > t_w$ and $p_u < p_v < p_w$, and under the condition $\frac{t_u - t_v}{p_v - p_u} \geq \frac{t_v - t_w}{p_w - p_v}$, we also have

$$\frac{t_u - t_v}{p_v - p_u} \geq \frac{t_u - t_w}{p_w - p_u} \geq \frac{t_v - t_w}{p_w - p_v}. \quad (4.11)$$

When $S_e (= \sum_{i=1}^e s_i) \leq \frac{t_u - t_w}{p_w - p_u}$, machine u dominates machines v and w , while for $S_e \geq \frac{t_u - t_w}{p_w - p_u}$, machine w dominates machines u and v . Therefore, machine v never dictates the makespan.

Remark 4.2. The following polynomially-time-bounded procedure can be used to apply the result of Proposition 4.3 to the machine set Ω' .

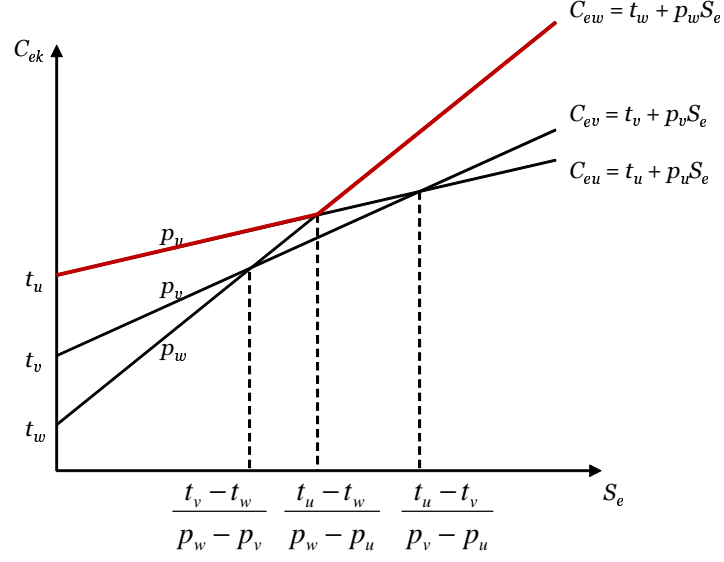


Figure 4.5: Illustration of completion times on three machines

Initialization: Rank and index all the subassembly machines in set Ω' based on the increasing order of their lot-detached setup times or on the decreasing order of their unit processing times. Let $u = 1$, $v = 2$, $w = 3$. Define a set $\bar{\Omega}'$, and let $\hat{\Omega}' = \emptyset$ at start.

Step 1: Calculate $\frac{t_u - t_v}{p_v - p_u}$ and $\frac{t_v - t_w}{p_w - p_v}$. If $\frac{t_u - t_v}{p_v - p_u} < \frac{t_v - t_w}{p_w - p_v}$, go to Step 2; otherwise, add machine v to $\hat{\Omega}'$, and go to Step 2.

Step 2: If $u = |\Omega'| - 2$, $v = |\Omega'| - 1$, and $w = |\Omega'|$, stop. Otherwise, we have the following three cases: case (i): $u < |\Omega'| - 2$, $v = |\Omega'| - 1$, and $w = |\Omega'|$, then go to step 3; case (ii): $u < |\Omega'| - 2$, $v < |\Omega'| - 1$, and $w = |\Omega'|$, then go to Step 4; case (iii): $u < |\Omega'| - 2$, $v < |\Omega'| - 1$, and $w < |\Omega'|$, go to Step 5.

Step 3: Let $\Omega' = \Omega' - \hat{\Omega}'$, and re-index the machines in Ω' . Let $u = u + 1$, $v = u + 1$, and $w = v + 1$, go to Step 1.

Step 4: Let $\Omega' = \Omega' - \hat{\Omega}'$, and re-index the machines in Ω' . Let $v = v + 1$ and $w = v + 1$, go to Step 1.

Step 5: Let $w = w + 1$, go to Step 1.

As a result of Proposition 4.3, denote the remaining set of subassembly machines by Ω'' . Without loss of generality, we assume that there are m machines in Ω'' . Let $\sigma_k = \frac{t_k - t_{k+1}}{p_{k+1} - p_k}$, where $k = 1, \dots, m - 1$. For the sake of convenience, let $\sigma_0 = 0$ and $\sigma_m = U$, and we have

$$\sigma_0 < \sigma_1 < \sigma_2 < \dots < \sigma_{m-1} < \sigma_m. \quad (4.12)$$

Note the strict inequalities in (4.12). This is afforded by the fact that if two successive σ -values are the same, then one of the associated machines will be eliminated by the result of Proposition 4.3. For illustration purpose, Figure 4.6 depicts the completion times, C_{ek} , for all sublots e , $e = 1, \dots, n$ on each of the subassembly machines in Ω'' .

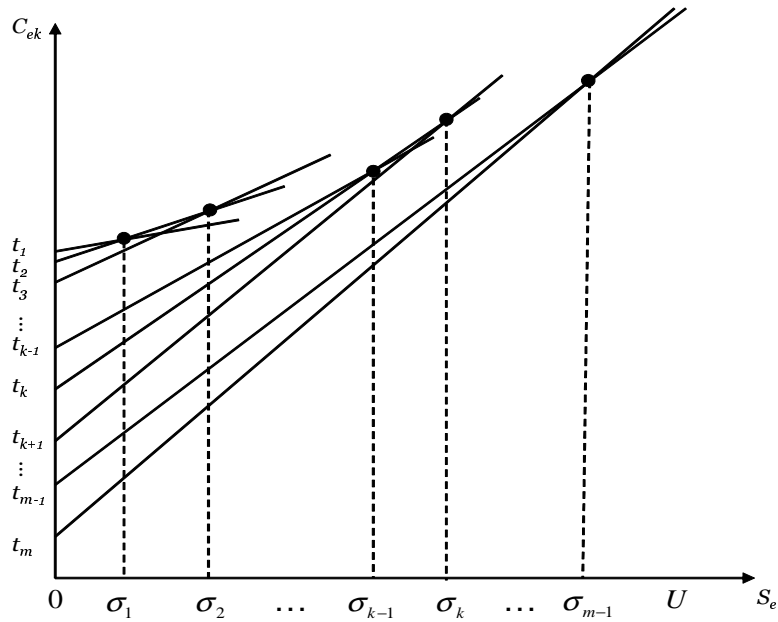


Figure 4.6: Illustration of completion times on m machines

4.3.2 Optimality Conditions

Next, we develop some necessary conditions satisfied by optimal subplot sizes.

Proposition 4.4. *In an optimal solution, if there exists a subplot e such that the cumulative size of the sublots up to subplot e satisfies $\sum_{i=1}^e s_i \in (\sigma_{k-1}, \sigma_k]$, for some subassembly machine k , then subplot e must be critical with respect to the subassembly machine k . In other words, the makespan,*

$$M = t_k + p_k \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i, \quad \forall e = 1, \dots, n. \quad (4.13)$$

Proof. We prove this by contradiction. Suppose such a subplot e is contained in an optimal solution Q but it is not critical for machine k . We have

$$\Delta = M(Q) - [t_k + p_k(s_1 + \dots + s_e) + p_A(s_e + \dots + s_n)],$$

for some positive Δ . Now, construct a solution Q' by setting

$$\begin{aligned} s'_e &= s_e(1 - \theta) + U(\theta), \\ s'_i &= s_i(1 - \theta), \quad \forall i \neq e, \end{aligned}$$

where $0 < \theta < \frac{\Delta}{\Delta + t_m + (p_m + p_A)U - M(Q)}$. Now suppose c' is the index of a critical subplot in schedule Q' . We have the following three cases: case (1). $S'_{c'} \in (\sigma_{k-1}, \sigma_k]$; case (2). $S'_{c'} \leq \sigma_{k-1}$; and case (3). $S'_{c'} > \sigma_k$. We discuss each of these cases next.

Case (1): $S'_{c'} \in (\sigma_{k-1}, \sigma_k]$, then, by the definition of the critical subplot, we have

$$M(Q') = t_k + p_k(s'_1 + \dots + s'_{c'}) + p_A(s'_{c'} + \dots + s'_n). \quad (4.14)$$

This case can be decomposed into three subcases as follows:

Case (1.1): If $c' > e$, we have

$$\begin{aligned} M(Q') &= t_k + p_k(s'_1 + \dots + s'_{c'}) + p_A(s'_{c'} + \dots + s'_n) \\ &= t_k + (1 - \theta)[p_k(s_1 + \dots + s_{c'}) + p_A(s_{c'} + \dots + s_n)] + \theta p_v U \\ &= (1 - \theta)[t_k + p_k(s_1 + \dots + s_{c'}) + p_A(s_{c'} + \dots + s_n)] + \theta(t_k + p_v U) \\ &\leq (1 - \theta)M(Q) + \theta(t_k + p_v U) \\ &< (1 - \theta)M(Q) + \theta M(Q). \\ &< M(Q) \end{aligned}$$

Case (1.2): If $c' = e$, we have

$$\begin{aligned}
M(Q') &= t_k + p_k(s'_1 + \cdots + s'_{c'}) + p_A(s'_{c'} + \cdots + s'_n) \\
&= t_k + (1 - \theta)[p_k(s_1 + \cdots + s_{c'}) + p_A(s_{c'} + \cdots + s_n)] + \theta[(p_k + p_A)U] \\
&= (1 - \theta)[t_k + p_k(s_1 + \cdots + s_{c'}) + p_A(s_{c'} + \cdots + s_n)] + \theta[t_k + (p_k + p_A)U] \\
&= (1 - \theta)(M(Q) - \Delta) + \theta[t_k + (p_k + p_A)U] \\
&< M(Q).
\end{aligned}$$

The above inequality holds as long as $0 < \theta < \frac{\Delta}{\Delta + t_m + (p_m + p_A)U - M(Q)} \leq \frac{\Delta}{\Delta + t_k + (p_k + p_A)U - M(Q)}$.

Case (1.3): $c' < e$, we have

$$\begin{aligned}
M(Q') &= t_k + p_k(s'_1 + \cdots + s'_{c'}) + p_A(s'_{c'} + \cdots + s'_n) \\
&= t_k + (1 - \theta)[p_k(s_1 + \cdots + s_{c'}) + p_A(s_{c'} + \cdots + s_n)] + \theta p_A U \\
&= (1 - \theta)[t_k + p_k(s_1 + \cdots + s_{c'}) + p_A(s_{c'} + \cdots + s_n)] + \theta(t_k + p_A U) \\
&\leq (1 - \theta)M(Q) + \theta(t_k + p_A U) \\
&< (1 - \theta)M(Q) + \theta M(Q) \\
&< M(Q)
\end{aligned}$$

Case (2): $S'_{c'} \leq \sigma_{k-1}$, then, by definition of the critical subplot, we have

$$M(Q') = t_u + p_u(s'_1 + \cdots + s'_{c'}) + p_A(s'_{c'} + \cdots + s'_n), \quad (4.15)$$

for some machine $u < k$.

Case (2.1): If $c' > e$, we have

$$\begin{aligned}
M(Q') &= t_u + p_u(s'_1 + \cdots + s'_{c'}) + p_A(s'_{c'} + \cdots + s'_n) \\
&= t_k + (1 - \theta)[p_u(s_1 + \cdots + s_{c'}) + p_A(s_{c'} + \cdots + s_n)] + \theta p_u U \\
&= (1 - \theta)[t_u + p_u(s_1 + \cdots + s_{c'}) + p_A(s_{c'} + \cdots + s_n)] + \theta(t_u + p_u U) \\
&\leq (1 - \theta)M(Q) + \theta(t_u + p_u U) \\
&< (1 - \theta)M(Q) + \theta M(Q) \\
&< M(Q)
\end{aligned}$$

Case (2.2): If $c' = e$, we have

$$\begin{aligned}
M(Q') &= t_u + p_u(s'_1 + \cdots + s'_{c'}) + p_A(s'_{c'} + \cdots + s'_n) \\
&= t_u + (1 - \theta)[p_u(s_1 + \cdots + s_{c'}) + p_A(s_{c'} + \cdots + s_n)] + \theta(p_u + p_A)U \\
&= (1 - \theta)[t_u + p_k(s_1 + \cdots + s_e) + p_A(s_e + \cdots + s_n)] + \theta(t_u + (p_u + p_A)U) \\
&= (1 - \theta)[M(Q) - \Delta] + \theta[t_u + (p_u + p_A)U] \\
&< M(Q)
\end{aligned}$$

The above inequality holds as long as $0 < \theta < \frac{\Delta}{\Delta + t_m + (p_m + p_A)U - M(Q)} \leq \frac{\Delta}{\Delta + t_u + (p_u + p_A)U - M(Q)}$.

Case (2.3): If $c' < e$, then we have

$$\begin{aligned}
M(Q') &= t_u + (1 - \theta)[p_u(s_1 + \cdots + s_{c'}) + p_A(s_{c'} + \cdots + s_n)] + \theta p_A U \\
&= (1 - \theta)[t_u + p_u(s_1 + \cdots + s_{c'}) + p_A(s_{c'} + \cdots + s_n)] + \theta[t_u + p_A U] \\
&\leq (1 - \theta)M(Q) + \theta(t_u + p_A U) \\
&< (1 - \theta)M(Q) + \theta M(Q) \\
&< M(Q).
\end{aligned}$$

Case (3): $S'_{c'} > \sigma_k$, then, by definition of the critical subplot, we have

$$M(Q') = t_u + p_u(s'_1 + \cdots + s'_{c'}) + p_A(s'_{c'} + \cdots + s'_n) \quad (4.16)$$

for some machine $u > k$

Case (3.1): If $c' > e$, we have

$$\begin{aligned}
M(Q') &= t_u + p_u(s'_1 + \cdots + s'_{c'}) + p_A(s'_{c'} + \cdots + s'_n) \\
&= t_u + (1 - \theta)[p_u(s_1 + \cdots + s_{c'}) + p_A(s_{c'} + \cdots + s_n)] + \theta p_u U \\
&= (1 - \theta)[t_u + p_u(s_1 + \cdots + s_{c'}) + p_A(s_{c'} + \cdots + s_n)] + \theta(t_u + p_u U) \\
&\leq (1 - \theta)M(Q) + \theta(t_u + p_u U) \\
&< (1 - \theta)M(Q) + \theta M(Q) \\
&< M(Q).
\end{aligned}$$

Case (3.2): If $c' = e$, we have

$$\begin{aligned}
M(Q') &= t_u + p_u(s'_1 + \cdots + s'_{c'}) + p_A(s'_{c'} + \cdots + s'_n) \\
&= t_u + (1 - \theta)[p_u(s_1 + \cdots + s_{c'}) + p_A(s_{c'} + \cdots + s_n)] + \theta(p_u + p_A)U \\
&= (1 - \theta)[t_u + p_u(s_1 + \cdots + s_e) + p_A(s_e + \cdots + s_n)] + \theta(t_u + (p_u + p_A)U) \\
&= (1 - \theta)[M(Q) - \Delta] + \theta[t_u + (p_u + p_A)U] \\
&< M(Q).
\end{aligned}$$

The above inequality holds as long as $0 < \theta < \frac{\Delta}{\Delta + t_m + (p_m + p_A)U - M(Q)} \leq \frac{\Delta}{\Delta + t_u + (p_u + p_A)U - M(Q)}$.

Case (3.3): If $c' < e$, then we have

$$\begin{aligned}
M(Q') &= t_u + (1 - \theta)[p_u(s_1 + \cdots + s_{c'}) + p_A(s_{c'} + \cdots + s_n)] + \theta p_A U \\
&= (1 - \theta)[t_u + p_u(s_1 + \cdots + s_{c'}) + p_A(s_{c'} + \cdots + s_n)] + \theta[t_u + p_A U] \\
&\leq (1 - \theta)M(Q) + \theta(t_u + p_A U) \\
&< (1 - \theta)M(Q) + \theta M(Q) \\
&< M(Q).
\end{aligned}$$

The above three cases show that a better schedule Q' can be constructed, which contradicts the assumption that schedule Q is an optimal schedule. \square

Note that if $\sum_{i=1}^e s_i = \sigma_{k-1}$, then, without loss of generality, we define subplot e to be critical with respect to $k - 1$ and not k . Proposition 4.4 also implies that, in order for every subplot to be critical, there should be neither an inserted idle time on the assembly and subassembly machines nor a waiting time of a subplot on any subassembly machine. We state this formally as follows.

Corollary 4.1. *For any two sublots s_e and s_f in an optimal schedule where $e < f$, if $\sum_{i=1}^e s_i \in (\sigma_{k_1-1}, \sigma_{k_1}]$ and $\sum_{i=1}^f s_i \in (\sigma_{k_2-1}, \sigma_{k_2}]$, for any two machines k_1 and k_2 , where $k_1 \leq k_2$, then the following equation holds:*

$$t_{k_1} + p_{k_1} \sum_{i=1}^e s_i + p_A \sum_{i=e}^f s_i = t_{k_2} + p_{k_2} \sum_{i=1}^f s_i + p_A s_f. \quad (4.17)$$

Proof. From Proposition 4.4, we have $M = t_{k_1} + p_{k_1} \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i$ for $\sum_{i=1}^e s_i \in (\sigma_{k_1-1}, \sigma_{k_1}]$ and $M = t_{k_2} + p_{k_2} \sum_{i=1}^f s_i + p_A \sum_{i=f}^n s_i$ for $\sum_{i=1}^f s_i \in (\sigma_{k_1-1}, \sigma_{k_1}]$. By equating and simplifying, we have $t_{k_1} + p_{k_1} \sum_{i=1}^e s_i + p_A \sum_{i=e}^f s_i = t_{k_2} + p_{k_2} \sum_{i=1}^f s_i + p_A s_f$. \square

Corollary 4.2. *For any two sublots s_e and s_f in an optimal schedule, where $e < f$, if both s_e and s_f satisfy $\sum_{i=1}^e s_i \in (\sigma_{k-1}, \sigma_k]$ and $\sum_{i=1}^f s_i \in (\sigma_{k-1}, \sigma_k]$ for some machine k , then the following equation holds*

$$s_f = s_e (q_k)^{f-e}, \quad (4.18)$$

where $q_k = p_A/p_k$.

Proof. This can be shown by induction. Suppose $f = e + 1$. By Corollary 4.1, we have $t_k + p_k \sum_{i=1}^e s_i + p_A \sum_{i=e}^n s_i = t_k + p_k \sum_{i=1}^{e+1} s_i + p_A \sum_{i=e+1}^n s_i$, which leads to $s_f = s_e q_k$. Let $s_{e+l} = s_e \cdot (q_k)^l$, $l = 1, \dots, r$. Once again, using Corollary 4.1, we can show that $s_{e+r+1} = s_e \cdot q_k^{r+1}$. Hence, $s_f = s_e q_k^{f-e}$. \square

From above, in an optimal solution, the criticality of a subplot is associated with a subassembly machine. However, a subassembly machine may or may not have a critical subplot associated with it. In case, there exists a critical subplot with respect to a machine, we call that subplot a pattern-switching subplot. For the case when more than one sublots are critical with respect to a machine, then the last of these sublots is a pattern-switching subplot. We denote a pattern-switching subplot for machine k by ρ_k . Let there be r machines, k_1, k_2, \dots, k_r , each of which has a pattern-switching subplot associated with it, and let $D = \{k_1, k_2, \dots, k_r\}$ denote the set of these machines. Also let $W = \{\rho_{k_1}, \rho_{k_2}, \dots, \rho_{k_r}\}$ be the set of pattern-switching sublots associated with the machines in D . Note that, $\rho_{k_r} = n$ and $k_r = m$. This follows by the fact that $\sum_{i=1}^n s_i = U$, and hence, the last subplot is critical with respect to the subassembly machine m . In view of the definition of D , Corollaries 4.1 and 4.2 can be stated as the following property, which is also the optimality condition.

Property 4.1. *In an optimal schedule, the following equations hold:*

$$\sum_{i=1}^n s_i = U. \quad (4.19)$$

$$\sigma_{k_{d-1}} < \sum_{i=1}^e s_i \leq \sigma_{k_d}, \quad \forall k_{d-1}, k_d \in D, \forall \rho_{k_{d-1}}, \rho_{k_d} \in W, \forall e = \rho_{k_{d-1}} + 1, \dots, \rho_{k_d}, \quad (4.20)$$

$$t_{k_{d-1}} + p_{k_{d-1}} \sum_{i=1}^{\rho_{k_{d-1}}} s_i + p_A s_{\rho_{k_{d-1}}} = t_{k_d} + p_{k_d} \sum_{i=1}^{\rho_{k_{d-1}}+1} s_i, \quad \forall k_{d-1}, k_d \in D, \forall \rho_{k_{d-1}}, \rho_{k_d} \in W, \quad (4.21)$$

$$s_{i+1} = s_i q_{k_d}, \quad \forall k_d \in D, \forall \rho_{k_d} \in W, \forall i = \rho_{k_{d-1}} + 1, \dots, \rho_{k_d} - 1. \quad (4.22)$$

Property 4.2. *The above conditions (4.19), (4.20), (4.21) and (4.22) lead to the following inequality:*

$$q_{k_d} s_{\rho_{k_{d-1}}} \leq s_{\rho_{k_{d-1}}+1} \leq q_{k_{d-1}} s_{\rho_{k_{d-1}}}, \quad \forall k_{d-1}, k_d \in D, \forall \rho_{k_{d-1}}, \rho_{k_d} \in W. \quad (4.23)$$

We can show this as follows. According to (4.20), we have $t_{k_{d-1}} + p_{k_{d-1}}(S_{\rho_{k_{d-1}}} + s_{\rho_{k_{d-1}}+1}) \leq t_{k_d} + p_{k_d}(S_{\rho_{k_{d-1}}} + s_{\rho_{k_{d-1}}+1})$. Since $t_{k_d} + p_{k_d}(S_{\rho_{k_{d-1}}} + s_{\rho_{k_{d-1}}+1}) = t_{k_{d-1}} + p_{k_{d-1}}S_{\rho_{k_{d-1}}} + p_A s_{\rho_{k_{d-1}}}$ by (4.21), we have $t_{k_{d-1}} + p_{k_{d-1}}(S_{\rho_{k_{d-1}}} + s_{\rho_{k_{d-1}}+1}) \leq t_{k_{d-1}} + p_{k_{d-1}}S_{\rho_{k_{d-1}}} + p_A s_{\rho_{k_{d-1}}}$. Hence, $s_{\rho_{k_{d-1}}+1} \leq q_{k_{d-1}} s_{\rho_{k_{d-1}}}$. Similarly, due to (4.20), we have $t_{\rho_{k_{d-1}}} + p_{\rho_{k_{d-1}}}S_{\rho_{k_{d-1}}} + p_A s_{\rho_{k_{d-1}}} \leq t_{\rho_{k_{d-1}}} + p_{\rho_{k_{d-1}}}(S_{\rho_{k_{d-1}}} + s_{\rho_{k_{d-1}}+1})$, which leads to $q_{k_d} s_{\rho_{k_{d-1}}} \leq s_{\rho_{k_{d-1}}+1}$.

The above two properties are illustrated using an example depicted in Figure 4.7. In the example, a production lot is to be split into 7 sublots for processing (at stage 1) by four subassembly machines $M1$, $M2$, $M3$ and $M4$, which are ordered and indexed in the decreasing order of their setup times. We assume that, in an optimal solution shown in Figure 4.7, $M1$, $M3$ and $M4$ are in set D , and they are denoted by k_1 , k_2 and k_3 , respectively. The pattern-switching sublots for the machines in set D are $\rho_{k_1} = 3$, $\rho_{k_2} = 5$ and $\rho_{k_3} = 7$, respectively. Note that, subassembly machine $M2$ is not in set D by the fact that $\sum_{i=1}^{\rho_{k_2}+1} s_i (= \sum_{i=1}^4 s_i) > \sigma_2$. By Corollary 4.2, the following geometric relationships hold among follows:

$$\begin{aligned} s_{i+1} &= q_1 s_i, & \text{for } i = 1, 2, \\ s_{i+1} &= q_3 s_i, & \text{for } i = 4, \\ s_{i+1} &= q_4 s_i, & \text{for } i = 6, \end{aligned}$$

which illustrate the optimality condition (4.22). For pattern-switching sublots 3 and 5, by Corollary 4.1, we have

$$t_1 + p_1 \sum_{i=1}^3 s_i + p_A s_3 = t_3 + p_2 \sum_{i=1}^4 s_i,$$

$$t_3 + p_3 \sum_{i=1}^5 s_i + p_A s_5 = t_4 + p_4 \sum_{i=1}^6 s_i,$$

which illustrate the optimality condition (4.21). The optimal value of s_i , $i = 1, \dots, 7$ are determined using the above relationships and the fact that $\sum_{i=1}^7 s_i = U$. Also, note that the example shows $p_1 s_4 \leq p_A s_3 \leq p_3 s_4$ and $p_3 s_6 \leq p_A s_5 \leq p_4 s_6$. By re-arranging, we have

$$q_3 s_3 \leq s_4 \leq q_1 s_3,$$

$$q_4 s_5 \leq s_6 \leq q_3 s_5,$$

which follow (4.23).

