Optimal and Approximate Algorithms for the Multiple-Lots-per-Carrier Scheduling and Integrated Automated Material Handling and Lot Scheduling Problems in 300mm Wafer Fabs

Lixin Wang

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

Doctor of Philosophy
in
Industrial and Systems Engineering

August 20, 2008

Blacksburg, Virginia

Keywords: Multiple Lots per Carrier Scheduling, Integrated AMHS and Lot Scheduling, Makespan, Total Completion Time, AMHS, 300mm Wafer Fabs

Optimal and Approximate Algorithms for the Multiple-Lots-per-Carrier Scheduling and Integrated Automated Material Handling and Lot Scheduling Problems in 300mm Wafer Fabs

Lixin Wang

(Abstract)

The latest generation of semiconductor wafer fabs produce Integrated Circuits (ICs) on silicon wafers of 300mm diameter. In this dissertation, we address the following two types of (new) scheduling problems that are encountered in this generation of wafer fabs: multiple-lots-per-carrier scheduling problem (MLCSP) and integrated automated material handling and lot scheduling problem (IMHLSP). We consider several variations of the MLCSP depending upon the number of machines used, the prevailing processing technology of the machines, and the type of objective functions involved. For the IMHLSP, we study two instances, one with infinite number of vehicles and the other with finite number of vehicles.

We begin by introducing a single-machine, multiple-lots-per-carrier with single-wafer-processing-technology scheduling problem for the objective of minimizing the total completion time (MLCSP1). The wafer carrier is a front-opening unified pod (FOUP) that can hold a limited number of wafers. The problem is easy to solve when all the lots are of the same size. For the case of different lot sizes, we first relax the carrier (FOUP) capacity and propose a dynamic programming-based algorithm, called RelaxFOUP-DP, which enables a quick determination of its optimal solution that serves as a lower bound for the problem with limited FOUP capacity. Then, a branch-and-bound algorithm, designated as MLCSP1-B&B, is developed that relies on the lower bound determined by the RelaxFOUP-DP algorithm. Numerical tests indicate that MLCSP1-B&B finds optimal solutions much faster than the direct solution of the MLCSP1 model by the

AMPL CPLEX 10.1 Solver. In fact, for the medium and low density problems, the MLCSP1-B&B algorithm finds optimal solutions at the starting node (node zero) itself.

Next, we consider a single-machine, multiple-lots-per-carrier with single-carrier-processing-technology scheduling problem for the objective of minimizing total completion time (MLCSP2). As for the case of MLCSP1, the optimal solution for the case in which all the lots are of the same size can be obtained easily. For the case of different lot sizes, we determine a lower bound and an upper bound for the problem and prove the worst-case ratios for them.

Subsequently we analyze a two-machine flow shop, multiple-lots-per-carrier with single-wafer-processing-technology scheduling problem for the objective of minimizing the makespan (MLCSP3). We first consider a relaxed version of this problem, and transform the original problem to a two-machine flow shop lot streaming problem. Then, we propose algorithms to find the optimal capacitated sublot sizes for the case of lots with (1) the same ratio of processing times, and, (2) different ratios of processing times on the machines. Since the optimal solutions obtained from the lot streaming problem may not be feasible to the MLCSP3, we develop heuristic methods based on the heuristic procedures for the bin packing problem. We develop four heuristic procedures for lots with the same ratio of processing times, and another four procedures for lots with different ratios of processing times on the machines. Results of our numerical experimentation are presented that show that our heuristic procedures generate almost optimal solutions in a matter of a few seconds.

Next, we address the integrated automated material handling and lot scheduling problem (IMHLSP) in the presence of infinite number of vehicles. We, first, propose a new strong hybrid model, which has the advantages of both segregate and direct models. In the segregate model, a job is always transferred to the stocker after its completion at a station, while in the direct model, it is transferred to the next machine in case that machine can accommodate the jobs; otherwise, the job will stay at current station. The decisions involved in the strong hybrid model are the sequence in which to process the lots and a selection between the segregate and direct models for each lot, whichever optimizes system performance. We show that, under certain conditions about the processing times of the lots, the problem can be approximated by the cases of either infinite buffer or zero-buffer at the machines. Hence, we consider all three cases of the IMHLSP in this chapter, namely, infinite buffer, zero-buffer, and limited buffer sizes. For the strong hybrid model with limited buffer size, we propose a branch-and-bound algorithm, which uses a modified Johnson's algorithm to determine a lower bound. Two upper bounds for this algorithm are also determined. Results of our numerical investigation indicate that our algorithm finds optimal solutions faster than the direct solution of the IMHLSP model by the AMPL CPLEX 10.1 Solver. Experimental results also indicate that for the same problem size, the times required to solve the IMHLSP model with interbay movements are larger than those for intrabay movements.

Finally, we investigate the IMHLSP in the presence of limited number of vehicles. Due to the complex nature of the underlying problem, we analyze small-size versions of this problem and develop algorithms for their solution. For some of these problems, we can find optimal solutions in polynomial time. Also, based on our analysis on small-size systems, we have shown why some real-time dispatching (RTD) rules used in real fabs are expected to perform well while not the others. In addition, we also propose some new and promising RTD rules based on our study.

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my advisor, Dr. Subhash C. Sarin, for his invaluable guidance, support, motivation and expertise throughout the entire progress of my dissertation research. I have learnt so much from working with him and taking classes from him. His patience has endured my obstinate anxiety during the countless hours in research meetings. I specially appreciate his great help and encouragement during the last several months while I was in transition from a student to a full time engineer. I want to offer my heart-felt thanks to my committee member, Dr. Kimberly Ellis, who has helped me in many ways, including and beyond my dissertation. It has been pleasant experience to be a teaching assistant for her. I owe much to my committee member, Dr. Barbara Fraticelli, for her kindness and encouragement during our discussions. She is such a considerate person. I also want to give thanks to my committee member, Dr. Robert Hendricks for his brain-storming style discussions with me about semiconductor industry. His opinions and observations from his years of experience in semiconductor industry have been very valuable.

I am extremely grateful to my friends in Blacksburg for making my life here so wonderful. I enjoyed being a roommate with Chengbin Zhu and Wenwei Zhong. Thanks to my good friends: Ming Chen, Weiping Chen, Ming Cheng, Ying Fu, Cheng Guo, Seonki Kim, Feng Li, Lingrui Liao, Yunkai Lu, Shunying Qiu, Xiangshan Tong, Yong Yang, Yuqiang Wang, Liming Yao, Xiaomei Zhu, Yueqin Zhao, …, thank you all for the good times we shared.

Finally, I am indebted to my family members for their unconditional love and constant support. I thank my wife (Lingling Zhuang) for resigning her job with Intel (Shanghai) and accompanying me to Blacksburg for years. She may not be conversant with my research area but she understands me and has always been there for me no matter what. I thank my lovely baby boy, Kenneth Yichen Wang. He brings happiness and joys to my life everyday. I thank my parents-in-law (Guizhang Zhuang and Cuihua Wang) for taking care of my newborn baby. Without them, it would have been impossible for me to finish my study in time. I appreciate my parents (Mingren Wang and Shibao Zhu) and grandfather (Zhangxiu Wang) for their understanding and tolerance in accepting my absence from home for all these years. I appreciate my sisters (Xuewei Wang and Xueping Wang) for taking care of my parents and grandfather when I have not been around. I would like to dedicate this dissertation to my family members, as an infinitesimal return for their love and support.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background and motivation

Semiconductor manufacturing is the process of creating integrated circuits (ICs), or chips, that are used in a variety of electrical and electronic devices. The process starts from a pure semiconductor material (the most commonly used material is silicon) and ends with packaged chips ready for shipment to customers. It consists of four main stages: wafer fabrication, wafer test, packaging and final test (see Figure 1.1). The manufacturing of wafers in which a cylindrical ingot (high purity crystalline silicon) is sliced with an inner diameter diamond coated blade and polished to form wafers, is usually done by a specialized company and is considered outside of these four stages. Wafer fabrication generally refers to the process of building integrated circuits on silicon wafers and is performed in highly specialized facilities referred to as fabs. Wafer testing takes place in between various processing steps and verifies if the wafers are good for acceptance or are damaged from previous processing steps, and need repair or are to be disposed of. Next, a wafer is broken into individual die and each undergoes requisite processes and becomes a device which is able to perform the functions for which it is designed. This is the packaging stage. Finally, the devices are subjected to a variety of electrical tests to determine if they function properly. At the packaging and final test stages, the processing units are individual die (chips), and not a wafer. Thus, wafer fabrication and wafer test are usually referred to as front-end processing while packaging and final test are called back-end processing (Plummer et al. 2000).

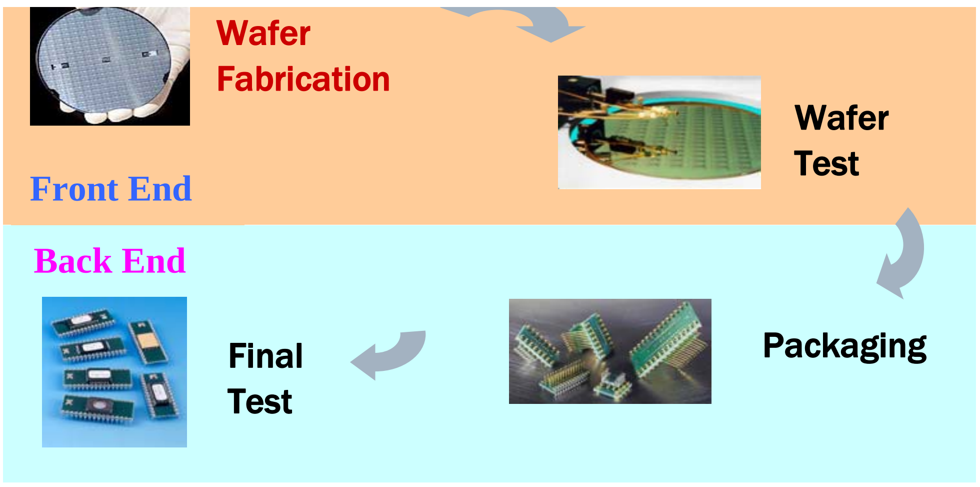

Figure 1.1: The stages of semiconductor manufacturing

Of the four stages, wafer fabrication requires the longest time (usually 6 to 8 weeks), uses the most resources, and is the most value-added stage. More importantly, wafer fabrication has been called one of the most complex processes in the manufacturing domain. A single product may contain as many as 1600 steps and a fab can produce as many as 200 products. Machines differ widely from each other. Some are capable of only serial processing, such as photolithography and etching, while others permit batch processing, such as furnaces and ovens used in diffusion, oxidation and ion implantation. Processing times can vary from 1 hour to 24 hours depending on the process involved, and sequence-dependent setups are not unusual. Wafers are built in layers with each layer undergoing sophisticated processing at multiple processors, like steppers. As these machines are rather expensive (up to 30 million dollars), there are only a very limited number of them in a fab. Therefore, wafer lots have to revisit these machines for the processing of different layers. This gives rise to the characteristic re-entrant flow, with wafers at different stages of processing having to compete for time on the same machine (see Figure 1.2). Specific requirements for the lots, such as lot dedication to ensure that all (or at least critical) layers of a lot are processed at the same stepper only add to the complexity of the situation.

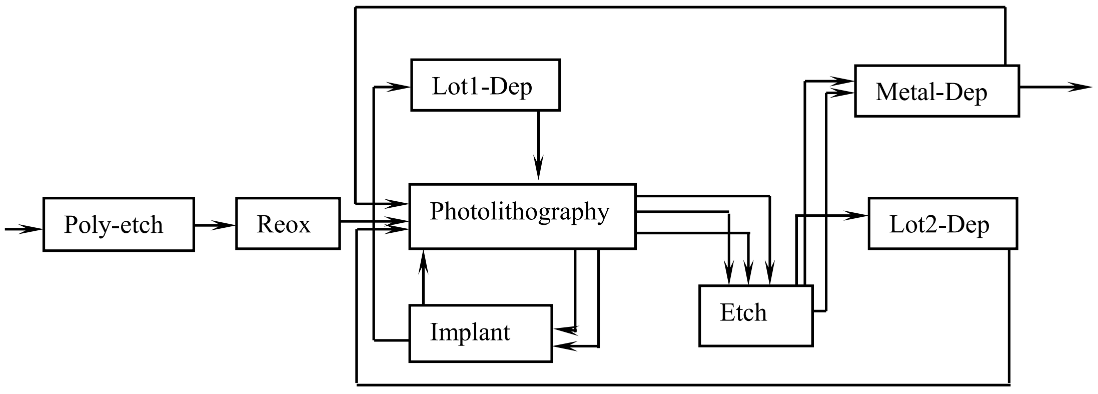

Figure 1.2: A simplified 16-step production process with re-entrant flow in wafer fabrication

The capital expense for a wafer fab (most of which is for equipment) can be as high as 3 billion dollars. Furthermore, the equipment is, typically, replaced with newer generations in 5~10 years. Thus, in order to gain profits from this huge investment, wafer fabs must utilize the equipment as much as possible. On the other hand, the semiconductor industry is also facing strong competition. Most of the players have identical abilities to undertake research and development of new products (except for Intel, which has unmatched advantage over its competitors (Wilson 2007)). Thus, to survive in the market, fabs must be able to reduce manufacturing costs. The following three methods have been implemented to reduce manufacturing costs in fabs.

(i) Achievement of economic of scale

When wafer size (the diameter of a wafer) increases, the number of dies or chips that are built on the wafer increases dramatically. Thus, the number of chips produced increases without using more wafers. Consequently, the cost per chip decreases. Semiconductor manufacturers have been consistently experiencing this trend (see Figure 1.3) (SEMATECH 2005). On the other hand, increment of wafer size also leads to increment in investment on the equipment in the fab (see Figure 1.4) (SEMATECH 2005). This further increases the pressure of reducing manufacturing cost. Thus, this method itself is not enough to drive cost down.

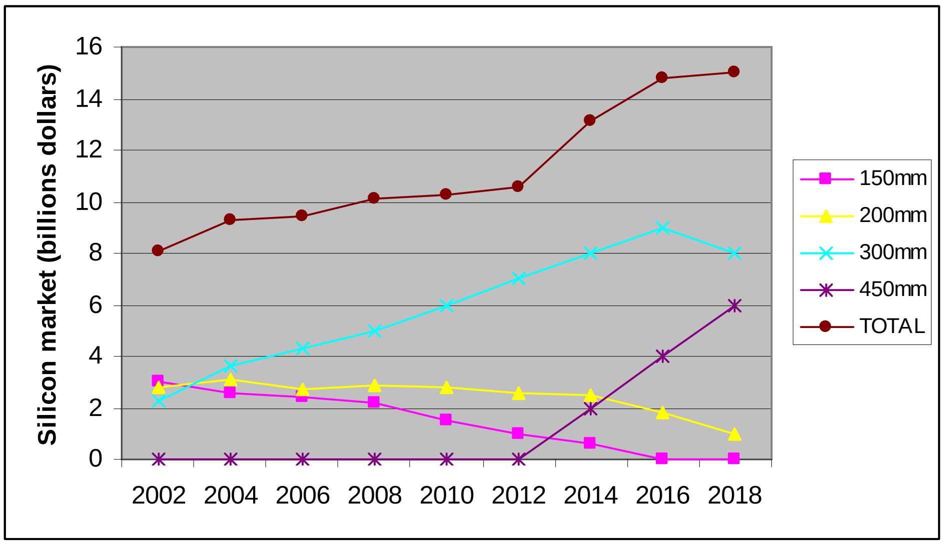

Figure 1.3: Market forecast by wafer size

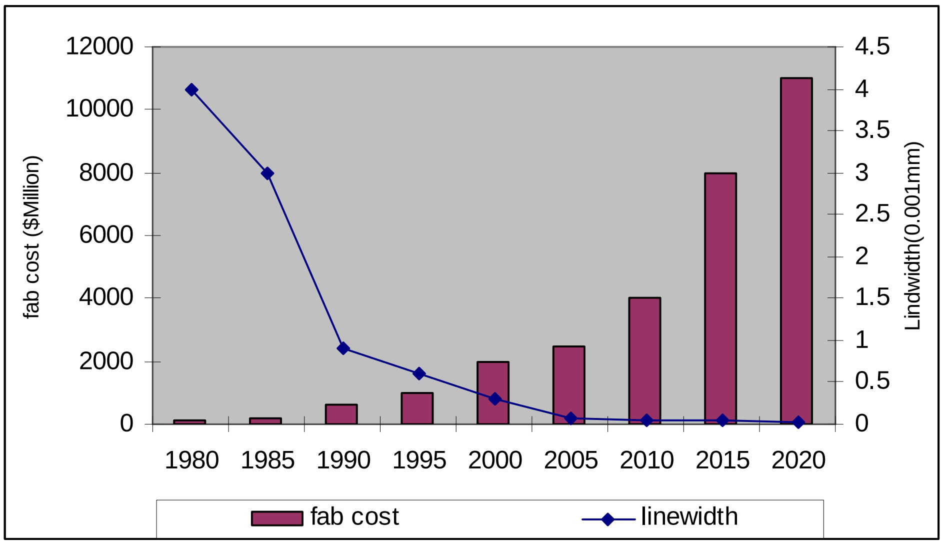

Figure 1.4: Increasing cost of wafer fabs (SEMATECH 2005)

(ii) Applications of management science and operations research techniques

The application of industrial engineering/operation research techniques in wafer fabrication started four decades ago, almost from the day the first fab in the world was built. For the implementation of these techniques, a decision-making hierarchy is used, which comprises of four levels: strategic planning, tactical planning, operational planning and shop floor scheduling and control (see Figure 1.5). The strategic planning level pertains to long term decision making (generally for 3-10 years). The types of decisions that are considered at this level include selection of the location of a facility, determination of the required capacity, selection of products and technology to use. The tactical planning level addresses decisions over a shorter time horizon (generally from 6 months to 3 years). It includes determination of workforce levels, process routings and production rates. The operational planning level covers decisions that are made for a planning horizon of 1 week - 6 months. These include allocation of jobs to machines and determination of lot sizes in which to process the jobs, overtime usage, and amount of subcontracting/outsourcing. The lowest level, belonging to the shop floor operations, mostly involves decisions pertaining to the scheduling and control of work over a period that spans from real-time to one week. These also include determination of processing lot sizes, lot release strategies, processing sequence as well as the dispatching of lots to the machines (determination of starting times).

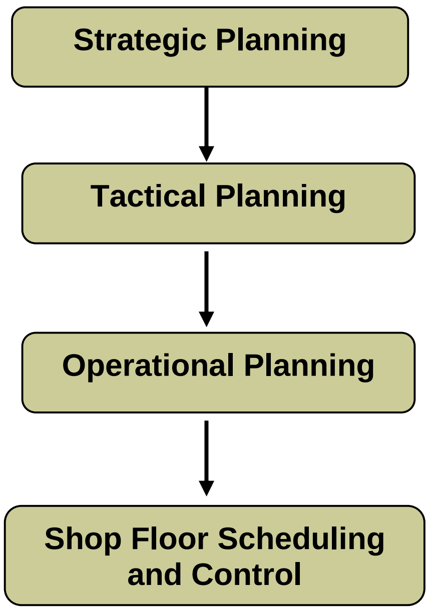

Figure 1.5: The decision making hierarchy in wafer fabs

(iii) Implementation of the Toyota Production System (TPS)

For more than 30 years, Toyota Motor Corporation has followed a production system that has enabled it to increase quality, double capacity, produce a wider variety of models in a given factory, and change the mix on a dime. In 2007, Toyota surpassed General Motors Corporation in becoming the largest automobile manufacturer in the world. But more importantly, Toyota's approach to mass production has produced bountiful profits. It became the first in the automobile history to gain profit over 10 billion dollars in one year. The Toyota Production System (TPS) is so successful that it has been widely applied in other industries, such as restaurant, aircraft manufacturing, banking and finance, among others.

After working hard on methods (i) and (ii) described above, and having not succeeded as expected, recently, scholars and executives found a glimmer of hope from the TPS. Christensen et al. (2008) have reported their innovative work to emulate the TPS, and have applied its principles to a logic fab belonging to an integrated device manufacturer (IDM). The four principle rules of the TPS, in summary, are (1) highly specify activities, (2) clearly define the transfer of material and information, (3) keep the pathway for every product and service simple and direct, and (4) detect and solve problems where and when they happen, using the scientific method (Spear and Bowen 1999). After 7 months, it is reported that the company was able to reduce the manufacturing cost per wafer by 12 percent and the cycle time by 67 percent, without investing in new equipment or changing product design.

It should be noted that there have not been many examples reported in the literature about successful implementation of the TPS in fabs, especially memory fabs. Logic and foundry fabs are high mix/high volume fabs, while a memory fab involves fewer products. The system which works well in a logic fab may not be promising in a memory fab, which is a low mix/high volume fab. However, we believe that the TPS will eventually bring new economics to semiconductor manufacturing while, at the current stage, it has just been a start.

1.2 Scope of dissertation

In this dissertation, we focus on the scheduling problems arising from the 300mm wafer fabs. Thus, it is a combination of methods (i) and (ii) (at the shop floor scheduling and control level of the decision making hierarchy) to reduce the manufacturing costs mentioned above. Two types of scheduling problems are addressed. The first of these is the multiple-lots-per-carrier scheduling problem (MLCSP). Since a 300mm wafer is much larger than a 200mm wafer, the size of a lot (a group of wafers processed in the fab as a non-split entity) may be less than the capacity of a carrier K (K=25 wafers). Therefore to reduce the number of carriers needed, which would save the cost of carriers, and more importantly, reduce the number of movements undertaken by the Automated Material Handling Systems (AMHS), one idea is to include more than one lot in a carrier, thus leading to the MLCSP. It involves decisions pertaining to which lots to include in the carrier as well as the sequence in which to process the carriers on the machines. Depending on the processing technology employed, number of machines involved, and the objective function used, the MLCSP can be further classified into four sub-problems: single-machine MLCSP with single-wafer-processing-technology and the objective of minimizing total completion time (MLCSP1), single-machine MLCSP with single-carrier-processing-technology and the objective of minimizing total completion time (MLCSP2), two-machine flow shop MLCSP with single-wafer-processing-technology and the objective of minimizing makespan (MLCSP3), and two-machine flow shop MLCSP with single-carrier-processing-technology and the objective of minimizing the makespan (MLCSP4). MLCSP4 is equivalent to the bin packing problem, thus it is not considered in this study. These four sub-problems are summarized in table 1.1.

Table 1.1 Descriptions of the four sub-problems of the MLCSP

Category	Number of machines	Machine processing technology	Performance measure	Presented in this study?
MLCSP1	1	Single wafer processing	Total completion time	Yes
MLCSP2	1	Single carrier processing	Total completion time	Yes
MLCSP3	2	Single wafer processing	Makespan	Yes
MLCSP4	2	Single carrier processing	Makespan	No

The second problem that we address is the integrated AMHS scheduling and lot scheduling problem (IMHLSP). A 300mm fab is also a fully automated fab where the AMHS is responsible for every movement of a lot from a machine to another machine or to a temporary storage (see Figure 1.6). Manual carrying of a carrier is minimized. The AMHS scheduling and lot scheduling is closely inter-related. Lot scheduling provides delivery requests and impacts operational control of the AMHS. The AMHS scheduling, especially vehicle scheduling, provides the release time for processing lots on downstream machines. Therefore, a schedule determined by integrating AMHS delivery issues and lot scheduling will perform better than that determined by considering these two aspects independently from the view point of overall fab performance metrics, such as cycle time, and throughput rate, among others. The IMHLSP can be further classified into two sub-problems: IMHLSP with infinite vehicle capacity and IMHLSP with finite vehicle capacity. If enough vehicles are operated in a 300mm fab, it can be viewed to operate under infinite vehicle capacity. Otherwise, we have the case of finite vehicle capacity. For both of these sub-problems, the decisions pertain to determining the sequence in which to process the lots on the machines and the movements of vehicles from one machine to another.

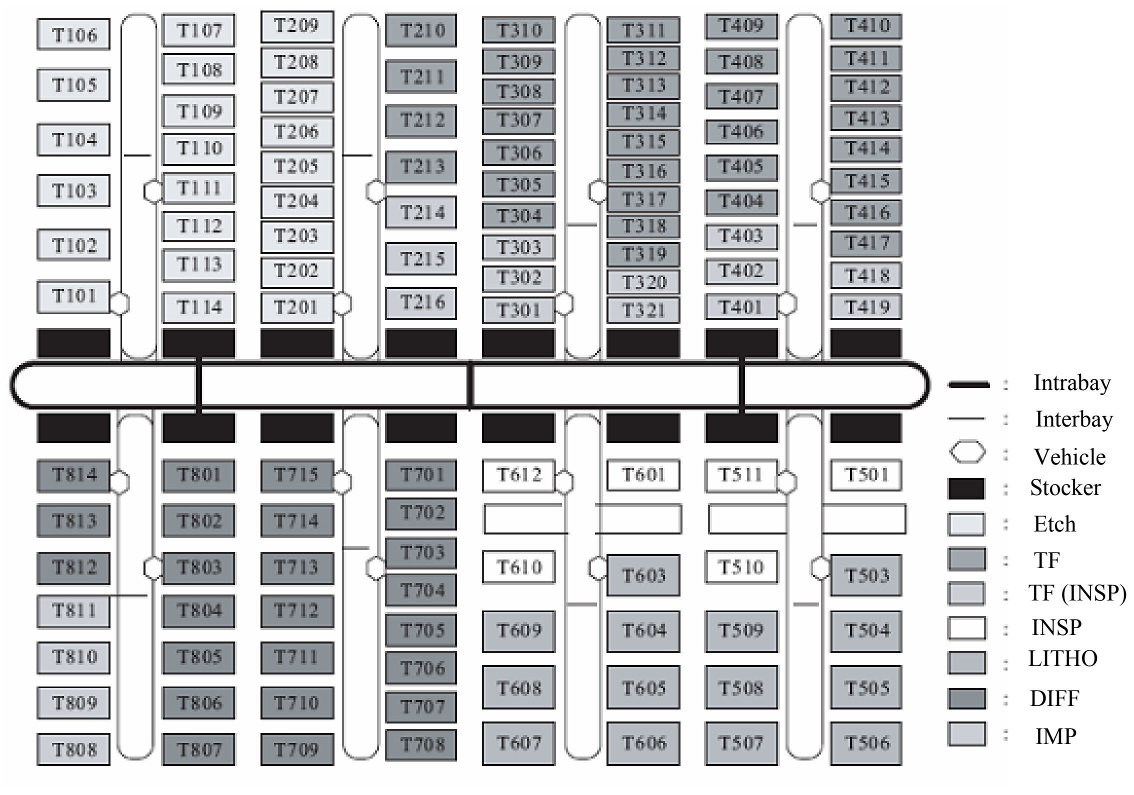

Figure 1.6 A simplified AMHS layout

1.3 Research methodology

This study mainly focuses on the scheduling level of the hierarchical structure described in Section 1.1. Therefore, we make the following two assumptions: (1) The fab planning has been in place. That is, the numbers of tools, tool layout, number of products and the desired fab throughput have all been determined. (2) The AMHS is at stable stage. This means that the AMHS is running well and does not need significant changes. The AMHS layout is not to be revised. The number of vehicles and vehicle routing are also known.

The scheduling problem in a 300mm fab is an extremely large-size problem. It involves hundreds of machines, thousands of processing steps, and work-in-process (WIP) of the order of tens of thousand of wafers. It is not realistic to study such a large-size problem as a whole. Thus, in this study, we start with a small-size problem. This small-size problem has one to two machines, and one to two vehicles. The advantage of focusing on the small-size problem is that it would enable identification of inherent properties that could potentially be exploited for the solution of large-size problems as well via the development of optimum seeking methods or real-time dispatching (RTD) rules. The graphical depiction of a wafer fab is shown in Figure 1.7. The fab, presented on the left, is approximated in our study by a system consisting of one or two machines. The solution methodology is depicted by blocks on the right side with their eventual implementation in the entire fab.

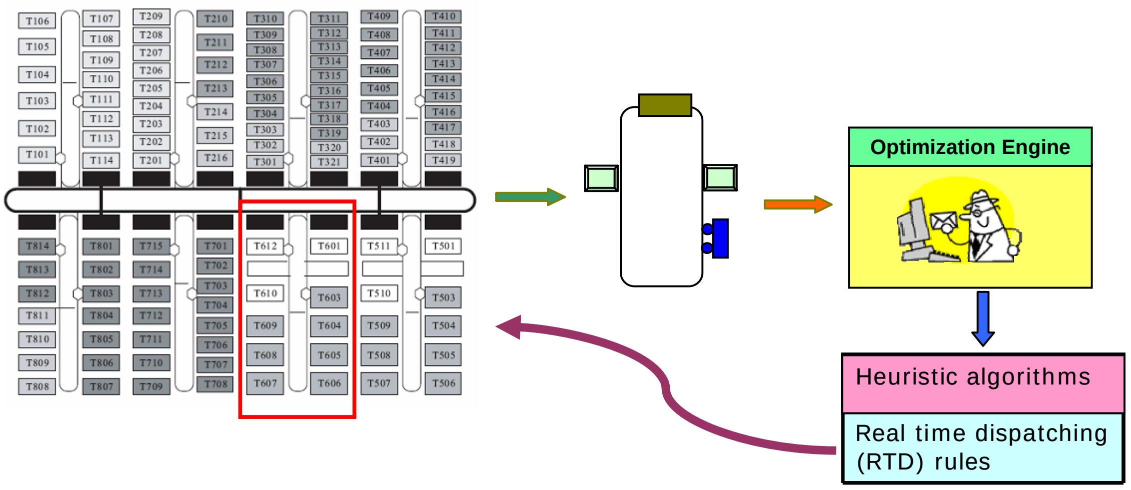

Figure 1.7 Schematic of the research methodologies in this study

1.4 Research objectives

The primary objective of our work is to provide insights and effective solution methodologies for the two types of new scheduling problems encountered in 300mm fabs, namely, MLCSP and IMHLSP. These two problems are faced in real-life environments and have drawn the attention of engineers and management teams. However, very few results have been reported on these problems. Through this dissertation research, we provide optimal or near-optimal solutions for these problems, which, at the same time, are easy to implement in real-life fabs. On the other hand, through this pioneering work, we hope to promote the interest of others as well to work further on these and related problems.

The specific objectives of our research work are as follows:

- To study the impact of including multiple lots in a carrier on the performance measure of total completion time and makespan, where the lots are processed either on a single machine or a two-machine flow shop, and the machines use either a single-wafer-processing technology or single-carrier-processing technology, and to develop efficient solution methodologies to determine an optimal formation of the carriers and the sequence in which to process them.
- To obtain insights into the integration of AMHS scheduling and lot scheduling for the cases with infinite and finite number of vehicles in the 300mm fabs, and to develop effective methodologies for solving small-size problems as well as to develop fast solution procedure for implementation in real-life fabs.

1.5 Organization of dissertation

The rest of this dissertation is organized as follows. In Chapter 2, we discuss the MLCSP1. After formulating the integer programming model, we present some structural properties as well as an efficient branch-and-bound algorithm to solve this problem. Chapter 3 addresses the MLCSP2. We determine a lower bound and upper bound for this problem. The worst case analyses are also presented for both of these bounds. In Chapter 4, we address the MLCSP3, which is first transformed to a two-machine flow shop lot streaming problem. After the lot streaming problem is solved optimally, heuristic procedures are used to transform the solutions obtained to a feasible solution to the original problem. We analyze the IMHLSP with infinite vehicle capacity in Chapter 5, and propose a new AMHS operation model, namely, the Strong Hybrid model. A branch-and-bound algorithm is presented to implement the Strong Hybrid model in a two-machine environment. In Chapter 6, we determine a closed-form expression for small-size IMHLSP for finite vehicle capacity, and recommend several real-time dispatching rules for real-size problems that rely on our analysis of the problem. Finally, we conclude this study in Chapter 7.

Chapter 2: Minimizing Total Completion Time for Single Machine MLCSP with Single-Wafer-Processing-Technology (MLCSP1)

2.1 Introduction

The latest generation of semiconductor wafer fabs produce Integrated Circuits (ICs) on silicon wafers of 300 mm diameter. The area of a 300 mm wafer is 2.25 times larger than that of an older generation 200mm wafer. Furthermore, 300mm wafers have over twice the usable area of 200mm wafers (please see Figure 2.1), thereby delivering up to 2.6 times the number of chips produced on a 200mm wafer (United Microelectronics Corporation 2008). This, in turn, requires use of fewer wafers to produce a certain number of chips.

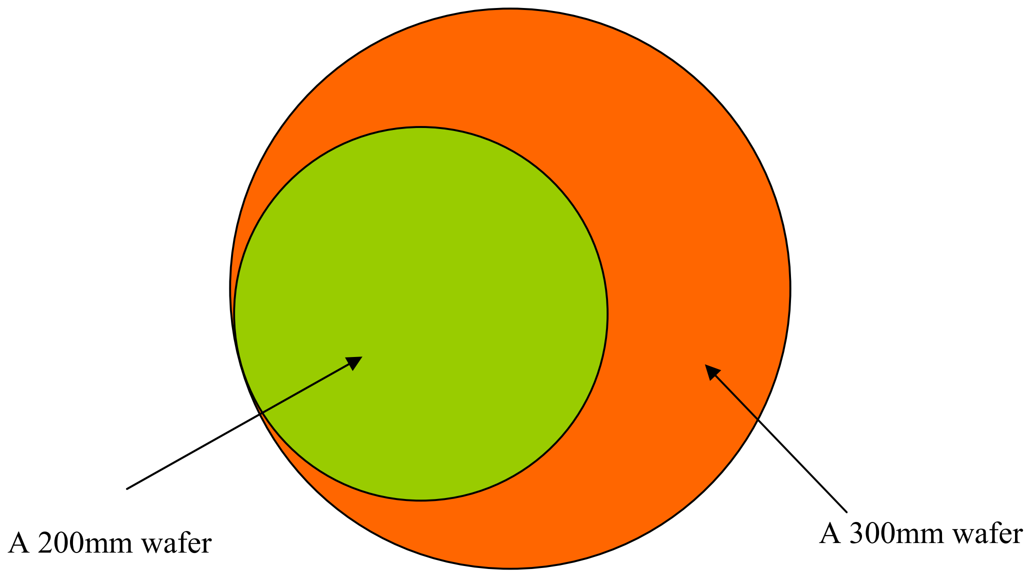

Figure 2.1: The usable area of a 300mm wafer is over twice that of a 200mm wafer

Generally, there are two types of wafer fabs, namely, high volume/low mix, and high volume/high mix. Most of the memory-chip manufacturers belong to the first category, and include manufacturers like Qimonda, Micron, Samsung, among others. A high volume/low mix fab produces several products with 40k to 50k WSPM (Wafer Shipment per Month). According to the ITRS (International Technology Roadmap for Semiconductors) (2006), a high mix fab can be characterized as follows: presence of more than 50 products with many in small lots of 1 – 10 wafers; and production of an average of less than 50 wafers between reticle changes for each lithography equipment. Examples of high volume/high mix fabs include ASIC (Application Specific Integration Circuit) fabs like Analog Devices, and foundries like TSMC (Taiwan Semiconductor Manufacturing Corporation).

The semiconductor industry has established an international standard to transfer wafers between tools in a 300 mm wafer fab. Wafers are carried by a FOUP (Front-Opening Unified Pod, Figure 2.2). This standard carrier has the capacity of 25 wafers. As noted earlier, in a high volume/high mix fab, there are many lots of small size with less than 25 wafers in them. If a FOUP holds only one such lot, it has great ramifications on the transactions and storage loading capabilities of the interbay and intrabay AMHS (Automated Material Handling System) (more information about the AMHS used in a 300mm fab can be found in Chapter 5). Besides, it reduces output of a workstation (as a result of more frequent loading/unloading operations performed).

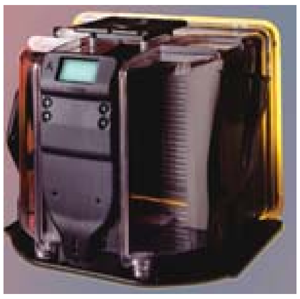

Figure 2.2: A front-opening unified pod (FOUP)

One of the ways to address this issue and improve the output of a workstation of a high volume/high mix fab is to use multiple-lots-per-carrier (ITRS 2006 and 300 mm Integrated Vision for Semiconductor Factories 1999), as shown in Figure 2.3. This strategy aims to minimize the problems described above by loading similar lots, with potentially different processing conditions but with the same process flow, into one carrier. There are two ways of mixing lots under this strategy. One way is to combine lots with the same process flow, but with partially different processing conditions such as metallization layers or wiring patterns. The second way is to combine more than one lot in a carrier, all of which have the same process flow, but some having additional process steps as optional processes. For the first alternative, a process equipment must have the capability to set different processing conditions (different recipes) for each lot or for a subset of the wafers in a carrier. For the second alternative, the process equipment used in the optional process step must have the capability to selectively process one or more wafers in a particular carrier, and leave the other wafers in the carrier unprocessed.

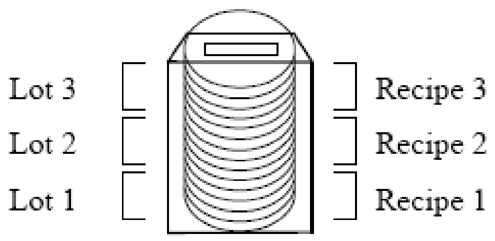

Figure 2.3: Multiple-lots-per-carrier

For the high volume/low mix fabs, there is no advantage of mixing production lots in a carrier. However, it is not uncommon to group engineering lots (i.e., new technology lots, or test lots) in a carrier in such a fab (Fu 2007). This is mainly due to small sizes of engineering lots (much less than 25 wafers, usually 1-3 wafers) as well as shorter completion time requirements.

We consider two types of dependencies between lot processing time and carrier processing time, namely, *single-wafer-processing* and *single-carrier-processing*. Single wafer processing occurs on various tools, such as a photolithography stepper, in which a single wafer is processed at-a-time. Thus, the processing time for a carrier depends on the total number of wafers of the lots in the carrier. Under single carrier processing, the carrier's processing time is independent of the number of wafers it holds. An example of single carrier processing is the operation of a wet sink, wherein the entire carrier (FOUP) of wafers is processed simultaneously when the carrier is submerged in a liquid solution.

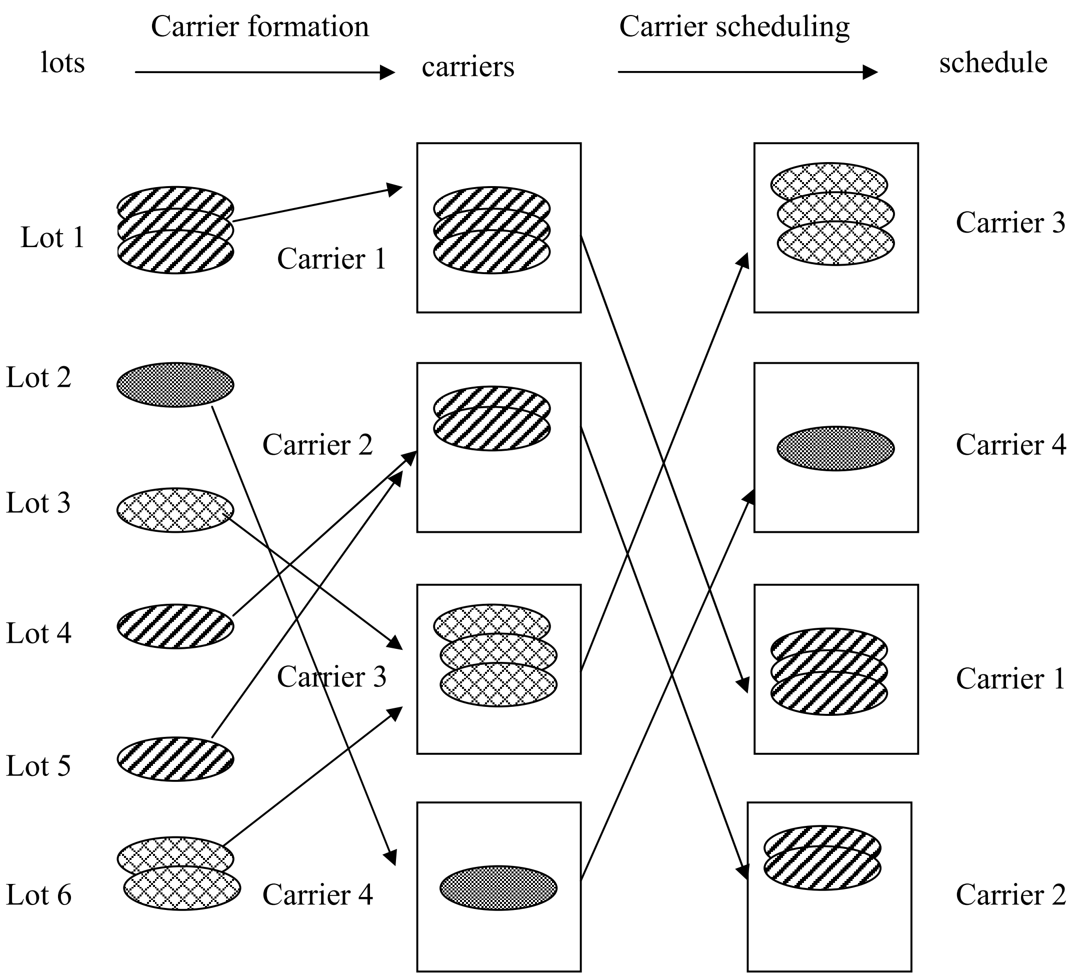

Figure 2.4: Illustration of the MLCSP

To tackle the multiple-lots-per-carrier scheduling problem (designated MLCSP) that is described above for the high volume/high mix fabs or for the special instances of the high volume/low mix fabs, we define it precisely as follows: given a number of partial lots, group them into a given or unrestricted number of carriers, and find a sequence in which to process the carriers on a machine in order to minimize a performance measure such as makespan or total completion time of the lots. The MLCSP is different from traditional scheduling problem in that it requires two types of decisions, namely, carrier formation and carrier sequencing. The formation of carriers from lots, and their sequence for processing by a machine (or a workstation) is depicted in Figure 2.4.

In this chapter, we address the single-machine MLCSP with single-wafer-processing, and designate it as MLCSP1. The performance measure that is considered is the total completion time of the lots. The single machine MLCSP with single-carrier-processing and for the objective of minimizing the total completion time, designated as MLCSP2, will be presented in Chapter 3, while the two-machine MLCSP with single-wafer-processing for the objective of minimizing the makespan, designated as MLCSP3, will be discussed in Chapter 4.

The rest of this chapter is organized as follows. In Section 2.2, we present the research work from the literature that is related to the MLCSP. We, then, formulate an integer programming model for this problem in Section 2.3. By relaxing the capacity of the FOUP, we first derive some useful properties. These properties are used to determine optimal solution for the case of same-size lots. For the case of different lot sizes, we propose a dynamic programming-based algorithm to obtain an optimal solution. This work is presented in Section 2.4. In the presence of limited FOUP capacity (i.e., the original problem), when all the lots are of the same size, the optimal solution for the case when the FOUP capacity is relaxed, is also optimal for this case. When the lots of different sizes are considered, the problem becomes difficult to solve, and we develop a branch-and-bound algorithm for its solution. It relies on a lower bound found by using the DP-algorithm presented in Section 2.4. This work is presented in Section 2.5. Results of our computational experimentation on the use of the proposed branch-and-bound method for the solution of our problem are presented in Section 2.6. In Section 2.7, we explore determination of optimal total cost if the number of carriers is not given. Finally, we conclude this chapter in Section 2.8.