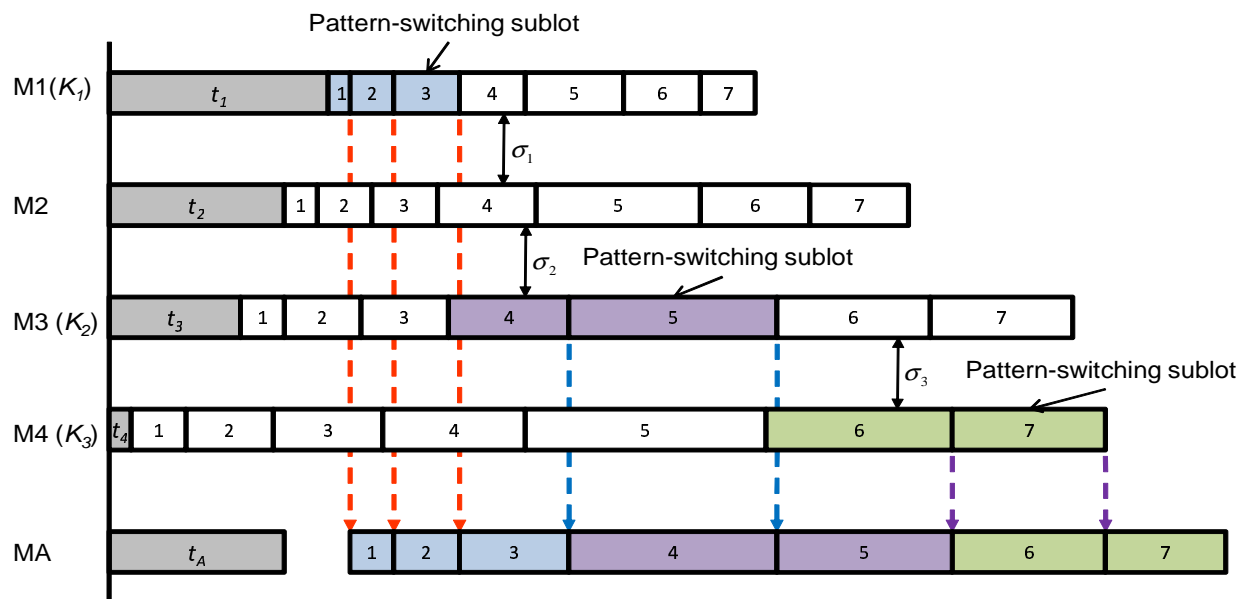


Figure 4.7: Illustration of optimality conditions

4.4 Algorithm for 2+1 Problem

In this section, we demonstrate the use of the optimality conditions developed above for solving the TSLSP for the case of $m = 2$. We denote this problem as the 2+1 problem, and assume that the two subassembly machines are indexed according to the decreasing order of their lot-detached setup times so that $0 < \sigma_1 < U$. From the previous section, the criticality of a subplot is associated with a subassembly machine. The number of machines, r , each of which has a pattern-switching subplot associated with it, satisfies $1 \leq r \leq 2$. We define subplot ρ_1 and ρ_2 as two pattern-switching sublots, if they exist, with respect to subassembly machines 1 and 2, respectively. Note that $\rho_2 = n$. We have the following two cases in accordance with (4.20) based on the location of ρ_1 .

Case 1: $r = 1$. In this case, we have $\rho_1 = 0$, that is, all the sublots are critical with respect to the subassembly machine 2. We shall have the following feasibility condition:

$$0 < \sigma_1 < S_i \leq U, \quad \forall i = 1, \dots, n. \quad (4.24)$$

Clearly, the optimal subplot sizes can be determined by using the geometric expressions developed in Trietsch (1987), as follows:

$$s_i = \begin{cases} \frac{(q_2)^{i-1} - (q_2)^i}{1 - (q_2)^n} U, \forall i = 1, \dots, n & \text{if } q_2 \neq 1, \\ \frac{U}{n}, & \text{otherwise.} \end{cases} \quad (4.25)$$

Case 2: $r = 2$. In this case, we have $1 \leq \rho_1 \leq n - 1$, that is, sublots 1 to ρ_1 are critical with respect to subassembly machine 1, and sublots $\rho_1 + 1$ to n are critical with respect to subassembly machine 2. We shall have the following feasibility condition:

$$0 < S_i \leq \sigma_1 < S_j \leq U, \quad \forall i = 1, \dots, \rho_1, \forall j = \rho_1 + 1, \dots, n. \quad (4.26)$$

Note that, for feasible solutions of the above two cases, we have

$$M_1 \leq M_2, \quad (4.27)$$

where M_1 and M_2 are the makespans obtained for Cases 1 and 2, respectively. This follows by the fact that an addition of a subassembly machine to the problem can only potentially increase the makespan. Therefore, we shall start our algorithm with Case 1. If the resulting solution from Case 1 satisfies the condition specified in (4.24), then the solution is optimal. Otherwise, we try Case 2, until we obtain a solution in which its corresponding feasibility condition (4.26) is satisfied.

Next, we will discuss a method for obtaining ρ_1 for Case 2. But first, we re-state the necessary optimality conditions of Property 4.1, for the $2 + 1$ problem.

Property 4.3. *In an optimal schedule for the $2 + 1$ problem, if $\rho_1 \geq 1$, the following relationships hold:*

$$\sum_{i=1}^n s_i = U. \quad (4.28)$$

$$0 < \sum_{i=1}^e s_i \leq \sigma_1 < \sum_{i=1}^f s_i \leq U, \quad \forall e = 1, \dots, \rho_1, \forall f = \rho_1 + 1, \dots, n, \quad (4.29)$$

$$t_1 + p_1 \sum_{i=1}^{\rho_1} s_i + p_A s_{\rho_1} = t_2 + p_2 \sum_{i=1}^{\rho_1+1} s_i, \quad (4.30)$$

$$s_{i+1} = s_i q_1, \quad \forall i = 1, \dots, \rho_1 - 1 \quad (4.31)$$

$$s_{i+1} = s_i q_2, \quad \forall i = \rho_1 + 1, \dots, n - 1 \quad (4.32)$$

Note that equations (4.28), (4.30), (4.31) and (4.32) involve $n + 1$ variables and n constraints. Therefore, by fixing ρ_1 , we can obtain a unique solution of subplot sizes. We also re-state Property 4.2 for the $2 + 1$ problem below.

Property 4.4. *The conditions (4.29), (4.30), (4.31) and (4.32) lead to the following inequality:*

$$q_2 s_{\rho_1} \leq s_{\rho_1+1} \leq q_1 s_{\rho_1}. \quad (4.33)$$

Next, we establish the sufficiency of these optimality conditions for the $2 + 1$ problem.

Proposition 4.5. *If there exists a ρ_1 , $1 \leq \rho_1 \leq n - 1$, for which the conditions (4.28), (4.29), (4.30), (4.31) and (4.32) hold, then the solution is optimal.*

Proof. We prove this proposition by contradiction. Denote by $Q = \{s_1, \dots, s_n\}$ a solution in which there is a pattern switching subplot ρ_1 so that (4.28), (4.29), (4.30), (4.31) and (4.32) are satisfied. Suppose there exists an optimal solution $Q' = \{s'_1, \dots, s'_n\}$ in which $\rho'_1 \neq \rho_1$ and $M(Q') < M(Q)$. Due to the criticality of the sublots 1 and n in both solutions, and the fact that $M(Q') < M(Q)$, we have

$$\begin{aligned} t_1 + p_1 s'_1 + p_A U &< t_1 + p_1 s_1 + p_A U \\ t_2 + p_2 U + p_A s'_n &< t_2 + p_2 U + p_A s_n \end{aligned}$$

which leads to $s'_1 < s_1$ and $s'_n < s_n$. Next, consider the following two cases: (i) $\rho'_1 < \rho_1$ and (ii) $\rho'_1 > \rho_1$.

Case (i): According to (4.31) and (4.32), we have $s'_i < s_i, \forall i = 1, \dots, \rho'_1$ and $s'_j < s_j, \forall j = \rho_1 + 1, \dots, n$. In addition, due to (4.33) for solution Q' , we have $s'_{\rho'+1} \leq q_1 s'_{\rho'}$. Because $s_{\rho'+1} = q_1 s_{\rho'}$ in Q (as $\rho'_1 < \rho_1$ by assumption), and $s'_{\rho'} < s_{\rho'}$ (from above), we have $s'_{\rho'+1} = q_1 s'_{\rho'} > q_1 s'_{\rho'_1} \geq s'_{\rho'_1+1}$, or $s'_{\rho'+1} < s_{\rho'+1}$. According to (4.32) for Q' and (4.31) for Q , we have $s'_{i+1} = q_2 s'_i, \forall i = \rho' + 1, \dots, \rho - 1$ and $s_{i+1} = q_1 s_i, \forall i = \rho' + 1, \dots, \rho - 1$. Because $q_2 < q_1$ and $s'_{\rho'+1} < s_{\rho'+1}$ (from above), we have $s'_i < s_i, \forall i = \rho'_1 + 1, \dots, \rho_1$, which leads to $\sum_{i=1}^n s'_i < \sum_{i=1}^n s_i = U$. This contradicts the feasibility of solution Q' .

Case (ii): According to (4.31) and (4.32), we have $s'_i < s_i, \forall i = 1, \dots, \rho_1$ and $s'_j < s_j, \forall j = \rho'_1 + 1, \dots, n$. In addition, due to (4.33) for solution Q' , we have $s'_{\rho'} \leq \frac{s'_{\rho'+1}}{q_2}$. Because $s_{\rho'} = \frac{s_{\rho'+1}}{q_2}$ in Q (as $\rho'_1 > \rho_1$ by assumption) and $s'_{\rho'+1} < s_{\rho'+1}$ (from above), we have $s_{\rho'_1} = \frac{s_{\rho'_1+1}}{q_2} > \frac{s'_{\rho'_1+1}}{q_2} \geq s'_{\rho'_1}$, or $s'_{\rho'} < s_{\rho'}$. According to (4.32) for Q' and (4.31) for Q , we have $s'_i = \frac{s'_{i+1}}{q_1}, \forall i = \rho + 1, \dots, \rho' - 1$ and $s_i = \frac{s_{i+1}}{q_2}, \forall i = \rho + 1, \dots, \rho' - 1$. Because $q_2 < q_1$ and $s'_{\rho'} < s_{\rho'}$ (from above), we have $s'_i < s_i, \forall i = \rho'_1 + 1, \dots, \rho_1$, which leads to $\sum_{i=1}^n s'_i < \sum_{i=1}^n s_i = U$. This contradicts the feasibility of solution Q' . \square

Next, we derive the close-form expressions for S_{ρ_1}, s_{ρ_1} and s_{ρ_1+1} . According to equations (4.28), (4.31) and (4.32), the subplot sizes s_{ρ_1} and s_{ρ_1+1} can be calculated in terms of S_{ρ_1} as

follows:

$$s_{\rho_1} = \begin{cases} \frac{(q_1)^{\rho_1-1} - (q_1)^{\rho_1}}{1 - (q_1)^{\rho_1}} S_{\rho_1}, & \text{if } q_1 \neq 1, \\ \frac{1}{\rho_1} S_{\rho_1}, & \text{otherwise.} \end{cases} \quad (4.34)$$

and,

$$s_{\rho_1+1} = \begin{cases} \frac{1 - q_2}{1 - (q_2)^{n-\rho_1}} (U - S_{\rho_1}), & \text{if } q_2 \neq 1, \\ \frac{1}{n - \rho_1} (U - S_{\rho_1}), & \text{otherwise.} \end{cases} \quad (4.35)$$

By substituting (4.34) and (4.35) into (4.30), $S_{\rho_1} (= \sum_{i=1}^{\rho_1} s_i)$ and ρ_1 can then be expressed by the following equations:

$$\begin{cases} t_1 + p_1 S_{\rho_1} + p_A \frac{(q_1)^{\rho_1-1} - (q_1)^{\rho_1}}{1 - (q_1)^{\rho_1}} S_{\rho_1} = t_2 + p_2 S_{\rho_1} + p_2 \frac{1 - q_2}{1 - (q_2)^{n-\rho_1}} (U - S_{\rho_1}), & \text{if } q_1 \neq 1, q_2 \neq 1 \\ t_1 + p_1 S_{\rho_1} + p_A \frac{(q_1)^{\rho_1-1} - (q_1)^{\rho_1}}{1 - (q_1)^{\rho_1}} S_{\rho_1} = t_2 + p_2 S_{\rho_1} + p_2 \frac{1}{n - \rho_1} (U - S_{\rho_1}), & \text{if } q_1 \neq 1, q_2 = 1 \\ t_1 + p_1 S_{\rho_1} + p_A \frac{1}{\rho_1} S_{\rho_1} = t_2 + p_2 S_{\rho_1} + p_2 \frac{1 - q_2}{1 - (q_2)^{n-\rho_1}} (U - S_{\rho_1}), & \text{if } q_1 = 1, q_2 \neq 1 \end{cases} \quad (4.36)$$

Note that it is not possible to have $q_1 = q_2 = 1$ since we can eliminate subassembly machine 2 in that case. By rearranging (4.36), we obtain the following expressions:

$$\begin{cases} S_{\rho_1} = \frac{p_2 \frac{1-q_2}{1-(q_2)^{n-\rho_1}} U - (t_1 - t_2)}{p_A \frac{(q_1)^{\rho_1-1} - (q_1)^{\rho_1}}{1 - (q_1)^{\rho_1}} + p_2 \frac{1-q_2}{1-(q_2)^{n-\rho_1}} - (p_2 - p_1)}, & \text{if } q_1 \neq 1, q_2 \neq 1 \\ S_{\rho_1} = \frac{p_2 \frac{1}{n-\rho_1} U - (t_1 - t_2)}{p_A \frac{(q_1)^{\rho_1-1} - (q_1)^{\rho_1}}{1 - (q_1)^{\rho_1}} + p_2 \frac{1}{n-\rho_1} - (p_2 - p_1)}, & \text{if } q_1 \neq 1, q_2 = 1 \\ S_{\rho_1} = \frac{p_2 \frac{1-q_2}{1-(q_2)^{n-\rho_1}} U - (t_1 - t_2)}{p_A \frac{1}{\rho_1} S_{\rho_1} + p_2 \frac{1-q_2}{1-(q_2)^{n-\rho_1}} - (p_2 - p_1)}, & \text{if } q_1 = 1, q_2 \neq 1 \end{cases} \quad (4.37)$$

Therefore, we can search for the value of ρ_1 and obtain the values of S_{ρ_1} , s_{ρ_1} and s_{ρ_1+1} . We then check whether (4.29) is satisfied. An algorithm to find an optimal solution is described next.

4.4.1 Algorithm 2+1

This is a polynomial-time algorithm of complexity $O(n)$, and it determines optimal subplot sizes for the 2 + 1 problem.

Step 1: Let $\rho_1 = 0$, calculate $S_i, \forall i = 1, \dots, n$ using geometric relationship described in (4.25). If (4.24) is satisfied, stop. Otherwise, let $\rho_1 = 1$.

Step 2: Calculate S_{ρ_1} using (4.37), and then, s_{ρ_1+1} using (4.35). If (4.29) is satisfied, that is, $0 < S_{\rho_1} \leq \sigma_1 < S_{\rho_1} + s_{\rho_1+1} < U$, then stop; otherwise, go to Step 3.

Step 3: Let $\rho_1 = \rho_1 + 1$, and go to Step 2.

Figure 4.8 provides a flow-chart for the above algorithm.

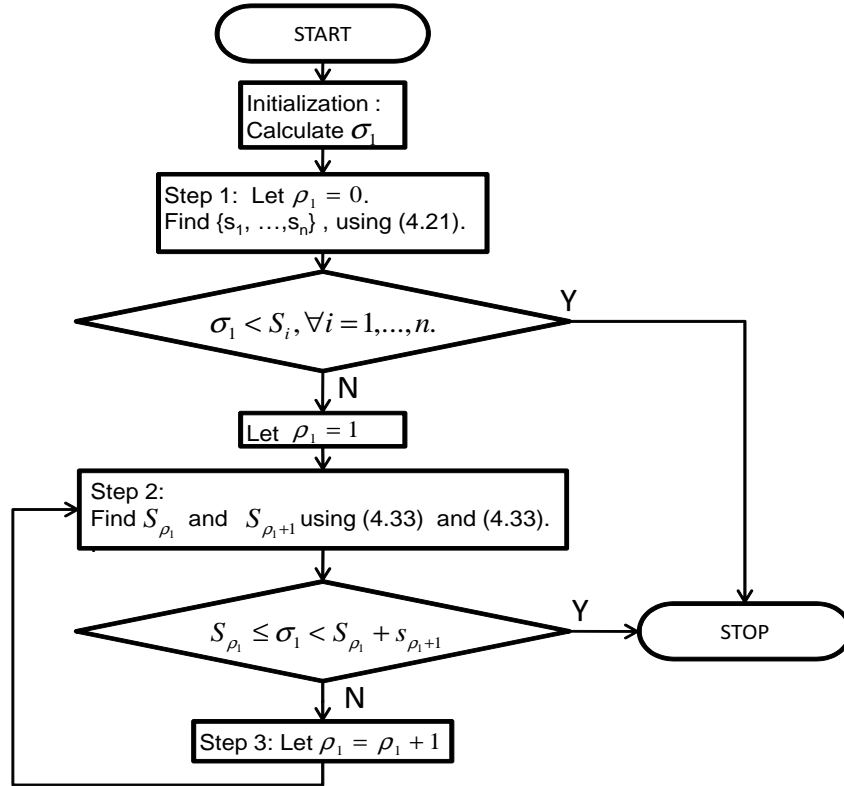


Figure 4.8: Flow-chart for Algorithm 2 + 1

4.4.2 Example 1

Consider the data shown in Table 4.1 for a 2 + 1 problem.

Table 4.1: Data for a 2+1 problem

Production lot & number of sublots:	U=120, n=6
Setup times:	$t_1 = 100, t_2 = 30, t_A = 10$
Unit processing times:	$p_1 = 2, p_2 = 3, p_A = 2.5$

We have,

$$\sigma_1 = \frac{t_1 - t_2}{p_2 - p_1} = \frac{100 - 30}{3 - 2} = 70.$$

We begin with step 1. Let $\rho_1 = 0$, we calculate the subplot size based on the geometric relationship between the subassembly machine 2 and the assembly machine. We have,

$$s_1 = \frac{1 - q_2}{1 - (q_2)^n} U = \frac{1 - 2.5/3}{1 - (2.5/3)^6} 120 = 30.0757,$$

which does not satisfy (4.24) (since $S_1(= s_1 = 30.0757) < \sigma_1(= 70)$). Therefore, let $\rho_1 = n - 1 = (6 - 1) = 5$. By step 2, we compute,

$$\begin{aligned} S_{\rho_1} = S_5 &= \frac{p_2 \frac{1-q_2}{1-(q_2)^{n-\rho_1}} U - (t_1 - t_2)}{p_A \frac{(q_1)^{\rho_1-1} - (q_1)^{\rho_1}}{1-(q_1)^{\rho_1}} + p_2 \frac{1-q_2}{1-(q_2)^{n-\rho_1}} - (p_2 - p_1)} \\ &= \frac{3 \frac{1-2.5/3}{1-(2.5/3)^1} 120 - 70}{2.5 \frac{(2.5/2)^4 - (2.5/2)^5}{1-(2.5/2)^5} + 3 \frac{1-2.5/3}{1-(2.5/3)^1} - 1} \\ &= \frac{290}{2.725} = 106.422. \end{aligned}$$

Apparently, (4.29) is not satisfied since $S_{\rho_1} > \sigma_1$. By step 3, we reduce ρ_1 , and we have $\rho_1 = 4$. We repeat step 2 to obtain

$$\begin{aligned} S_{\rho_1} = S_4 &= \frac{3 \frac{1-2.5/3}{1-(2.5/3)^2} 120 - 70}{2.5 \frac{(2.5/2)^3 - (2.5/2)^4}{1-(2.5/2)^4} + 3 \frac{1-2.5/3}{1-(2.5/3)^2} - 1} \\ &= \frac{126.3636}{1.3138} = 85.20, \end{aligned}$$

which again violates (4.29). Once again, we reduce ρ_1 . We have $\rho_1 = 3$, and then, we calculate the following,

$$\begin{aligned} S_{\rho_1} = S_3 &= \frac{3 \frac{1-2.5/3}{1-(2.5/3)^3} 120 - 70}{2.5 \frac{(2.5/2)^2 - (2.5/2)^3}{1-(2.5/2)^3} + 3 \frac{1-2.5/3}{1-(2.5/3)^3} - 1} \\ &= \frac{72.4176}{1.24} = 59.78, \end{aligned}$$

which satisfies $S_{\rho_1} \leq \sigma_1$ in (4.29). We then determine,

$$\begin{aligned} s_{\rho_1+1} = s_4 &= \frac{1 - q_2}{1 - (q_2)^{n-\rho_1}} (U - S_{\rho_1}) \\ &= \frac{1 - 2.5/3}{1 - (2.5/3)^3} (120 - 59.78) \\ &= 23.82. \end{aligned}$$

Furthermore, $S_{\rho_1} + s_{\rho_1+1} > \sigma_1$ in (4.29) is satisfied. Hence, we find an optimal solution. The subplot sizes in the optimal solution are

$$\begin{aligned} s_1 &= \frac{1 - q_1}{1 - (q_1)^3} \cdot 59.78 = 15.68 \\ s_2 &= q_1 s_1 = 19.6 \\ s_3 &= q_1 s_2 = 24.5 \\ s_4 &= 23.82 \\ s_5 &= q_2 s_4 = 19.85 \\ s_6 &= q_2 s_5 = 16.54. \end{aligned}$$

The makespan,

$$M = t_1 + p_1 S_3 + p_3 (U - S_3) = 30 + 3 * 120 + 2.5 * 16.54 = 431.36.$$

The schedule (including the completion time of each subplot) of the above solution to the example is illustrated in Figure 4.9.

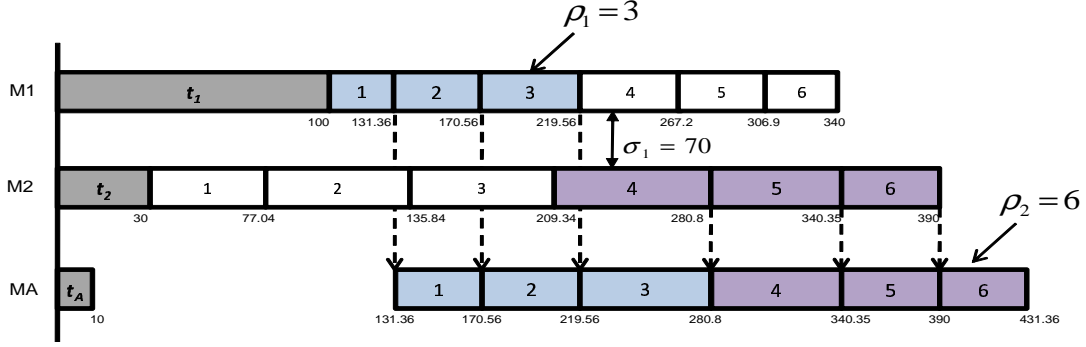


Figure 4.9: Illustration of the optimal solution to Example 1

4.5 Algorithm for 3+1 Problem

In this section, we study the TSLSP for $m = 3$, and designate it as 3 + 1 problem. Assume the subassembly machines have been indexed according to the decreasing order of their lot-detached setup times so that we have $0 < \sigma_1 < \sigma_2 < U$. As shown in Section 4.3, the criticality of a subplot is associated with a subassembly machine. The number of machines, r , each of which has a pattern-switching subplot associated with it, satisfies $1 \leq r \leq 3$. We define sublots ρ_1 , ρ_2 and ρ_3 as three pattern-switching sublots, if they exist. Note that $\rho_3 = n$. We have the following four cases in accordance with (4.20), based on the locations of ρ_1 and ρ_2 .