2.2 Literature review

The semiconductor industry has transited from a 200 mm fab to a 300 mm fab. As a result, there are many new scheduling problems related to the 300 mm fabs that are still open. To the best of our knowledge, there has not been any study conducted on the MLCSP problem. However, a similar problem, called multiple-orders-per-job (MOJ) problem, has been addressed in the context of a 300 mm fab by several researchers. Also, the traditional batching and scheduling problem, which has been addressed extensively in the literature, is related to the MLCSP. In addition, the carrier formation part of the MLCSP problem is similar to the bin packing problem. We, briefly, review work on these problems next.

2.2.1 Multiple-Order-per-Job (MOJ) problem

The Multiple-Order-per-Job (MOJ) problem has been addressed by Qu (2004). He developed a nonlinear integer programming model for the general MOJ problem, which was then reformulated as a linear integer programming (IP) model for the objective of minimizing the total weighted tardiness in a single-machine environment. A number of heuristic approaches, including genetic algorithm (GA) and tabu search (TS), were employed due to computational intractability of the IP model. He has presented experimental results, which demonstrate that their heuristic approaches can find good quality solutions in a reasonable amount of computational time. Similar results can be found in Qu and Mason (2004a, 2004b and 2005), and Kutanoglu et al. (2004).

In a follow-up work, Erramilli and Mason (2006) investigated the MOJ problem in a single batch-processing machine environment. The objective of their study was to minimize the total weighted tardiness of orders. They developed a mixed integer programming model for the problem. A new simulated annealing-based heuristic is used, which is demonstrated to solve the problem within 4% of optimality in a few minutes.

Laub et al. (2007) have extended the MOJ problem to a two-machine flow shop. A heuristic procedure was proposed to minimize the makespan. First, the heuristic procedure relaxes the constraint that the entire order can only be assigned to one job. The relaxed problem becomes a two-machine lot streaming problem, which can be solved easily. The optimal solution to the lot streaming problem is a lower bound for the original problem. Using the heuristic procedure, the orders are re-assigned to the jobs so that one entire order is assigned to only one job. It has been shown that this heuristic procedure obtains solutions within 2% of the lower bound.

It should be noted that even in a foundry fab, it cannot be guaranteed that all or a large percent of customer orders will be less than 25 wafers. When a customer order is larger than 25 wafers, we can not simply separate it into two parts: the splinters (leftover items after an order is divided by 25) and the entire carriers (consisting of 25 wafers), and then, schedule them separately, with the splinters possibly grouped with other splinters. Thus, the MOJ problem actually addresses a scenario which may have limited realistic applications.

The MOJ problem also assumes that all the orders require the same processing times. However, for the MLCSP, lots in the same carrier may require different processing conditions (and as a result different processing times) as described in section 2.1 and studied in Chapter 4. Thus, even though the MLCSP is different from the MOJ problem, yet the MOJ problem can provide useful insights for our study of the MLCSP.

2.2.2 Batching and scheduling problem

Another type of problem which is related to the MLCSP is batching and scheduling problem. The motivation for batching the jobs is to gain efficiency because it may be cheaper or faster to process the jobs in a batch than to process them individually. In a *family scheduling model*, jobs are partitioned into families according to their similarity so that no setup is required for a job if it belongs to the family of the previously processed jobs (see Potts and Kovalyov (2000), and Webster and Baker (1995)). In this model, a batch is a maximal set of jobs that are scheduled contiguously on a machine and share a setup.

There are two variants of the family scheduling model depending on when the jobs become available (either for dispatching to a customer or for processing on the next machine). Under *batch availability*, a job becomes available only when the complete batch to which it belongs has been processed. An alternative assumption is *job availability* (i.e., *item availability*), in which a job becomes available immediately after its processing is completed.

Another batching situation is when *a batching machine* is capable of processing several jobs simultaneously. This is called *batch processing model*. Our MLCSP is similar to batch processing model. However, it differs in the following three ways:

(1) the capacity of a batching machine is the maximum number of jobs that can be processed at any one time while the capacity of a carrier is the maximum number of wafers (not lots) the carrier can hold
(2) in a batching machine, each job occupies the same capacity while in MLCSP, each lot will have different lot sizes
(3) in a batching machine, the processing time of the batch is fixed and is independent of the number of jobs in the batch. This is called a *fixed* batch processor. For any regular performance measure, it is desirable to use batches of the maximum possible size for as long as possible. Such a schedule is called a *full-batch* schedule. A more complicated version is that in which the processing time of a batch is the longest among the processing times of the jobs in the batch. For MLCSP, the carrier processing time is either a fixed amount (single-carrier-processing) or is the sum of the processing times of the wafers in the carrier (single-wafer-processing).

In the literature, one special type of batch processing model, in which jobs have non-identical sizes and no setup required in the context of a single machine environment, has been studied. When makespan is considered, Damodaran et al. (2006) have proposed a genetic algorithm for this problem, and have applied it to a set of randomly generated instances. The results obtained using their algorithm were compared with those obtained using a simulated annealing approach of Melouk et al. (2004) and a commercial solver. The results indicate that the proposed algorithm is able to arrive at near-optimal makespan values with shorter run times than those for the other approaches.

For this same type of batch processing model, when the total completion time or mean flow times are considered, Uzsoy (1994) has presented a branch-and-bound procedure. However, the computational burden of this algorithm rapidly becomes excessive. Azizoglu and Webster (2000 and 2001) have proposed a branch-and-bound algorithm to minimize the total weighted completion time. This algorithm has been shown to generate optimal solutions to problems of up to 25 jobs in reasonable cpu times. Using results from the bin-packing problems, Uzsoy (1994) and Ghazvini and Dupont (1998) have suggested a number of heuristics for these problems. Computational experiments show that they are capable of rapidly obtaining near-optimal solutions. When incompatible job families are considered, Dobson and Nambimadom (2001) have presented an integer programming formulation, and have used it to generate a lower bound using a partial LP relaxation. Several heuristics have also been compared and analyzed in detail.

Besides the MOJ problem, batching and scheduling problem is another type of problem that is relevant to the MLCSP. Though they are different in several ways, the methodologies commonly used to analyze the batching and scheduling problem, such as branch-and-bound algorithm and dynamic programming (DP), can be adopted to our research.

2.2.3 Bin packing (BP) problem

The bin packing (BP) problem can be stated as follows: given a positive integer number of bins of capacity W and a list of n items of integer sizes $L = \{l_1, l_2, ..., l_n\}$ ($0 \le l_i \le W$), assign the items to the bins so that the capacity of the bins is not exceeded and the number of bins used is minimized.

The bin-packing problem belongs to the class of NP-hard problems (Garey and Johnson 1979), and it is not likely that a polynomial time algorithm can be found to solve this problem optimally. Coffman et al (1996) have provided an excellent survey of the methods used for this problem. Three commonly used methods are as follows:

(1) Exact method: Martello and Toth (1990) have proposed a branch-and-bound procedure Martello and Toth Procedure (MTP) which has become a basic reference for use in most comparative studies. Scholl et al. (1997) proposed another exact method BISON (bin packing solution procedure) which makes use of several bounds, reduction procedures, heuristics, and a branch-and-bound procedure using a new branching scheme. Later, Schwerin and Wascher (1999) showed that the MTP provides significantly better results by using the bound derived from the cutting stock problem. Valerio de Carvalho (1999) presented an exact algorithm based on column generation and branch-and-bound. Vanderbeck (1999) proposed yet another column generation-based exact algorithm for the cutting stock problem and has shown its effectiveness for some classes of bin packing problem instances.

(2) Heuristics: Simchi-Levi (1994) has shown that the first-fit decreasing (FFD) and the best-fit decreasing (BFD) heuristics have an absolute performance ratio of 1.5, and that this value is the best possible for the bin-packing problem.

(3) Metaheuristics: Hubscher and Glover (1994) have proposed a tabu search with influential diversification algorithm for bin packing problem, while, Falkenauer (1996) has described a hybrid grouping genetic algorithm (HGGA) for this problem.

As described in Section 2.1, the bin packing problem is a part of the MLCSP problem for carrier formation. The presence of the bin packing problem within the context of the MLCSP makes it a more difficult problem to address; and it is thus more amenable for the use of a heuristic for its solution.

2.3 Model formulation and development of some structural properties

We make the following assumptions. Processing time per wafer is identical for different lots, which means that all the lots belong to the same product type. Setup time is negligible in comparison to the processing time. The objective function is to minimize the total completion time for lots, i.e., the average cycle time, which is a critical performance measure for any manufacturing system, and especially for a semiconductor manufacturer.

Parameters:

N: total number of lots

L: total number of carriers available to be formed, obviously $L < N$

s_i: size for lot i (i.e., number of wafers), $i = 1, \ldots, N$

w_i: weight for lot i, $i = 1, \ldots, N$

K: FOUP carrier capacity (usually, $K = 25$ wafers)

ρ: processing time per wafer at the machine

C_i: completion time for lot i, $i = 1, \ldots, N$

F_k: completion time for carrier k, $k = 1, \ldots, L$

n_k: number of lots in carrier k, $k = 1, \ldots, L$

J_k: set of lots forming carrier k, $k = 1, \ldots, L$

H: a big positive number

O: set of all lots available to be scheduled

P: set of already scheduled lots, $P \subseteq O$

N_P: number of lots belonging to P

L_P: number of carriers formed by P

θ_1: cost per time unit of completion time

θ_2: cost per carrier for transportation

Decision Variables:

$$X_{ik} = \begin{cases} 1, & \text{if lot } i \text{ is assigned to carrier } k,\ i=1,\ldots,N,\ k=1,\ldots,L \\ 0, & \text{otherwise.} \end{cases}$$

$$Y_k = \begin{cases} 1, & \text{if carrier } k \text{ is not empty,} \\ 0, & \text{otherwise.} \end{cases}$$

We can formulate a mathematical model for MLCSP1 as follows.

<u>Model MLCSP1</u>

Minimize $\sum_{i=1}^{N} C_i$

Subject to

$$\sum_{k=1}^{L} X_{ik} = 1, \quad \forall i = 1, \ldots, N \tag{2.1}$$

$$\sum_{i=1}^{N} (X_{ik} * s_i) \le K, \quad \forall k = 1, \ldots, L \tag{2.2}$$

$$F_k = \sum_{l=1}^{k} \sum_{i=1}^{N} (X_{il} * s_i \rho), \ k=1, 2, \ldots, L \tag{2.3}$$

$$C_i \ge F_k - H(1 - X_{ik}), \quad \forall i = 1, \ldots, N, \ k=1, \ldots, L \tag{2.4}$$

(We can let H equal to the maximum possible mackespan value, i.e., $H = \sum_{i=1}^{N} s_i \rho$)

$$X_{ik} \text{ binary, } C_i, \ F_k \ge 0, \quad \forall i = 1, \ldots, N, \ k=1, \ldots, L \tag{2.5}$$

Constraint set (2.1) assigns each lot to one carrier only, and constraint set (2.2) makes sure that FOUP capacity K is not violated. For single-wafer-processing-technology, the processing time of a carrier is determined by the sum of the processing times of the lots contained in it, and all the lots contained in it have the same completion time. These are enforced by Constraint sets (2.3) and (2.4), respectively. Note that each carrier is identical to each other. Thus in the model, we can use arbitrary sequence of carriers, *1,* *2, …, L*. The decision variable X_{ik} automatically enforces the best carrier formation and sequencing.

Model MLCSP1 is an integer programming model involving a large number *H*. As such, it is not computationally efficient to solve this model directly. Next, we develop some structural properties that afford an effective and efficient methodology for the solution of the MLCSP1.

Proposition 2.1: In the optimal solution for MLCSP1, each carrier contains at least one lot.

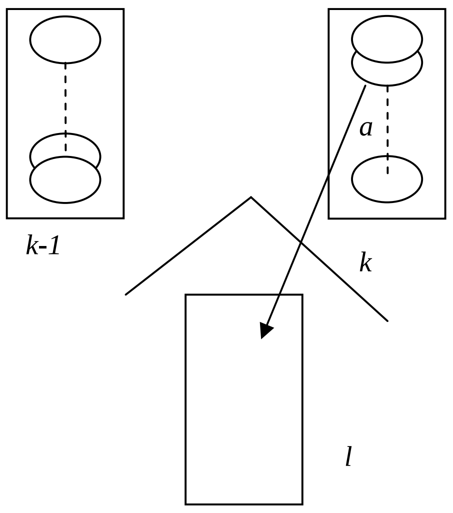

Figure 2.5: Schedule *S'* with insertion of carrier *l* in front of carrier *k*

Proof: This can be proved by contradiction. Suppose in the optimal schedule *S*, there is one empty carrier *l*. Thus, there is at least one carrier containing more than one lot since $L \le N$. Without loss of generality, we construct another schedule S' by inserting the empty carrier *l* in front of carrier *k*, which has more than one lot (see Figure 2.5). Remove one lot, *a,* from carrier *k* and put it in carrier *l*. Since this change does not affect the completion times of the lots in other carriers except for those in *l* and *k*, we will only need to consider the change in the sum of the completion times for the lots in carriers *l* and *k*. Let F_{k-1} represent completion time of carrier *k*-1. The sum of the completion times of the lots in *S*,

$$sum1(l,k) = n_k * (\sum_{i \in J_k} s_i \rho + F_{k-1}) .$$

While for S', the sum of the completion times of the lots,

$$sum2(l,k) = (s_a \rho + F_{k-1}) + (n_k - 1) * (s_a \rho + F_{k-1} + \sum_{i \in J_k \backslash a} s_i \rho) .$$

Note that $sum1(l,k) - sum2(l,k) = \sum_{i \in J_k \backslash a} s_i \rho > 0$, which contradicts the fact that S is optimal. ◘

Corollary 2.1: A lower bound (LB) on the total completion time can be obtained by using N carriers, i.e., with each carrier containing exactly one lot.

It is known that the shortest processing time first (SPT) rule minimizes the total completion time for single machine. We can, thus, arrange the lots by non-decreasing order of their sizes, and re-index them by *j*=1, 2, …, *N*. The lower bound can, thus, be calculated as

$$LB = \sum_{j=1}^{N} (N - j + 1) * s_j \rho .$$

From Proposition 2.1, we can pictorially represent the relationship between the number of carriers used and the resulting total completion time (see Figure 2.6). However, note that the transportation task for the AMHS increases with an increment in the number of carriers. Therefore, usually, the number of carriers to be configured, *L*, is given, and $L < N$. Such a L can be determined by using historical Manufacturing Execution System (MES) data in the fab.

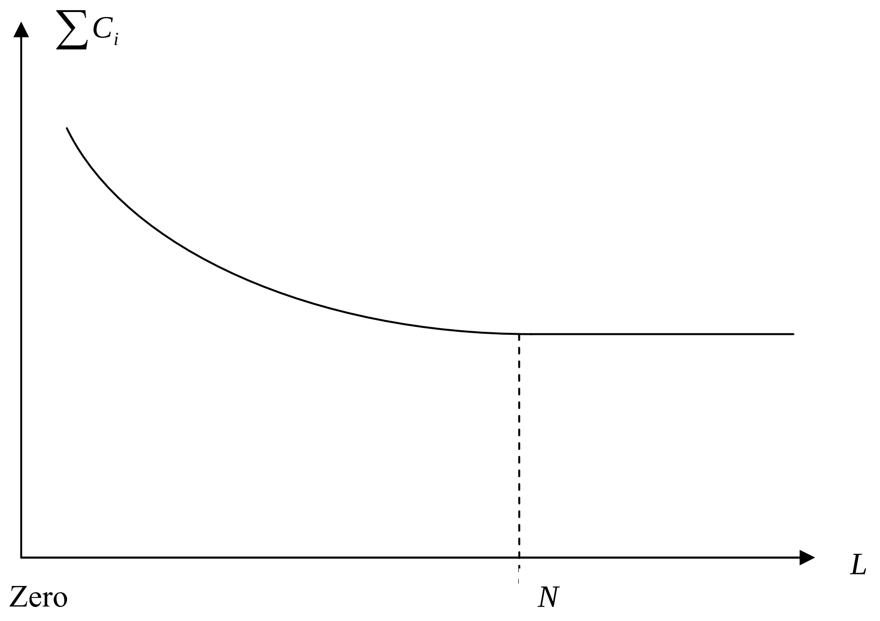

Figure 2.6: The relationship between the number of carriers and total completion time

Proposition 2.2: The MLCSP1 problem is NP-hard.

Proof: We can reformulate the mathematical model as

Minimize $$\sum_{i=1}^{N}\sum_{k=1}^{L}(X_{ik} * \sum_{l=1}^{k}\ \sum_{i=1}^{N}(X_{il} * s_i \rho))$$

Subject to

$$\sum_{k=1}^{L} X_{ik} = 1 \quad \forall i = 1, 2, \ldots, N \tag{2.6}$$

$$\sum_{i=1}^{N}(X_{ik} * s_i) \le K \quad \forall k = 1, 2, \ldots, L \tag{2.7}$$

Note that it is a quadratic generalized assignment problem (GAP), and a quadratic assignment problem is known to be a NP-hard problem (Fisher et al. 1986). ◘

Proposition 2.3: In an optimal solution, the carriers appear in non-decreasing order of the ratios of the total time required to process the wafers contained in a carrier and the number of lots in the corresponding carrier, i.e., $\frac{\sum_{i \in J_j} s_i \rho}{n_j}$.

Proof: By definition, $\sum C_i = \sum_{j=1}^{L} (n_j * F_j)$. For each carrier *j*, we have completion time of F_j, processing time of $\sum_{i \in J_j} s_i \rho$ and "weight" of n_j. The result follows by the fact that the shortest weighted processing time (SWPT) sequence minimizes the total weighted completion time for a single machine (see Pinedo 2005). ◘

In accordance with Proposition 2.3, once each carrier has been configured, the optimal sequence of these carriers can be determined easily.

Note that, in accordance with Proposition 2.1, the lots are spread over all *L* carriers in an optimal solution. Therefore, in an optimal solution obtained by relaxing FOUP capacity, FOUP capacity (i.e., 25 wafers) will not likely be violated severely. Thus, if the optimal solution obtained for the FOUP capacity-relaxed problem is not optimal, it provides a good lower bound for the original (i.e., FOUP capacitated) MLCSP problem.

2.4 A problem with infinite FOUP capacity

2.4.1 Some structural properties

Proposition 2.4: In an optimal solution, $\max_{i \in J_m} s_i \le \min_{j \in J_n} s_j$, where carrier *m* immediately preceds carrier *n*.

Proof: Suppose in an optimal schedule *S*, carrier *m* immediately precedes carrier *n*, and there exist two lots *a* and *b*, $a \in J_m$, $b \in J_n$, such that $s_a > s_b$. Since the carriers have unlimited capacities, we can exchange lots *a* and *b* (see Figure 2.7). Call this schedule S'. We want to show S' to be better than *S*, which would contradict the fact that *S* is optimal. Since a switch of these two lots does not affect the completion times of other carriers, we only need to consider the sum of completion times for lots in carriers *m* and *n*. Let F_l be the completion time of carrier *l,* which immediately precedes carrier *m* in the optimal solution.

Before the switch,

$$\begin{aligned} sum1(m,n) &= n_m(F_l + \rho\sum_{i\in J_m} s_i) + n_n(F_l + \rho\sum_{i\in J_m} s_i + \rho\sum_{j\in J_n} s_j) \\ &= n_m(F_l + \rho\sum_{\substack{i\in J_m\\ i\notin a}} s_i + \rho s_a) + n_n(F_l + \rho\sum_{i\in J_m} s_i + \rho\sum_{j\in J_n} s_j). \end{aligned}$$

After the switch,

$$\begin{aligned} sum2(m,n) &= n_m(F_l + \rho\sum_{\substack{i\in J_m\\ i\notin a}} s_i + \rho s_b) + n_n(F_l + \rho\sum_{\substack{i\in J_m\\ i\notin a}} s_i + \rho s_b + \rho\sum_{\substack{j\in J_n\\ j\notin b}} s_j + \rho s_a) \\ &= n_m(F_l + \rho\sum_{\substack{i\in J_m\\ i\notin a}} s_i + \rho s_b) + n_n(F_l + \rho\sum_{i\in J_m} s_i + \rho\sum_{j\in J_n} s_j). \end{aligned}$$

We have, $sum1(m,n) - sum2(m,n) = \rho n_m(s_a - s_{b)} > 0$. Thus, S' is strictly better than *S*. ◘

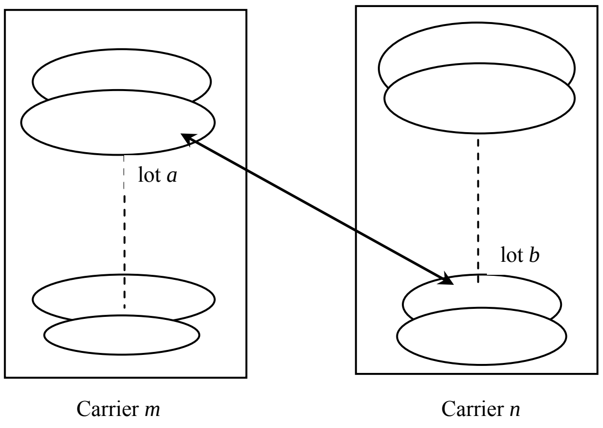

Figure 2.7: A switch of lot *a* and lot *b* between carriers *m* and *n*

Corollary 2.2: In an optimal solution, starting from the first carrier, the lots appear in the carriers in the non-decreasing order of their sizes.