Case 1: $r = 1$. In this case, we have $\rho_1 = 0$ and $\rho_2 = 0$. Then, all the sublots are critical with respect to the subassembly machine 3. We shall have the following feasibility condition:

$$0 < \sigma_1 < \sigma_2 < S_i \leq U, \quad \forall i = 1, \dots, n \quad (4.38)$$

Clearly, the optimal subplot sizes can be determined by using the following geometric expressions,

$$s_i = \begin{cases} \frac{(q_3)^{i-1} - (q_3)^i}{1 - (q_3)^n} U, \forall i = 1, \dots, n & \text{if } q_3 \neq 1, \\ \frac{U}{n}, & \text{otherwise.} \end{cases} \quad (4.39)$$

Case 2: $r = 2$, $1 \leq \rho_1 \leq n - 1$ and $\rho_2 = 0$. Then, sublots 1 to ρ_1 are critical with respect to the subassembly machine 1, and sublots $\rho_1 + 1$ to n are critical with respect to

the subassembly machine 3. We shall have the following feasibility condition:

$$0 < S_i \leq \sigma_1 < \sigma_2 < S_j \leq U, \quad \forall i = 1, \dots, \rho_1, \forall j = \rho_1 + 1, \dots, n. \quad (4.40)$$

Case 3: $r = 2$, $1 \leq \rho_2 \leq n - 1$ and $\rho_1 = 0$. Then, sublots 1 to ρ_2 are critical with respect to the subassembly machine 2, and sublots $\rho_2 + 1$ to n are critical with respect to the subassembly machine 3. We shall have the following feasibility condition:

$$0 < \sigma_1 < S_i \leq \sigma_2 < S_j \leq U, \quad \forall i = 1, \dots, \rho_2, \forall j = \rho_2 + 1, \dots, n. \quad (4.41)$$

We can use Algorithm 2 + 1 to solve for Cases 2 and 3.

Case 4: $r = 3$. In this case, we have $1 \leq \rho_1 < \rho_2 \leq n - 1$. Then, sublots 1 to ρ_1 are critical with respect to the subassembly machine 1, sublots $\rho_1 + 1$ to ρ_2 are critical with respect to the subassembly machine 2, and sublots $\rho_2 + 1$ to n are critical with respect to the subassembly machine 3. We shall have the following feasibility condition:

$$0 < S_i \leq \sigma_1 < S_j \leq \sigma_2 < S_l \leq U, \quad \forall i = 1, \dots, \rho_1, \forall j = \rho_1 + 1, \dots, \rho_2, \forall l = \rho_2 + 1, \dots, n \quad (4.42)$$

Note that, for feasible solutions to all of the above four cases, we have,

$$M_1 \leq M_2 < M_3 \leq M_4, \quad (4.43)$$

where M_1 , M_2 , M_3 and M_4 are the makespan values obtained for the four cases, respectively. The relationship $M_1 \leq M_2 \leq M_4$ or $M_1 \leq M_3 \leq M_4$ follows by the fact that an addition of a subassembly machine to the problem can only potentially increase the makespan. To show the relationship $M_2 < M_3$, suppose s_1^2 and s_1^3 are the first sublots in the solution for Case 2 and Case 3, respectively. We have $s_1^2 \leq \sigma_1 < s_1^3$ according to (4.40) and (4.41). Furthermore,

$$\begin{aligned} M_2 &= t_1 + p_1 s_1^2 + p_A U \leq t_1 + p_1 \sigma_1 + p_A U, \\ M_3 &= t_2 + p_2 s_1^3 + p_A U > t_2 + p_2 \sigma_1 + p_A U. \end{aligned}$$

Due to $t_1 + p_1 \sigma_1 + p_A U = t_2 + p_2 \sigma_1 + p_A U$ by definition of σ_1 , we have $M_2 < M_3$.

Therefore, we shall start our algorithm for Cases 1, 2, 3, 4 in that order until we obtain a solution with its corresponding optimality conditions (4.19), (4.20), (4.21) and (4.22) satisfied. For Case 4, we shall select a solution with the minimum makespan, among those solutions that satisfy the optimality conditions.

Next, we discuss the method to find ρ_1 and ρ_2 for Case 4. The necessary optimality conditions in Property 4.1 for the 3 + 1 problem are re-stated as follows.

Property 4.5. *In an optimal schedule for the 3+1 problem, if $1 \leq \rho_1 < \rho_2 \leq n - 1$, the following relationships hold:*

$$\sum_{i=1}^n s_i = U, \quad (4.44)$$

$$0 < \sum_{i=1}^e s_i \leq \sigma_1 < \sum_{i=1}^f s_i \leq \sigma_2 < \sum_{i=1}^g s_i \leq U, \quad \forall e = 1, \dots, \rho_1, \forall f = \rho_1 + 1, \dots, \rho_2, \quad (4.45)$$

$$\forall g = \rho_2 + 1, \dots, n,$$

$$t_1 + p_1 \sum_{i=1}^{\rho_1} s_i + p_A s_{\rho_1} = t_2 + p_2 \sum_{i=1}^{\rho_1+1} s_i, \quad (4.46)$$

$$t_2 + p_2 \sum_{i=1}^{\rho_2} s_i + p_A s_{\rho_2} = t_3 + p_3 \sum_{i=1}^{\rho_2+1} s_i, \quad (4.47)$$

$$s_{i+1} = s_i q_1, \quad \forall i = 1, \dots, \rho_1 - 1, \quad (4.48)$$

$$s_{i+1} = s_i q_2, \quad \forall i = \rho_1 + 1, \dots, \rho_2 - 1, \quad (4.49)$$

$$s_{i+1} = s_i q_3, \quad \forall i = \rho_2 + 1, \dots, n. \quad (4.50)$$

Note that equations (4.44), (4.46), (4.47), (4.48), (4.49) and (4.50) involve $n + 2$ variables and n constraints. Therefore, if ρ_1 and ρ_2 are given, we can find a unique solution.

According to relationships (4.44), (4.46), (4.47), (4.48), (4.49) and (4.50), s_{ρ_1} , s_{ρ_1+1} , s_{ρ_2} and s_{ρ_2+1} can be expressed in terms of S_{ρ_1} and S_{ρ_2} as follows:

$$s_{\rho_1} = AS_{\rho_1}, \quad (4.51)$$

$$s_{\rho_1+1} = BS_{\rho_2} - S_{\rho_1}, \quad (4.52)$$

$$s_{\rho_2} = CS_{\rho_2} - S_{\rho_1}, \quad (4.53)$$

$$s_{\rho_2+1} = D(U - S_{\rho_2}) \quad (4.54)$$

where A, B, C, D are represented by

$$A = \begin{cases} \frac{(q_1)^{\rho_1-1} - (q_1)^{\rho_1}}{1 - (q_1)^{\rho_1}} & \text{if } q_1 \neq 1, \\ \frac{1}{\rho_1}, & \text{otherwise.} \end{cases}$$

$$B = \begin{cases} \frac{1 - q_2}{1 - (q_2)^{\rho_2-\rho_1}} & \text{if } q_2 \neq 1, \\ \frac{1}{\rho_2 - \rho_1}, & \text{otherwise.} \end{cases}$$

$$C = \begin{cases} \frac{(q_2)^{\rho_2-\rho_1-1} - (q_2)^{\rho_2-\rho_1}}{1 - (q_2)^{\rho_2-\rho_1}} & \text{if } q_2 \neq 1, \\ \frac{1}{\rho_2 - \rho_1}, & \text{otherwise.} \end{cases}$$

$$D = \begin{cases} \frac{1 - q_3}{1 - (q_3)^{n-\rho_2}} & \text{if } q_3 \neq 1, \\ \frac{1}{n - \rho_2}, & \text{otherwise.} \end{cases}$$

By substituting (4.51) and (4.52) into (4.46), and (4.53) and (4.54) into (4.47), S_{ρ_1} and S_{ρ_2} can then be expressed by the following equations:

$$p_2 \cdot B \cdot (S_{\rho_2} - S_{\rho_1}) = p_A \cdot A \cdot S_{\rho_1} + [(t_1 - t_2) - (p_2 - p_1)S_{\rho_1}], \quad (4.55)$$

$$p_3 \cdot D \cdot (U - S_{\rho_2}) = p_A \cdot C \cdot (S_{\rho_2} - S_{\rho_1}) + [(t_2 - t_3) - (p_3 - p_2)(S_{\rho_2} - S_{\rho_1})]. \quad (4.56)$$

Therefore, we can search for the values of ρ_1 and ρ_2 and obtain $s_{\rho_1}, s_{\rho_1+1}, s_{\rho_2}, s_{\rho_2+1}, S_{\rho_1}$ and S_{ρ_2} such that (4.42) is satisfied. This is formally stated in the following algorithm.

4.5.1 Algorithm 3+1

This is a polynomial-time algorithm of complexity $O(n^2)$, and it determines optimal subplot sizes for the 3 + 1 problem.

Initialization: Let $\Psi = \emptyset$. Let $Q(\rho_1, \rho_2)$ and $M(Q)$ denote a solution associated with ρ_1 and ρ_2 and its corresponding makespan, respectively. Calculate σ_1, σ_2 and σ_{13} , where $\sigma_{13} = \frac{t_1 - t_3}{p_3 - p_1}$.

Step 1: (Determine subplot sizes by assuming that only machine 3 is dominant). Let $\rho_1 = \rho_2 = 0$, and determine subplot sizes $\{s_1, \dots, s_n\}$ using (4.39). If $0 < \sigma_1 < \sigma_2 < S_i \leq U, \forall i = 1, \dots, n$, stop; if $0 < \sigma_{13} < S_i, \forall i = 1, \dots, n$, then it is an optimal solution of the 2 + 1 problem involving machines 1 and 3 (see Case 1, Section 4.4), go to Step 3; otherwise go to Step 2.

Step 2: (Determine subplot sizes by assuming only machines 1 and 3 are dominant). Let $\rho_2 = 0$ and $1 \leq \rho_1 \leq n - 1$, and determine $\{s_1, \dots, s_n\}$ using Algorithm 2 + 1. If $0 < S_i \leq \sigma_1 < \sigma_2 < S_j \leq U, \forall i = 1, \dots, \rho_1, \forall j = \rho_1 + 1, \dots, n$, in (4.40) is satisfied, stop; otherwise, go to Step 3.

Step 3: (Determine subplot sizes by assuming only machines 2 and 3 are dominant). Let $\rho_1 = 0$ and $1 \leq \rho_2 \leq n - 1$, and determine $\{s_1, \dots, s_n\}$ using Algorithm 2 + 1. If $0 < \sigma_1 < S_i \leq \sigma_2 < S_j \leq U, \forall i = 1, \dots, \rho_2, \forall j = \rho_2 + 1, \dots, n$, in (4.41) is satisfied, stop; otherwise, set $\rho_1 = 1$ and $\rho_2 = 2$, and go to Step 4.

Step 4: (Determine subplot sizes by assuming machines 1, 2 and 3 are dominant). Calculate S_{ρ_1} and S_{ρ_2} using (4.55) and (4.56), and then $s_{\rho_1}, s_{\rho_1+1}, s_{\rho_2}, s_{\rho_2+1}$ using (4.51), (4.52), (4.53) and (4.54), respectively. If $0 < S_i \leq \sigma_1 < S_j \leq \sigma_2 < S_l \leq U, \forall i = 1, \dots, \rho_1, \forall j = \rho_1 + 1, \dots, \rho_2, \forall l = \rho_2 + 1, \dots, n$, in (4.42) is satisfied, add solution $Q(\rho_1, \rho_2)$ to set Ψ . In any case, go to Step 5.

Step 5: If $\rho_1 = n - 2$ and $\rho_2 = n - 1$, go to Step 6; otherwise, we have two possibilities, case (i): $\rho_1 < n - 2$ and $\rho_2 = n - 1$; case (ii): $\rho_1 < \rho_2 < n - 1$. For case (i), let $\rho_1 = \rho_1 + 1$ and $\rho_2 = \rho_1 + 1$ and go to Step 4. For case (ii), let $\rho_2 = \rho_2 + 1$ and go to Step 4.

Step 6: Select a solution Q from the set Ψ such that $M(Q)$ is minimal. Stop.

Figure 4.10 provides a flow-chart for the above algorithm.

4.5.2 Example 2

Consider the data shown in Table 4.2 for a 3 + 1 problem.

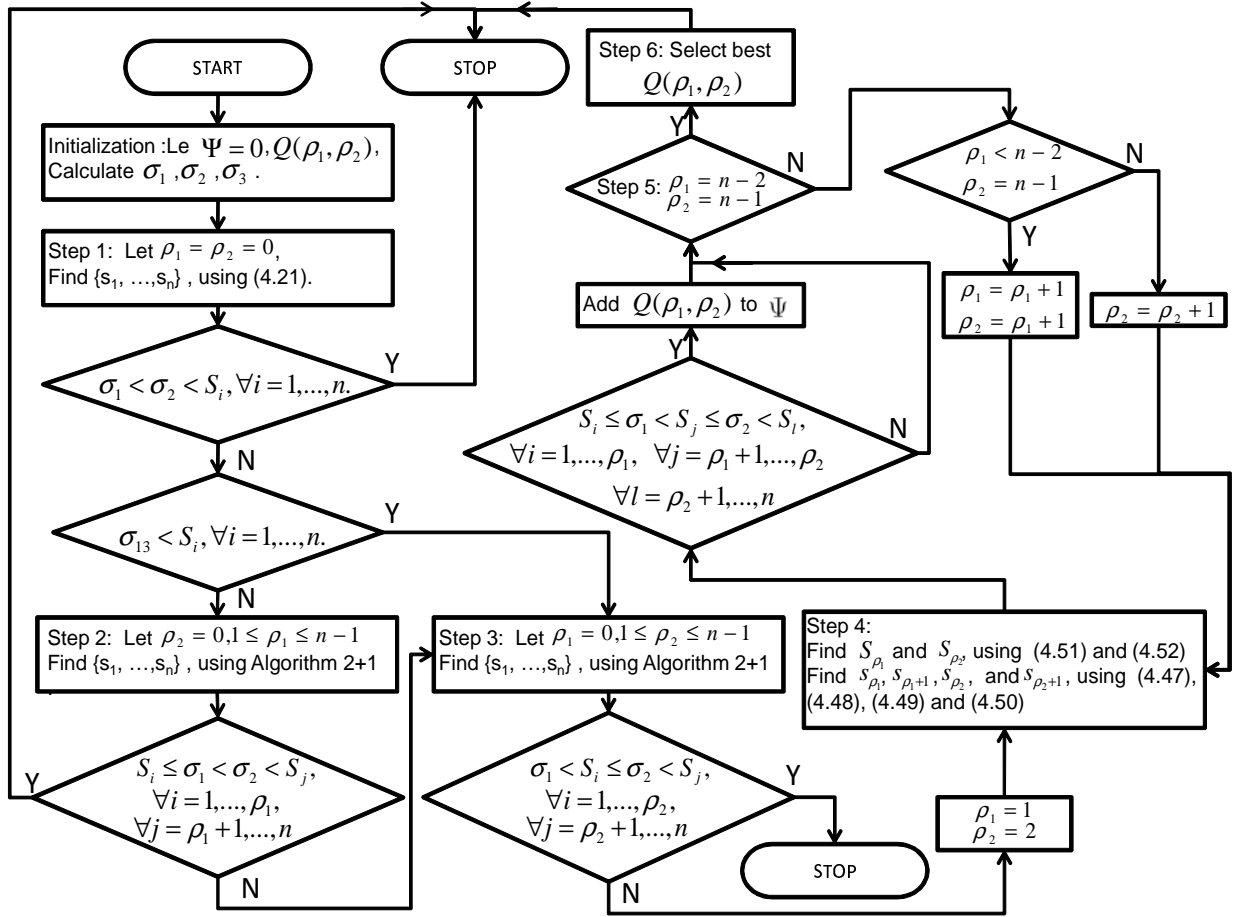


Figure 4.10: Flow-chart for Algorithm 3 + 1

Table 4.2: Data for a 3+1 problem

Production lot & number of sublots	U=120	n=8		
Setup times	$t_1 = 100$	$t_2 = 80$	$t_3 = 30$	$t_A = 10$
Unit processing times	$p_1 = 2$	$p_2 = 3$	$p_3 = 4$	$p_A = 2.5$

We have

$$\sigma_1 = \frac{t_1 - t_2}{p_2 - p_1} = \frac{100 - 80}{3 - 2} = 20,$$

$$\sigma_2 = \frac{t_2 - t_3}{p_3 - p_2} = \frac{80 - 30}{4 - 3} = 50,$$

$$\sigma_{13} = \frac{t_1 - t_3}{p_3 - p_1} = \frac{100 - 30}{4 - 2} = 35.$$

We begin with Step 1. Let $\rho_1 = \rho_2 = 0$, we calculate the subplot size based on the geometric relationship between the subassembly machine 3 and the assembly machine. We have

$$\begin{aligned} s_1 &= 46.0727, & s_2 &= 28.7954, & s_3 &= 17.9972, & s_4 &= 11.2482, \\ s_5 &= 7.0301, & s_6 &= 4.3939, & s_7 &= 2.7462, & s_8 &= 1.7163, \end{aligned}$$

which does not satisfy $0 < \sigma_1 < \sigma_2 < S_1 \leq U$ since $S_1 (= s_1 = 46.0727) < \sigma_2$. However, $\sigma_{13} < s_1$ holds. Therefore, we go to Step 3 and consider $\rho_1 = 0$ and $1 \leq \rho_2 \leq n - 1$. Using Algorithm 2 + 1, we obtain that when $\rho_2 = 1$, the optimality conditions hold true. We have

$$\begin{aligned} S_1 &= \frac{p_3 \frac{1-q_3}{1-(q_3)^{n-\rho_2}} U - (t_2 - t_3)}{p_A \frac{(q_2)^{\rho_2-1} - (q_2)^{\rho_2}}{1-(q_2)^{\rho_2}} + p_3 \frac{1-q_3}{1-(q_3)^{n-\rho_1}} - (p_3 - p_2)} \\ &= \frac{4 \frac{1-2.5/3}{1-(2.5/3)^7} 120 - 50}{2.5 \frac{(2.5/2)^0 - (2.5/2)^1}{1-(2.5/2)^1} + 4 \frac{1-2.5/4}{1-(2.5/4)^7} - 1} \\ &= \frac{136.96}{3.058} = 44.7885, \\ s_2 &= \frac{1 - q_3}{1 - (q_3)^7} (120 - 44.7885) \\ &= 29.2957, \end{aligned}$$

which is feasible to $0 < \sigma_1 < S_i \leq \sigma_2 < S_j \leq U$ in (4.41). We, then, stop and calculate other subplot sizes. They are:

$$\begin{aligned} s_1 &= 44.7885, & s_2 &= 29.2957, & s_3 &= q_3 s_2 = 18.3098, & s_4 &= q_3 s_3 = 11.4436, \\ s_5 &= q_3 s_4 = 7.15226, & s_6 &= q_3 s_5 = 4.47017, & s_7 &= q_3 s_6 = 2.79385, & s_8 &= q_3 s_7 = 1.74616. \end{aligned}$$

And, the makespan value is

$$M = t_3 + p_3 U + p_4 s_8 = 30 + 4 * 120 + 2.5 * 1.74616 = 514.365.$$

The schedule (including the completion time of each subplot) for the above example is illustrated in Figure 4.11.

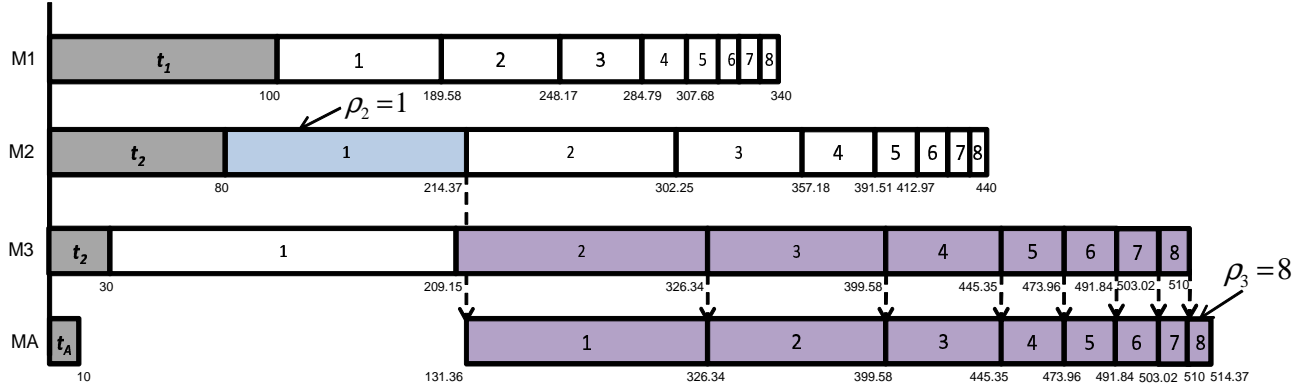


Figure 4.11: Illustration of the optimal solution to Example 2

4.6 Integer-size Sublots

To determine integer subplot sizes, we can extend the method to that end proposed by Trietsch (1987) for a two-machine flow shop lot streaming problem. To achieve feasibility, we must complete subplot i on m subassembly machines no later than the latest start time for that subplot on the assembly machine A . That is,

$$t_k + p_k S_i \leq M - p_A(U - S_{i-1}), \quad \forall k = 1, \dots, m \quad (4.57)$$

or

$$S_i \leq \min \left\{ \min_{1 \leq k \leq m} \left\{ \frac{M - t_k - p_A(U - S_{i-1})}{p_k} \right\}, U \right\} \quad (4.58)$$

The S_i values are the largest integer values permitted by the inequality (4.58). To calculate it, let

$$S'_i = \min_{1 \leq k \leq m} \left\{ \frac{M - t_k - p_A(U - S_{i-1})}{p_k} \right\}. \quad (4.59)$$

We have

$$S_i = \min \left\{ \lfloor S'_i \rfloor, U \right\}. \quad (4.60)$$

By rounding down S_i , $\forall i = 1, \dots, n$, we may get $S_n < U$. In that case, the makespan M does not accommodate all the items and needs to be incremented. We need to select a subplot to round up its size to the nearest integer. Let

$$S_i'' = \min \left\{ \lceil S_i' \rceil, U \right\}. \quad (4.61)$$

Let Δ_i represent the time difference incurred by rounding S_i' up to its nearest integer, and it is given by

$$\Delta_i = \max_{1 \leq k \leq m} \{t_k + p_k S_i''\} - \max_{1 \leq k \leq m} \{t_k + p_k S_i'\}. \quad (4.62)$$

The new makespan value, \hat{M} , can then be incremented by a minimum amount of Δ_i as follows,

$$\hat{M} = M + \min_{1 \leq i \leq n} \{\Delta_i\}. \quad (4.63)$$

We, then, repeat the process until we obtain $S_n = U$. The corresponding algorithm to determine optimal, integer subplot sizes is given as follows.

4.6.1 Algorithm -Integer Sublot Sizes

Step 1: Let $S_0 = 0$, $i = 1$.

Step 2: For $i = 1, \dots, n$. Calculate S_i' , S_i , S_i'' , and Δ_i for each i using (4.59), (4.60), (4.61) and (4.62), respectively.

Step 3: If $S_n = U$, stop. Otherwise, find a new value of M by using (4.63). Go to Step 2.

An example is used to illustrate the above procedure in the next section.

4.6.2 Example 3

Consider the data depicted in Table 4.2. We have already found the fractional optimal solution, which is shown as follows:

$$s_1 = 45.7886, \quad s_2 = 29.2957, \quad s_3 = 18.3098, \quad s_4 = 11.4436,$$

$$s_5 = 7.1523, \quad s_6 = 4.4702, \quad s_7 = 2.7939, \quad s_8 = 1.7462.$$

Iteration 1: We first calculate S'_i , S''_i and S_i and Δ_i , as shown in Table 4.3.