Proposition 2.5: In an optimal solution, $\frac{\sum_{j\in J_n} s_j}{s_b} \le n_m + 1$ for a lot $b \in J_n$, where carrier *m* immediately precedes carrier *n*.

Proof: Assume the contrary; that is, in an optimal schedule, *S*, with two consecutive carriers *m* and *n*, *m* preceding *n*, $\frac{\sum_{j\in J_n} s_j}{s_b} > n_m + 1$ for a lot $b \in J_n$. Since FOUP has unlimited capacity, construct another schedule S' by removing lot *b* from carrier *n* and including it in carrier m (see Figure 2.8). Again, F_l is defined as before, and we consider the net change in completion times of carriers *m* and *n* only as those of the others are unaffected.

Before the change, we have

$$\begin{aligned} sum1(m,n) &= n_m(F_l + \rho\sum_{i\in J_m} s_i) + n_n(F_l + \rho\sum_{i\in J_m} s_i + \rho\sum_{j\in J_n} s_j) \\ &= n_m(F_l + \rho\sum_{i\in J_m} s_i) + n_n(F_l + \rho\sum_{i\in J_m} s_i + \rho\sum_{\substack{j\in J_n\\ j\neq b}} s_j + \rho s_b). \end{aligned}$$

After the change, we have

$$\begin{aligned} sum2(m,n) &= (n_m+1)(F_l + \rho\sum_{i\in J_m} s_i + \rho s_b) + (n_n - 1)(F_l + \rho\sum_{i\in J_m} s_i + \rho s_b + \rho\sum_{\substack{j\in J_n\\ j\notin b}} s_j) \\ &= n_m(F_l + \rho\sum_{i\in J_m} s_i) + n_m s_b \rho + (F_l + \rho\sum_{i\in J_m} s_i) + s_b\rho + n_n(F_l + \rho\sum_{i\in J_m} s_i + \rho\sum_{j\in J_n} s_j) \\ &\quad - (F_l + \rho\sum_{i\in J_m} s_i + \rho\sum_{j\in J_n} s_j). \end{aligned}$$

Moreover,

$$\begin{aligned} sum1(m,n) - sum2(m,n) &= -n_m s_b \rho - (F_l + \rho \sum_{i \in J_m} s_i) - s_b \rho + (F_l + \rho \sum_{i \in J_m} s_i) + \rho \sum_{j \in J_n} s_j \\ &= -\rho(n_m + 1)s_b + \rho \sum_{j \in J_n} s_j > 0, \end{aligned}$$

implying a contradiction. ◘

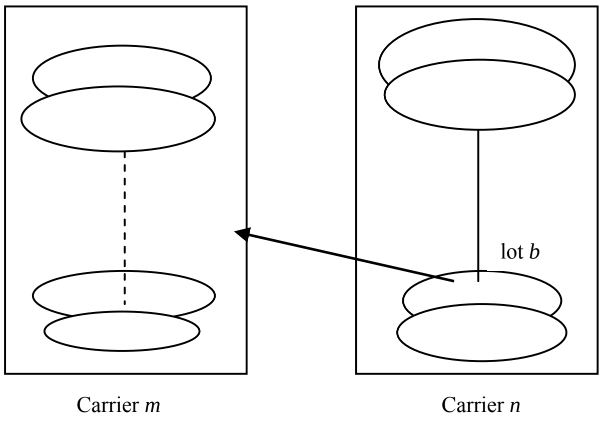

Figure2.8: Schedule *S'* with lot *b* removed from carrier *n* and included in carrier *m*

Corollary 2.3 For two consecutive carriers *m* and *n*, in an optimal solution, where carrier *m* precedes *n*, , the following relationship holds: $n_n \leq n_m + 1$.

Proof: It follows from Proposition 2.5, and the fact that $n_n \leq \frac{\sum_{j \in J_n} s_j}{\min_{j \in J_n} s_j}$. ◘

Note that the only situation under which $n_n = \frac{\sum_{j \in J_n} s_j}{\min_{j \in J_n} s_j}$ would hold is when all the lots in carrier *n* are of the same size.

Proposition 2.6 $n_{m\min} = \left\lceil \dfrac{(N - \sum_{i=1}^{m-1} n_i) - \dfrac{(L-m)(L-m+1)}{2}}{(L-m+1)} \right\rceil$, for m=1,…, L-1.

Proof: Based on the fact that $n_m + 1 \geq n_n$, with carrier *m* immediately preceding carrier *n,* the minimum n_m value is achieved when $n_{k+1} = n_k + 1$, *k*=*m, m+1, …N-1*. Thus, the total number of lots after carrier *m*, including carrier *m* is

$$\sum_{i=m}^{L} n_i = \sum_{i=m}^{L} (n_{m\min} + (i-m)) = n_{m\min}(L-m+1) + \frac{(L-m)(L-m+1)}{2}.$$

Since $\sum_{i=1}^{m-1} n_i + n_{m\min}(L-m+1) + \dfrac{(L-m)(L-m+1)}{2} = N$, we have

$n_{m\min} = \left\lceil \dfrac{(N - \sum_{i=1}^{m-1} n_i) - \dfrac{(L-m)(L-m+1)}{2}}{(L-m+1)} \right\rceil$ as $n_{m\min}$ can only be integer. ◘

Proposition 2.7: In an optimal solution, $\dfrac{\sum_{j \in J_n} s_j}{s_a} \geq n_m - 1$ for a lot $a \in J_m$ and for two consecutive carriers *m* and *n*, with *m* preceding *n*.

Proof: Assume the contrary; that is, in an optimal solution *S*, two consecutive carriers *m* and *n*, with *m* preceding *n*, $\dfrac{\sum_{j \in J_n} s_j}{s_a} < n_m - 1$ for a lot $a \in J_m$. Since the FOUP capacity is relaxed, construct another schedule S', by removing lot *a* from carrier *m* and including it in carrier *n* (see Figure 2.9). Before the change, we have

$$sum1(m,n) = n_m(F_l + \rho\sum_{i\in J_m} s_i) + n_n(F_l + \rho\sum_{i\in J_m} s_i + \rho\sum_{j\in J_n} s_j)$$

$$= n_m(F_l + \rho\sum_{\substack{i\in J_m\\ i\neq a}} s_i + \rho s_a) + n_n(F_l + \rho\sum_{i\in J_m} s_i + \rho\sum_{j\in J_n} s_j).$$

After the change, we have

$$sum2(m,n) = (n_m - 1)(F_l + \rho\sum_{\substack{i\in J_m\\ i\neq a}} s_i) + (n_n + 1)(F_l + \rho\sum_{\substack{i\in J_m\\ i\neq a}} s_i + \rho\sum_{j\in J_n} s_j + \rho s_a)$$

$$= n_m(F_l + \rho\sum_{\substack{i\in J_m\\ i\neq a}} s_i) - (F_l + \rho\sum_{\substack{i\in J_m\\ i\neq a}} s_i) + n_n(F_l + \rho\sum_{i\in J_m} s_i + \rho\sum_{j\in J_n} s_j) + (F_l + \rho\sum_{i\in J_m} s_i + \rho\sum_{j\in J_n} s_j).$$

Consequently,

$$sum1(m,n) - sum2(m,n) = n_m s_a \rho + (F_l + \rho\sum_{\substack{i\in J_m\\ i\neq a}} s_i) - (F_l + \rho\sum_{\substack{i\in J_m\\ i\neq a}} s_i + \upsilon s_a + \rho\sum_{j\in J_n} s_j)\rho$$

$$= (n_m - 1)s_a\rho - \rho\sum_{j\in J_n} s_j > 0,$$

implying contradiction. ◘

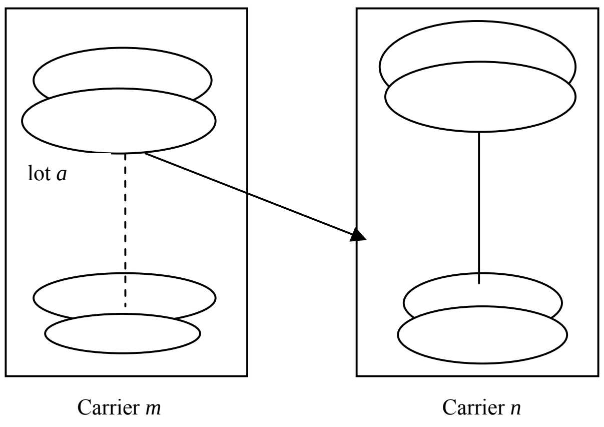

Figure 2.9: Schedule *S'* with a lot *a* removed from carrier *m* and included in carrier *n*

Corollary 2.4: If all the lots are of the same size, then $n_n \geq n_m - 1$.

Proof: For a lot $i \in J_m$, $\frac{\sum_{j \in J_n} s_j}{s_i} = n_n \geq n_m - 1$ by Proposition 2.7. ◘

Proposition 2.8: In an optimal solution, $\frac{\sum_{j \in J_n} s_j + \sum_{k=m+1}^{n-1} \sum_{i \in J_k} s_i}{s_b} \leq n_m + \sum_{k=m+1}^{n-1} n_k + 1$ for a lot $b \in J_n$, where the carriers are sequenced in the order *m, m+1, m+2, …, n-2, n-1, n.*

Proof: Assume the contrary; that is, in an optimal solution *S*, $\frac{\sum_{j \in J_n} s_j + \sum_{k=m+1}^{n-1} \sum_{i \in J_k} s_i}{s_b} > n_m + \sum_{k=m+1}^{n-1} n_k + 1$ for a lot $b \in J_n$. Construct another schedule S' by removing a lot *b* from carrier *n* and including it in carrier *m* (see Figure 2.10). Before the change, we have

$$sum1(m \sim n) = n_m (F_l + \rho \sum_{i \in J_m} s_i) + \sum_{k=m+1}^{n-1} n_k (F_l + \rho \sum_{i \in J_m} s_i + \rho \sum_{l=m+1}^{k} \sum_{i \in J_l} s_i)$$

$$+ n_n (F_l + \rho \sum_{k=m+1}^{n-1} \sum_{i \in J_k} s_i + \rho \sum_{i \in J_m} s_i + \rho \sum_{j \in J_n} s_j).$$

After the change, we have

$$sum2(m \sim n) = (n_m + 1)(F_l + \rho \sum_{i \in J_m} s_i + \rho s_b) + \sum_{k=m+1}^{n-1} n_k (F_l + \rho \sum_{i \in J_m} s_i + \rho s_b + \rho \sum_{l=m+1}^{k} \sum_{i \in J_{lk}} s_i)\rho$$

$$+ (n_n - 1)(F_l + \rho \sum_{i \in J_m} s_i + \rho \sum_{k=m+1}^{n-1} \sum_{i \in J_k} s_i + \rho s_b + \rho \sum_{\substack{j \in J_n \\ j \neq b}} s_j).$$

Consequently,

$$sum1(m \sim n) - sum2(m \sim n) = -n_m s_b \rho - (F_l + \rho \sum_{i \in J_m} s_i + \rho s_b) - \rho \sum_{k=m+1}^{n-1} n_k s_b + (F_l + \rho \sum_{i \in J_m} s_i)$$
$$+ \rho \sum_{k=m+1}^{n-1} \sum_{i \in J_k} s_i + \rho \sum_{j \in J_n} s_j$$
$$= -(n_m + \sum_{k=m+1}^{n-1} n_k + 1) s_b \rho + \sum_{k=m+1}^{n-1} \sum_{i \in J_k} s_i \rho + \sum_{j \in J_n} s_j \rho > 0,$$

implying a contradiction. ◘

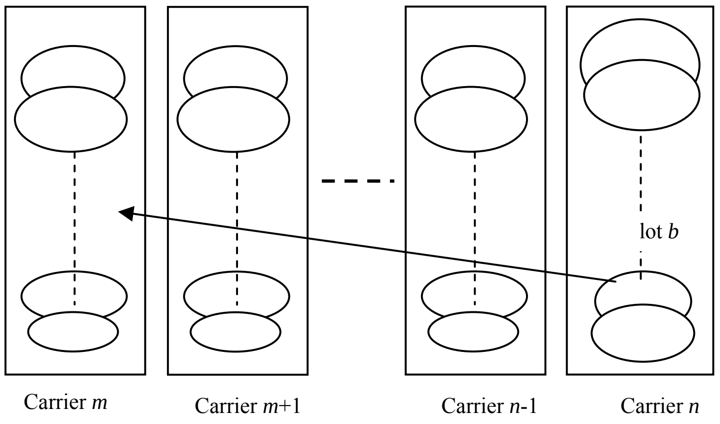

Figure2.10: Schedule *S'* with lot *b* removed from carrier *n* and included in carrier *m*

Corollary 2.5: If all the lots are of the same size, then $n_n \le n_m + 1$, where carrier *m* precedes carrier *n*.

Proof: It is based on the fact that $\dfrac{\sum_{j \in J_n} s_j + \sum_{k=m+1}^{n-1} \sum_{i \in J_k} s_i}{s_b} = n_n + \sum_{k=m+1}^{n-1} n_k \le n_m + \sum_{k=m+1}^{n-1} n_k + 1$ for a lot $b \in J_n$. ◘

Note that this result is different from Corollary 2.3 since *m* and *n* need not be adjacent.

Proposition 2.9: In an optimal solution, $\dfrac{\sum_{j\in J_n} s_j + \sum_{k=m+1}^{n-1} \sum_{i\in J_k} s_i}{s_a} \geq n_m + \sum_{k=m+1}^{n-1} n_k - 1$ for a lot $a \in J_m$ where the carriers are sequenced in the order *m, m+1, m+2, …, n-2, n-1, n*.

Proof: Assume the contrary; that is, in an optimal solution, $\dfrac{\sum_{j\in J_n} s_j + \sum_{k=m+1}^{n-1} \sum_{i\in J_k} s_i}{s_a} < n_m + \sum_{k=m+1}^{n-1} n_k - 1$ for a lot $a \in J_m$. Construct another schedule S', by removing lot *a* from carrier m and including it in carrier *n* (see Figure 2.11). Before the change, we have

$$sum1(m \sim n) = n_m (F_l + \rho \sum_{i\in J_m} s_i) + \sum_{k=m+1}^{n-1} n_k (F_l + \rho \sum_{i\in J_m} s_i + \rho \sum_{l=m+1}^{k} \sum_{i\in J_l} s_i)$$

$$+ (F_l + \rho \sum_{k=m+1}^{n-1} \sum_{i\in J_k} s_i + \rho \sum_{i\in J_m} s_i + \rho \sum_{j\in J_n} s_j).$$

After the change, we have

$$sum2(m \sim n) = (n_m - 1)(F_l + \rho \sum_{i\in J_m} s_i - \rho s_a) + \sum_{k=m+1}^{n-1} n_k (F_l + \rho \sum_{i\in J_m} s_i - \rho s_a + \rho \sum_{l=m+1}^{k} \sum_{i\in J_l} s_i)$$

$$+ (n_n + 1)(F_l + \rho \sum_{i\in J_m} s_i + \rho \sum_{k=m+1}^{n-1} \sum_{i\in J_k} s_i + \rho s_a + \rho \sum_{\substack{j\in J_n \\ j\notin a}} s_j)\rho.$$

Consequently,

$$sum1(m\sim n) - sum2(m\sim n) = (F_l + \rho\sum_{i\in J_m} s_i - \rho s_a) + n_m s_a \rho + \sum_{k=m+1}^{n-1} n_k s_a \rho -$$
$$(F_l + \rho\sum_{i\in J_m} s_i + \rho\sum_{k=m+1}^{n-1}\sum_{i\in J_k} s_i + \rho s_a + \rho\sum_{\substack{j\in J_n\\ j\notin a}} s_j)$$
$$= (n_m - 1 + \sum_{k=m+1}^{n-1} n_k) s_a \rho - (\sum_{j\in J_n} s_j + \sum_{k=m+1}^{n-1}\sum_{i\in J_k} s_i)\rho > 0,$$

implying a contradiction. ◘

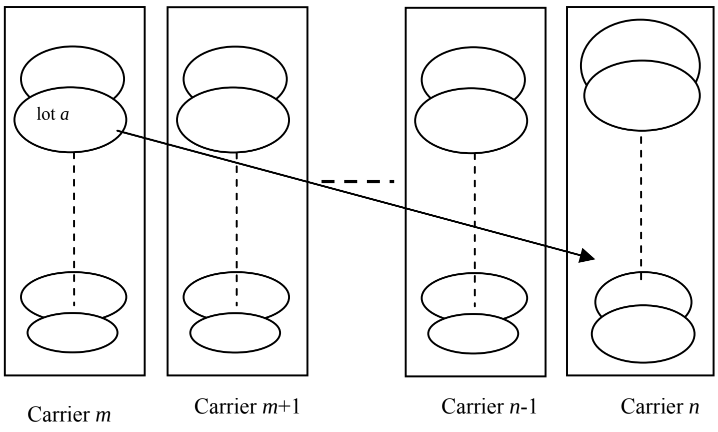

Figure 2.11: Schedule *S'* with a lot *a* removed from carrier *m* and included in carrier *n*

Corollary 2.6: If all the lots are of the same size, $n_n \geq n_m - 1$, with carrier *m* preceding carrier *n*.

Proof: Note that for a lot $a \in J_m$, we have

$$\frac{\sum_{j\in J_n} s_j + \sum_{k=m+1}^{n-1}\sum_{i\in J_k} s_i}{s_a} = n_n + \sum_{k=m+1}^{n-1} n_k \geq n_m + \sum_{k=m+1}^{n-1} n_k - 1.$$

2.4.2 Optimal solution for the lots of the same size

Proposition 2.10: If all the lots are of the same size, then, in an optimal solution, there are only at most two possible values of n_1. Furthermore, if n_1 is fixed, then there are only at most two common possible values of n_i, for all *i*=2, …, *L*.

Proof: From Corollaries 2.5 and 2.6, we have $n_m - 1 \le n_n \le n_m + 1$, $\forall$ *m*=1, 2, …..*L-1*, *n* =2, 3, ….*L*, with *m* preceding *n*. For first carrier, we have

$$n_{1\max} + (L-1)(n_{1\max} - 1) = N \text{, and}$$

$$n_{1\min} + (L-1)(n_{1\min} + 1) = N .$$

This results in $\frac{N-L+1}{L} \le n_1 \le \frac{N+L-1}{L}$. Since $\frac{N+L-1}{L} - \frac{N-L+1}{L} = \frac{2L-2}{L} < 2$, and n_1 is integer, there can be at most two values that n_1 can take, i.e, $n_{1\min} = \left\lceil \frac{N+1}{L} \right\rceil - 1$ and $n_{1\max} = \left\lceil \frac{N-1}{L} \right\rceil + 1$.

Based on Corollary 2.5 and 2.6, if $n_1 = n_{1\max}$, n_i can only take at most two possible values $n_i = n_1$ or $n_i = n_1 - 1$; if $n_1 = n_{1\min}$, n_i can only take at most two possible values $n_i = n_1$ or $n_i = n_1 + 1$. ◘

The distribution of the number of lots for each carrier is shown in Figure 2.12.

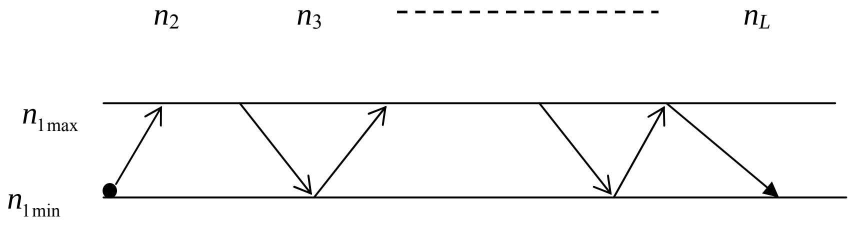

Figure 2.12: The distribution of the number of lots in each carrier

Corollary 2.7: If all the lots are of the same size and $\frac{N}{L}$ is integer, then, in the optimal solution, $n_m = \frac{N}{L}$, m =1, 2, …, L.

Proof: Suppose the contrary, that is, there is one carrier *i*, $n_i \neq \frac{N}{L}$. Without loss of generality, let $n_i < \frac{N}{L}$. Since *N* is constant, there must be at least one carrier *j*, $n_j > \frac{N}{L}$. Since $\frac{N}{L}$ is integer, $n_j - n_i \geq 2$. Thus, the condition from Corollaries 2.5 and 2.6, $n_m - 1 \leq n_n \leq n_m + 1$, $\forall$ *m*=1, 2, …, *L-1*, *n* =2, 3, …, *L*, with *m* preceding *n*, is violated, implying a contradiction. ◘

Proposition 2.11: If all the lots are of the same size, then, n_i, $\forall i$ =1,…, *L*, can take any of the two possible values without impacting the total completion time provided $\sum_{i=1}^{L} n_i = N$.

Proof: As all the lots must be allocated, if we reduce the number of lots in carrier i from n_i to $n_i - 1$, then there must be a carrier j for which the number of lots is increased from n_j to $n_j + 1$ (see Figure 2.13).