Table 4.3: Iteration 1

i	S'_i	S''_i	S_i	Δ_i
1	$\min\{57.183, 44.788, 46.091\} = 44.788$	45	44	$215 - 214.365 = 0.635$
2	$\min\{112.183, 81.445, 73.591\} = 73.591$	74	73	$326 - 324.465 = 1.635$
3	$\min\{148.433, 105.622, 91.916\} = 91.716$	92	91	$398 - 396.865 = 1.135$
4	$\min\{170.933, 120.622, 102.966\} = 102.966$	103	102	$442 - 441.865 = 0.135$
5	$\min\{184.683, 129.788, 109.841\} = 109.841$	110	109	$470 - 469.365 = 0.635$
6	$\min\{193.433, 135.622, 114.216\} = 114.216$	115	114	$490 - 486.865 = 3.135$
7	$\min\{199.683, 139.788, 117.341\} = 117.341$	118	117	$502 - 499.365 = 2.635$
8	$\min\{203.433, 142.288, 119.216\} = 119.216$	120	119	$510 - 506.865 = 3.135$

Since $S_8 (= 119) < U$, we increase the makespan as $\hat{M} = M + \min_{1 \leq i \leq n} \{\Delta_i\} = 514.365 + 0.135 = 514.5$.

Iteration 2: Similarly, we find S'_i , S''_i and S_i and Δ_i using the new makespan obtained in Iteration 1 as shown in Table 4.4.

Table 4.4: Iteration 2

i	S'_i	S''_i	S_i	Δ_i
1	$\min\{57.75, 45.167, 46.375\} = 45.167$	46	45	$218 - 215.5 = 2.5$
2	$\min\{114, 82.667, 74.5\} = 74.5$	75	74	$330 - 328 = 2$
3	$\min\{150.25, 106.833, 92.625\} = 92.625$	93	92	$402 - 400.5 = 1.5$
4	$\min\{172.75, 121.833, 103.875\} = 103.875$	104	103	$446 - 445.5 = 0.5$
5	$\min\{186.5, 131, 110.75\} = 110.75$	111	110	$474 - 473 = 1$
6	$\min\{195.25, 136.833, 115.125\} = 115.125$	116	115	$494 - 490.5 = 3.5$
7	$\min\{201.5, 141, 118.25\} = 118.25$	119	118	$506 - 503 = 3$
8	$\min\{205.25, 143.5, 120.125\} = 120.125$	121	120	$514 - 510.5 = 3.5$

Since $S_8 = U$, stop. The integer subplot sizes can be calculated as

$$\begin{aligned} s_1 = S_1 - S_0 = 45, & \quad s_2 = S_2 - S_1 = 29, & \quad s_3 = S_3 - S_2 = 18, & \quad s_4 = S_4 - S_3 = 11, \\ s_5 = S_5 - S_4 = 7, & \quad s_6 = S_6 - S_5 = 5, & \quad s_7 = S_7 - S_6 = 3, & \quad s_8 = S_8 - S_7 = 2. \end{aligned}$$

And, the makespan is 514.5.

4.7 Concluding Remarks

In this chapter, we have discussed the lot streaming problem for a single lot that is to be processed in a two-stage assembly system. This system is different from the flow shop machine configuration typically considered in the literature for lot streaming problems. We have presented some machine dominance properties, which result in reducing the problem size. The inherent structural properties of the problem are then used in obtaining an optimality condition, which affords development of an effective algorithm for its solution. We present polynomial-time algorithms for the $2 + 1$ and $3 + 1$ problems. A polynomial-time algorithm has also been presented to obtain integer-size subplot.

An important finding in this chapter has been the identification of multiple pattern-switching sublots that constitute an optimal solution. Due to an interaction between lot-detached setup time and cumulative processing time for each subassembly machine, multiple subassembly machines may dominate the makespan at different times in the processing of the sublots on them. In other words, the criticality of sublots changes from one machine to another depending upon the number of items that have been processed.

Chapter 5

Multiple-Lot Lot Streaming in a Two-stage Assembly System

In this chapter, we address a lot streaming problem in a two-stage assembly system involving multiple lots, where each lot represents a unique product type, for the objective of minimizing the makespan. We call this problem as a two-stage multiple-lot, lot streaming problem (TSMLSP). The problem contains two decisions, namely, lot splitting and lot sequencing. The interaction between these two decisions makes the problem a difficult one to analyze. Some basic results derived for the TSMLSP using the properties of the two-stage single-lot lot streaming problem are presented. For solution methodology, a branch-and-bound-based methodology is developed that relies on effective lower bounds and dominance properties, which are also established. The next step is to perform computational experimentation to determine efficacy of the branch-and-bound-based methodology for our problem.

5.1 Introduction

Lot streaming is the process of using transfer batches to move completed portions of a product lot to downstream machines so that their operations can be undertaken in an overlapping fashion. Kalir and Sarin (2000) have shown that the potential benefits of lot streaming with

respect to three commonly-used performance measures, namely, makespan, mean flow time and average work-in-process. Our work in this chapter extends the two-stage single-lot, lot streaming problem (TSLSP) presented in Chapter 4 to the multiple-lot case. In the presence of multiple lots, we need to simultaneously consider sizing of the sublots and sequencing of the lots. Baker (1995) and Centinkaya and Kayaligil (1992) have shown that unit-sized sublots are optimal for the problem with no setup time, and they have solved the resulting sequencing problem using a modification of Johnson's algorithm (see Johnson (1954)). For the case with lot-detached setup times and subplot transfer times, Vickson (1995) has shown that the subplot sizing and lot sequencing problems are independent. They have derived optimal subplot sizes for the subplot sizing problem, and have solved the lot sequencing problem using Johnson's algorithm (see Johnson (1954)). For the case with lot-detached setup and removal times, Centinkaya (1994) has also shown that the subplot sizing problem and sequencing problems are independent, and furthermore, the optimal subplot sizes are geometric. A sequence of lots is determined using a modification of Johnson's algorithm based on run-in and run-out times (see Johnson (1954)). Sriskandarajah and Wagneur (1999) have addressed the multiple-lot, lot streaming problem in a no-wait two-machine flow shop, and they have proved that the subplot sizing and lot sequencing problems are independent. The optimal continuous subplot sizes are geometric in this case as well and the optimal sequence can be obtained using an algorithm proposed by Gilmore and Gomory (1964). Kalir and Sarin (2003) have considered a problem with subplot-attached setups. They have presented solution procedures for the case when subplot sizes for all the lots are the same and for the case when the subplot sizes are unequal. For the former case, their algorithm iterates over all possible values of subplot size, and it sequences the lots using a modified Johnson's algorithm. For the latter case, they have proposed a two-phase procedure in which the construction phase determines the sequence using a modified Johnson's algorithm and the improvement phase re-optimizes subplot sizes based on the sequence obtained. The iteration continues until no improvement can be made.

Scheduling problem in the two-stage assembly system that we consider in this chapter has been considered by Lee et al. (1993), Hariri and Potts (1997) and Sun et al. (2003).

However, they do not include the streaming of the lots. Lee et al. (1993) have studied a 3-machine assembly scheduling problem. They have shown that their problem is strongly NP-hard, and have identified the special cases of the problem that are solvable in polynomial time. In addition, they have presented several heuristics and their respective worst-case bounds. Hariri and Potts (1997) have extended the problem to a machine configuration involving arbitrary number of subassembly machines. They have developed a branch-and-bound algorithm for the problem, and have provided the computational experience with the use of this method. Sun et al. (2003) have considered a 3-machine assembly scheduling problem and have presented several heuristics to address the worst-case scenarios presented in the literature.

Our work in this chapter is different from the work presented in the literature in that we consider a lot streaming problem for processing multiple lots in a two-stage assembly system. The configuration of this assembly system is illustrated in Figure 5.1. The first stage of this system consists of multiple, parallel machines where subassemblies are prepared with one subassembly-type on each machine for each production lot. These subassemblies, are then, assembled into final products at the second stage. An example for such an assembly system in which there are two production lots of 50 and 40 items is shown in Figure 5.1. Lot-detached setup is incurred on every machine at both stages. For the example, these values are assumed to be 40, 40 and 60 units for lot 1 on the subassembly machines, and 10 units on the assembly machine. Similar number for lot 2 are 30, 20 and 30 units for setups on the subassembly machines, and 60 units on the assembly machine. The processing times for lot 1 are 1.5, 1 and 1 units per item on the subassembly machines and 1 unit on the assembly machine. For lot 2, the processing times are 1, 2 and 0.5 units per item on the subassembly machines and 2 units on the assembly machine. Figure 5.1(a) depicts schedule 1 in which the processing of lot 1 precedes that of lot 2. The subplot sizes used for lot 1 are 10 and 40, and the subplot sizes used for lot 2 are 10, 10 and 20. Schedule 2 is shown in Figure 5.1(b) in which the processing sequence of the lots is altered, while the subplot sizes used for both the lots are as in Schedule 1. Due to a change in the sequence in which the lots

are processed, the makespan reduces from 295 to 230. Note that (see Figure 5.1(c)), if we change the subplot sizes for lot 1 to 20 and 30, and keep the sequence to be lot 1 followed by lot 2, the makespan further reduces from 230 to 220. Our aim is to determine the sequence of the lots and subplot sizes of each lot so as to minimize makespan.

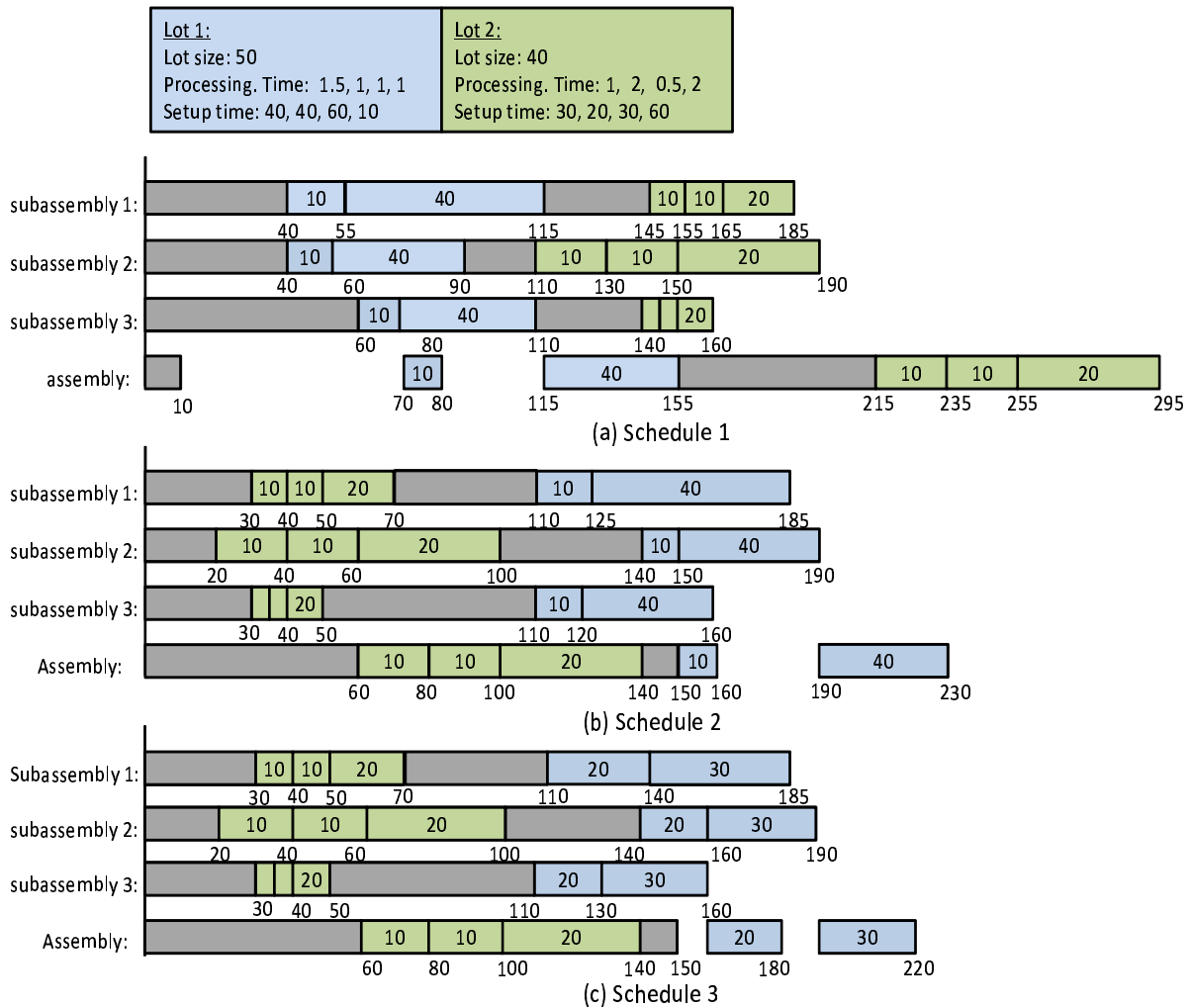


Figure 5.1: Example depicting streaming of multiple lots in a two-stage assembly system

This chapter is organized as follows. In Section 5.2, we describe the TSMLSP and present its useful properties. A mixed integer programming formulation for the TSMLSP is presented in Section 5.3. In Section 5.4, we propose a branch-and-bound methodology, which includes

the development of various lower and upper bounds, and dominance rules based on the optimality conditions for the two-stage single-lot, lot streaming problem to help in curtailing nodes in the branch-and-bound tree. Our next step is to perform computational test to demonstrate the efficacy of the proposed branch-and-bound algorithm. Some concluding remarks are made in Section 5.6.

5.2 Problem Description and Basic Properties

The TSMLSP can formally be described as follows. There are N production lots to be processed in a two-stage assembly system. Each lot j consists of U_j items. There is a set, Ω , of M subassembly machines at the first stage and an assembly machine at the second stage. The per unit processing time for the items of a lot can vary over the machines, and is designated by p_{jk} , for lot j on machine k . Similarly, the unit processing time is different for different lots on the assembly machine, and is designated by p_{jA} for lot j . Lot-detached setup times are incurred before each lot j starts its processing on subassembly machine k , $k \in \Omega$, and the assembly machine A , and they are denoted by t_{jk} , $k \in \Omega$ and t_{jA} , respectively. We assume that all the machines (these at stage 1 and stage 2) in the problem use the same number of sublots, n_j , for lot j , $j = 1, \dots, N$. In addition, we make the following assumptions: (i) all the machines are available at time zero; (ii) sublot sizes are continuous; (iii) the processing of a sublot on machine A can be started only after sufficient number of its components have finished processing at the first stage; (iv) sublot intermingling is not allowed, that is, once a machine starts processing a lot, it has to finish all the items in that lot before beginning to process the next lot. The objective is to determine: (1) the sequence in which to process the production lots; and (2) the sublot sizes of each lot j on the subassembly and assembly machines so as to minimize the makespan, that is, the completion time of the last sublot of the last lot on the assembly machine A . We use the following notation.

Parameters:

N - Number of production lots.

- M - Number of subassembly machines.
- U_j - Number of items in lot j , $j = 1, \dots, N$.
- n_j - Number of sublots of lot j , $j = 1, \dots, N$.
- t_{jk} - Detached-setup time of lot j on subassembly machine k , $k = 1, \dots, M$.
- t_{jA} - Detached-setup time of lot j on the assembly machine.
- p_{jk} - Unit processing time of lot j on subassembly machine k , $k = 1, \dots, M$.
- p_{jA} - Unit processing time of lot j on the assembly machine.

Variables:

- x_{ijk} = 1, if lot j is sequenced in position i of the sequence on subassembly machine k , and 0, otherwise.
- x_{ijA} = 1, if lot j is sequenced in position i of the sequence on assembly machine A , and 0, otherwise.
- s_{juk} - Size of subplot u of lot i on subassembly machine k .
- s_{juA} - Size of subplot u of lot i on assembly machine A .
- C_{ik} - Completion time of a lot on subassembly machine k if it is sequenced in position i .
- C_{iA} - Completion time of a lot on the assembly machine A if it is sequenced in position i .

Before presenting a mathematical formulation for the TSMLSP, we first show the following properties, which help in curtailing the type of sequence and the subplot sizes that we need to consider.

Property 5.1. *There exists an optimal schedule in which the sequences of lots are the same on all the machines.*

Let π_k be a sequence on subassembly machine k , and π_A be a different sequence on assembly machine A . It is easy to see that we can alter the the sequence π_k on each subassembly machine k to conform to π_A without worsening the makespan. Consequently, we can drop the subscripts “ k ” and “ A ” from the notation of the sequencing variables, namely, x_{ijk} and x_{ijA} .

Property 5.2. *For a given sequence of lots, there exist optimal subplot sizes for each lot such that the completion time of each lot is minimized.*

Since there is no idle time in between the processing of the production lots on each subassembly machine, the minimization of makespan for TSMLSP can be considered as the minimization of total idle time on the assembly machine A . For a given sequence of lots, minimization of the total idle time on the assembly machine A is equivalent to minimizing the completion time of each lot in the sequence. Consequently, the result from TSLSP (see Chapter 4) regarding the nature of the sublots is still valid. Hence, we have the following property.

Property 5.3. *There exists an optimal schedule in which each lot is split into consistent sublots.*

Due to this property, we can drop the subscripts “ k ” and “ A ” from the notation of subplot sizes, namely s_{iek} and s_{ieA} . This property also establishes the fact that if a permutation sequence of lots is given, the optimality conditions (Property (4.1) of Chapter 4) developed for the TSLSP, will be valid for the TSMLSP as well. To accommodate these properties in the TSMLSP, we will re-state them in Section 5.4.

5.3 A Mixed Integer Programming Formulation

Our formulation of a model for the **TSMLSP** is as follows:

TSMLSP-MIP:

$$\text{Minimize} \quad C_{NA} \quad (5.1a)$$

$$\text{subject to} \quad \sum_{i=1}^N x_{ij} = 1, \quad \forall j = 1, \dots, N, \quad (5.1b)$$

$$\sum_{j=1}^N x_{ij} = 1, \quad \forall i = 1, \dots, N, \quad (5.1c)$$

$$C_{ik} \geq C_{i-1,k} + \sum_{j=1}^N (t_{jk} + p_{jk}U_j)x_{ij}, \quad \forall i = 1 \dots, N, \forall k = 1, \dots, M, \quad (5.1d)$$

$$C_{iA} \geq C_{i-1,A} + \sum_{j=1}^N (t_{jA} + p_{jA}U_j)x_{ij}, \quad \forall i = 1 \dots, N, \quad (5.1e)$$

$$C_{iA} + (1 - x_{ij}) \left((p_{jk} + p_{jA})U_j \right) \geq C_{i-1,k} \\ + t_{jk}x_{ij} + p_{jk} \sum_{u=1}^e s_{ju} + p_{jA} \sum_{u=e}^{n_j} s_{ju}, \quad \forall i, j = 1 \dots, N, \forall e = 1, \dots, n_j, \\ \forall k = 1, \dots, M, \quad (5.1f)$$

$$\sum_{u=1}^{n_j} s_{ju} = U_j, \quad \forall j = 1, \dots, N, \quad (5.1g)$$

$$s_{ju} \geq 0, \quad \forall j = 1, \dots, N, \forall u = 1, \dots, n_j, \quad (5.1h)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j = 1, \dots, N. \quad (5.1i)$$

Constraints (5.1b) and (5.1c) are the assignment constraints that ensure that each lot is assigned to a position and each position is allocated to only one lot, respectively, in a permutation of lots. Constraints (5.1d) and (5.1e) ensure that a machine (at stage 1 and stage 2) can process only a single production lot at a time. Constraints (5.1f) ensure that the completion time of each lot j , if assigned to position i on the assembly machine, can be no less than the completion time of each of its sublots on assembly machine A . Specifically, if $x_{ij} = 1$, we have

$$C_{iA} \geq C_{i-1,k} + t_{jk} + p_{jk} \sum_{u=1}^e s_{ju} + p_{jA} \sum_{u=e}^{n_j} s_{ju},$$

which enforces that the completion time of lot j in position i on assembly machine A is greater than or equal to the total time required to complete lot j , taking into consideration its processing on all subassembly machines k , $k = 1, \dots, M$. If $x_{ij} = 0$, we have

$$C_{iA} + (p_{jk} + p_{jA})U_j \geq C_{i-1,k} + p_{jk} \sum_{u=1}^e s_{ju} + p_{jA} \sum_{u=e}^{n_j} s_{ju}.$$

Since $(p_{jk} + p_{jA})U_j \geq p_{jk} \sum_{u=1}^e s_{ju} + p_{jA} \sum_{u=e}^{n_j} s_{ju}$, for any $e = 1, \dots, n_j$, the above constraint is redundant. Constraints (5.1g) ensure that the sum of the subplot sizes of lot j is equal to

the number of items in lot j . Constraints (5.1h) represent the non-negativity of subplot sizes, and Constraints (5.1i) represent the binary restriction of the assignment variables.

5.4 A Branch-and-Bound-based Methodology for the TSMLSP

In this section, we propose a branch-and-bound-based methodology for the TSMLSP. We first present the mathematical expression of makespan, which is used in the sequel.

5.4.1 Expression of Makespan

We first give an expression of the makespan for a given sequence of lots that relies on the properties stated in Section 5.2. Given a feasible schedule π , let $\pi(u)$, $u = 1, \dots, N$, denote the lot located at position u in solution π . Then, the makespan can be defined by

$$M(\pi) = \max \left\{ \max_{1 \leq k \leq M} \left\{ \max_{1 \leq u \leq N} \left\{ \max_{1 \leq e \leq n_{\pi(u)}} \{C_{uek}(\pi)\} \right\} \right\}, \sum_{j=1}^N (t_{jA} + p_{jA}U_j) \right\}, \quad (5.2)$$

where $C_{uek}(\pi)$ is the completion time of subplot e of the lot in position u on machine k , and is given as follows:

$$C_{uek}(\pi) = \sum_{v=1}^{u-1} \left(t_{\pi(v)k} + p_{\pi(v)k}U_{\pi(v)} \right) + \left(t_{\pi(u)k} + \sum_{w=1}^e p_{\pi(u)k}S_{\pi(u)w} + \sum_{w=e}^{n_{\pi(u)}} p_{\pi(u)A}S_{\pi(u)w} \right) + \sum_{v=u+1}^N \left(t_{\pi(v)A} + p_{\pi(v)A}U_{\pi(v)} \right). \quad (5.3)$$

From the definition of makespan, we have the following inequality

$$M(\pi) \geq C_{uek}(\pi), \quad \forall k = 1, \dots, M, \quad \forall u = 1, \dots, N, \quad \forall e = 1, \dots, n_{\pi(u)}. \quad (5.4)$$

A critical subplot is defined as a subplot for which the equality holds in (5.4) for some machine k , while a critical lot is a lot to which the critical subplot belongs. Note that, if the equality holds for some u , e and k , we call that lot, sequenced in position u , and its subplot e to be

critical with respect to machine k . Also, if a lot is critical, all its sublots are critical based on the criticality of sublots in TSLSP. For instance, if a production lot at position c in solution π is critical with respect to machine k , we have

$$M(\pi) = C_{cek} = \sum_{v=1}^{c-1} \left(t_{\pi(v)k} + p_{\pi(v)k} U_{\pi(v)} \right) + \left(t_{\pi(c)k} + \sum_{u=1}^e p_{\pi(c)k} s_{\pi(c)u} + \sum_{u=e}^{n_{\pi(c)}} p_{\pi(c)A} s_{\pi(c)u} \right) + \sum_{v=c+1}^N \left(t_{\pi(v)A} + p_{\pi(v)A} U_{\pi(v)} \right), \quad \forall e = 1, \dots, n_{\pi(c)}. \quad (5.5)$$

Also note that, when the makespan, $M(\pi) = \sum_{j=1}^N (t_{jA} + p_{jA} U_j)$, then there will be no critical sublots and lots.

Let τ_i denote such a partial sequence containing a set of lots that have been scheduled upto position i in the permutation, and τ'_i denote the set of lots yet to be scheduled. In the branch-and-bound tree (see Figure 5.2), a node at level i represents a subproblem in which a partial sequence τ_i has been fixed, and the remaining sequence needs to be determined among the lots in set τ'_i in order to minimize the makespan. We denote such a subproblem by $P_{\tau_i}^i$.

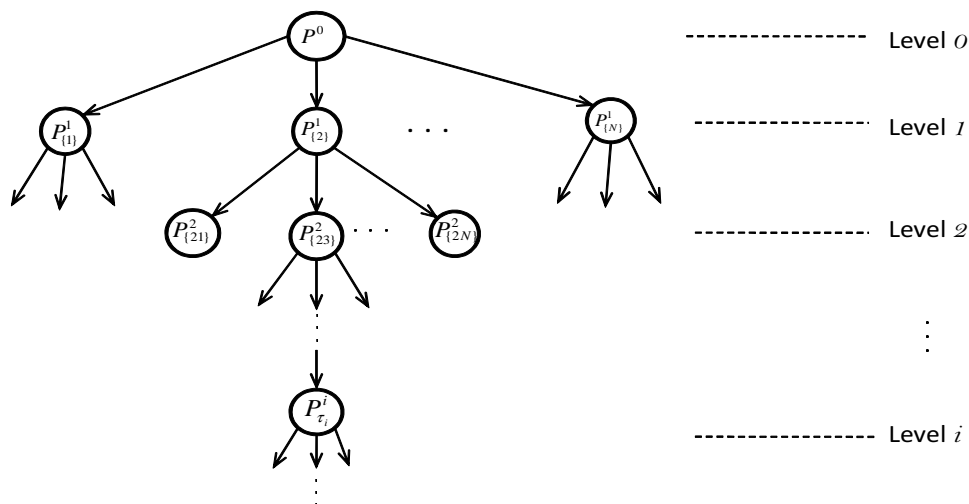


Figure 5.2: The branch-and-bound tree for the TSMLSP

Let $C_k(\tau_i)$ and $C_A(\tau_i)$ denote the completion times of the lot at position i on the sub-assembly machine k and the assembly machine A , respectively. At a level i of the branch-and-bound tree, each node j at level i of this tree corresponds to a partial permutation τ_i in which lots have been sequenced in the first i positions. Let τ'_i be the set of remaining lots. If π is a complete sequence built from τ_i as a sequence of lots in its first i positions, then the makespan of such a sequence is given by

$$M(\pi) = \max \left\{ \max_{1 \leq k \leq M} \left\{ \max_{i+1 \leq u \leq N} \left\{ \max_{1 \leq e \leq n_{\pi(u)}} \{C_{uek}(\pi)\} \right\} \right\}, C_A(\tau_i) + \sum_{j \in \tau'_i} (t_{jA} + p_{jA}U_j) \right\}, \quad (5.6)$$

where C_{uek} for any position u after i is described as follows:

$$C_{uek}(\pi) = C_k(\tau_i) + \sum_{v=i+1}^{u-1} \left(t_{\pi(v)k} + p_{\pi(v)k}U_{\pi(v)} \right) + \left(t_{\pi(u)k} + \sum_{w=1}^e p_{\pi(u)k}S_{\pi(u)w} + \sum_{w=e}^{n_{\pi(u)}} p_{\pi(u)A}S_{\pi(u)w} \right) + \sum_{v=u+1}^N \left(t_{\pi(v)A} + p_{\pi(v)A}U_{\pi(v)} \right). \quad (5.7)$$

Note that $M(\pi) \geq C_{uek}(\pi)$, $\forall k = 1, \dots, M, \forall u = 1, \dots, n, \forall e = 1, \dots, \pi(u)$. In case, the equality in the above expression holds for a lot at position c with respect to some machine k , then that lot $\pi(c)$ and its sublots are critical.

5.4.2 Determination of Lower and Upper Bounds

Let $C_A^0(\tau_i, j)$ be the completion time of lot j on machine A if it is sequenced in position $i+1$, assuming that lot j is processed immediately on machine A after the largest among its completion times on subassembly machines. Similarly, where $C_A^0(\pi - \{j\}, j)$ is the completion time of lot j if it is sequenced in the last position assuming lot j is processed immediately on machine A after the largest among its completion times on subassembly machines. We

have the following lower bounds.

$$\text{Lower Bound 1:} \quad LB^1(\tau_i) = C_A(\tau_i) + \sum_{j \in \tau'_i} (t_{jA} + p_{jA}U_j), \quad (5.8)$$

$$\text{Lower Bound 2:} \quad LB^2(\tau_i) = \min_{j \in \tau'_i} \left\{ C_A^0(\tau_i, j) + \sum_{\substack{u \in \tau'_i, \\ u \neq j}} (t_{uA} + p_{uA}U_u) \right\}, \quad (5.9)$$

$$\text{Lower Bound 3:} \quad LB^3(\tau_i) = \min_{j \in \tau'_i} \left\{ C_A^0(\pi - \{j\}, j) \right\}, \quad (5.10)$$

Note that $LB^1(\tau_i)$, $LB^2(\tau_i)$ and $LB^3(\tau_i)$ are machine-based lower bounds. Next, we present a lower bound based on a relaxed problem of the original problem. Let $\pi_k^*(\tau'_i)$ denote an optimal sequence of the lots in set τ'_i for a two-machine system that contains the subassembly machine k and the assembly machine A . Let $C_k(\pi_k^*(\tau'_i))$ denote the optimal makespan obtained for this relaxed problem. We have the following lower bound

$$\text{Lower Bound 4:} \quad LB^4(\tau_i) = \max_{1 \leq k \leq M} \left\{ C_k(\tau_i) + C_k(\pi_k^*(\tau'_i)) \right\}. \quad (5.11)$$

We can use the modified Johnson's algorithm on two-machine problem containing machine k and machine A to obtain $\pi_k^*(\tau'_i)$ and $C_k(\pi_k^*(\tau'_i))$ as follows. First, we state the following property (see Centinkaya (1994)).

Property 5.4. *The optimal subplot sizes for a lot $j \in \tau'_i$ in $\pi_k^*(\tau'_i)$ can be obtained by solving the single lot streaming problem in a two-machine flow shop containing machine k and machine A , irrespective of the lot sequence.*

Therefore, the subplot sizes s_{juk}^* for each lot j in solution $\pi_k^*(\tau'_i)$ are geometric, and can be obtained as follows:

$$s_{juk}^* = \begin{cases} \frac{(q_k)^{u-1} - (q_k)^u}{1 - (q_k)^{n_j}}, & \text{if } q_k \neq 1, \\ \frac{U}{n_j}, & \text{otherwise.} \end{cases} \quad \forall k = 1, \dots, M, \forall j \in \tau'_i, \forall u = 1, \dots, n_j, \quad (5.12)$$

where $q_{jk} = \frac{p_{jA}}{p_{jk}}$. With the subplot sizes determined, we then calculate run-in and run-out

times for each lot as follows:

$$RI_{jk} = \max \left\{ 0, t_{jk} + p_{jk} s_{j1k}^* - t_{jA} \right\}, \quad \forall j \in \tau'_i, \forall k = 1, \dots, M \quad (5.13)$$

$$RO_{jA} = \max \left\{ p_{jA} s_{j,n_j,k}^*, (t_{jA} - t_{jk}) + U_j(p_{jA} - p_{jk}) \right\}, \quad \forall j \in \tau'_i, \forall k = 1, \dots, M \quad (5.14)$$

The modified Johnson's rule based on the concept of run-in and run-out times, which is similar to the algorithm proposed in Centinkaya (1994), is as follows:

Proposition 5.1. *In a two-machine flow shop with machines k and A , lot u precede lot v in solution $\pi_k^*(\tau'_i)$ if the following is true*

$$\min\{RI_{uk}, RO_{vA}\} \leq \min\{RI_{vk}, RO_{uA}\}, \quad \forall u, v \in \tau'_i \quad (5.15)$$

Next, we develop an upper bound which can be obtained at each node. Given a partial sequence τ_i , an upper bound can be computed by

$$UB(\tau_i) = \min_{1 \leq k \leq M} \left\{ C \left(\tau_i + \pi_k^*(\tau'_i) \right) \right\}, \quad (5.16)$$

where $\pi_k^*(\tau'_i)$ is a sequence obtained by using the modified Johnson's rule on a two machine problem (containing subassembly machine k and the assembly machine) over the set τ'_i . Note that $C \left(\tau_i + \pi_k^*(\tau'_i) \right)$ can be obtained by solving a linear programming model for the permutation sequence $\tau_i + \pi_k^*(\tau'_i)$.

5.4.3 Development of Dominance Rules

In this section, we present some dominance rules that afford reduction in branching of the branch-and-bound tree. But first, we derive some necessary results from the TSLSP.

From Properties 5.2 and 5.3, we have the fact that if a permutation of lots is given, then the problem can be decomposed into N subproblems and each subproblem is a TSLSP problem. Therefore, the optimality conditions developed for the TSLSP are still valid for the TSMLSP. Given a partial permutation τ_i and set τ'_i containing the remaining lots, we consider the problem of assigning a lot to position $i + 1$. For any lot $j \in \tau'$, the criticality of

a subplot of lot j is associated with respect to a subassembly machine (see Proposition 4.4). Let $D(\tau_i, j) = \{k_1, k_2, \dots, k_r\}$ be a set of dominant machines, each of which has a pattern-switching subplot associated with it; and let $W(\tau_i, j) = \{\rho_{jk_1}, \rho_{jk_2}, \dots, \rho_{jk_r}\}$ be the sequence of the pattern-switching sublots in lot j . Suppose all of the subassembly machines are ordered in their nondecreasing values of unit processing time p_{jk} for lot j , and re-indexed from 1 to M . For the sake of convenience, we now restate the optimality condition in Property 4.1 as follows:

Property 5.5. *Given a partial permutation τ_i , there exists an optimal subplot-size solution for a lot $j \in \tau'$ at position i in which the following conditions hold:*

$$\sum_{u=1}^{n_j} s_{ju} = U_j, \quad (5.17)$$

$$\sigma_{jk_{d-1}} < \sum_{u=1}^e s_{ju} \leq \sigma_{jk_d}, \forall k_{d-1}, k_d \in D(\tau_i, j), \forall \rho_{jk_{d-1}}, \rho_{jk_d} \in W(\tau_i, j), \forall e = \rho_{k_{d-1}} + 1, \dots, \rho_{jk_d}, \quad (5.18)$$

$$\left(C_{k_{d-1}}(\tau_i) + t_{jk_{d-1}} \right) + p_{jk_{d-1}} \sum_{u=1}^{\rho_{jk_{d-1}}} s_{ju} + p_{jA} s_{\rho_{jk_{d-1}}} = \left(C_{k_d}(\tau_i) + t_{k_d} \right) + p_{jk_d} \sum_{u=1}^{\rho_{jk_{d-1}}+1} s_{ju}, \quad \forall k_{d-1}, k_d \in D(\tau_i, j), \forall \rho_{k_{d-1}}, \rho_{k_d} \in W(\tau_i, j), \quad (5.19)$$

$$s_{j,u+1} = s_{ju} q_{jk_d}, \quad \forall k_d \in D(\tau_i, j), \forall \rho_{j,k_d} \in W(\tau_i, j), \forall u = \rho_{jk_{d-1}} + 1, \dots, \rho_{jk_d} - 1. \quad (5.20)$$

Note that we treat $\left(C_{i-1,k_{d-1}} + t_{jk_{d-1}} \right)$ and $\left(C_{i-1,k_d} + t_{k_d} \right)$ as the lot-detached setup time for lot j to be scheduled in position i . Similarly, we also have the following property.

Property 5.6. *Given a partial permutation τ_i , the above conditions (5.18), (5.19) and (5.20) lead to the following inequality for lot j :*

$$q_{jk_d} s_{\rho_{jk_{d-1}}} \leq s_{j\rho_{jk_{d-1}}+1} \leq q_{jk_{d-1}} s_{j\rho_{jk_{d-1}}}, \forall k_{d-1}, k_d \in D(\tau_i, j), \forall \rho_{k_{d-1}}, \rho_{k_d} \in W(\tau_i, j). \quad (5.21)$$

5.4.3.1 Properties for the First and the Last Sublots

Next, we find lower and upper bounds of the first and last sublots of a lot by solving problem P_j^k , which designates a reduced (relaxed) problem of splitting a single lot j to minimize the

makespan for a two-machine flow shop consisting of subassembly machine k and the assembly machine A , in the absence of lot-detached setups. Problem P_j^k can be easily solved, and its optimal subplot sizes are geometric in nature as shown by Trietsch (1987) and Potts and Baker (1989). Hence, for problem P_j^k , the sizes of the first subplot and the last subplot for a given lot j for machine k and A can be represented by

$$s_{j1}^k = \begin{cases} \frac{1 - q_{jk}}{1 - (q_{jk})^{n_j}} U_j, & \text{if } q_k \neq 1, \\ \frac{U_j}{n_j}, & \text{otherwise.} \end{cases} \quad \forall k = 1, \dots, M, \forall j = 1, \dots, N, \quad (5.22)$$

and,

$$s_{jn_j}^k = \begin{cases} \frac{(q_{jk})^{n_j-1} - (q_{jk})^{n_j-1}}{1 - (q_{jk})^{n_j-1}} U_j, & \text{if } q_k \neq 1, \\ \frac{U}{n_j}, & \text{otherwise.} \end{cases} \quad \forall k = 1, \dots, M, \forall j = 1, \dots, N. \quad (5.23)$$

Without loss of generality, suppose all of the subassembly machines are ordered in their nondecreasing order of unit processing times p_{jk} , $k = 1, \dots, M$, for lot j , and re-indexed from 1 to M . We have

$$s_{j1}^1 \leq s_{j1}^2 \leq \dots \leq s_{j1}^k \leq \dots \leq s_{j1}^M, \quad \text{and} \quad (5.24)$$

$$s_{jn_j}^1 \geq s_{jn_j}^2 \geq \dots \geq s_{jn_j}^k \geq \dots \geq s_{jn_j}^M, \quad (5.25)$$

since $q_{j1} \geq q_{j2} \geq \dots \geq q_{jm}$ (see (5.22) and (5.23)). Therefore, define lb_{j1} , ub_{j1} , lb_{jn_j} and ub_{jn_j} as follows:

$$lb_{j1} = \min_{1 \leq k \leq M} \{s_{j1}^k\} = s_{j1}^1, \quad (5.26)$$

$$ub_{j1} = \max_{1 \leq k \leq M} \{s_{j1}^k\} = s_{j1}^M, \quad (5.27)$$

$$lb_{jn_j} = \min_{1 \leq k \leq M} \{s_{jn_j}^k\} = s_{jn_j}^M, \quad (5.28)$$

$$ub_{jn_j} = \max_{1 \leq k \leq M} \{s_{jn_j}^k\} = s_{jn_j}^1. \quad (5.29)$$

Next, we will show the expressions given above are, in fact, the lower and upper bounds of the first and last subplot sizes of a lot as conjectured. To that end, we have the following

result with respect to the first and the last sublots s_{j1} and s_{jn_j} in an optimal solution to the original problem.

Proposition 5.2. *Given a partial permutation τ_i , if lot j is assigned to position $i + 1$, then there exists an optimal subplot-size solution for lot j in which the following inequalities are satisfied:*

$$lb_{j1} \leq s_{j1} \leq ub_{j1}, \quad \text{and} \quad lb_{jn_j} \leq s_{jn_j} \leq ub_{jn_j}, \quad \forall j = 1, \dots, N \quad (5.30)$$

Proof. We prove this result by contradiction. Let $Q_j^1 = \{s_{j1}^1, \dots, s_{jn_j}^1\}$ and $Q_j^M = \{s_{j1}^M, \dots, s_{jn_j}^M\}$ be two optimal solutions for relaxed problems P_j^1 and P_j^M , respectively. We have the following geometric relationships:

$$s_{ju+1}^1 = q_{j1} s_{ju}^1, \quad \forall u = 1, \dots, n_j - 1, \quad (5.31)$$

$$s_{ju+1}^M = q_{jm} s_{ju}^M, \quad \forall u = 1, \dots, n_j - 1, \quad (5.32)$$

Let solution Q_j^* be an optimal subplot-size solution $\{s_{j1}^*, \dots, s_{jn_j}^*\}$ for lot j in position $i + 1$. Let $D(\tau_i)$ be the set of r dominant machines, each of which has a pattern-switching subplot associated with it; and let $W(\tau, j) = \{\rho_{jk_1}, \rho_{jk_2}, \dots, \rho_{jk_r}\}$ be a sequence of the pattern-switching sublots of lot j . By the order of subassembly machines in the non-decreasing order of unit processing time, we have the following relationship among the processing time ratios:

$$q_{jm} \leq q_{jk_r} \leq \dots \leq q_{jk_1} \leq q_{j1}. \quad (5.33)$$

We have the following four cases: (i) $s_{j1}^* < s_{j1}^1 (= lb_{j1})$; (ii) $s_{j1}^* > s_{j1}^M (= ub_{j1})$; (iii) $s_{jn_j}^* < s_{jn_j}^M (= lb_{jn_j})$; (iv) $s_{jn_j}^* > s_{jn_j}^1 (= ub_{jn_j})$. We analyze each of these cases next.

Case (i) $s_{j1}^* < s_{j1}^1 (= lb_{j1})$.

Due to the geometric relationships among the sublots from 1 to ρ_{jk_1} in both Q^* and Q^1 , we have $s_{j,u+1}^* = q_{jk_1} s_{ju}^*$ and $s_{j,u+1}^1 = q_{j1} s_{ju}^1$, for $u \in [1, \rho_{jk_1} - 1]$. By the assumption $s_{j1}^* < s_{j1}^1$ and $q_{jk_1} \leq q_{j1}$ (see (5.33)), we have $s_{ju}^* < s_{ju}^1$, for $u \in [1, \rho_{jk_1}]$. For sublots $\rho_{jk_1} + 1$ in Q_j^* and Q_j^1 , we have $s_{j,\rho_{k_1}+1}^* \leq q_{jk_1} s_{j\rho_{k_1}}^*$ by (5.21) and $s_{j,\rho_{k_1}+1}^1 = q_{j1} s_{j\rho_{k_1}}^1$ by (5.31), respectively. By the fact that $s_{j\rho_{k_1}}^* < s_{j\rho_{k_1}}^1$ and $q_{jk_1} \leq q_{j1}$, we have $s_{j,\rho_{k_1}+1}^* < s_{j,\rho_{k_1}+1}^1$. Hence, we have

$s_{ju}^* < s_{ju}^1$, for subplot u , $\forall u \in [1, \rho_{jk_1} + 1]$. We can use similar arguments to show $s_{ju}^* < s_{ju}^1$ for a subplot u in ranges $[\rho_{k_1} + 1, \rho_{k_2} + 1]$, $[\rho_{k_2} + 1, \rho_{k_3} + 1]$, \dots , $[\rho_{k_{r-1}} + 1, \rho_{k_r}]$, sequentially. This leads to $\sum_{u=1}^{n_j} s_{ju}^* < U_j$, which contradicts the feasibility of Q_j^* .

Case (ii) $s_{j1} > s_{j1}^M (= ub_{j1})$.

Due to the geometric relationships for sublots from 1 to ρ_{jk_1} in both Q^* and Q^M , we have $s_{j,u+1}^* = q_{jk_1} s_{ju}^*$ and $s_{j,u+1}^M = q_{jm} s_{ju}^M$, for $u \in [1, \rho_{jk_1} - 1]$. By the assumption $s_{j1}^* > s_{j1}^M$ and $q_{jk_1} \geq q_{jm}$ (see (5.33)), we have $s_{ju}^* > s_{ju}^M$, for $u \in [1, \rho_{jk_1}]$. For sublots $\rho_{jk_1} + 1$ in Q_j^* and Q_j^M , we have $s_{j,\rho_{k_1}+1}^* \geq q_{jk_2} s_{j\rho_{k_1}}^*$ by (5.21) and $s_{j,\rho_{k_1}+1}^M = q_{jm} s_{j\rho_{k_1}}^M$ by (5.32), respectively. By the fact that $s_{j\rho_{k_1}}^* > s_{j\rho_{k_1}}^M$ and $q_{jk_2} \geq q_{jm}$, we have $s_{j,\rho_{k_1}+1}^* > s_{j,\rho_{k_1}+1}^M$. Hence, we have $s_{ju}^* > s_{ju}^M$, for subplot u , $\forall u \in [1, \rho_{jk_1} + 1]$. We can use similar arguments to show $s_{ju}^* > s_{ju}^M$ for a subplot u in ranges $[\rho_{k_1} + 1, \rho_{k_2} + 1]$, $[\rho_{k_2} + 1, \rho_{k_3} + 1]$, \dots , $[\rho_{k_{r-1}} + 1, \rho_{k_r}]$, sequentially. This leads to $\sum_{u=1}^{n_j} s_{ju}^* > U_j$, which contradicts the feasibility of Q_j^* .

Case (iii) $s_{jn_j} < s_{jn_j}^M (= lb_{jn_j})$.

Due to the geometric relationships for sublots from $\rho_{jk_{r-1}} + 1$ to $\rho_{jk_r} (= n_j)$ in both Q^* and Q^M , we have $s_{ju}^* = \frac{s_{j,u+1}^*}{q_{jk_r}}$ and $s_{ju}^M = \frac{s_{j,u+1}^M}{q_{jm}}$, for $u \in [\rho_{jk_{r-1}} + 1, \rho_{jk_r} - 1]$. By the assumption $s_{jn_j}^* < s_{jn_j}^M$ and $q_{jk_r} \geq q_{jm}$ (see (5.33)), we have $s_{ju}^* < s_{ju}^M$, for $u \in [\rho_{jk_{r-1}} + 1, \rho_{jk_r}]$. For sublots $\rho_{jk_{r-1}}$ in Q_j^* and Q_j^M , we have $s_{j,\rho_{k_{r-1}}}^* \leq \frac{s_{j,\rho_{k_{r-1}}+1}^*}{q_{jk_r}}$ by (5.21) and $s_{j,\rho_{k_{r-1}}}^M = \frac{s_{j,\rho_{k_{r-1}}+1}^M}{q_{jm}}$ by (5.32), respectively. By the fact that $s_{j,\rho_{k_{r-1}}+1}^* < s_{j,\rho_{k_{r-1}}+1}^M$ and $q_{jk_r} \geq q_{jm}$, we have $s_{j,\rho_{k_{r-1}}}^* < s_{j,\rho_{k_{r-1}}}^M$. Hence, we have $s_{ju}^* < s_{ju}^M$, for subplot u , $\forall u \in [\rho_{jk_{r-1}}, \rho_{jk_r}]$. We can use similar arguments to show $s_{ju}^* < s_{ju}^M$ for a subplot u in ranges $[1, \rho_{k_1}]$, $[\rho_{k_1}, \rho_{k_2}]$, \dots , $[\rho_{k_{r-2}}, \rho_{k_{r-1}}]$, sequentially, in the reverse order. This leads to $\sum_{u=1}^{n_j} s_{ju}^* < U_j$, which contradicts the feasibility of Q_j^* .

Case (iv) $s_{jn_j} > s_{jn_j}^1 (= ub_{jn_j})$.