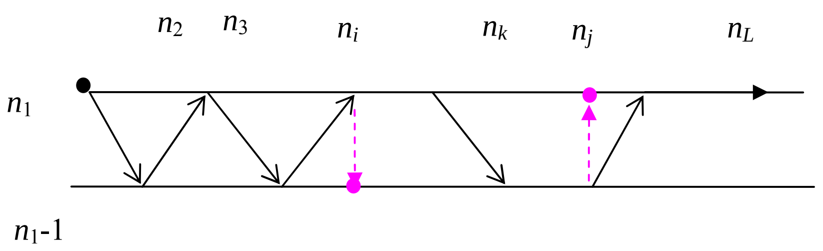

Figure 2.13: Illustration of the impact due to a change in carrier size

Clearly, this change only affects the completion times of the carriers between *i* and *j*. The total completion time between carriers *i* to *j*, before the change,

$$C_{i->j} = (F_{i-1} + n_i s\rho)n_i + \sum_{k=i+1}^{j-1} (F_{i-1} + \sum_{l=i}^{k} n_l s\rho)n_k + (F_{i-1} + \sum_{l=i}^{j} n_l s\rho)n_j \ .$$

After the change, the total completion time,

$$C'_{i->j} = [F_{i-1} + (n_i - 1)s\rho](n_i - 1) + \sum_{k=i+1}^{j-1} [F_{i-1} + (\sum_{l=i}^{k} n_l - 1)s\rho]n_k$$
$$+ (F_{i-1} + \sum_{l=i}^{j} (n_l - 1)s\rho + s\rho)(n_j + 1).$$

Consequently,

$$C_{i->j} - C'_{i->j} = s\rho - 2n_i s\rho + (n_i s\rho + n_j s\rho) = (n_i + n_j + 1 - 2n_i)s\rho = 0.$$

Thus, the total completion time is not affected if the number of lots, n_i in a carrier *i*, is reduced by one, and n_j in another carrier *j* is increased by one. This is true for any two carriers *i* and *j*. ◘

Proposition 2.12: If all the lots are of the same size, any solution that satisfies $n_m - 1 \le n_n \le n_m + 1$, $\forall$ *m*=1, 2, …, *L-1*, *n* =2, 3, …, *L*, with *m* preceding *n*, is the optimal solution.

Proof: Based on Corollaries 2.5 and 2.6, we have $n_m - 1 \le n_n \le n_m + 1$, $\forall$ *m*=1, 2, …, *L-1*, *n* =2, 3, …, *L*, with *m* preceding *n*. From Proposition 2.11, it follows that any schedule that satisfies this condition has the same total completion time. Thus, this condition will lead to optimal solutions. ◘

One of the optimal solutions if all of the lots are of the same size, is $n_k = \left\lceil \frac{N}{L} \right\rceil$ for carrier *k* =1, …, $N - L\left\lfloor \frac{N}{L} \right\rfloor$, and $n_k = \left\lfloor \frac{N}{L} \right\rfloor$ for carrier *k*= $N - L\left\lfloor \frac{N}{L} \right\rfloor + 1$, …, *L*, and the total completion time,

$$\sum_{i=1}^{N} C_i = \left(\left\lceil \frac{N}{L} \right\rceil s\rho\right)\left(\sum_{k=1}^{N-L\left\lfloor \frac{N}{L} \right\rfloor}(L-k+1)\right) + \left(\left\lfloor \frac{N}{L} \right\rfloor \rho s\right)\left(\sum_{k=N-L\left\lfloor \frac{N}{L} \right\rfloor+1}^{L}(L-k+1)\right)\Bigg).$$

2.4.3 An algorithm for the case of lots of different sizes

In accordance with Proposition 2.4, lots appear in the non-decreasing order of their sizes in an optimal solution. Based on this property, we can propose a dynamic programming algorithm to find an optimal schedule. We designate this algorithm as **RelaxFOUP-DP**.

RelaxFOUP-DP Algorithm

The lots are arranged in the non-decreasing order of their sizes. Each stage of this multiple stage decision process is a lot *k*, *k*=1,…, *N*. The output state at a stage is the number of carriers configured so far, designated as *b*, *b*=1, …, *L*. The decision at stage *k* is to determine a lot *l* so that when lots $l+1$ to *k* are contained in the last carrier (carrier *b*), the total completion time of the lots from 1 to *k* is minimized. We use forward recursion. Therefore, if *b* is the output state at stage *k*, and the decision is to group lots from *l*+1 to *k* in carrier *b*, then the number of carriers as the output of stage *l* is *b*-1.

Let $\alpha_k = s_1 + s_2 + ... + s_k$, $1 \le k \le N$, and note that $\alpha_1 < \alpha_2 < ... < \alpha_N$. For $0 \le l < k$, let $F(k,l)^b$ indicate the total completion time of a solution in which the first *l* lots are contained in *b*-1 carriers and lots *l+1,…, K* are included in the last carrier *b* (see Figure 2.14). Also, let G_k^b indicate the smallest total completion time when the first *k* lots are included in *b* carriers. We seek the determination of G_N^L.

$$\underbrace{s_1 \quad s_2 \quad ------ \quad s_l}_{\text{Previous carriers}} \qquad \underbrace{s_{l+1} \quad ----- \quad s_k}_{\text{Last carrier}}$$

Figure 2.14: The forward DP recursion

We have,

$G_k^b = \min_{l<k} F(k,l)^b$, $1 \le k \le N$, $1 \le b \le L$

$F(k,l)^b = G_l^{b-1} + \alpha_k (k-l)$.

And, the recursion equation is

$$G_k^b = \min_{l<k} \{G_l^{b-1} + \alpha_k (k-l)\}, \quad 1 \le k \le N, \ b > 1$$

Example:

Table 2.1: An example to illustrate RelaxFOUP-DP algorithm

Lot index	1	2	3	4	5	6	7	8
Lot Size	3	3	4	5	6	7	7	8
Processing time	$\rho = 1$							
No. of carriers	$L = 4$							

$G_1^1 = 1(3) = 3$

$G_2^2 = \{G_1^1 + (2-1)(3+3)\} = \{3+6\} = 9$

$G_2^1 = \{2(3+3)\} = 12$

$G_3^1 = \{3(3+3+4)\} = 30$

$$G_3^2 = \min\{G_1^1 + (3-1)(3+4), G_2^1 + (3-2)(4)\} = \min\{3+14, 12+4\} = 16$$

$$G_3^3 = \{G_2^2 + (3-2)(4)\} = \{9+4\} = 13$$

$$G_4^1 = \{4(3+3+4+5)\} = 60$$

$$\begin{aligned} G_4^2 &= \min\{G_1^1 + (4-1)(3+4+5), G_2^1 + (4-2)(4+5), G_3^1 + (4-3)(5)\} \\ &= \min\{3+36, 12+18, 30+5\} = 30 \end{aligned}$$

$$G_4^3 = \min\{G_2^2 + (4-2)(4+5), G_3^2 + (4-3)(5)\} = \min\{9+18, 16+5\} = 21$$

$$G_4^4 = \{G_3^3 + (4-3)(5)\} = \{13+5\} = 18$$

$$G_5^1 = \{5(3+3+4+5+6)\} = 126$$

$$\begin{aligned} G_5^2 &= \min\{G_1^1 + (5-1)(3+4+5+6), G_2^1 + (5-2)(4+5+6), G_3^1 + (5-3)(5+6), G_4^1 + (5-4)(6)\} \\ &= \min\{3+72, 12+45, 30+22, 60+6\} \\ &= 52 \end{aligned}$$

$$\begin{aligned} G_5^3 &= \min\{G_2^2 + (5-2)(4+5+6), G_3^2 + (5-3)(5+6), G_4^2 + (5-4)(6)\} \\ &= \min\{9+45, 16+22, 30+6\} \\ &= 36 \end{aligned}$$

$$G_5^4 = \min\{G_3^3 + (5-3)(5+6), G_4^3 + (5-4)(6)\} = \min\{13+22, 21+6\} = 27$$

$$G_6^1 = \{6(3+3+4+5+6+7)\} = 168$$

$$\begin{aligned} G_6^2 &= \min\{G_1^1 + (6-1)(3+4+5+6+7), G_2^1 + (6-2)(4+5+6+7), G_3^1 + (6-3)(5+6+7), \\ &\quad G_4^1 + (6-4)(6+7), G_5^1 + (6-5)(7)\} \\ &= \min\{3+125, 12+88, 30+54, 60+26, 126+7\} \\ &= 84 \end{aligned}$$

$$\begin{aligned} G_6^3 &= \min\{G_2^2 + (6-2)(4+5+6+7), G_3^2 + (6-3)(5+6+7), G_4^2 + (6-4)(6+7), \\ &\quad G_5^2 + (6-5)(7)\} \\ &= \min\{9+88, 16+54, 30+26, 52+7\} \\ &= 56 \end{aligned}$$

$$\begin{aligned} G_6^4 &= \min\{G_3^3 + (6-3)(5+6+7), G_4^3 + (6-4)(6+7), G_5^3 + (6-5)(7)\} \\ &= \min\{13+54, 21+26, 36+7\} \\ &= 43 \end{aligned}$$

$$G_7^1 = \{7(3+3+4+5+6+7+7)\} = 245$$

$$\begin{aligned}G_7^2 &= \min\{G_1^1 + (7-1)(3+4+5+6+7+7), G_2^1 + (7-2)(4+5+6+7+7), G_3^1 + \\ &\quad (7-3)(5+6+7+7), G_4^1 + (7-4)(6+7+7), G_5^1 + (7-5)(7+7), G_6^1 + (7-6)(7)\} \\ &= \min\{3+192, 12+145, 30+100, 60+60, 126+28, 168+7\} \\ &= 120\end{aligned}$$

$$\begin{aligned}G_7^3 &= \min\{G_2^2 + (7-2)(4+5+6+7+7), G_3^2 + (7-3)(5+6+7+7), G_4^2 + (7-4)(6+7+7), \\ &\quad G_5^2 + (7-5)(7+7), G_6^2 + (7-6)(7)\} \\ &= \min\{9+145, 16+100, 30+60, 52+28, 84+7\} \\ &= 80\end{aligned}$$

$$\begin{aligned}G_7^4 &= \min\{G_3^3 + (7-3)(5+6+7+7), G_4^3 + (7-4)(6+7+7), G_5^3 + (7-5)(7+7), G_6^3 + (7-6)(7)\} \\ &= \min\{13+100, 21+60, 36+28, 56+7\} \\ &= 63\end{aligned}$$

$$G_8^1 = \{8(3+3+4+5+6+7+7+8)\} = 312$$

$$\begin{aligned}G_8^2 &= \min\{G_1^1 + (8-1)(3+4+5+6+7+7+8), G_2^1 + (8-2)(4+5+6+7+7+8), \\ &\quad G_3^1 + (8-3)(5+6+7+7+8), G_4^1 + (8-4)(6+7+7+8), G_5^1 + (8-5)(7+7+8), \\ &\quad G_6^1 + (8-6)(7+8), G_7^1 + (8-7)(8)\} \\ &= \min\{3+280, 12+222, 30+165, 60+112, 126+66, 168+30, 245+8\} \\ &= 172\end{aligned}$$

$$\begin{aligned}G_8^3 &= \min\{G_2^2 + (8-2)(4+5+6+7+7+8), G_3^2 + (8-3)(5+6+7+7+8), \\ &\quad G_4^2 + (8-4)(6+7+7+8), G_5^2 + (8-5)(7+7+8), G_6^2 + (8-6)(7+8), G_7^2 + (8-7)(8)\} \\ &= \min\{9+222, 16+165, 30+112, 52+66, 84+30, 120+8\} \\ &= 114\end{aligned}$$

$$\begin{aligned}G_8^4 &= \min\{G_3^3 + (8-3)(5+6+7+7+8), G_4^3 + (8-4)(6+7+7+8), G_5^3 + (8-5)(7+7+8), \\ &\quad G_6^3 + (8-6)(7+8), G_7^3 + (8-7)(8)\} \\ &= \min\{13+165, 21+112, 36+66, 56+30, 80+8\} \\ &= 86\end{aligned}$$

The optimal solution is (3 3)(4 5)(6 7) (7 8) with total completion time = 86.

We can improve computational effectiveness of the above DP algorithm by making use of Propositions 2.3, 2.5, 2.6 and 2.7.

2.5 A problem with finite FOUP capacity

2.5.1 Optimal solution for the MLCSP1 with lots of the same size

Proposition 2.13: When all the lots are of the same size, the MLCSP1 problem is solved in polynomial time.

Proof: By Proposition 2.12, if all the lots are of the same size, any solution that satisfies $n_m - 1 \le n_n \le n_m + 1$, $\forall$ *m*=1, 2, …, *L-1*, *n* =2, 3, …, *L*, with *m* preceding *n*, is the optimal solution. We want to show that such a solution is feasible when the FOUP has a finite capacity.

Suppose the contrary, that is, there exists such a solution that satisfies $n_m - 1 \le n_n \le n_m + 1$, $\forall$ *m*=1, 2, …, *L-1*, *n*=2, 3, …, *L*, with *m* preceding *n*, in which there exists a carrier *k*, with $n_k s > K$. There are two possible cases:

(1) n_k is the minimum among all n_i, that is, $n_i \ge n_k$, $\forall$ *i*=1, …, *L* and $i \ne k$. We have $\sum_{i=1}^{L} n_i \ge L * n_k s > L * K$. This means that the sum of the wafers in all the lots is larger than the available FOUP capacity. The MLCSP1 problem becomes infeasible, implying a contradiction.

(2) n_k is the maximum value. From Proposition 2.10, the minimum carrier size $n_{\min} s = (n_k - 1)s > K - s$ (since $n_k s > K$). Thus, $n_k = \left\lceil \frac{K}{s} \right\rceil$ and $n_k - 1 = \left\lfloor \frac{K}{s} \right\rfloor$. We have $\sum_{i=1}^{L} n_i > L * \left\lfloor \frac{K}{s} \right\rfloor$. Since lots are not allowed to split over different carriers, the MLCSP1 problem is infeasible, implying a contradiction. ◘

2.5.2 A branch-and-bound algorithm for the MLCSP1 with lots of different sizes

In case the **RelaxFOUP-DP** algorithm generates a feasible solution to the original problem, then it solves the problem directly. We can identify some situations for which the **RelaxFOUP-DP** algorithm is likely to generate an optimal solution without violating the FOUP capacity.

(1) The number of available carriers is close to the number of lots (i.e, L is close to N).

Since an optimal solution will use L carriers, when L is closer to N, the number of lots in a carrier will be close to one. In such a case, it is unlikely that the carrier capacity will be violated in the solution generated by the **RelaxFOUP-DP** algorithm.

(2) The values of s_i , for all i, are close to one another, and small compared to the value of K.

Since $(n_{m-1}-1)\max_{j\in J_{m-1}} s_j \le \sum_{i\in J_m} s_i \le (n_{m-1}+1)\min_{i\in J_m} s_i$ by Propositions 2.5 and 2.7, the variation in the value of $\sum_{i\in J_m} s_i$ is restricted to be $2s_i$. Thus, the **RelaxFOUP-DP** algorithm will generate solutions in which the number of wafers in each carrier would be close to each other, and hence, would not violate K.

Note that the **RelaxFOUP-DP** algorithm can generate infeasible solutions in which the carrier size may violate the FOUP capacity constraint. But, it gives a good lower bound. In this section, we propose a branch-and-bound algorithm to find an optimal solution for the MLCSP1 problem for different lot sizes. We designate the algorithm as **MLCSP1-B&B**.

Enumeration scheme

We can possibly use two methods, called Method I and Method II, to add a new lot to the already scheduled lots. In Method I, a new lot is put into a new carrier and attached to the end of the partial schedule. In Method II, a new lot is inserted into the last carrier (see Figure 2.15).

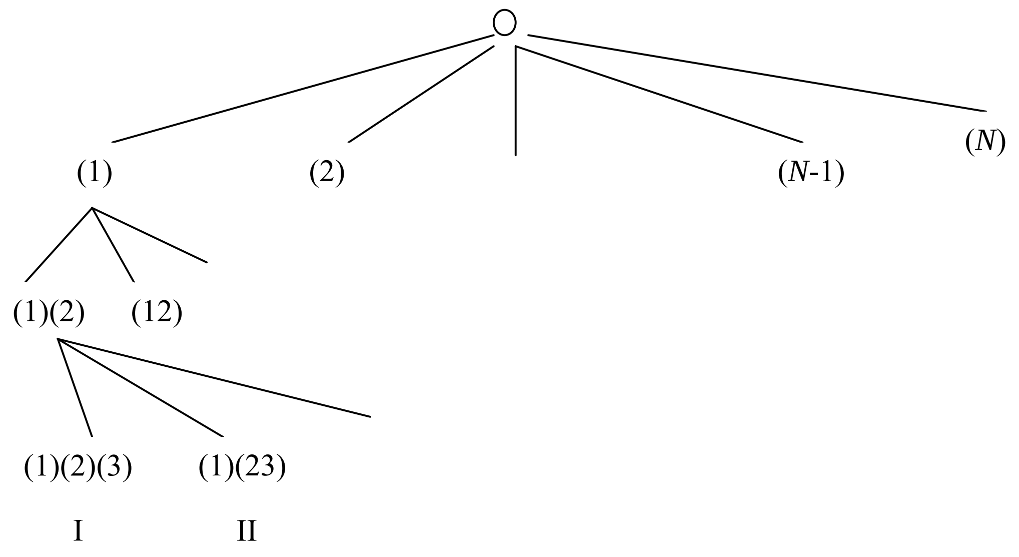

Figure2.15: Methods I and II for adding a lot to a carrier

Upper bounds

In view of Corollary 2.2, we can formulate the following greedy heuristic to generate a feasible solution for MLCSP1.

RelaxFOUP-Greedy Heuristic

Step 1: Arrange the lots in the non-decreasing order of their sizes.

Step 2*:* Starting from the first carrier, fill each carrier with lots in the order determined in Step 1 such that the FOUP capacity is not violated.

Lower bounds

A lower bound for the unscheduled lots can be obtained by the following three ways.

$LB1\{O \setminus P\}$: Total completion time generated by $L' = N - N_P$ carriers.

$LB2\{O \setminus P\}$: Total completion time generated by unscheduled set of lots $\{O \setminus P\}$ configured in leftover carriers, $L - L_P$ by applying the **RelaxFOUP-DP** algorithm.

$LB3\{O \setminus P\}$: For the unscheduled set of lots $\{O \setminus P\}$, we find the biggest common factor γ among lot sizes s_i, $i \in \{O \setminus P\}$. We, then, split all the lots into smaller lots with lot size of γ and the weight of $\frac{\gamma}{s_i}$. This problem, thus, becomes a MLCSP with lots of the same size and the objective of minimizing the total weighted completion time. The optimal solution determined is a lower bound for the original problem.

Continuing with the development of the **MLCSP1-B&B** algorithm, note that the result of Proposition 2.1 holds for the MLCSP1 under the objective of minimizing the total weighted completion time as well. This follows by the fact that the result of Proposition 2.1 is developed for a special case with $w_i = 1$, $\forall i = 1$, …, *N*. Hence, each carrier for this problem will consist of at least one lot.

It is well-known that the shortest weighted processing time first (SWPT) rule minimizes the total weighted completion time for the single machine problem. Thus, we have the following result.

Proposition 2.14: In the optimal solution, the carriers appear in the non-decreasing order of their total lot size to total weight ratios, i.e., $\frac{\sum_{i\in J_m} s_i}{\sum_{i\in J_m} w_i}$, *m*=1, …, *L*.

Proposition 2.15: When FOUP capacity is relaxed, in an optimal solution, for a lot *i* from carrier *m* and a lot *j* from carrier *n*, with carrier *m* immediately preceding carrier *n*, $\max_{i\in J_m} \frac{s_i}{w_i} \le \min_{j\in J_n} \frac{s_j}{w_j}$, if weights are agreeable, i.e., $s_i \le s_j$ implies $w_i \ge w_j$.

Proof: We can prove this result by contradiction using pair-wise interchange. Suppose there exists an optimal solution in which a lot *a* from carrier *m* and a lot *b* from carrier *n* are such that $s_a \ge s_b$ and $w_a \le w_b$, The total weighted completion time of this solution is as follows:

$$sum1(m,n) = (\sum_{i\in J_m} w_i)(\sum_{i\in J_m} s_i\rho) + (\sum_{j\in J_n} w_j)(\sum_{i\in J_m} s_i\rho + \sum_{j\in J_n} s_j\rho).$$

If we interchange lot *a* and *b*, we have

$$sum2(m,n) = (\sum_{i\in J_m} w_i - w_a + w_b)(\sum_{i\in J_m} s_i - s_a + s_b)\rho + (\sum_{j\in J_n} w_j - w_b + w_a)(\sum_{i\in J_m} s_i - s_a + s_b + \sum_{j\in J_n} s_j - s_b + s_a)\rho.$$

Consequently,

$$sum1(m,n) - sum2(m,n) = (w_b - w_a)(\sum_{j\in J_n} s_j\rho + s_a\rho - s_b\rho) + (s_a\rho - s_b\rho)(\sum_{i\in J_m} w_i\rho) > 0.$$

This implies that the given solution is not optimal, a contradiction. ◘

Corollary 2.9: If all the lots are of the same size, then in the optimal solution, the lots appear in the non-increasing order of their weights.

Based on Corollary 2.9, we can propose a DP-based algorithm, **RelaxFOUP-DP-Weighted** algorithm, which is very similar to the **RelaxFOUP-DP** algorithm. The result of Proposition 2.14 helps in reducing computational efforts. The solution determined by **RelaxFOUP-DP-Weighted** algorithm is $LB3\{O \setminus P\}$.

Then, the lower bound for Method I can be computed as follows:

$$LB\{P\} = \sum_{i \in P} C_i + \max(LB1\{O \setminus P\}, LB2\{O \setminus P\}, LB3\{O \setminus P\}) + (N - N_P) * \sum_{i \in P} C_i .$$

We do not derive lower bound for Method II addition directly. But, once a Method II addition is executed, the Method I lower bound analysis is used for its child nodes to determine a lower bound.

Fathoming strategies

We have the following strategies for Method I and Method II

Fathoming strategies for Method I addition:

A node is fathomed if

(1) $\dfrac{\sum_{i \in J_{m-1}} s_i}{n_{m-1}} \le \dfrac{\sum_{j \in J_m} s_j}{n_m}$ does not hold for neighborhood carriers *m-1* and *m*, for *m*=2, .., *L*.

(2) $s_a \le s_b$ does not hold when a lot *a* from carrier *m-1* and a lot *b* from carrier *m* can be exchanged without violating FOUP capacity for neighborhood carriers *m-1* and *m*, for *m*=2, …, *L*.