Due to the geometric relationships for sublots from $\rho_{jk_{r-1}} + 1$ to $\rho_{jk_r} (= n_j)$ in both Q^* and Q^1 , we have $s_{ju}^* = \frac{s_{j,u+1}^*}{q_{jk_r}}$ and $s_{ju}^1 = \frac{s_{j,u+1}^1}{q_{j1}}$, for $u \in [\rho_{jk_{r-1}} + 1, \rho_{jk_r} - 1]$. By the assumption $s_{jn_j}^* > s_{jn_j}^1$ and $q_{jk_r} \leq q_{j1}$ (see (5.33)), we have $s_{ju}^* > s_{ju}^1$, for $u \in [\rho_{jk_{r-1}} + 1, \rho_{jk_r}]$. For sublots $\rho_{jk_{r-1}}$ in Q_j^* and Q_j^1 , we have $s_{j,\rho_{k_{r-1}}}^* \geq \frac{s_{j,\rho_{k_{r-1}}+1}^*}{q_{jk_{r-1}}}$ by (5.21) and $s_{j,\rho_{k_{r-1}}}^1 = \frac{s_{j,\rho_{k_{r-1}}+1}^1}{q_{j1}}$ by (5.31), respectively. By the fact that $s_{j,\rho_{k_{r-1}}+1}^* > s_{j,\rho_{k_{r-1}}+1}^1$ and $q_{jk_{r-1}} \leq q_{j1}$, we have $s_{j,\rho_{k_{r-1}}}^* > s_{j,\rho_{k_{r-1}}}^1$. Hence, we have $s_{ju}^* > s_{ju}^1$, for subplot u , $\forall u \in [\rho_{jk_{r-1}}, \rho_{jk_r}]$. We can use similar

arguments to show $s_{ju}^* > s_{ju}^1$ for a subplot u in ranges $[1, \rho_{k_1}]$, $[\rho_{k_1}, \rho_{k_2}]$, \dots , $[\rho_{k_{r-2}}, \rho_{k_{r-1}}]$, sequentially, in the reverse order. This leads to $\sum_{u=1}^{n_j} s_{ju}^* > U_j$, which contradicts the feasibility of Q_j^* . \square

5.4.3.2 Dominance Rules

Proposition 5.3. (DM1) *Let τ_i and $\hat{\tau}_i$ be two partial solutions up to position i . If $\hat{\tau}_i = \tau_i$, and their corresponding completion times on the assembly machine A are such that $C_A(\tau_i) \leq C_A(\hat{\tau}_i)$, then there exists an optimal solution schedule which does not start with $\hat{\tau}_i$.*

Proof. Due to $\hat{\tau}_i = \tau_i$, we have the same partial completion time on any subassembly machine k , that is $C_k(\hat{\tau}_i) = C_k(\tau_i)$. Moreover, due to the fact that $C_A(\tau_i) \leq C_A(\hat{\tau}_i)$, the completion time on machine A incurred by a schedule that starts with τ_i is no larger than that of a schedule that starts with $\hat{\tau}_i$. \square

Proposition 5.4. (DM2) *Given completion times $C_k(\tau_i)$ and $C_A(\tau_i)$ of τ_i , if there exists a lot f , $f \in \tau_i'$, such that*

$$t_{fA} + p_{fA}U_f - (t_{fk} + p_{fk}U_f) \geq \max_{j \in \tau_i' - \{f\}} \left\{ p_{jk}(ub_{j1} - lb_{j1}) \right\}, \quad \forall k = 1, \dots, M, \quad (5.34)$$

and

$$C_k(\tau_i) + t_{fk} + p_{fk}ub_{f1} + p_{fA}U_f + \sum_{j \in \tau_i' - \{f\}} (t_{jA} + p_{jA}U_j) \leq LB(\tau_i), \quad \forall k = 1, \dots, M, \quad (5.35)$$

then there exists an optimal schedule in which lot f is sequenced in position $i + 1$.

Proof. We prove the result by contradiction. Suppose there exists an optimal solution π^* in which lot f is sequenced in position f' , where $f' > i + 1$. Let π be a solution obtained by moving lot f from position f' to position $i + 1$. For π , the makespan can be represented by

$$\begin{aligned} M(\pi) = \max \left\{ C_k(\tau_i) + \sum_{j=i+1}^{c-1} (t_{\pi(j)k} + p_{\pi(j)k}U_{\pi(j)}) + (t_{\pi(c)k} + p_{\pi(c)k}S_{\pi(c)1} + p_{\pi(c)A}U_{\pi(c)}) \right. \\ \left. + \sum_{j=c+1}^N (t_{\pi(j)A} + p_{\pi(j)A}U_{\pi(j)}), C_A(\tau_i) + \sum_{j=i+1}^N (t_{\pi(j)A} + p_{\pi(j)A}U_{\pi(j)}) \right\} \end{aligned} \quad (5.36)$$

for some c ($c = i + 1, \dots, N$) and k ($k = 1, \dots, M$). For the position c , we have the following cases: (1) c does not exist from $i + 1$ to N ; (2) $c = i + 1$; (3) $i + 2 \leq c \leq f'$; case (4): $c \geq f' + 1$. We consider each of these cases next.

Case (1): c does not exist from $i + 1$ to N .

For this case, the makespan can be represented by

$$M(\pi) = C_A(\tau_i) + \sum_{j=i+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right). \quad (5.37)$$

Then π is an optimal solution because it is equal to $LB^1(\tau_i)$ in (5.8).

Case (2): $c = i + 1$.

The subplot f in position c , is critical. We have

$$\begin{aligned} M(\pi) &= C_k(\tau_i) + t_{fk} + p_{fk} s_{f1} + p_{fA} U_f + \sum_{j=i+2}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right) \\ &\leq C_k(\tau_i) + t_{fk} + p_{fk} u_{bf1} + p_{fA} U_f + \sum_{j=i+2}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right). \end{aligned}$$

In view of (5.35), we have $M(\pi) \leq LB(\tau_i)$, which indicates that π is an optimal solution.

Case (3): $i + 2 \leq c \leq f'$.

Based on (5.2) and (5.4), for solution π^* , we have

$$\begin{aligned} M(\pi^*) &\geq C_k(\tau_i) + \sum_{j=i+1}^{c-2} \left(t_{\pi^*(j)k} + p_{\pi^*(j)k} U_{\pi^*(j)} \right) \\ &\quad + \left(t_{\pi^*(c-1)k} + p_{\pi^*(c-1)k} s_{\pi^*(c-1)1} + p_{\pi^*(c-1)A} U_{\pi^*(c-1)} \right) + \sum_{j=c}^N \left(t_{\pi^*(j)A} + p_{\pi^*(j)A} U_{\pi^*(j)} \right). \end{aligned} \quad (5.38)$$

Since f is in position $i + 1$ in π , and it is in a position $f' \geq c$ in π^* , and the lots in position $i + 1$ to $c - 2$ in π^* are identical to those in positions $i + 2$ to $c - 1$ in π , we have

$$\sum_{j=i+1}^{c-2} \left(t_{\pi^*(j)k} + p_{\pi^*(j)k} U_{\pi^*(j)} \right) = \sum_{j=i+1}^{c-1} \left(t_{\pi(j)k} + p_{\pi(j)k} U_{\pi(j)} \right) - (t_{fk} + p_{fk} U_f). \quad (5.39)$$

Furthermore, for the lots in position c to N in π^* and lots in $c + 1$ to N in π , we have

$$\sum_{j=c}^N \left(t_{\pi^*(j)A} + p_{\pi^*(j)A} U_{\pi^*(j)} \right) = \sum_{j=c+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right) + (t_{fA} + p_{fA} U_f). \quad (5.40)$$

Note that $\pi^*(c-1) = \pi(c)$ which results in,

$$t_{\pi^*(c-1)k} + p_{\pi^*(c-1)A} U_{\pi^*(c-1)} = t_{\pi(c)k} + p_{\pi(c)A} U_{\pi(c)}. \quad (5.41)$$

By substituting (5.39), (5.40) and (5.41) into (5.38), we have

$$\begin{aligned} M(\pi^*) &\geq C_k(\tau_i) + \sum_{j=i+1}^{c-1} \left(t_{\pi(j)k} + p_{\pi(j)k} U_{\pi(j)} \right) - (t_{fk} + p_{fk} U_f) \\ &\quad + \left(t_{\pi(c)k} + p_{\pi(c)k} s_{\pi^*(c-1)1} + p_{\pi(c)A} U_{\pi(c)} \right) + \sum_{j=c+1}^N \left(t_{\pi(j)A} + p_{\pi(j),A} U_{\pi(j)} \right) + (t_{fA} + p_{fA} U_f) \\ &\geq C_k(\tau_i) + \sum_{j=i+1}^{c-1} \left(t_{\pi(j)k} + p_{\pi(j)k} U_{\pi(j)} \right) + \left(t_{\pi(c)k} + p_{\pi(c)A} U_{\pi(c)} \right) + \sum_{j=c+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right) \\ &\quad + (t_{fk} + p_{fk} U_f) - (t_{fA} + p_{fA} U_f) + p_{\pi(c)A} lb_{\pi^*(c-1)1}. \end{aligned} \quad (5.42)$$

As $lb_{\pi^*(c-1)1} = lb_{\pi(c)1}$ because $\pi^*(c-1) = \pi(c)$ (as noted above), and using (5.34), we have

$$\begin{aligned} M(\pi^*) &\geq C_k(\tau_i) + \sum_{j=i+1}^{c-1} \left(t_{\pi(j)k} + p_{\pi(j)k} U_{\pi(j)} \right) + \left(t_{\pi(c)k} + p_{\pi(c)k} ub_{\pi(c)1} + p_{\pi(c)A} U_{\pi(c)} \right) \\ &\quad + \sum_{j=c+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right) \end{aligned}$$

In view of (5.36), we have $M(\pi) \leq M(\pi^*)$, which indicates that solution π is at least as good as π^* .

Case (4): when $c \geq f' + 1$, the movement of lot f from position $i + 1$ to position f' can only create a chance for idle time to increase before position c on machine A . Therefore, we have $M(\pi^*) \geq M(\pi)$, which again indicates that solution π is at least as good as π^* . \square

Proposition 5.5. (DM3) *Given machine availability times $C_k(\tau_i)$, $k = 1, \dots, M$ and $C_A(\tau_i)$, if there exists a lot f such that*

$$t_{fk} - t_{fA} + p_{fk} U_f - p_{fA} U_f \geq \max_{j \in \tau'_i - \{f\}} \left\{ p_{jA} (ub_{jn_j} - lb_{jn_j}) \right\}, \quad \forall k = 1, \dots, M, \quad (5.43)$$

and

$$C_k(\tau_i) + \sum_{j \in \tau'_i} \left(t_{jk} + p_{jk} U_j \right) + p_{fA} ub_{f,n_f} \leq LB(\tau_i), \quad \forall k = 1, \dots, M, \quad (5.44)$$

then there exists an optimal schedule in which lot f is sequenced last.

Proof. We prove the result by contradiction. Suppose there exists an optimal solution π^* in which lot f is sequenced in position f' , where $f' < N$. Let π be a solution obtained by moving lot f from position f' to the last position N . For π , the makespan can be represented by

$$M(\pi) = \max \left\{ C_k(\tau_i) + \sum_{j=i+1}^{c-1} \left(t_{\pi(j)k} + p_{\pi(j)k} U_{\pi(j)} \right) + \left(t_{\pi(c)k} + p_{\pi(c)k} U_{\pi(c)} + p_{\pi(c)A} s_{\pi(c)n_{\pi(c)}} \right) \right. \\ \left. + \sum_{j=c+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right), C_A(\tau_i) + \sum_{j=i+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right) \right\}, \quad (5.45)$$

for some c ($c = i + 1, \dots, N$) and k ($k = 1, \dots, M$). We have the following cases: (1) c does not exist from $i + 1$ to N ; (2) $c = N$; (3) $f' \leq c \leq N - 1$; (4) $c \leq f' - 1$. We consider each of these cases next.

Case (1): c does not exist from $i + 1$ to N .

For this case, the makespan can be represented by

$$M(\pi) = C_A(\tau_i) + \sum_{j=i+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right).$$

Then π is an optimal solution because the makespan is equal to $LB^1(\tau_i)$.

Case (2): $c = N$.

The subplot f in position c is critical. We have

$$M(\pi) = C_k(\tau_i) + \sum_{j=i+1}^{N-1} \left(t_{\pi(j)k} + p_{\pi(j)k} U_{\pi(j)} \right) + \left(t_{fk} + p_{fk} U_f + p_{fA} s_{fn_f} \right) \\ = C_k(\tau_i) + \sum_{j=i+1}^N \left(t_{\pi(j)k} + p_{\pi(j),k} U_{\pi(j)} \right) + \left(p_{fA} s_{fn_f} \right) \\ \leq C_k(\tau_i) + \sum_{j \in \tau'_i} \left(t_{\pi(j)k} + p_{\pi(j)k} U_{\pi(j)} \right) + \left(p_{fA} ub_{fn_f} \right),$$

In view of (5.44), we have $M(\pi) \leq LB(\tau_i)$, which indicates that π is an optimal solution.

Case (3): $f' \leq c \leq N - 1$.

Based on (5.2) and (5.4), for solution π^* , we have

$$\begin{aligned} M(\pi^*) &\geq C_k(\tau_i) + \sum_{j=i+1}^c \left(t_{\pi^*(j)k} + p_{\pi^*(j)k} U_{\pi^*(j)} \right) \\ &\quad + \left(t_{\pi^*(c+1)k} + p_{\pi^*(c+1)k} U_{\pi^*(c+1)} + p_{\pi^*(c+1)A} S_{\pi^*(c+1)n_{\pi^*(c+1)}}^* \right) + \sum_{j=c+2}^N \left(t_{\pi^*(j)A} + p_{\pi^*(j)A} U_{\pi^*(j)} \right). \end{aligned} \quad (5.46)$$

Since f is in position N in π , and it is in position $f' \leq c$ in π^* , and the lots in positions $c+2$ to N in π^* are identical to those in positions $c+1$ to $N-1$ in π , we have

$$\sum_{j=c+2}^N \left(t_{\pi^*(j)A} + p_{\pi^*(j)A} U_{\pi^*(j)} \right) = \sum_{j=c+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right) - (t_{fA} + p_{fA} U_f). \quad (5.47)$$

Furthermore, for the lots in positions $i+1$ to c in π^* and in positions $i+1$ to $c-1$ in π , we have

$$\sum_{j=i+1}^c \left(t_{\pi^*(j)k} + p_{\pi^*(j)k} U_{\pi^*(j)} \right) = \sum_{j=i+1}^{c-1} \left(t_{\pi(j)k} + p_{\pi(j)k} U_{\pi(j)} \right) + (t_{fk} + p_{fk} U_f). \quad (5.48)$$

Note that $\pi^*(c+1) = \pi(c)$, which results in,

$$t_{\pi^*(c+1),k} + p_{\pi^*(c+1),k} U_{\pi^*(c+1)} = t_{\pi(c),k} + p_{\pi(c),k} U_{\pi(c)}. \quad (5.49)$$

By substituting (5.55), (5.56) and (5.57) into (5.54), we have

$$\begin{aligned} M(\pi^*) &\geq C_k(\tau_i) + \sum_{j=i+1}^{c-1} \left(t_{\pi(j)k} + p_{\pi(j)k} U_{\pi(j)} \right) + (t_{fk} + p_{fk} U_f) \\ &\quad + \left(t_{\pi(c)k} + p_{\pi(c)k} U_{\pi(c)} + p_{\pi(c)A} S_{\pi^*(c+1)n_{\pi^*(c+1)}}^* \right) + \sum_{j=c+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right) - (t_{fA} + p_{fA} U_f) \\ &\geq C_k(\tau_i) + \sum_{j=i+1}^{c-1} \left(t_{\pi(j)k} + p_{\pi(j)k} U_{\pi(j)} \right) + \left(t_{\pi(c)k} + p_{\pi(c)k} U_{\pi(c)} \right) + \sum_{j=c+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right) \\ &\quad + (t_{fk} + p_{fk} U_f) - (t_{fA} + p_{fA} U_f) + p_{\pi(c)A} lb_{\pi^*(c+1)n_{\pi^*(c+1)}} \end{aligned}$$

AS $lb_{\pi^*(c+1)n_{\pi^*(c+1)}} = lb_{\pi(c)n_{\pi(c)}}$ because $\pi^*(c+1) = \pi(c)$ (as noted above), and using (5.43), we have

$$M(\pi^*) \geq C_k(\tau_i) + \sum_{j=i+1}^{c-1} \left(t_{\pi(j)k} + p_{\pi(j)k} U_{\pi(j)} \right) + \left(t_{\pi(c)k} + p_{\pi(c)k} U_{\pi(c)} + p_{\pi(c)A} ub_{\pi(c)n_{\pi(c)}} \right) \\ + \sum_{j=c+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right).$$

In view of (5.45), we have $M(\pi^*) \geq M(\pi)$, which indicates that solution π is at least as good as π^* .

Case (4): for $c \leq f' - 1$, the movement of lot f from position N to position f' can only create a chance for idle time to increase after position c on machine A . Therefore, we have $M(\pi^*) \geq M(\pi)$, implying that solution π is at least as good as π^* . \square

Proposition 5.6. (DM4) *Given machine availability times $C_k(\tau_i)$, $k = 1, \dots, M$ and $C_A(\tau_i)$, if there exist two lots f and g such that*

$$t_{fA} - t_{fk} + p_{fA} U_f - p_{fk} U_f \geq \max_{j \in \tau'_i - \{f\}} \left\{ p_{jk} (ub_{j1} - lb_{j1}) \right\}, \quad \forall k = 1, \dots, M, \quad (5.50)$$

and

$$C_k(\tau_i) + t_{fk} + p_{fk} ub_{f1} - t_{fA} \leq \max \left\{ C_k(\tau_i) + t_{gk} + p_{gk} lb_{g1} - t_{gA}, C_A(\tau_i) \right\}, \quad \forall k = 1, \dots, M, \quad (5.51)$$

then there exists an optimal schedule in which lot g is not sequenced in position $i + 1$.

Proof. We prove the result by contradiction. Suppose that π^* is an optimal solution in which lot g is sequenced in position $i + 1$ and lot f is sequenced in position f' , where $f' \geq i + 2$. Let π be obtained from π^* by moving lot f from position f' to position $i + 1$. The makespan for π can be represented by

$$M(\pi) = \max \left\{ C_k(\tau_i) + \sum_{j=i+1}^{c-1} \left(t_{\pi(j)k} + p_{\pi(j)k} U_{\pi(j)} \right) + \left(t_{\pi(c)k} + p_{\pi(c)k} s_{\pi(c)1} + p_{\pi(c)A} U_{\pi(c)} \right) \right. \\ \left. + \sum_{j=c+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right), C_A(\tau_i) + \sum_{j=i+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right) \right\} \quad (5.52)$$

If $M(\pi) = C_A(\tau_i) + \sum_{j=i+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right)$, then π is an optimal solution because it is equal to $LB^1(\tau)$. Otherwise, we have the following cases: (1) $c = i + 1$; (2) $c = i + 2$; (3) $f' \geq c \geq i + 3$; and (4) $c \geq f' + 1$. We consider each of these cases next.

Case (1): $c = i + 1$.

The subplot f at position c is critical, the makespan for π can be re-written as

$$\begin{aligned} M(\pi) &= C_k(\tau_i) + t_{fk} + p_{fk} s_{f1} + p_{fA} U_f + \sum_{j=i+2}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right), \\ &= C_k(\tau_i) + t_{fk} + p_{fk} s_{f1} - t_{fA} + \sum_{j=i+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right) \end{aligned}$$

For solution π^* , we have

$$\begin{aligned} M(\pi^*) &\geq \max \left\{ C_k(\tau_i) + t_{gk} + p_{gk} s_{g1}^* + p_{gA} U_g + \sum_{j=i+2}^N \left(t_{\pi^*(j)A} + p_{\pi^*(j)A} U_{\pi^*(j)} \right), \right. \\ &\quad \left. C_A(\tau_i) + \sum_{j=i+1}^N \left(t_{\pi^*(j)A} + p_{\pi^*(j)A} U_{\pi^*(j)} \right) \right\} \\ &= \max \left\{ C_k(\tau_i) + t_{gk} + p_{gk} s_{g1}^* + p_{gA} U_g - t_{gA} - p_{gA} U_g, C_A(\tau_i) \right\} \\ &\quad + \sum_{j=i+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right) \\ &= \max \left\{ C_k(\tau_i) + t_{gk} + p_{gk} s_{g1}^* - t_{gA}, C_A(\tau_i) \right\} + \sum_{j=i+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right) \\ &\geq \max \left\{ C_k(\tau_i) + t_{gk} + p_{gk} l_{b_{g1}} - t_{gA}, C_A(\tau_i) \right\} + \sum_{j=i+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right) \end{aligned}$$

In view of (5.51), we have $M(\pi^*) \geq M(\pi)$, which indicates that solution π is at least as good as π^* .

Case (2): $c = i + 2$.

The makespan for π can be re-written as

$$M(\pi) = C_k(\tau_i) + t_{fk} + p_{fk} U_f + t_{gk} + p_{gk} s_{g1} + p_{gA} U_g + \sum_{j=i+3}^N \left(t_{\pi(j)A} + p_{\pi(j)A} U_{\pi(j)} \right),$$

$$\begin{aligned}
&= C_k(\tau_i) + t_{fk} - t_{fA} + t_{gk} - t_{gA} + (p_{fk} - p_{fA})U_f + p_{gk}s_{g1} + \sum_{j=i+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A}U_{\pi(j)} \right) \\
&\leq C_k(\tau_i) + t_{fk} - t_{fA} + t_{gk} - t_{gA} + (p_{fk} - p_{fA})U_f + p_{gk}ub_{g1} + \sum_{j=i+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A}U_{\pi(j)} \right)
\end{aligned}$$

Based on (5.50), we have

$$M(\pi) \leq C_k(\tau_i) + t_{gk} + p_{gk}lb_{g1} - t_{gA} + \sum_{j=i+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A}U_{\pi(j)} \right)$$

For solution π^* , we have

$$\begin{aligned}
M(\pi^*) &\geq C_k(\tau_i) + t_{gk} + p_{gk}s_{g1}^* + p_{gA}U_g + \sum_{j=i+2}^N \left(t_{\pi^*(j)A} + p_{\pi^*(j)A}U_{\pi^*(j)} \right) \\
&\geq C_k(\tau_i) + t_{gk} + p_{gk}lb_{g1} - t_{gA} + \sum_{j=i+1}^N \left(t_{\pi^*(j)A} + p_{\pi^*(j)A}U_{\pi^*(j)} \right) \quad (5.53)
\end{aligned}$$

This leads to $M(\pi^*) \geq M(\pi)$, which indicates that solution π is at least as good as π^* .

Case (3): $f' \geq c \geq i + 3$.

Based on (5.2) and (5.4), for solution π^* , we have

$$\begin{aligned}
M(\pi^*) &\geq C_k(\tau_i) + \sum_{j=i+1}^{c-2} \left(t_{\pi^*(j)k} + p_{\pi^*(j)k}U_{\pi^*(j)} \right) + \left(t_{\pi^*(c-1)k} + p_{\pi^*(c-1)k}s_{\pi^*(c-1)1}^* + p_{\pi^*(c-1)A}U_{\pi^*(c-1)} \right) \\
&\quad + \sum_{j=c}^N \left(t_{\pi^*(j)A} + p_{\pi^*(j)A}U_{\pi^*(j)} \right). \quad (5.54)
\end{aligned}$$

Since f is in position $i + 1$ in π , and it is in a position $f' \geq c$, and the lots in positions $i + 1$ to $c - 2$ in π^* are identical to those in positions $i + 2$ to $c - 1$, we have

$$\sum_{j=i+1}^{c-2} \left(t_{\pi^*(j)k} + p_{\pi^*(j)k}U_{\pi^*(j)} \right) = \sum_{j=i+1}^{c-1} \left(t_{\pi(j)k} + p_{\pi(j)k}U_{\pi(j)} \right) - (t_{fk} + p_{fk}U_f). \quad (5.55)$$

Furthermore, for the lots in positions c to N in π^* and $c + 1$ to N in π , we have

$$\sum_{j=c}^N \left(t_{\pi^*(j)A} + p_{\pi^*(j)A}U_{\pi^*(j)} \right) = \sum_{j=c+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A}U_{\pi(j)} \right) + (t_{fA} + p_{fA}U_f). \quad (5.56)$$

Note that $\pi^*(c-1) = \pi(c)$, which results in,

$$t_{\pi^*(c-1)k} + p_{\pi^*(c-1)A}U_{\pi^*(c-1)} = t_{\pi(c)k} + p_{\pi(c)A}U_{\pi(c)} \quad (5.57)$$

By substituting (5.55), (5.56) and (5.57) into (5.54), we have

$$\begin{aligned} M(\pi^*) &\geq C_k(\tau_i) + \sum_{j=i+1}^{c-1} \left(t_{\pi(j)k} + p_{\pi(j)k}U_{\pi(j)} \right) - (t_{fk} + p_{fk}U_f) \\ &\quad + \left(t_{\pi(c)k} + p_{\pi(c)k}S_{\pi^*(c-1)1}^* + p_{\pi(c)A}U_{\pi(c)} \right) + \sum_{j=c+1}^N \left(t_{\pi(j)A} + p_{\pi(j),A}U_{\pi(j)} \right) + (t_{fA} + p_{fA}U_f) \\ &\geq C_k(\tau_i) + \sum_{j=i+1}^{c-1} \left(t_{\pi(j)k} + p_{\pi(j)k}U_{\pi(j)} \right) + \left(t_{\pi(c)k} + p_{\pi(c)A}U_{\pi(c)} \right) + \sum_{j=c+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A}U_{\pi(j)} \right) \\ &\quad + (t_{fk} + p_{fk}U_f) - (t_{fA} + p_{fA}U_f) + p_{\pi(c)A}lb_{\pi^*(c-1)1} \end{aligned} \quad (5.58)$$

As $lb_{\pi^*(c-1)1} = lb_{\pi(c)1}$ because $\pi^*(c-1) = \pi(c)$ (as noted above), and using (5.34), we have

$$\begin{aligned} M(\pi^*) &\geq C_k(\tau_i) + \sum_{j=i+1}^{c-1} \left(t_{\pi(j)k} + p_{\pi(j)k}U_{\pi(j)} \right) + \left(t_{\pi(c)k} + p_{\pi(c)k}ub_{\pi(c)1} + p_{\pi(c)A}U_{\pi(c)} \right) \\ &\quad + \sum_{j=c+1}^N \left(t_{\pi(j)A} + p_{\pi(j)A}U_{\pi(j)} \right) \end{aligned}$$

In view of (5.52), we have $M(\pi) \leq M(\pi^*)$, which indicates that solution π is at least as good as π^* .