(3) $\sum_{j \in J_m} s_j \ge (n_{m-1} - 1) \max_{i \in J_{m-1}} s_i$ does not hold when a lot *b* from carrier *m* can be included in carrier *m-1* without violating FOUP capacity for neighborhood carriers *m-1* and *m*, for *m*=2, …, *L*.

(4) $\sum_{j \in J_m} s_j \le (n_{m-1} + 1) \min_{j \in J_m} s_j$ does not hold when a lot *a* from carrier *m-1* can be included in carrier *m* without violating FOUP capacity for neighborhood carriers *m-1* and *m*, for *m*=2, …, *L*.

(5) $\frac{\sum_{i \in O \setminus P} s_i}{K} > L - L_P$.

(6) $N - N_P < L - L_p$.

(7) $N - N_P = L - L_p$, in which case we find the optimal solution for that node.

(8) the solution generated by the **RelaxFOUP-DP** algorithm is feasible to the original problem, because it is an optimal solution for that node.

(9) the solution generated by **RelaxFOUP-DP-Weighted** is feasible to the original problem, in which case it is an optimal solution for that node.

(10) $LB\{P\} \ge CurrentBestSolution$

Fathoming strategies for Method II addition:

A node is fathomed if

(11) it violates FOUP capacity *K*.

(12) the index of the last lot is not the largest among the scheduled set *P*.

With fathom strategy (12), we can eliminate duplication by indexing the lots in the non-decreasing order of lot sizes as sequencing of lots in a carrier is irrelevant

The **MLCSP1-B&B** algorithm can now be presented as follows.

MLCSP1-B&B Algorithm

Step 1: Arrange the lots in the non-decreasing order of sizes.

Step 2: At node 0 (root node), create a list of active nodes, L, $L = \{0 : \text{root node}\}$.

Step 3: Determine lower bound $LB\{P\}$ and upper bound $UB\{P\}$ at node 0. If the solution to the lower bound is feasible to the original problem, go to step 8; otherwise let $z^* = UB\{P\}$ and $\bar{z} = LB\{P\}$. If $z^* = \bar{z}$, go to step 8.

Step 4: Partition root node into its children nodes, and update list L.

Step 5: Check the node list L. If $L = \varnothing$, go to step 8; otherwise, extract the first node k from L and make it a current node (depth-first node selection strategy is used).

Step 6: At node j, apply fathoming strategies (1) to (10) if it is a Method I addition, and apply fathoming strategies (11) –(12) if it is a Method II addition. If the current node is fathomed, go to step 7; otherwise, determine $LB\{P\}$ and $UB\{P\}$; update z^* and $\bar{z}$ if necessary; if $z^* = \bar{z}$, go to step 8; otherwise, go to step 7.

Step 7: If at node k, scheduled set of lots P does not contain N lots, partition node k into its children nodes, designate this set of nodes by ψ_k (note that we have Method I and Method II for the generation of nodes), and add ψ_k to L, go to step 5; otherwise if node k contains N lots, node k cannot be portioned further, and go to step 5

Step 8: We have found an optimal solution with objective value z^*. Stop.

2.6 Numerical experimentation

In this section, we test the performance of **MLCSP1-B&B** with the direct solution of the problem by using the AMPL CPLEX Solver (version 10.1). The performance measure is cpu time required to find the optimal solution. The summary of the test data is contained in Table 2.2. We test five cases with 10, 15, 20, 23, 25 lots. For each case, three L values are used, namely, small, medium and large, which result in schedules with high, medium and low densities (density is the ratio of number of lots over number of carriers). Thus, there are a total of $5 \times 3 = 15$ data sets. For each data set, we generate 20 problem instances with randomly generated lot sizes.

Table 2.2: Data used in numerical experimentation for problem MLCSP1

Number of lots (N)		10, 15, 20, 23, 25
Number of carriers available (L)	High density case	$\left\lceil \frac{N}{K/\bar{s}} \times 100\% \right\rceil$
	Medium density case	$\max\{\left\lceil \frac{N}{K/\bar{s}} \times 100\% \right\rceil + 1, \left\lceil \frac{N}{K/\bar{s}} \times 140\% \right\rceil\}$
	Low density case	$\max\{\left\lceil \frac{N}{K/\bar{s}} \times 100\% \right\rceil + 2, \left\lceil \frac{N}{K/\bar{s}} \times 180\% \right\rceil\}$
Lot size (s_i)		Uniform distribution [1,10] †
Average lot size ($\bar{s}$)		5.5
Processing time per wafer (carrier) ρ		This value does not affect the solution, and is assumed to be 1 time unit.
Carrier capacity (K)		25 wafers
H		Sum of the processing times of all lots, $\sum_{i=1}^{N} s_i$

(† see ITRS 2006 Factory Integration)

We coded **MLCSP1-B&B** by using Excel VBA (version 2003). All numerical tests were performed on a Dell computer with Pentium 4 processor (2.8GHz) in the Electronics Manufacturing Research (EMR) Lab at Virginia Tech.

The average cpu times required to find the optimal solutions for our algorithm and the AMPL CPLEX 10.1 Solver are presented in Table 2.3. It can be seen that for each data set, **MLCSP1-B&B** algorithm obtains optimal solution much faster than the CPLEX Solver. In fact, the CPLEX Solver cannot solve the problem involving 15 lots and 6 carriers, and all problems involving 20 or more lots, within the allowable cpu time of 14,400 seconds. This is due to two reasons. The first reason is the use of a large positive number H in the integer programming (IP) model, MLCSP1, which impacts the tightness of the model. The second reason is that, when the number of carriers (L) increases, the number of binary variables X_{ik} increases as well, which leads to longer cpu time needed for the solver to find an optimal solution. On the contrary, an increment of L affords algorithm **RelaxFOUP-DP** in generating solutions without violating the FOUP capacity constraint. In other words, the lower bound generated by **RelaxFOUP-DP** tends to be an optimal solution. Thus, **MLCSP1-B&B** algorithm has the tendency of solving the problem at node 0 itself, thereby, resulting in a sharp decrement in average cpu time with increase in the value of L. This phenomenon can be observed for all instances of number of lots used (see Table 2.3).

Table2.3: The average cpu times required to find optimal solutions by MLCSP1-B&B and the AMPL CPLEX 10.1 solver

Data set	Number of Lots (*N*)	Number of Carriers (*L*)	Average cpu time (seconds)	
			MLCSP1-B&B	CPLEX 10.1 Solver
1	10	3	0.18	0.32
2	10	4	0.06	1.61
3	10	5	0.06	8.87
4	15	4	7.57	29.50
5	15	5	1.60	650.54
6	15	6	0.06	>14400
7	20	5	418.63	>14400
8	20	7	19.23	>14400
9	20	8	0.06	>14400
10	23	6	2429.5	>14400
11	23	8	0.16	>14400
12	23	10	0.11	>14400
13	25	6	4767.78	>14400
14	25	8	0.13	>14400
15	25	10	0.10	>14400

In order to further investigate the impact of the number of carriers (i.e., different densities for the same number of lots), additional tests, were run, with different combinations of number of lots (N) and number of carriers (L). Percentage of times in which our algorithm solves the problem at node 0 for the 20 problem instances tested is summarized in Table 2.4. The number of lots (N) considered are 10, 30, 50, 70, 90 and 100. For each value of N, different numbers of carriers (L) were used resulting in different densities.

Note that when the number carriers is low (i.e., the density is high), our algorithm is unlikely to generate optimal solution at node 0, which implies that algorithm **RelaxFOUP-DP** leads to a solution, which violates the FOUP capacity. For this case, lower bounds generated by other algorithms such as **RelaxFOUP-DP-Weighted** are more efficient and **MLCSP1-B&B** can solve problems with number of lots up to 25. However, with slight decrement of density (i.e., increment of number of carriers), the percentage of times an optimal solution is obtained at node 0 increases sharply (see Figure 2.16). Note that in Figure 2.16, the percentage is the ratio of number of carriers over the base value of the number of carriers $\left[\frac{N}{\overline{K/s}} \times 100\%\right]$.

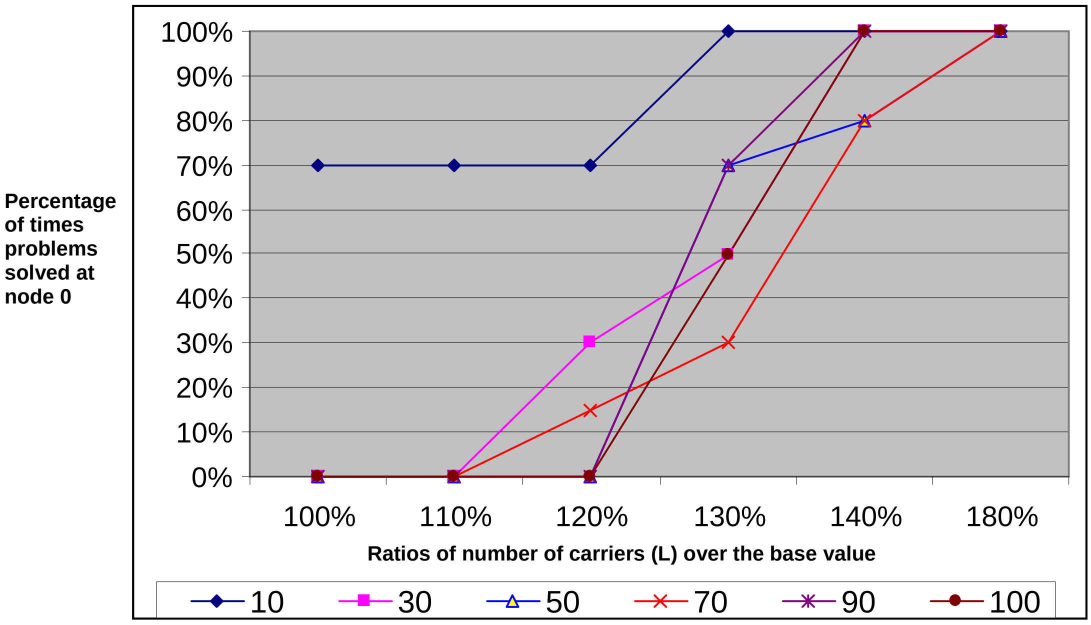

Figure 2.16: **Percentage of times MLCSP1-B&B solves the problem at node 0 as with increasing number of carriers (*L*)**

Table 2.4: Percentage of times MLCSP1-B&B solves the problem at node 0

Number of Lots (N)	Ratios of number of carriers over the base value†	Number of Carriers (L)	Percentage of times an optimal solution is obtained at node 0††
10	100%	3	70%
10	110%	3	70%
10	120%	3	70%
10	130%	4	100%
10	140%	4	100%
10	180%	5	100%
30	100%	8	0%
30	110%	8	0%
30	120%	9	30%
30	130%	10	50%
30	140%	11	100%
30	180%	13	100%
50	100%	13	0%
50	110%	14	0%
50	120%	15	0%
50	130%	16	70%
50	140%	17	80%
50	180%	22	100%
70	100%	17	0%
70	110%	19	0%
70	120%	21	15%
70	130%	22	30%
70	140%	24	80%
70	180%	31	100%
90	100%	22	0%
90	110%	24	0%
90	120%	26	0%
90	130%	29	70%
90	140%	31	100%
90	180%	39	100%
100	100%	25	0%
100	110%	27	0%
100	120%	29	0%
100	130%	32	50%
100	140%	34	100%
100	180%	44	100%

(† The base value is $\left\lceil \frac{N}{K/\bar{s}} \right\rceil$. ††The ratio of the number of instances over the 20 instances in which the instances were solved at node 0.)

2.7 Determination of optimal number of carriers *L* considering transportation cost

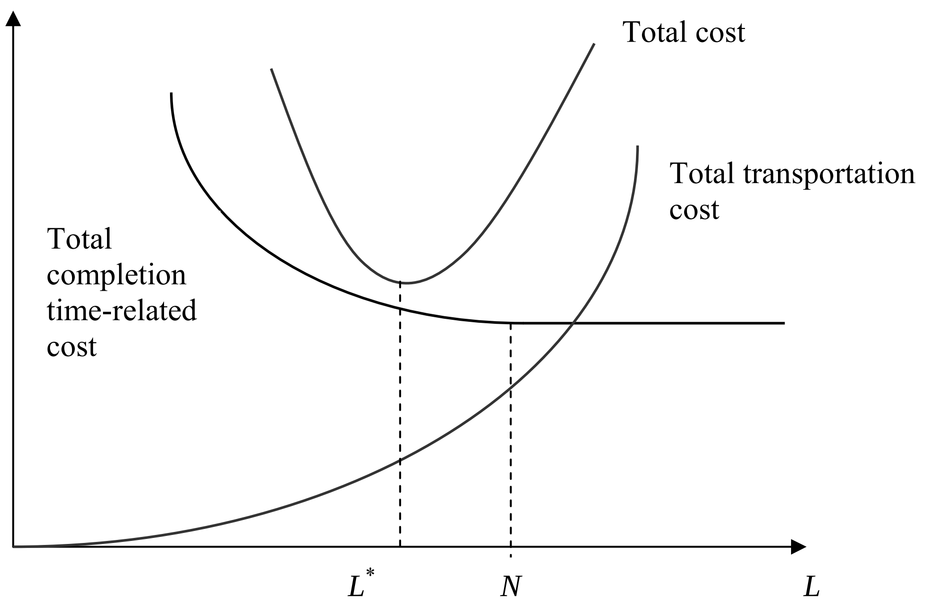

Figure 2.17: Relationship between total completion time-related, transportation, and total costs with number of carriers *L*

If we consider the transportation cost per carrier, and we are also able to determine the cost related to total completion time, a relationship between total cost and number of carriers, *L*, can be determined as shown in Figure 2.17.

We can determine the optimal value of *L* by using the following model for a given *L*, and then, by enumerating over *L*.

Minimize $$\theta_1 \sum_{i=1}^{N} C_i + \theta_2 * \sum_{k=1}^{N} Y_k$$

Subject to

$$\sum_{k=1}^{N} X_{ik} = 1 \qquad \forall i = 1, \ldots, L \tag{2.8}$$

$$\sum_{i=1}^{N}(X_{ik} * S_i) \le Y_k * K \quad \forall k = 1, 2, \ldots, L \tag{2.9}$$

$$C_i \ge F_k - H(1 - X_{ik}) \quad \forall i = 1, \ldots, N, \text{ and } k = 1, \ldots, L \tag{2.10}$$

$$F_k = \sum_{l=1}^{k} \sum_{i=1}^{N}(X_{il} * s_i \rho), \ k = 1, 2, \ldots, L \tag{2.11}$$

$$X_{ik}, \ Y_k \text{ binary; } C_i, \ F_k \ \ge 0, \ \forall \ i = 1, \ldots, N, \text{ and } k = 1, \ldots, L \tag{2.12}$$

We can determine a lower bound of the minimal total cost by applying the **RelaxFOUP-DP** algorithm. Also, a reasonable range for L is from $L_{\min} = \left\lceil \frac{\sum_{i=1}^{N} s_i}{K} \right\rceil$ to $L_{\max} = N$.

2.8 Conclusions

In this chapter, we have addressed a single machine, multiple-lot-per-carrier with single-wafer processing technology scheduling problem for the objective of minimizing the total completion time (MLCSP1). We have first formulated this problem as an integer programming model. Due to the complexity involved in solving this model directly, we analyze variations of this problem in order to determine some inherent structural properties. To that end, we, first, study the problem with relaxed FOUP capacity. For this problem, when all the lots are of the same size, the optimal solution can be obtained easily. However, for the case of different lot sizes, we propose a dynamic programming-based algorithm, **RelaxFOUP-DP.** For the problem with limited FOUP capacity, when all the lots are of the same size, we show that the optimal solution is the same as that for the relaxed problem. When the lots are of different sizes, a branch-and-bound algorithm, **MLCSP1-B&B**, is developed that relies on the **RelaxFOUP-DP** algorithm**.**

Numerical tests indicate that **MLCSP1-B&B** finds optimal solutions much faster than the direct solution of the MLCSP1 model using the AMPL CPLEX 10.1 Solver. The CPLEX Solver cannot solve problems with 15 lots or more within the allowable cpu time of

14,400 seconds. In particular, our computational results show that, with an increment in the number of carriers, L, the cpu time required by the CPLEX Solver increases sharply while that required by **MLCSP1-B&B** for these instances falls dramatically. In fact, for the medium and low density problems, the **RelaxFOUP-DP** algorithm finds the optimal solution at node 0 itself whereas the AMPL CPLEX Solver is unable to solve these problem instances within the allowable cpu time.

Chapter 3: Minimizing Total Completion Time for Single Machine MLCSP with Single-Carrier-Processing-Technology (MLCSP2)

3.1 Introduction

In this chapter, we address the single-machine MLCSP for single-carrier-processing, designated MLCSP2. In contrast to the MLCSP for single-wafer-processing discussed in Chapter 2, the processing time of a carrier is not dependent on the number of wafers in that carrier, but it is a fixed value. For a detailed description of the MLCSP2, please see Chapter 2.

This chapter is organized as follows. The mathematical formulation for the MLCSP2 is presented in Section 3.2. We derive some useful properties of this problem and show in Section 3.3 that the problem can be solved in polynomial time when all the lots are of the same size. In Section 3.4, we determine a lower bound and upper bound for this problem, and the worst case analysis of the bounds is also presented in this section. We, finally, conclude the chapter in Section 3.5.

3.2 A mathematical model for the MLCSP2

Parameters

N: total number of lots
L: total number of available carriers, obviously $L < N$
s_i: size of lot i (number of wafers), i =1, 2, …, N
w_i: weight for lot i
K: FOUP capacity (usually, K= 25)
ρ: processing time per carrier
C_i: completion time of lot i, i =1, 2, …, N

$\sum C_i$: total completion time

n_j : number of lots in carrier *j*

J_j : set of lots forming carrier *j*

O : set of all lots available to be scheduled

P : set of the already scheduled lots, $P \subseteq O$

N_P : number of lots belonging to set *P*

L_P : number of carriers formed by the set of lots in *P*

Decision Variables

$$X_{ik} = \begin{cases} 1, & \text{if lot } i \text{ is assigned to carrier } k, \\ 0, & \text{otherwise.} \end{cases}$$

$$Y_k = \begin{cases} 1, & \text{if carrier } k \text{ is not empty} \\ 0, & \text{otherwise.} \end{cases}$$

An integer linear programming model for this problem has been presented by Qu (2004). However, it uses a large positive number, *M*, in the model, which makes the model computationally inefficient. We present another method that does not rely on the use of a large positive number. This is afforded by formulating the MLCSP2 as a special case of a generalized assignment problem (GAP). Each carrier is assumed to be a sink, and each lot is considered as a resource (see Figure 3.1). The cost of assigning one lot to a carrier *k* is $k\rho$.

Model MLCSP2

Minimize $\sum_{i=1}^{N} C_i$

Subject to

$$\sum_{k=1}^{N} X_{ik} = 1 \quad \forall i = 1, \ldots, N \tag{3.1}$$

$$\sum_{i=1}^{N} (X_{ik} * s_i) \le K \quad \forall k = 1, \ldots, N \tag{3.2}$$

$$C_i = \sum_{k=1}^{N} (X_{ik} * k\rho) \ \ \forall i = 1, \ldots, N \tag{3.3}$$

$$X_{ik} \text{ binary, } \forall i = 1, \ldots, N, \text{ and } \forall k = 1, \ldots, N \tag{3.4}$$

Constraint set (3.1) permits assignment of a lot to only one carrier, and FOUP capacity is enforced by constraint set (3.2). For MLCSP2, processing time of a lot (and the completion time) is the same as that of the carrier the lot is contained in. This is represented by constraint set (3.3). Note that each carrier is identical to each other. Thus in the model, we can use arbitrary sequence of carriers, *1, 2, …, L*. The decision variable X_{ik} automatically enforces the best carrier formation and sequencing.

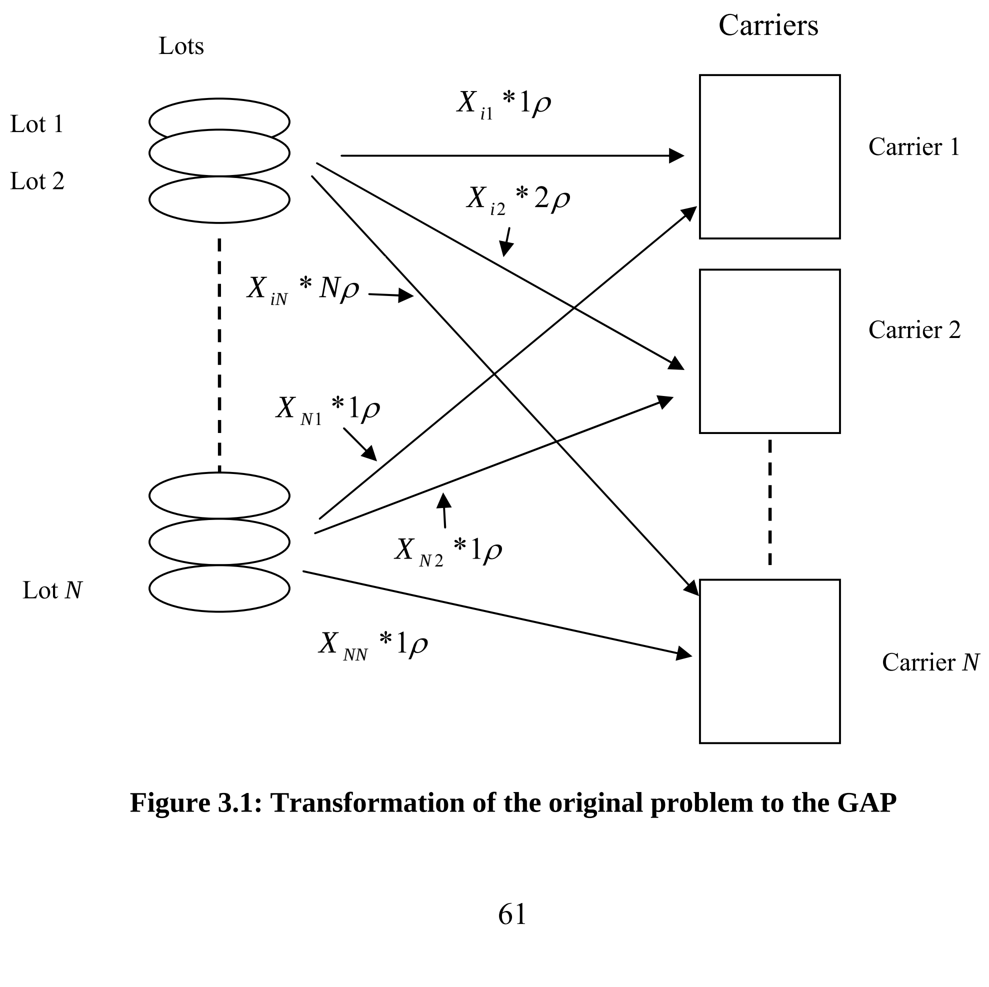

Figure 3.1: Transformation of the original problem to the GAP

The largest value of L is N. The above model will determine the optimal number of carriers that is required to minimize the total completion time.