Case (4): When $c \geq f' + 1$, the movement of lot f from position $i + 1$ to position f' can only create a chance for idle time to increase before position c on machine A . Therefore, we have $M(\pi^*) \geq M(\pi)$, which again indicates that solution π is at least as good as π^* . \square

5.4.4 Branch-and-Bound-based Algorithm

The proposed branch-and-bound algorithm relies on the lower and upper bounds, and dominance rules developed in Sections 5.4.2 and 5.4.3. We use the depth-first branching method in the algorithm. Given a node corresponding to a partial sequence τ_i , dominance rule **DM1** is used to determine whether or not to fathom this node. If a node is not fathomed as a result

of rule DM1, the proposed lower bounds are then calculated and compared with the best incumbent objective value to further determine whether to fathom this node. If the node is not fathomed, we use (i) dominance rule **DM2** to determine whether to fix an appropriate lot at position $i + 1$, (ii) dominance rule **DM3** to determine whether to fix a lot at the last position, hence, to eliminate further consideration for that lot in other positions, and (iii) dominance rule **DM4** to determine whether or not to eliminate a lot to be sequenced at position $i + 1$. If the current node is fathomed, we backtrack to continue the branching. The upper bound of the objective value is updated once a better incumbent solution is found. Figure 5.3 depicts the flowchart of the proposed branch-and-bound algorithm.

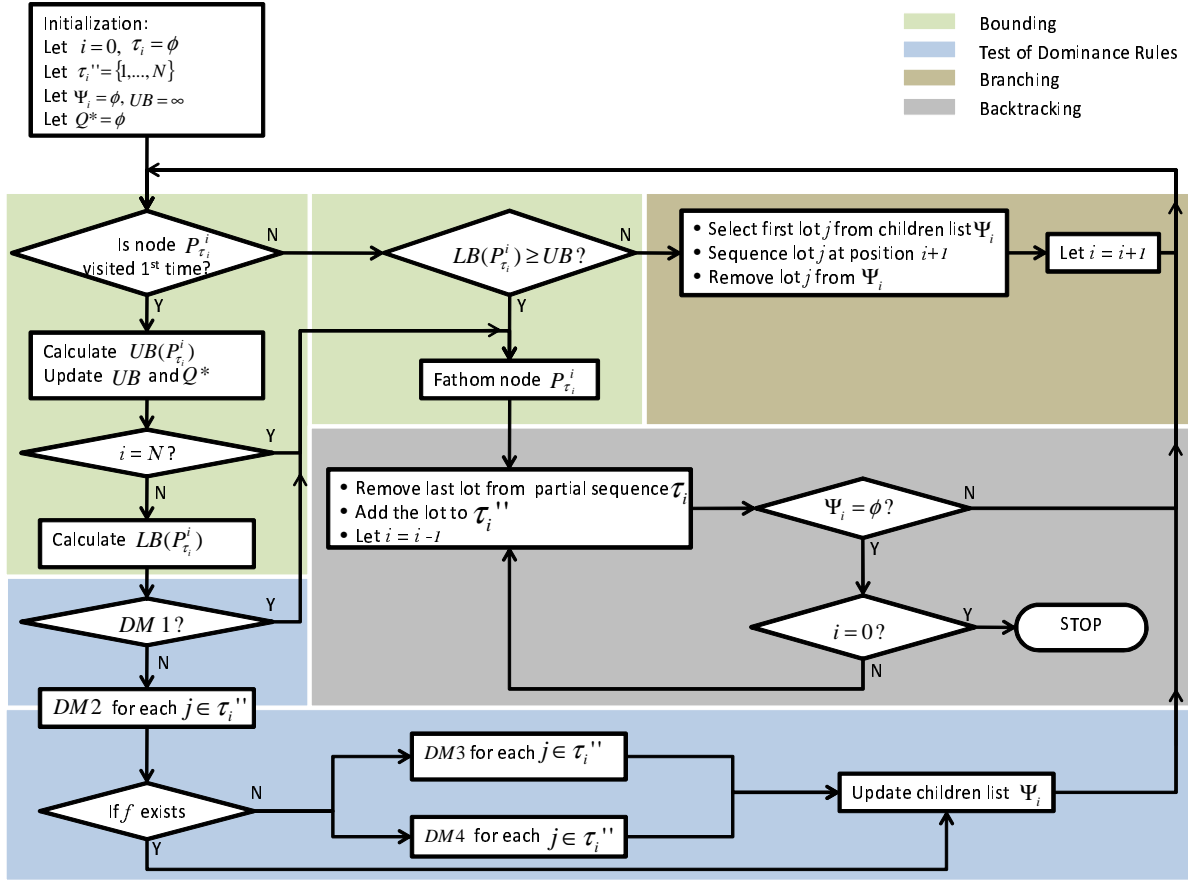


Figure 5.3: Flowchart for the proposed branch-and-bound approach

5.5 Computational Experimentation

In this section, we conduct numerical experimentation to study the computational effectiveness of the mixed integer formulations (TSMLSP-MIP) and the proposed branch-and-bound method (TSMLSP-BB). TSMLSP-BB was coded in C#, and all the runs for both TSMLSP-MIP and TSMLSP-BB were implemented on an Intel Xeon 3.6 GHz computer. CPLEX Solver (version 10.1) was used for solving both the subplot-sizing subproblem at each node of TSMLSP-BB and the TSMLSP-MIP formulation directly.

5.5.1 Computational Test of TSMLSP-BB

First, we present results of our computational experimentation to demonstrate the effectiveness of the proposed TSMLSP-BB method. In particular, our aim is to test the efficacy of using various dominance rules (i.e., DM1, DM2, DM3 and DM4). Note that, if inequalities (5.34), (5.43) and (5.50) hold, then the dominance rules, DM2, DM3 and DM4, respectively, are more likely to be applied successfully. To that end, we vary the per unit processing time on the assembly machine in contrast to those on the subassembly machines in order to generate different combinations of the applications of inequalities (5.34), (5.43) and (5.50). Three problem-sets are generated using uniform distributions for number of lots (N), number of subassembly machines (M), number of items in a lot j (U_j), number of sublots for processing lot j (n_j), and processing time of a job on a subassembly machine (p_{jk}) and assembly machine (p_{jA}), as shown in Table 5.2.

Table 5.2: Sets of problem instances

Problem set	N	M	U_j	n_j	p_{jk}	p_{jA}
1	(20, 50, 100)	(3, 6, 9)	$U(10, 100)$	$U(1, 10)$	$U(50, 100)$	$U(25, 75)$
2	(20, 50, 100)	(3, 6, 9)	$U(10, 100)$	$U(1, 10)$	$U(50, 100)$	$U(50, 100)$
3	(20, 50, 100)	(3, 6, 9)	$U(10, 100)$	$U(1, 10)$	$U(50, 100)$	$U(75, 125)$

For each combination of N and M , 20 problem instances were generated randomly by using the uniform distributions shown in Table 5.2. Note that, the three problem sets differ

due to the different range of p_{jA} values used in relation to the range of p_{jk} values. We kept the range of the p_{jk} values the same to determine the impact of differences among the processing times at the subassembly and assembly machines. Note that in Set 1, the average value of p_{jA} is less than the average of p_{jk} , in Set 2, they are the same, while in Set 3, the average value of p_{jA} is greater than that of p_{jk} . Also, for all problem sets, the same uniform distributions were used to generate values of U_j , n_j and p_{jk} . Moreover, to clearly determine the impact of the inequalities (5.34), (5.43) and (5.50), the lot-detached setup times at all the machines were set to zero. For a test run, we use a time limit of 500 seconds to terminate the computations.

Table 5.3 depicts computational results for the problem instances of the three problem sets obtained by using TSMLSP-BB, which was implemented with the proposed lower and upper bounding procedures. For each problem instance, five combinations of dominance rules, namely, No DMs, DM2, DM3, DM4 and All DMs were tested for their performance. For each combination, information on four criteria, namely, average computational time in seconds (ACT), average number of nodes explored before the algorithm stops (ANN), number of problem solved at the root node (NSR) and number of unsolved problems (NU) were obtained to evaluate the computational performance. The ACT and ANN values are also plotted in Figure 5.4.

Referring to Table 5.3 and Figure 5.4, it can be observed that: (i) for each combination of dominance rules, the computational effort, generally, increases with the number of lots and the number of subassembly machines; (ii) many problem instances are solved at the root node of our branch-and-bound method, due to the tightness of the lower and upper bounds used. Note that Problem Set 1 has the largest number of instances solved at the root node, and Problem Set 2 has fewer instances that were solved at the root node, while Problem Set 3 has the least number of instances solved at the root node; (iii) dominance rules DM2, DM3 and DM4 are most successfully applied in solving the instances of Problem Set 3, which is indicated by the the average cpu time (ACT) required and the number of nodes generated, which are greatly reduced, while it is not so for instances of Problem Set 1. For instances of

Table 5.3: Computational results of TSMLSP-BB

Set	N	M	No DMs				DM2				DM3				DM4				All DMs			
			ACT	ANN	NSR	NU	ACT	ANN	NSR	NU	ACT	ANN	NSR	NU	ACT	ANN	NSR	NU	ACT	ANN	NSR	NU
1	20	3	25.42	617	19	1	25.41	617	19	1	25.33	617	19	1	25.51	617	19	1	25.41	617	19	1
		6	50.86	620	18	2	50.86	616	18	2	50.86	617	18	2	50.86	626	18	2	50.85	626	18	2
		9	101.25	708	16	4	101.21	743	16	4	101.21	749	16	4	101.19	753	16	4	101.24	738	16	4
	50	3	51.05	1097	18	2	51.07	1075	18	2	51.08	1059	18	2	50.98	1087	18	2	51.06	1045	18	2
		6	52.2	406	18	2	52.33	431	18	2	52.18	439	18	2	52.19	443	18	2	52.19	429	18	2
		9	3.85	1	20	0	4.25	1	20	0	4.26	1	20	0	4.08	1	20	0	4.04	1	20	0
	100	3	27.05	410	19	1	26.98	415	19	1	26.97	415	19	1	26.97	415	19	1	27.05	361	19	1
		6	29.31	89	19	1	29.34	88	19	1	29.35	90	19	1	29.33	89	19	1	29.36	83	19	1
		9	56.93	11	18	2	56.98	11	18	2	57.31	11	18	2	56.97	11	18	2	56.79	11	18	2
2	20	3	143.77	1108	12	5	141.82	1131	12	5	66.86	600	12	2	141.8	1132	12	5	66.73	601	12	2
		6	194.18	573	9	6	194.29	630	9	6	184.58	604	9	6	191.94	573	9	6	188.39	545	9	6
		9	182.37	923	11	7	181.72	965	11	7	182.21	947	11	7	181.8	924	11	7	182.39	926	11	7
	50	3	77.82	134	12	1	76.92	135	12	1	25.16	24	12	0	61.57	111	12	1	25.84	24	12	0
		6	98.72	211	15	2	98.93	212	15	2	98.6	52	15	2	98.3	238	15	2	98.22	52	15	2
		9	55.59	18	17	1	55.62	18	17	1	55.7	18	17	1	55.65	18	17	1	55.6	18	17	1
	100	3	123.3	62	11	2	123.35	62	11	2	90.77	45	11	0	101.42	51	11	1	90.75	45	11	0
		6	79	19	17	3	78.86	18	17	3	79.39	19	17	3	79.15	18	17	3	79.22	18	17	3
		9	81.64	13	17	3	81.85	13	17	3	81.81	13	17	3	81.85	13	17	3	81.87	13	17	3
3	20	3	6.93	16	6	0	2.79	7	7	0	4.84	11	6	0	4.59	11	6	0	2.22	5	6	0
		6	17.56	19	3	0	13.89	16	3	0	12.68	14	3	0	13.5	15	3	0	11.05	13	3	0
		9	35.18	24	2	0	32.34	22	2	0	22.89	16	4	0	30.13	21	2	0	21.85	15	3	0
	50	3	48.31	44	3	0	2.19	2	3	0	31.22	29	3	0	22.79	21	3	0	2.16	2	3	0
		6	113.07	49	1	0	77.65	34	1	0	93.21	41	1	0	75.18	33	1	0	66.71	30	1	0
		9	206.69	54	0	0	158.35	44	1	0	200.39	54	0	0	149.77	42	0	0	168.97	44	1	0
	100	3	205.1	92	2	0	15.8	8	2	0	143.69	57	2	0	80.47	31	3	0	12.46	6	2	0
		6	476.03	102	0	2	330.41	65	0	13	394.79	85	0	1	303.18	61	0	1	295.1	58	0	1
		9	504.67	71	0	20	456.28	58	0	18	478.41	64	1	19	478.68	64	0	10	431.07	54	1	17

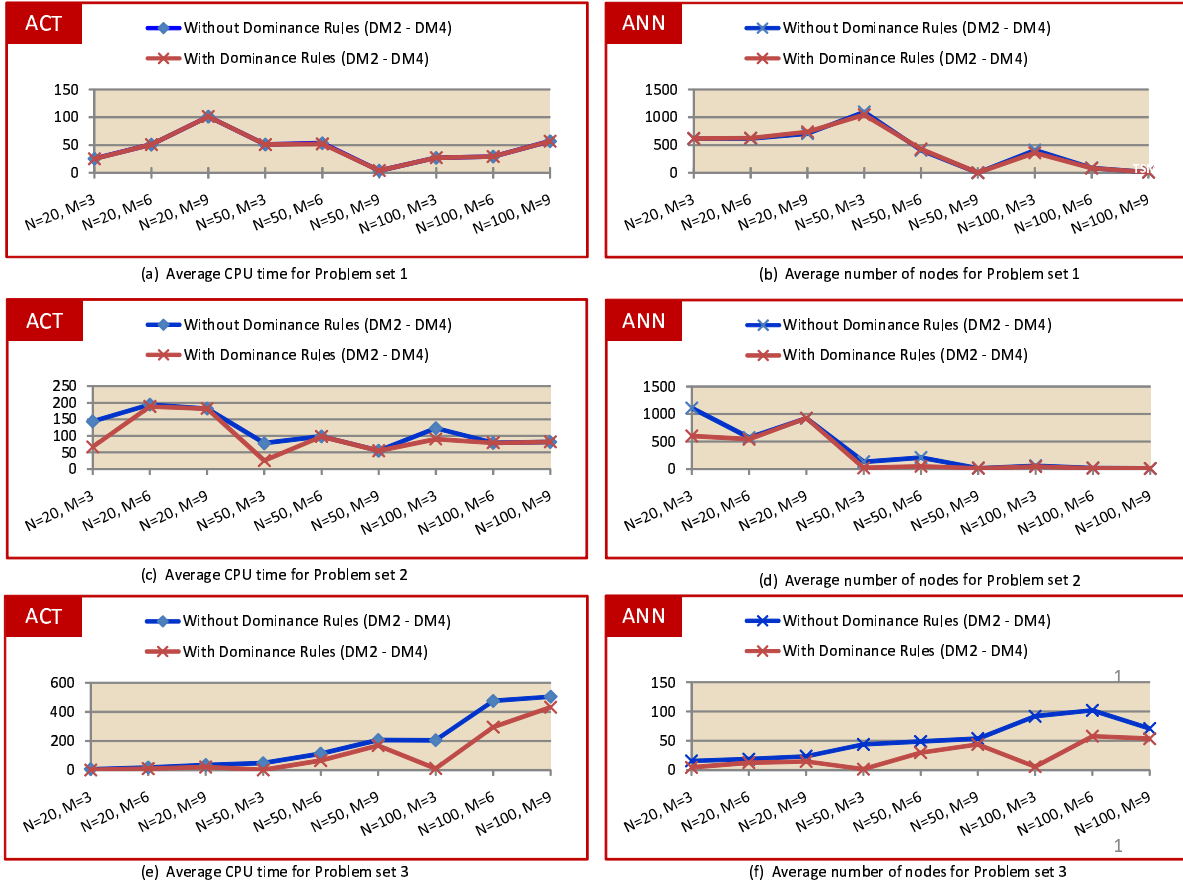


Figure 5.4: The ACT values and the ANN explored by TSMLSP-BB

Problem Set 2, the dominance rules DM2, DM3, DM4 have mixed effectiveness, some instances require less computational effort due to the use of dominance rules, while others do not. This is expected due to the identical average values of p_{jk} and p_{jA} ; (iv) for instances in Problem Set 3, DM2 is generally more effective than the other two dominance rules (DM3 and DM4); (v) when the number of subassembly machines increases, effectiveness of the dominance rules decreases, which can be seen from the ACT and ANN values for instances of Problem Set 2. This is also as expected since more subassembly machines lead to lower possibility for inequalities (5.34), (5.43) and (5.50) to hold true; (vi) there is no significant extra computational time incurred due to the use of dominance rules in contrast to the savings obtained by the elimination of nodes (see Problem Set 1 and 2). Consequently, it is

fairly effective and efficient to use all dominance rules (note the values depicted in the last column in relation to those shown in the others).

5.5.2 Comparison of TSMLSP-BB and TSMLSP-MIP

Next, we compare the computational effort required for implementing the TSMLSP-BB method with the direct solution of the TSMLSP-MIP formulation using CPLEX 10.1. As in the experimentation conducted in Section 5.5.1, we vary the per unit processing time on the assembly machine in contrast to those on the subassembly machines in order to generate different combinations of the applications of inequalities (5.34), (5.43) and (5.50). Three problem sets are generated, as shown in Table 5.4. For a test run, we use a time limit of 500 seconds to terminate the computations.

Table 5.4: Sets of problem instances

Problem set	N	M	U_j	n_j	p_{jk}	p_{jA}
4	(5, 15, 25)	(3, 6, 9)	$U(10, 100)$	$U(1, 10)$	$U(50, 100)$	$U(25, 75)$
5	(5, 15, 25)	(3, 6, 9)	$U(10, 100)$	$U(1, 10)$	$U(50, 100)$	$U(50, 100)$
6	(5, 15, 25)	(3, 6, 9)	$U(10, 100)$	$U(1, 10)$	$U(50, 100)$	$U(75, 125)$

Table 5.5 gives computational results for solving the TSMLSP-MIP formulation directly and by the TSMLSP-BB method. We compare the two approaches with respect to five criteria, AGR(Average Gap at Root node), ACT, ANN, NSR and NU. The ACT and ANN values for for both the methods are plotted in Figure 5.5. Note that the problem instances of Set 6 require the least computational effort for both TSMLSP-MIP and TSMLSP-BB, while the instances of Set 5 require the greatest computational effort for every criteria listed. It is also revealed that TSMLSP-BB, generally, requires much less computational effort to solve the same instances than that required by the direct solution of the TSMLSP-MIP with respect to all criteria. TSMLS-BB not only requires less cpu time and number of nodes explored, but also it solves more problem instances at the root node and leaves fewer instances unsolved within the pre-specified time limit of 500 seconds. This is also indicated by the AGR values in solving both formulations. Consequently, we can conclude that the

TSMLSP-BB method outperforms the direct solution of the TSMLSP-MIP formulation using CPLEX 10.1.

Table 5.5: Comparison of TSMLSP-MIP with TSMLSP-BB

Set	N	M	TSMLSP-MIP					TSMLSP-BB				
			AGR	ACT	ANN	NSR	NU	AGR	ACT	ANN	NSR	NU
4	5	3	11.38%	1.87	12	1	0	0.001%	0.45	11	18	0
		6	11.44%	3.31	11	0	0	0.001%	1.45	18	15	0
		9	11.43%	5.34	12	0	0	0.001%	0.58	3	19	0
	15	3	7.31%	6.07	80	0	0	0.001%	0.29	1	20	0
		6	7.12%	41.19	2482	0	1	0.001%	75.51	899	16	3
		9	7.39%	168.87	4062	0	4	0.001%	50.91	432	18	2
	25	3	4.31%	284.21	1015	0	3	0.001%	25.45	664	19	1
		6	5.12%	483.04	397	0	18	0.001%	25.99	346	19	1
		9	4.39%	500.67	35	0	20	0.001%	51.52	402	18	2
5	5	3	6.34%	0.19	20	0	0	0.61%	0.32	4	11	0
		6	6.94%	0.36	28	0	0	0.76%	1.14	8	9	0
		9	7.65%	0.53	28	0	0	0.46%	1.66	10	12	0
	15	3	3.61%	445.24	30526	0	17	1.26%	76.42	1158	10	3
		6	4.90%	476.97	19162	0	19	0.81%	167.52	848	7	6
		9	5.75%	437.67	12196	0	17	0.49%	178.96	719	11	7
	25	3	2.15%	477.61	11670	0	18	0.40%	33.49	213	9	1
		6	3.32%	496.95	3305	0	19	0.32%	186.69	771	10	5
		9	2.78%	500.57	579	0	20	0.30%	84.89	315	15	3
6	5	3	2.67%	0.17	14	0	0	0.62%	0.2	2	10	0
		6	5.29%	0.21	22	0	0	0.66%	0.71	4	8	0
		9	6.11%	0.31	28	0	0	0.60%	1.13	5	10	0
	15	3	0.16%	87.06	12621	0	3	0.08%	1.07	3	7	0
		6	0.20%	182.89	11261	0	4	0.14%	5.48	9	3	0
		9	0.26%	274.84	7624	0	9	0.14%	9.49	10	3	0
	25	3	0.08%	173.07	6362	0	5	0.05%	1.88	4	4	0
		6	0.10%	379.4	4368	0	13	0.07%	11.6	11	3	0
		9	0.11%	440.04	2542	0	16	0.07%	34.91	22	2	0

5.5.3 Computational Test of TSMLSP-BB for Large-size Problems

In this section, we present computational results on the efficacy of using the TSMLSP-BB method for large-size problem instances. We use problem instances involving 300 to 1000 lots. As in the experimentation conducted in Section 5.5.1, we vary the per unit processing times on the assembly machines in contrast to those on the subassembly machines in order to

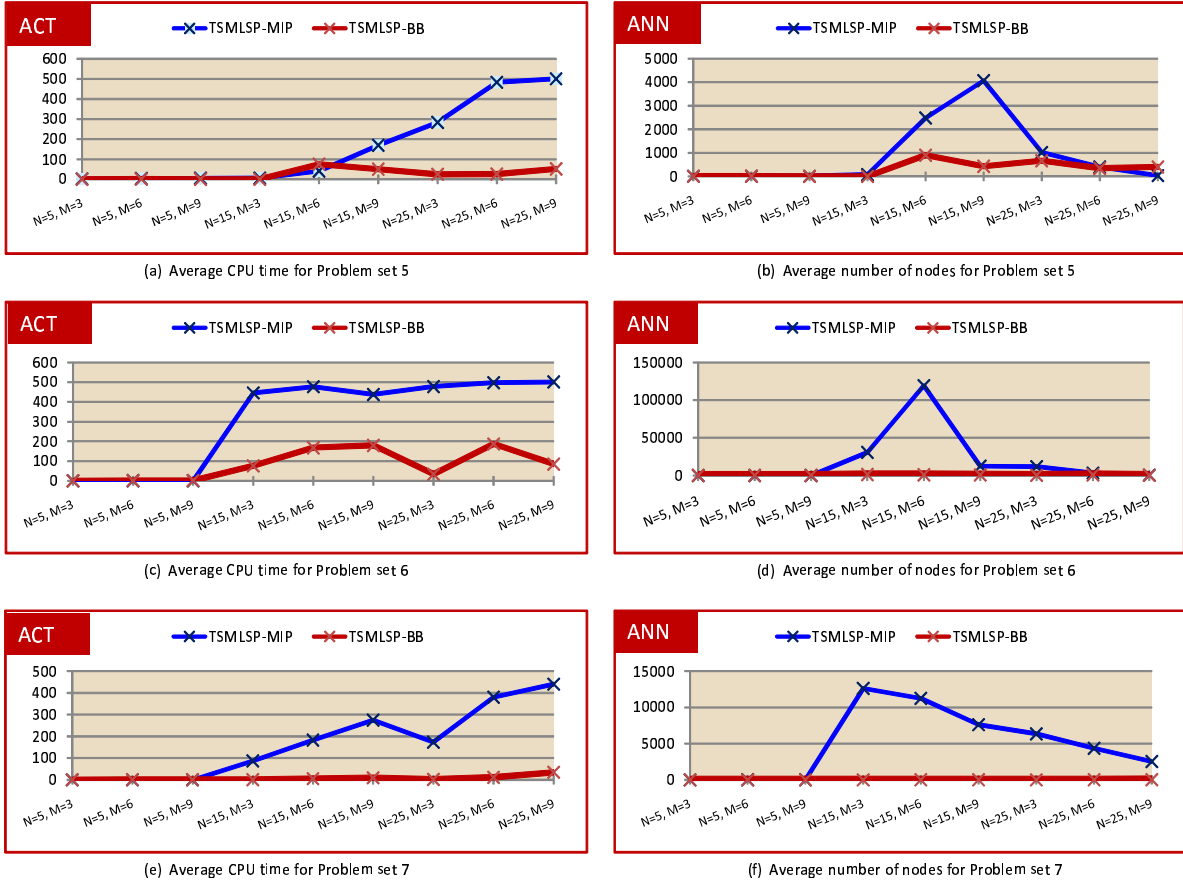


Figure 5.5: Computational comparison of TSMLSP-BB and TSMLSP-MIP with respect to the ACT values and the ANN explored

generate different combinations of the applications of inequalities (5.34), (5.43) and (5.50). Three problem sets are generated as shown in Table 5.6. For each test run, we use a time limit of 500 seconds to terminate the computations.