3.3 Structural properties

Proposition 3.1: Problem MLCSP2 is NP-hard.

Proof: We can rewrite model MLCSP2 as follows:

Minimize $$\sum_{i=1}^{N}\sum_{k=1}^{N}(X_{ik} * k\rho)$$

Subject to

$$\sum_{k=1}^{N} X_{ik} = 1 \quad \forall i = 1, \ldots, N \tag{3.5}$$

$$\sum_{i=1}^{N}(X_{ik} * s_i) \le K \quad \forall k = 1, \ldots, N \tag{3.6}$$

$$X_{ik} \text{ binary}, \ \forall i = 1, \ldots N, \ \forall k = 1, \ldots, N \tag{3.7}$$

Note that this is a generalized assignment problem (GAP) which is a well-known NP-hard problem (Fisher et al. 1986). ◘

Qu (2004) in his paper claims that an optimal solution to the MLCSP2 will utilize the minimum number of carriers (FOUPs). However, this claim is not true. We show this by the following counter example. First, note that the problem that minimizes the number of FOUPs used is the well-known bin-packing problem for which we have the following formulation.

Bin packing problem

Minimize $\sum_{k=1}^{N} Y_k$

Subject to

$$\sum_{k=1}^{N} X_{ik} = 1, \quad \forall i = 1, \ldots, N \tag{3.8}$$

$$\sum_{i=1}^{N} (X_{ik} * s_i) \le Y_k * K, \quad \forall k = 1, \ldots, N \tag{3.9}$$

$$X_{ik} \text{ binary}, \ \forall i = 1, \ldots \text{N}, \ \forall k = 1, \ldots, N \tag{3.10}$$

Clearly, the bin packing problem is different from our problem as the objective functions are different. Consider the data shown in Table 3.1 (*K*=25).

Table3.1: Data for a counter example ($\rho = 1$)

Lot	1	2	3	4	5	6	7	8	9	10	11	12
Lot size	8	19	13	5	15	10	15	22	3	17	5	12

The optimal solution to the bin packing problem is shown in Table 3.2. It uses 6 carriers. The total completion time = 2(1+2+3+4+5+6)=42.

Table3.2: An optimal solution to the bin packing problem

Bin	1	2	3	4	5	6
Lots	5,11	3,12	1,10	6,7	2,4	8,9

The optimal solution to our problem is given in Table 3.3. Although it uses more carriers, yet the total completion time =4(1)+2(2)+2(3)+1(4+5+6+7)=36, which is less than that for the solution that minimizes the number of carriers used.

Table3.3: An optimal solution to the original problem

Bin	1	2	3	4	5	6	7
Lots	4,9,11,12	3,6	1,7	5	10	8	2

Proposition 3.2: In an optimal solution to the MLCSP2, the carriers are arranged in the non-increasing order of the number of lots contained in them.

Proof : Let *n* be number of carriers formed, and n_i be number of lots in carrier *i*, for *i* =1, 2, …, *n*. The total completion time, $\sum_{i=1}^{N} C_i == 1\rho * n_1 + 2\rho * n_2 + ... + n\rho * n_n$, which is minimized if n_1, n_2, …, n_n are in non-increasing order. ◘

Corollary 3.1: If the objective is to minimizes the total weighted completion time, then the optimal sequence is obtained by arranging the carriers in the non-increasing order of $\sum_{i \in J_i} w_i$.

Based on Proposition 3.2, we can derive the following greedy heuristic to solve the MLCSP2.

Greedy Heuristics (GH):

Step 1: Arrange the lots in the non-decreasing order of lot sizes.

Step 2: Fill the carriers with the lots selected in the order determined in Step 1

Note that this procedure need not give an optimal solution because in an optimal solution, the smallest lot may not be scheduled in the first carrier, and the largest lot may not be scheduled in the last carrier.

Proposition 3.3: In an optimal solution, the smallest lot in a carrier should be larger than any empty space in the preceding carriers, i.e.,

$K - \sum_{i \in J_l} (X_{il} * s_i) < X_{ik} * s_i$, $\forall i, l < k, k = 2,3,4,....N$, and $X_{ik} = 1$.

Proof: If such a situation does not exist then, clearly, the total completion time is improved by moving the appropriate lot to an earlier carrier with enough remaining space to accommodate it as this move does not affect other carriers. ◘

Corollary 3.2 In an optimal schedule, the largest of the empty spaces in the first *n* carriers, $n \le L$, should be smaller than the smallest lot size in the remaining carriers, i.e.,

$\max_{k \le n}(K - \sum_{i \in J_k} (X_{ik} * s_i) < \min_{\substack{j \in J_l \\ L > l > n}} s_j$, $\forall n = 1, \ldots L$.

Corollary 3.3: (A polynomially solvable case). If all the lots are of the same size *s*, then there exists an optimal solution in which the carriers are filled as much as possible.

Proof: This result directly follows from Proposition 3.3. ◘

If we let maximum number of lots, which can be included in one carrier to be *A*, number of carriers with maximum number of lots to be *D,* and the leftover lots which can not be included in the first *D* carriers to be *B*, then we have $A = \left\lfloor \frac{K}{s} \right\rfloor$, $B = N - \left\lfloor \frac{N}{A} \right\rfloor$, $D = \left\lfloor \frac{N}{A} \right\rfloor$.

Thus $\sum_{i=1}^{N} C_i = \sum_{n=1}^{D} A * n\rho + B * (D+1)\rho$.

3.4 Determination and analysis of a lower bound and an upper bound

3.4.1 Determination of a lower bound

A lower bound for the unscheduled set of lots $\{O \setminus P\}$ can be obtained from a relaxed problem, which is formulated as follows.

For the unscheduled set $\{O \setminus P\}$, let γ be the largest common factor among all the lots. We split each lot into sublots of size γ, with each sublot having the weight of $\frac{\gamma}{s_i}$.

The algorithm to obtain an optimal solution for this relaxed problem is as follows.

Algorithm OpRelax:

Step 1: Arrange the lots in the non-decreasing sequence of lot sizes.

Step 2: Split each lot into $\frac{s_i}{\gamma}$ sublots, each of size γ

Step 3: Fill the FOUPs with the sublots defined in Step 2 and selected in the order determined in Step 1 as much as possible.

The optimality of the solution obtained for this relaxed problem follows by Corollary 3.1.

Let the total completion time obtained by the **OpRelax** algorithm be denoted by $LB\{O \setminus P\}$. For partial schedule P under Method I addition (see Chapter 2, Section 2.5), the lower bound can be computed as

$$LB\{P\} = \sum_{i \in P} C_i + LB\{O \setminus P\} + (N - N_P) * (L_P \rho) .$$

Proposition 3.4: $LB\{O \setminus P\} \geq 0.5 * \min \sum_{i \in O \setminus P} C_i$

Proof: Note that in the optimal solution obtained using the **OpRelax** algorithm for the relaxed problem, in the first carrier and last carrier, there can be at most one lot each of which is not totally contained in them, while in the other carriers, there can be at most two lots that are not totally contained in them (see Figure 3.2). One way to heuristically adjust the schedule to ensure that all the lots are totally contained in the carriers is as follows.

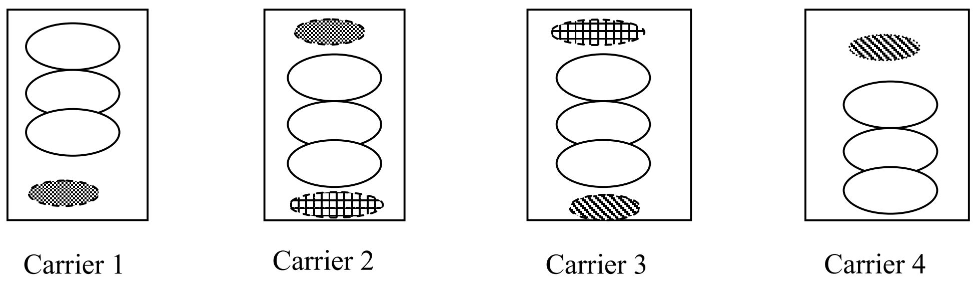

Figure3.2: An example of the optimal solution for the relaxed problem (L' =4)

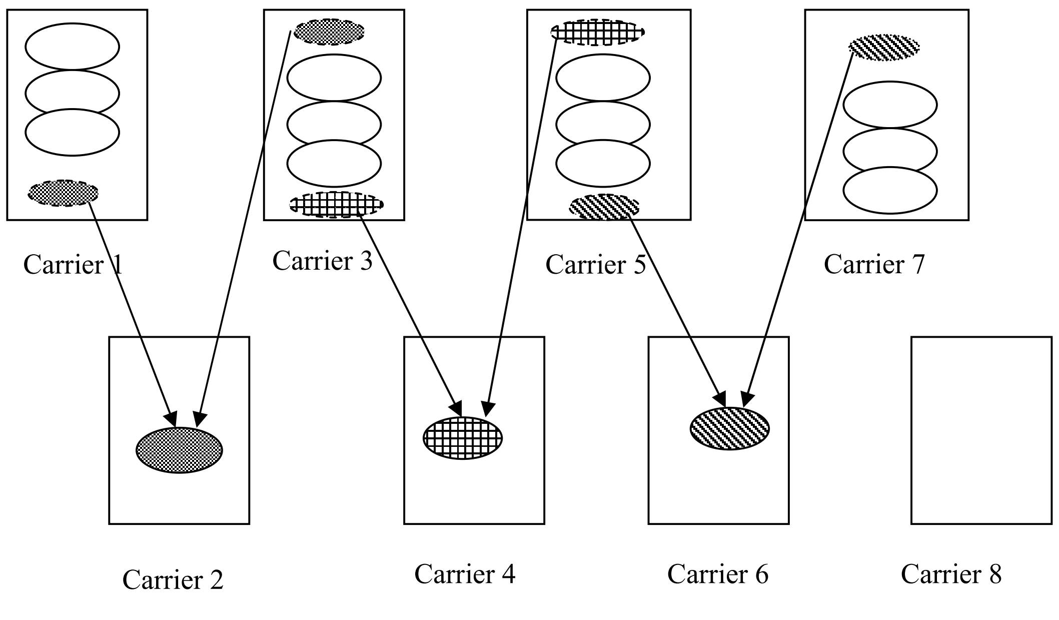

Figure 3.3: An example for the adjustment of the optimal solution for the relaxed problem ($L' = 4$)

Suppose L' carriers are used for the lot set $\{O \setminus P\}$ and we can insert one dummy carrier between any two neighborhood carriers as shown in Figure 3.3. Thus, there are *2L'* carriers formed totally. For carrier $2k-1$, *k*=1, …, *L'*, we move out the partial lot in the carrier and include it in its neighboring carrier $2k-2$ or $2k$ depending on whether the other part of the partial lot is in the carrier that is before or after it.

Let Δw_{2k-1}^{a} be the total weights of the sublots in carrier *2k-1*, which are put in carrier *2k-2*, Δw_{2k-1}^{b} be the total weights of sublots in carrier *2k-1* which are put in carrier *2k*, for *k*=1, …, *L'*. Let the total completion time after adjustment be $\sum_{i\in\{O\setminus P\}} C'_i$, we have

$$\begin{aligned}\sum_{i\in\{O\setminus P\}} C'_i &= \sum_{k=1}^{L'}(2k-1)\rho\Big(\sum_{i\in J_{2k-1}} w_i - \Delta w_{2k-1}^{a} - \Delta w_{2k-1}^{b}\Big) + \sum_{k=1}^{L'}[2k\rho\Delta w_{2k-1}^{b} + 2(k-1)\rho\Delta w_{2k-1}^{a})] \\ &\le \sum_{k=1}^{L'}\ [2k\rho \sum_{i\in J_{2k-1}} w_i] \\ &= 2 * LB\{O \setminus P\}\end{aligned}$$

Since $\sum_{i\in\{O\setminus P\}} C'_i \ge \min \sum_{i\in\{O\setminus P\}} C_i$, this implies that $2 * LB\{O \setminus P\} \ge \min \sum_{i\in\{O\setminus P\}} C_i$. ◘

3.4.2 Determination of an upper bound

Let $\sum_{i\in\{O\setminus P\}} C''_i$ indicate the total completion time obtained by the **GH** heuristic described in Section 3.3. This is an upper bound for the MLCSP2, indicated by $UB\{O \setminus P\}$, that is, $UB\{O \setminus P\} = \sum_{i\in\{O\setminus P\}} C''_i$.

Proposition 3.5 The GH heuristic has the worst case ratio of 2, i.e. $\dfrac{\sum_{i\in\{O\setminus P\}} C''_i}{\min \sum_{i\in\{O\setminus P\}} C_i} \le 2$.

Proof: From Proposition 3.4, we know $\sum_{i\in\{O\setminus P\}} C'_i \le 2 * LB\{O \setminus P\} \le 2\min \sum_{i\in\{O\setminus P\}} C_i$. Therefore, $\dfrac{\sum_{i\in\{O\setminus P\}} C'_i}{\min \sum_{i\in\{O\setminus P\}} C_i} \le 2$. Thus, we only need to prove $\sum_{i\in\{O\setminus P\}} C''_i \le \sum_{i\in\{O\setminus P\}} C'_i$.

Note that the solution obtained from the **GH** heuristic and the adjusted solution as shown in Figure 3.3 have the same sequence (the non-decreasing order of sizes) of lots. But, the **GH** heuristic tends to fill the carriers as much as possible. Thus, by Proposition 3.2, we have $\sum_{i\in\{O\setminus P\}} C''_i \le \sum_{i\in\{O\setminus P\}} C'_i$. ◘

Similarly, we can derive an upper bound for partial schedule *P* to be

$$UB\{P\} = \sum_{i\in P} C_i + UB\{O \mid P\} + (N - N_P) * (L_P \rho) .$$

3.5 Conclusions

In this chapter, we have addressed a single-machine, multiple-lots-per-carrier and single-carrier-processing technology scheduling problem for the objective of minimizing the total completion time (MLCSP2). We have first formulated this problem as a generalized assignment problem (GAP) instead of an integer programming model using a big number *M* as shown in the literature. We then analyze the problem and determine some inherent structural properties. Based on the results, the optimal solution for the case of equal-size lots can be obtained easily. For the case of different lot sizes, we determine a lower bound and an upper bound for the problem. It is shown that the worst case of the lower bound is 0.5 of the optimal solution and that of the upper bound is 2 times of the optimal solution.

Chapter 4: Minimizing Makespan for a 2-machine Flow Shop MLCSP with Single-Wafer-Processing (MLCSP3)

4.1 Introduction

In the single-carrier-processing problem, the processing time for a carrier is independent of the wafers contained in a carrier. An optimal schedule for the single-machine MLCSP would, therefore, tend to pack the lots into as few carriers as possible if the objective is to minimize the makespan. This problem is, thus, equivalent to the bin packing problem. This is also true for flow shops involving two or more machines (Laub et al. 2007). The bin packing problem is a well-known NP-hard problem. For an overview of work reported on the bin packing problem, see Coffman et al. (1996).

The single-wafer-processing problem is different from the single-carrier-processing problem as wafers are processed one-at-a-time. However, Laub et al. (2007) have proved NP-hardness of this problem for the single machine case under the criterion of minimizing the makespan. In this chapter, we address the 2-machine flow shop MLCSP for single-wafer-processing, and designate it by MLCSP3. As described in Chapter 2, we consider the following two scenarios: lots requiring identical processing times on each machine, and lots requiring different processing times on each machine. We develop methodologies to solve problems for both of these scenarios.

4.2 Methodology for the solution of MLCSP3

Parameters:

N: number of lots

L: number of available carriers, $L \le N$

M: number of machines

s_i: size for lot i (number of wafers), i =1, 2, …, N

K: FOUP (carrier) capacity (usually, K= 25)

ρ_{ij} : processing time per wafer of lot *i* on machine *j*

c_i : ratio of the processing time per wafer for lot *i* on machine 2 to that of machine 1,

$$c_i = \frac{\rho_{i2}}{\rho_{i1}}$$

Variables:

n_l : number of lots in carrier *l*

J_l : set of lots forming carrier *l*

C_{kj} : completion time of the carrier scheduled for processing at position *k* on machine *j*, *k* =1, 2, …, *L*, *j*=1,2,…, *M*

$C_{\max}$: makespan

$$X_{ik} = \begin{cases} 1, & \text{if lot } i \text{ is assigned for processing by carrier } k, \\ 0, & \text{otherwise.} \end{cases}$$

We can formulate a mixed integer programming model for the MLCSP3 as follows.

Model MLCSP3-1

Minimize $C_{\max}$

Subject to

$$C_{\max} \geq C_{kj},\ k=1,2,\ \ldots,\ L,\ j=1,2,\ \ldots,\ M \tag{4.1}$$

$$\sum_{k=1}^{L} X_{ik} = 1,\ i=1,\ 2,\ \ldots,\ N \tag{4.2}$$

$$\sum_{i=1}^{N} (X_{ik} s_i) \leq K,\ k=1,\ 2,\ \ldots,\ L \tag{4.3}$$

$$C_{11} \geq \sum_{i=1}^{N} (X_{i1} s_i \rho_{i1}) \tag{4.4}$$

$$C_{kj+1} \geq C_{kj} + \sum_{i=1}^{N} (X_{ik} s_i \rho_{kj+1}),\ k=1,2, \ldots, L,\ j=1, 2, \ldots, M\text{-}1 \tag{4.5}$$

$$C_{k+1,j} \geq C_{kj} + \sum_{i=1}^{N} (X_{ik+1} s_i \rho_{kj}),\ k=1,2,.., L\text{-}1,\ j=1, 2, \ldots, M \tag{4.6}$$

$$X_{ik} = \{0,1\};\ C_{kj} \geq 0,\ k=1, 2, \ldots, L,\ j=1, 2, \ldots, M \tag{4.7}$$

Note that the above formulation is developed for general number of machines, *M*. Constraint set (4.2) ensures that a lot is assigned to only one carrier. Constraint set (4.3) captures the capacity of a carrier. Constraint sets (4.4), (4.5), and (4.6) capture relationships among the completion times of the lots on the machines. Constraint set (4.4) ensures that the lot in the first position is not completed until all of its items have been processed. Constraint set (4.5) asserts that a lot at position *k* cannot start processing on machine *j+1* until it has finished processing on machine *j*. Similarly, a lot at position *k+1* cannot start processing on machine *j* until the lot at position *k* has finished processing on the same machine. The binary and non-negativity constraints are represented by constraint set (4.7). Constraint set (4.1) in conjunction with the objective function captures the makespan minimization objective.

Proposition 4.1: There exists an optimal solution in which all *L* carriers are used.

Proof: Suppose there does not exist any optimal schedule in which all *L* carriers are used. Then, there must exist an optimal schedule, *S,* in which not all *L* carriers are used. In other words, there must be at least one carrier in *S* that contains more than one lot and at least one carrier which is empty since $L \leq N$. Designate a carrier containing more than one lot by *l*, and an empty lot by *k* (see Figure 4.1). We can construct another schedule S' by removing one lot θ from carrier *l* and putting it into the empty carrier *k*, and then, inserting carrier *k* between carriers $l-1$ and *l* (see Figure 4.2). For schedule *S*, we have

$$C_{l1} = C_{l-1,1} + \sum_{i \in J_l} s_i \rho_{i1}\ ,\ C_{l2} = \max\{C_{l-1,2}, C_{l1}\} + \sum_{i \in J_l} s_i \rho_{i2}\ .$$

And, for schedule S', we have

$C'_{k1} = C'_{l-1,1} + s_\theta \rho_{\theta 1}$, $C'_{k2} = \max\{C'_{l-1,2}, C'_{k1}\} + s_\theta \rho_{\theta 2}$, $C'_{l1} = C'_{k1} + \sum_{\substack{i \in J_l \\ i \neq \theta}} s_i \rho_{i1}$, and

$$C'_{l2} = \max\{C'_{k2}, C'_{l1}\} + \sum_{\substack{i \in J_l \\ i \neq \theta}} s_i \rho_{i2}$$

$$= \max\{C'_{l-1,2} + s_\theta \rho_{\theta 2} + \sum_{\substack{i \in J_l \\ i \neq \theta}} s_i \rho_{i2}, C'_{l-1,1} + s_\theta \rho_{\theta 1} + s_\theta \rho_{\theta 2} + \sum_{\substack{i \in J_l \\ i \neq \theta}} s_i \rho_{i2}, C'_{l1} + \sum_{\substack{i \in J_l \\ i \neq \theta}} s_i \rho_{i2}\}.$$

Note that $C_{l1} = C'_{l1}$. Moreover, $C_{l2} \geq C'_{l2}$. If $C'_{l-1,2} (= C_{l-1,2}) \geq C_{l1} (= C'_{l1})$, then $C_{l2} = C'_{l2}$; otherwise, lot θ starts processing on machine 2 sooner than C'_{l1}, and it results in $C_{l2} > C'_{l2}$. Besides, carriers *l*+1, *l*+2 …, *L* remain unaffected in S' except that they may finish processing the lots earlier than those in *S*. Therefore, the makespan of schedule S' can not be worse than that of an optimal schedule *S*. This contradicts the assumption that there is no optimal schedule in which all *L* carriers are used. ◘

In view of the above result, we can, now, assume that all *L* carriers are used for processing the wafers.