Table 5.6: Sets of problem instances

Problem set	N	M	U_j	n_j	p_{jk}	p_{jA}
7	(300 ~ 1000)	(3, 6, 9)	$U(10, 100)$	$U(1, 10)$	$U(50, 100)$	$U(25, 75)$
8	(300 ~ 1000)	(3, 6, 9)	$U(10, 100)$	$U(1, 10)$	$U(50, 100)$	$U(50, 100)$
9	(300 ~ 1000)	(3, 6, 9)	$U(10, 100)$	$U(1, 10)$	$U(50, 100)$	$U(75, 125)$

Table 5.7 presents computational results for solving the various problem instances by the TSMLSP-BB method. Five criteria, namely, AGR, ACT, ANN, NSR and NU were used to

Table 5.7: Computational experimentation for TSMLSP-BB

Set	N	M	AGR	ACT	ANN	NSR	NU	N	M	AGR	ACT	ANN	NSR	NU
7	300	3	0%	83.95	1	20	0	700	3	0%	19.82	1	20	0
		6	0.001%	41.92	2	19	1		6	0%	42.01	1	20	0
		9	0.001%	76.25	2	18	2		9	0.001%	84.47	1	19	1
	400	3	0%	11.08	1	20	0	800	3	0%	22.43	1	20	0
		6	0%	23.50	1	20	0		6	0%	47.54	1	20	0
		9	0.001%	64.65	1	19	1		9	0%	71.27	1	20	0
	500	3	0.001%	14.05	1	19	1	900	3	0.001%	47.06	1	19	1
		6	0%	54.35	1	20	0		6	0.001%	71.11	1	19	1
		9	0.001%	71.09	1	19	1		9	0.001%	99.92	1	19	1
	600	3	0%	16.79	1	20	0	1000	3	0.001%	48.69	1	19	1
		6	0%	35.59	1	20	0		6	0.001%	53.07	1	19	1
		9	0.001%	105.81	1	18	2		9	0.001%	85.41	1	19	1
8	300	3	0.001%	149.35	14	14	5	700	3	0.001%	167.58	8	14	6
		6	0.001%	143.49	6	15	5		6	0.001%	137.89	3	16	4
		9	0.001%	104.76	3	17	3		9	0%	67.21	1	20	0
	400	3	0.001%	139.24	9	14	5	800	3	0.001%	200.34	7	13	7
		6	0.001%	78.30	2	18	2		6	0.001%	155.73	2	16	4
		9	0.001%	90.59	2	18	2		9	0.001%	114.66	1	19	1
	500	3	0.001%	139.18	8	15	5	900	3	0.002%	252.97	7	11	9
		6	0.001%	105.19	3	17	3		6	0.001%	184.09	1	18	2
		9	0%	47.97	1	20	0		9	0.001%	144.18	1	18	2
	600	3	0.002%	212.61	12	12	8	1000	3	0.001%	179.24	4	14	6
		6	0.001%	84.60	2	18	2		6	0.001%	141.13	1	17	3
		9	0.001%	129.22	2	17	3		9	0.001%	137.74	1	19	1
9	300	3	0.002%	13.79	1	2	0	700	3	0.001%	43.47	2	0	3
		6	0.002%	199.09	12	0	7		6	0.001%	142.84	2	0	2
		9	0.003%	448.62	14	0	17		9	0.001%	428.75	2	0	9
	400	3	0.001%	237.54	1	2	0	800	3	0.001%	53.91	2	0	0
		6	0.002%	146.24	5	0	3		6	0.001%	153.71	2	0	2
		9	0.002%	455.61	10	0	10		9	0.001%	420.18	4	0	13
	500	3	0.001%	32.53	1	1	0	900	3	0.001%	60.79	2	0	0
		6	0.001%	133.31	3	0	3		6	0.001%	167.76	2	0	1
		9	0.002%	417.31	8	0	8		9	0.001%	429.45	4	0	8
	600	3	0.001%	37.23	2	0	0	1000	3	0.001%	61.31	2	0	0
		6	0.001%	167.65	4	0	14		6	0.001%	166.57	2	0	2
		9	0.001%	445.36	7	0	9		9	0.001%	484.25	4	0	13

evaluate the computational performance, where the values of ACT, NSR and NU are also plotted in Figure 5.6. We observe that the proposed branch-and-bound method, TSMLSP-BB, is able to solve large problem instances within a reasonable time limit. Specifically, note

that (i) the values of AGRs, the average gap at root node, are very small for all problem instances; (ii) a large portion of problem instances in Set 7 and Set 8 are solved at root node; and (iii) the Set 9 problem instances have the fewest number of problem instances solved at the root node and the greatest number of unsolved problem instances within the time limit of 500 seconds. Note that this behavior is identical to that obtained in Table 5.3 for $N = 20, 50$ and 100 . A greater number of unsolved problem, especially for problem instances in Set 9, is due to an upper limit on cpu time (500 second) used. Also, in lieu of our observation made for problems in Set 3 in Table 5.3, it is expected that the proposed dominance rules will be effective in solving the problem instances in Set 9, beyond the root node. Consequently, the TSMLSP-BB method, with proposed bounding procedure and dominance properties, is an effective approach for solving large-size problem. The success of the TSMLSP-BB is largely due to the tightness of the lower- and upper-bounds and the dominance properties at each node.

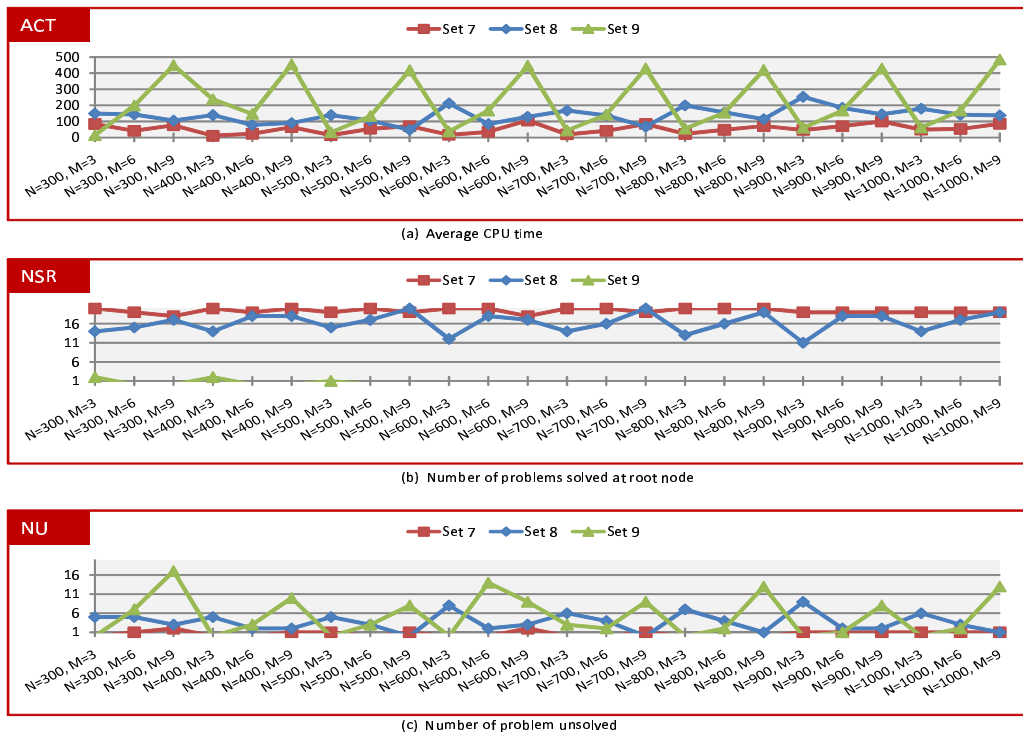


Figure 5.6: The ACT, NSR and NU values for TSMLSP-BB

5.6 Concluding Remarks

In this chapter, we have discussed a multiple-lot, lot streaming problem for a two-stage assembly system. This system consists of M parallel machines at stage 1 and a single assembly machine at stage 2. Such a system for the processing of the lots has been considered by Lee et al. (1993) and Hariri and Potts (1997); however, they do not include streaming of the lots, which we have incorporated in our analysis, and it adds another level of complexity to the problem. We use an important property derived from the single-lot, lot streaming problem for two-stage assembly system (TSLSP) (see Chapter 4), to determine lower and upper bounds on the sizes of the first and last sublots of a lot. Dominance properties are derived based on these lower and upper bounds on subplot sizes. Furthermore, we propose a branch-and-bound method for the solution of our problem. Various lower and upper bounds on the makespan values are determined including the one based on a modified-Johnson algorithm for the lot streaming problem involving scheduling and splitting of multiple lots in a two-machine flow shop. Computational experimentation is conducted, and it shows the efficacy of the proposed branch-and-bound methodology, lower and upper bounds, and the individual and joint use of dominance rules in solving both the small- and large-size problem instances. Our proposed solution method also outperforms the direct solution of the problem using CPLEX.

Chapter 6

Summary and Conclusion

This dissertation has been devoted to the modeling, analysis and development of solution approaches for some optimization-related problems encountered in industrial and manufacturing settings. We address four problems, namely, a high multiplicity asymmetric traveling salesman problem (HMATSP), a primary pharmaceutical manufacturing scheduling problem (PPMSP), and single-lot and multi-lot, lot streaming problems in a two-stage assembly system. We have developed a polynomial length formulation for the HMATSP and have shown its effectiveness in solving problem instances of various sizes. In that aspect, our formulation is the first of its kind. Our formulation for the PPMSP is also effective in solving a real-life instance of the problem. The single-lot and multiple-lot, lot streaming problem that we address are also new because of the machine configuration considered, which is a two-stage assembly system instead of a flow shop that is typically considered for these problems in the literature. We address these problems in Chapters 2, 3, 4 and 5.

In Chapter 2, we have considered a special type of traveling salesman problem called “High Multiplicity Asymmetric Traveling Salesman Problem” (HMATSP). In this version of the ATSP, a node can be visited multiple times. We have proposed a new formulation for this problem, which embraces a flow-based subtour elimination structure, and establish its validity for this problem. This model is, then, incorporated as a substructure in a formulation for the lot-sizing problem involving parallel machines and sequence-dependent setup costs,

also known as the “Chesapeake Problem”. Computational results have been presented to demonstrate the efficacy of our modeling approach for both the generic HMATSP and its application within the construct of the Chesapeake Problem.

In Chapter 3, we have investigated an integrated lot-sizing and scheduling problem encountered in a primary pharmaceutical manufacturing company. This problem entails determination of production lot sizes of multiple products and a sequence in which to process the products on the machines, which can process lots (batches) of a fixed size (due to limited capacity of containers) in the presence of sequence-dependent setup time/costs. This problem is generic in nature and arises in other industrial environments as well. The consideration of batches of final products, in addition to those for the intermediate products, which comprise a final product, further complicates the problem. We present a novel unifying model and a column generation-based optimization approach for this class of lot-sizing and sequencing problems. For the proposed two-stage column generation approach, the lot-sizing decision is considered at the first stage followed by the sequencing of production lots at the second stage. Computational experience is first provided by using randomly generated data sets to test the performances of several variants of our proposed approach. The efficacy of the best of these variants is further demonstrated by applying it to the real-life data collected with the collaboration of a pharmaceutical manufacturing company.

In Chapter 4, we address a single-lot, lot streaming problem for a two-stage assembly system. As noted earlier, this assembly system is different from the traditional flow shop configuration. It consists of m parallel subassembly machines at stage 1, each of which is devoted to the production of a component. A single assembly machine at stage 2, then, assembles a product after components (one each from the subassembly machines at the first stage) have been completed. Lot-detached setups are encountered on the machines at the first and second stages. Given a fixed number of transfer batches (or sublots) from each of the subassembly machines at stage 1 to the assembly machine at stage 2, our problem is to find subplot sizes so as to minimize the makespan. We develop optimality conditions to determine subplot sizes for the general problem, and present polynomial-time algorithms

to determine optimal subplot sizes for the assembly system with two and three subassembly machines at stage 1.

In Chapter 5, we extend the above single-lot, lot streaming problem for the two-stage assembly system to multiple lots, but still, for the objective of minimizing the makespan. Due to the presence of multiple lots, we need to address the issue of the sequencing of the lots along with lot-splitting, a fact which adds to the complexity of the problem. Some results derived for the single-lot version of this problem have successfully been generalized for this case. We develop a branch-and-bound-based methodology for this problem. It relies on effective lower bounds and dominance properties, which are also derived. Finally, we present results of computational experimentation to demonstrate the effectiveness of our branch-and-bound-based methodology. In particular, our computational experimentations show that our branch-and-bound method is very effective in solving small as well as large-size problem instances as compared with the direct solution of this problem using CPLEX 10.1. A vast majority of problems can be solved to optimality at root node itself by our method. For those which cannot be solved to optimality at root node, our dominance properties further help in obtaining the optimal solution within a reasonable amount of cpu time.

For future work, there are several directions that can be followed. First of all, effective approaches can be developed based on our polynomial-length formulations for the HMATSP and Chesapeake problems. Secondly, the column generation-based method that we have developed for the PPMSP can further be explored to generate appropriate columns by exploiting some inherent structural properties. Thirdly, there are two viable directions for future work with regard to the lot streaming problem that we have considered. These are: development of an algorithm for the general $m + 1$ problem for the single-lot problem that we presented in Chapter 4, and consideration of a more general assembly system than the two-stage system that we have focused on in this dissertation.

Bibliography

- Amentano, V. A., P. M. Franca, F. M. B. de Toledo. 1999. A network flow model for the capacitated lot-sizing problem. *OMEGA* **27** 275–284.
- Asano, M., H. Ohta. 1996. Single machine scheduling using dominance relation to minimize earliness subject to ready and due times. *International Journal of Production Economics* **44** 35–43.
- Asano, M., H. Ohta. 1999. Scheduling with shutdowns and sequence dependent setup times. *International Journal of Production Research* **37** 1661–1676.
- Baker, K. R. 1995. Lot streaming in the two-machine flow shop with setup times. *Annals of Operations Research* **57** 1–11.
- Baker, T., J. A. Muckstadt. 1989. The ches problems. *Technical Paper, Chesapeake Decision Sciences, Inc.* .
- Balakrishnan, N., J. J. Kanet, Sri. V. Sridharan. 1998. Early/tardy scheduling with sequence dependent setups on uniform parallel machines. *Computers and Operations Research* **26** 127–141.
- Barany, I., T. J. Van Roy, L. A. Wolsey. 1984. Strong formulations for multi-item capacitated lot-sizing. *Management Science* **30** 1255–1261.
- Barnes, J. W., L. K. Vanston. 1981. Scheduling jobs with linear delay penalties and sequence dependent setup costs. *Operations Research* **29** 146–154.

- Belvaux, G., L. A. Wolsey. 2001. Modeling practical lot-sizing problems as mixed integer programs. *Management Science* **47** 993–1007.
- Billington, P. S., J. O. McClain, L. J. Thomas. 1983. Mathematical programming approaches to capacity-constrained mrp systems: review formulation and problem reduction. *Management Science* **29** 1126–1141.
- Bitran, G. R., S. M. Gilbert. 1990. Sequencing production on parallel machines with two magnitudes of sequence-dependent setup cost. *Journal of Manufacturing Operations Management* **3** 24–52.
- Bukchin, J., M. Tzur, M. Jaffe. 2002. Lot splitting to minimize average flow time in a two-machine flowshop. *IIE Transactions* **34** 953–970.
- Cattrysse, D., J. Maes, L. N. van Wassenhove. 1990. Set partitioning and column generation heuristic for capacitated dynamic lot-sizing. *European Journal of Operational Research* **46** 38–47.
- Centinkaya, F. C. 1994. Lot streaming in a two-stage flow shop with set-up, processing and removing times separated. *Journal of Operations Research Society* **45**(12) 1445–1455.
- Centinkaya, F. C., M. S. Kayaligil. 1992. Unit sized transfer batch scheduling with setup times. *Computers and Industrial Engineering* **22**(2) 177–182.
- Chen, J., G. Steiner. 1999. Discrete lot streaming in 2 machine flow shops. *Information Systems and Operations Research* **37**(2) 160–173.
- Chen, W. H., J. M. Thizy. 1990. Analysis of relaxations for the multi-item capacitated lot-sizing problem. *Annals of Operations Research* **26** 29–72.
- Cosmadakis, S. S., C. H. Papadimitriou. 1984. The traveling salesman problem with many visits to few cities. *SIAM Journal on Computing* **13** 99–108.

- Dastidar, S. G., R. Nagi. 2005. Scheduling injection molding operations with multiple resource constraints and sequence dependent setup times and costs. *Computers and Operations Research* **32** 2987–3005.
- Diaby, M., H. C. Bahl, M. H. Karwan, S. Zionts. 1992. A lagrangean relaxation approach for very large scale capacitated lot-sizing. *Management Sciences* **38** 1329–1339.
- Eppen, G. D., R. K. Martin. 1987. Solving multi-item lot-sizing problems using variable redefinition. *Operations Research* **35** 832–848.
- Fleischmann, B. 1994. The discrete lot-sizing and scheduling problem with sequence-dependent setup costs. *European Journal of Operational Research* **75** 395–404.
- Gilmore, P. C., R. E. Gomory. 1964. Sequencing a one state-variable machine: a solvable case of the traveling salesman problem. *Operations Research* **12** 665–679.
- Grigoriev, A., J. van de Klundert. 2006. On the high multiplicity traveling salesman problem. *Discrete Optimization* **3** 50–62.
- Gupta, D., T. Magnusson. 2005. The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times. *Computers and Operations Research* **32** 727–747.
- Hariri, A. M. A., C. N. Potts. 1997. A branch and bound algorithm for the two-stage assembly scheduling problem. *European Journal of Operational Research* **103** 547–556.
- Hasse, K., A. Kimms. 2000. Lot sizing and scheduling with sequence-dependent setup costs and times and efficient rescheduling opportunities. *International Journal of Production Economics* **66** 159–169.
- Hindi, K. S. 1995. Computationally efficient solution of multi-item capacitated lot sizing problems. *Computers and Industrial Engineering* **28** 115–135.

- Johnson, S. M. 1954. Optimal two- and three-stage production schedule with setup times included. *Naval Research Logistics* **1**(1) 61–68.
- Kalir, A., S. C. Sarin. 2000. Evaluation of potential benefits of lot streaming in flow-shop systems. *International Journal of Production Economics* **66** 131–142.
- Kalir, A., S. C. Sarin. 2003. Constructing near optimal schedules for the flow shop lot streaming problem with subplot attached setups. *Journal of Combinatorial Optimization* **7** 23–44.
- Kang, S., K. Malik, L. J. Thomas. 1999. Lotsizing and scheduling on parallel machines with sequence-dependent setup costs. *Management Science* **45** 273–289.
- Lee, C-Y, T. C. E. Cheng, B. M. Lin. 1993. Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem. *Management Science* **39**(5) 616–625.
- Leung, J. M., T. L. Magnanti, R. Vachani. 1989. Facets and algorithms for capacitated lot sizing. *Mathematical Programming* **45** 331–359.
- Manne, A. 1960. On the job-shop scheduling problem. *Operations Research* **8** 219–223.
- Meyr, H. 2002. Simultaneous lotsizing and scheduling on parallel machines. *European Journal of Operational Research* **139** 277–292.
- Pinedo, M. 2002. *Scheduling: theory, algorithms and systems*. Prentice Hall, Englewood Cliffs, NJ.
- Potts, C. N., K. R. Baker. 1989. Flow shop scheduling with lot streaming. *Operation Research Letters* **8** 297–303.
- Rabadi, G., M. Mollaghasemi, G. Anagnostopoulos. 2004. A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. *Computers and Operations Research* **31** 1727–1751.
- Sarin, S. C., P. Jaiprakash. 2007. *Flow shop lot streaming*. Springer, New York, New York.

- Sarin, S. C., H. D. Sherali, A. Bhootra. 2005. New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints. *Operations Research Letters* **33** 62–70.
- Sarin, S. C., H. D. Sherali, P. Tsai. 2008. Hot strip rolling scheduling: a precedence constrained multiple-asymmetric traveling salesman problem approach. *Virginia Polytechnic Institute and State University, working paper* .
- Sen, A., E. Topalglu, O. Benli. 1998. Optimal streaming of a single job in a two stage flow shop. *European Journal of Operational Research* **110** 42–62.
- Shah, N. 2004. Pharmaceutical supply chains: key issues and strategies for optimisation. *Computers and Chemical Engineering* **28** 929–941.
- Sherali, H. D., W. P. Adams. 1990. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal of Discrete Mathematics* **3** 83–106.
- Sherali, H. D., W. P. Adams. 1994. A hierarchy of relaxations and convex hull characteristics for mixed-integer zero-one programming problems. *Discrete Applied Mathematics* **52** 83–106.
- Sherali, H. D., S. C. Sarin, P. F. Tsai. 2006. A class of lifted path and flow-based formulations for the asymmetric traveling salesman problem with and without precedence constraints. *Discrete Optimization* **3** 20–32.
- Sriskandarajah, C., E. Wagneur. 1999. Lot streaming and scheduling multiple products in 2-machine no-wait flowshops. *IIE Transactions* **31** 695–707.
- Sun, X., K. Morizawa, H. Nagasawa. 2003. Powerful heuristics to minimize makespan in fixed, 3-machine, assembly-type flowshop scheduling. *European Journal of Operational Research* **146** 498–516.

- Thizy, J. M., L. N. van Wassenhove. 1985. Lagrangean relaxation for the multi-item capacitated lot-sizing problem. *IIE Transactions* **17** 308–313.
- Trietsch, D. 1987. Optimal transfer lots for batch manufacturing. *Manuscript presented at the ORSA/TIMS Conference* .
- Trietsch, D., K. R. Baker. 1993. Basic techniques for lot streaming. *Operations Research* **41**(6) 1065–1076.
- Vickson, R. G. 1995. Optimal lot streaming for multiple products in a two-machine flow shop. *European Journal of Operations Research* **85** 556–575.
- Wong, R. T. 1980. Integer programming formulations of the traveling salesman problem. *Proceedings of the IEEE international conference on circuits and computers, Part I*. New York, NY, 149–152.
- Yalaoui, F., C. Chu. 2003. An efficient heuristic approach for parallel machine scheduling with job-splitting and sequence-dependent setup times. *IIE Transactions* **35** 183–190.
- Zhu, X., W. E. Wilhelm. 2006. Scheduling and lot sizing with sequence-dependent setup: A literature review. *IIE Transactions* **38** 987–1007.