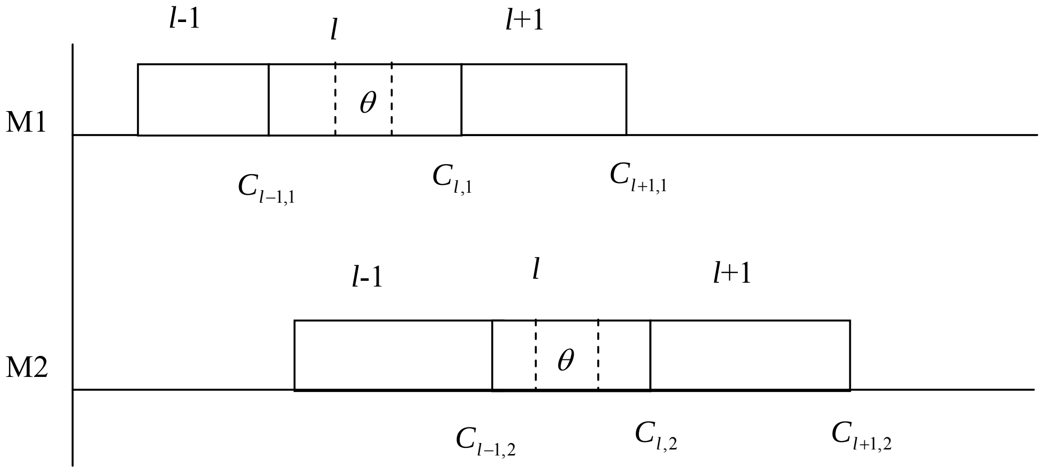

Figure 4.1: Schedule *S* with an empty carrier

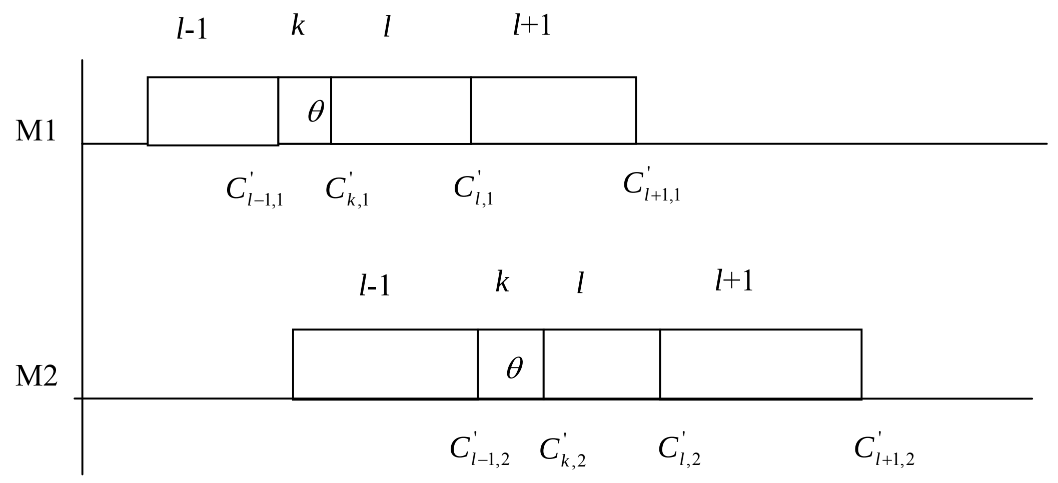

Figure 4.2: Schedule S' when none of the carriers is empty

4.2.1 An equivalent lot streaming problem

In Model MLCSP3-1 above, constraint sets (4.1), (4.4), (4.5), and (4.6) capture processing of the carriers in the flow shop. Constraint sets (4.2) and (4.3), which make this problem difficult to solve, are the bin packing type of constraints. If we relax these constraints, each lot can be broken into continuous-size sublots and assigned to multiple carriers with infinite capacities. With this relaxation, the original problem becomes an equivalent lot streaming problem in which we have one single "lot", being the aggregation of all the lots available for scheduling, and L "sublots", which are number of available carriers.

As described in Chapter 2, different lots may belong to a product, and thus, require the same processing time. Therefore, we aggregate lots into products. Then, different products are aggregated into one lot. We can, then, formulate this lot streaming problem as a linear programming model. To that end, we introduce additional notation.

Parameters:
I: number of products,

U_i, total number of wafers belonging to product *i*, *i*=1, 2, …, *I*

U : total number of wafers available to schedule, $U = \sum_{i=1}^{I} U_i$

ρ_{ij} : processing time per wafer of product *i* on machine *j* (note that, this variable is identical to that defined in Model MLCSP3-1)

<u>Variables:</u>

sl_{il} : number of wafers for product type *i* in sublot *l*, *i*=1, 2, …, *I*, *l*=1, 2, …, *L*

sl_l : number of wafers in sublot *l*, $sl_l = \sum_{i=1}^{I} sl_{il}$, *l*=1, 2, …, *L*

SL_l : set of wafers forming sublot *l*

C_{lj} : completion time of sublot *l* on machine *j*

T_{lj} : processing time of sublot *l* on machine *j*, $T_{lj} = \sum_{i=1}^{I} sl_{il}\rho_{ij}$, *l*=1, 2, …, *L*, *j*=1, 2, .., *M*

<u>Model MLCSP3-2</u>

Minimize $C_{\max}$

Subject to

$$C_{\max} \geq C_{LM} \tag{4.8}$$

$$\sum_{l=1}^{L} sl_{il} = U_i \text{, } i=1, 2, .., I \tag{4.9}$$

$$C_{l,j+1} \geq C_{l,j} + \sum_{i=1}^{I} sl_{il}\rho_{ij+1} \text{, } l=1, 2, \ldots, L, j=1, 2, .., M\text{-}1 \tag{4.10}$$

$$C_{l+1,j} \geq C_{l,j} + \sum_{i=1}^{I} sl_{il+1}\rho_{ij} \text{, } l=1, 2, \ldots, L\text{-}1, j=1, 2, \ldots, M \tag{4.11}$$

$$C_{1,1} \geq \sum_{i=1}^{I} sl_{i1}\rho_{i1} \tag{4.12}$$

$sl_{il} \geq 0$, i=1, 2, …, I, l=1, 2, …, L; $C_{lj} \geq 0$, l=1, 2, .., L, j=1, 2, .., M (4.13)

Constraint set (4.9) ensures that the wafers belonging to each product are, in fact, assigned to sublots. Constraint set (4.10) enforces that completion time of sublot l on machine j+1 is greater than or equal to its completion time on machine j plus its processing time on machine j+1. Similarly, constraint set (4.11) captures the fact that the completion time of sublot l+1 on machine j is greater than or equal to the completion time of sublot l on machine j plus its processing time on the same machine. Constraint set (4.12) makes sure that the completion time of the first sublot on first machine is the sum of the processing time of the wafers it contains. The non-negativity constraints are represented by constraint set (4.13). Constraint set (4.8) in conjunction with the objective function captures the makespan minimization objective.

4.2.2 Solution methodology for MLCSP3-2

Methodologies for various flow shop lot streaming problems are presented in Sarin and Jaiprakah (2007). Closed-form expressions for optimal sizes of a given number of sublots for a two-machine flow shop lot streaming problem are provided by Potts and Baker (1989). However, for MLCSP3-2 when m=2, the resulting optimal sublot sizes may be greater than the carrier capacity. Therefore, additional steps are required to determine optimal configuration of sublots for the capacitated case. However, a capacitated optimal solution to the lot streaming problem still violates the requirement that a lot cannot be split into different carriers. But, it does provide a good lower bound. We use a heuristic procedure to adjust the capacitated optimal solution, and find a good solution for the original 2-machine flow shop MLCSP (MLCSP3).

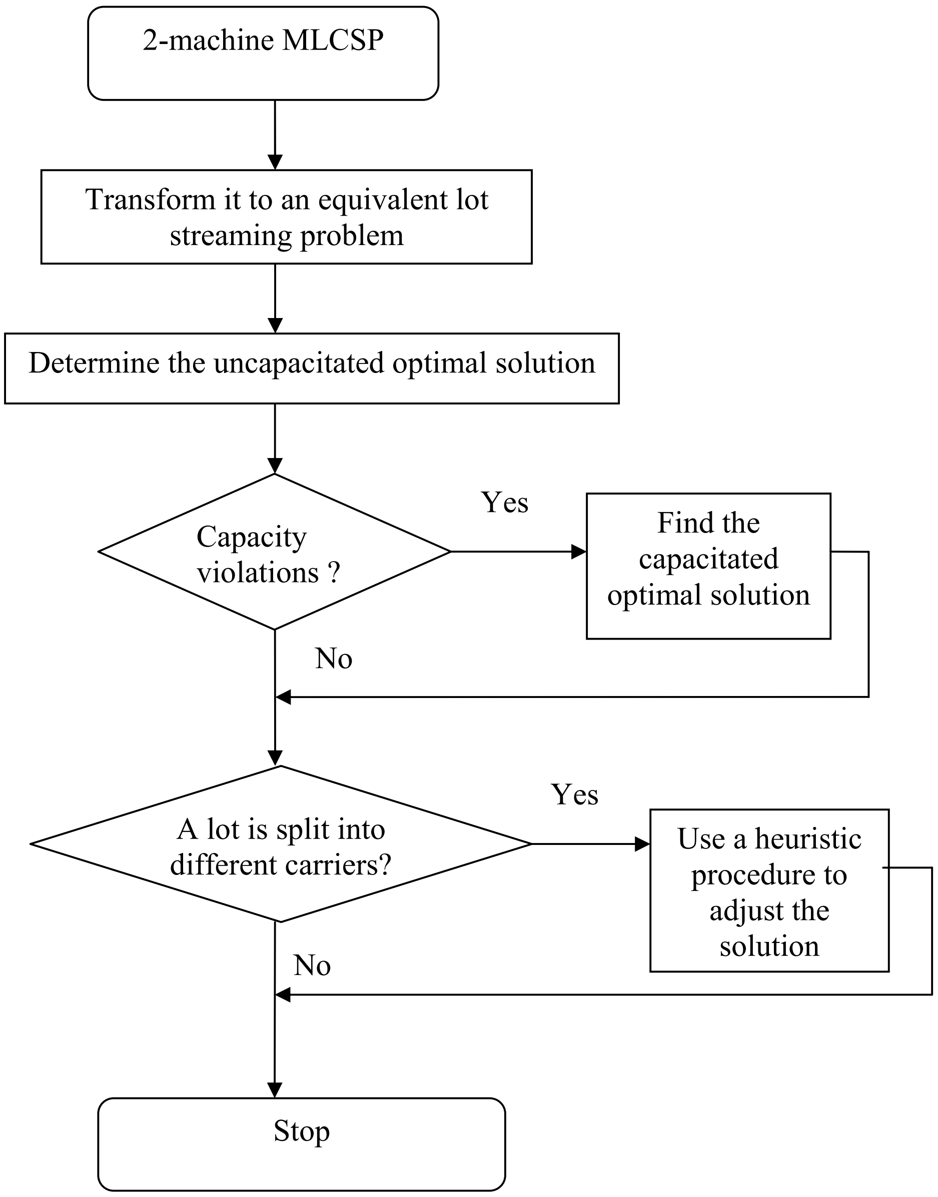
2-machine MLCSP
Transform it to an equivalent lot streaming problem
Determine the uncapacitated optimal solution
Capacity violations ?
Yes
Find the capacitated optimal solution
No
A lot is split into different carriers?
Yes
Use a heuristic procedure to adjust the solution
No
Stop

Figure 4.3: Flow chart of the heuristic procedure for the MLCSP3

The steps of our overall heuristic procedure for the solution of the MLCSP3 are as follows (please also see Figure 4.3).

Step 1: Transform the original MLCSP3 to an equivalent lot streaming problem.
Step 2: Determine the uncapacitated optimal solution for the lot streaming problem.

Step 3: Check whether any sublot violates carrier capacity constraint. If not, we get an optimal solution; go to step 4. Otherwise, find the capacitated optimal solution.

Step 4: Check whether any lot is split into different carriers. If not, we get an optimal solution; go to step 5. Otherwise, use a heuristic procedure to adjust the solution and assign wafers belonging to a lot to one sublot (carrier).

Step 5: Stop.

The rest of this chapter is organized as follows. The approaches to find the uncapacitated and capacitated optimal solutions are presented in Section 4.3 for lots requiring identical processing times on the machine. Section 4.4 addresses the case of the lots requiring different processing times on the machine. In Section 4.5, we present heuristics for adjusting a capacitated optimal solution in order to assign wafers belonging to a lot to one sublot. Results of our computational experimentation are presented in Section 4.6. We, finally, conclude the chapter in Section 4.7

4.3 Lots with identical processing times ($\rho_{ij} = \rho_j$, $c_i = c$)

4.3.1 Uncapacitated sublot sizes

Proposition 4.2 (Potts and Baker 1989). For the 2-machine one-lot lot streaming problem, optimal sublot sizes are geometric in nature, as follows:

$$sl_l = U\frac{c^{l-1}(c-1)}{c^L - 1},\ l=1, 2, \ldots, L.$$

The optimal makespan is

$$C_{\max} = U\rho_1 \frac{c^{L+1} - 1}{c^L - 1}.$$

The geometric sublot sizes for $c > 1$ are shown in Figure 4.4. Note that they increase in size with sublot number. On the other hand, the sublot sizes will be decreasing with sublot number when $c < 1$. The schedule with geometric sublot sizes forms a compact block structure, which means that there is no ide time between the processing of the sublots on the first and second machines (see Figure 4.5).

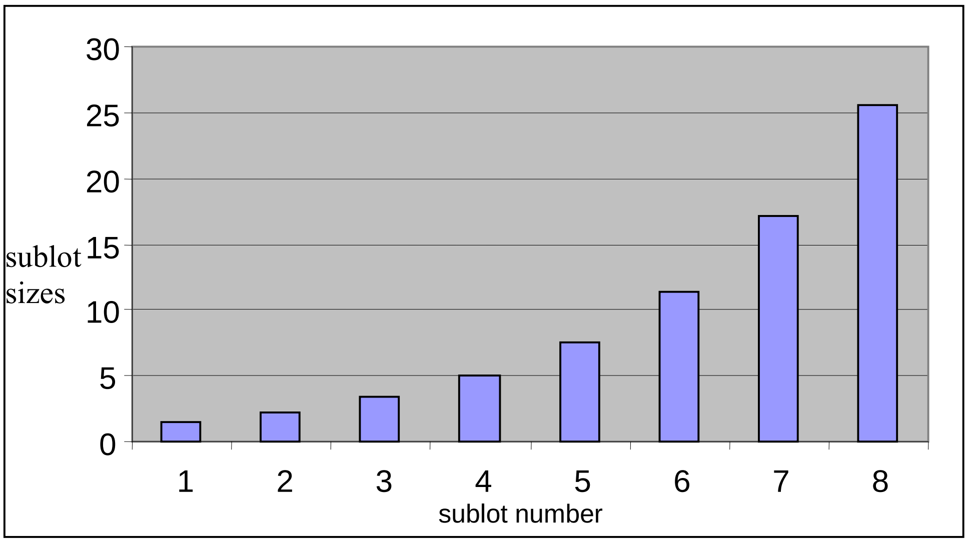

Figure 4.4: Sublot sizes for *c*=1.25 and *L*=8

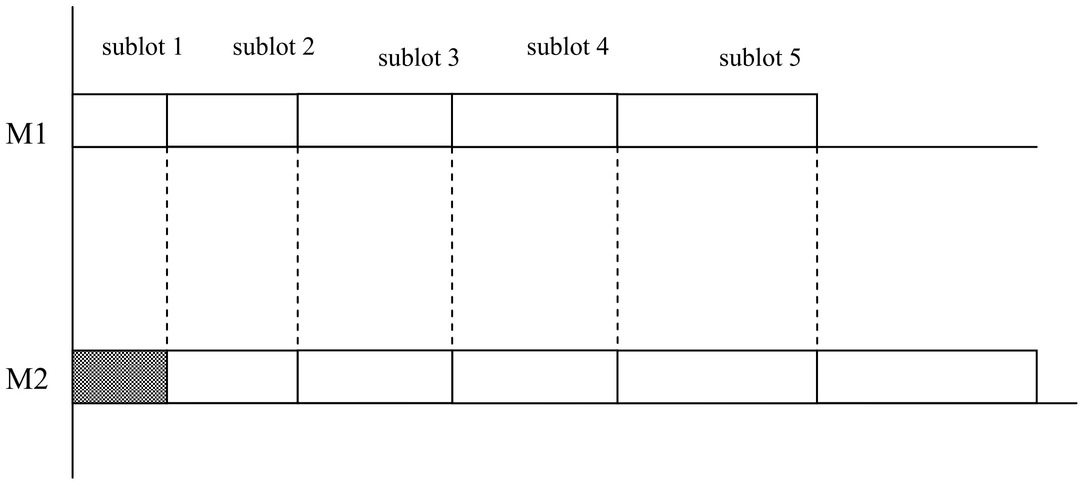

Figure 4.5: Compact block structure of optimal sublot sizes

4.3.2 Capacitated sublot sizes

Proposition 4.3 (Laub et al. 2007). For the 2-machine one-lot lot streaming problem with $c > 1$, the capacitated optimal sublot sizes are given by,

$$sl_l = \begin{cases} (U - bK)\dfrac{c^{l-1}(c-1)}{c^{L-b}-1}, & \text{for } 1 \le l \le L-b, \\ \text{K}, & \text{for } L-b < l \le L, \end{cases}$$

where *b* is the number of sublots of size *K*, and it is determined as follows:

$$b = \begin{cases} 0, & \text{if } U\dfrac{c^{L-1}(c-1)}{c^L-1} \le K, \\ \max\left\lceil \{x \mid (U - xK)\dfrac{c^{L-1-x}(c-1)}{c^{L-x}-1} > K\} \right\rceil, & \text{if } U\dfrac{c^{L-1}(c-1)}{c^L-1} > K, \end{cases}$$

and $\lceil x \rceil$ indicates the smallest integer larger than *x*.

The resulting makespan is given by,

$$C_{\max} = (U - bK)\rho_1 \frac{c^{L-b+1}-1}{c^{L-b}-1} + bK\rho_2 .$$

For a proof of the above result, please see Laub et al. (2007). An illustration of the type of optimal capacitated optimal sublot sizes can be found in Figure 4.6. Note that sublots 7 and 8 have been adjusted to meet the carrier capacity of 25 wafers.

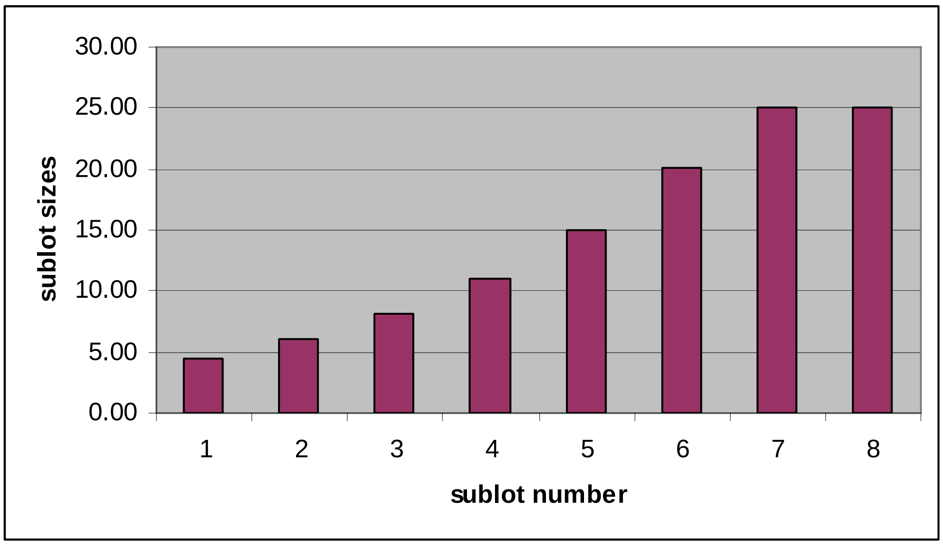

Figure 4.6: An illustration of the optimal capacitated sublot sizes with *c*=1.25 and *L*=8

From the classical scheduling theory, the reverse result holds when $c<1$, that is, the solution is read in the reverse order starting from machine 2 to machine 1. Thus, we have the capacitated optimal sublot sizes given by,

$$sl_l = \begin{cases} K, & for\ 1 \le l \le b, \\ (U-bK)\dfrac{c^{l-1}(c-1)}{c^{L-b}-1}, & for\ b < l \le L, \end{cases}$$

where *b* is the number of sublots of size *K*, and it is determined as follows:

$$b = \begin{cases} 0, & if\ U\dfrac{c-1}{c^L-1} \le K, \\ \max\left\lceil \{x \mid (U-xK)\dfrac{c-1}{c^{L-x}-1} > K\} \right\rceil, & if\ U\dfrac{c-1}{c^L-1} > K, \end{cases}$$

The resulting makespan is given by,

$$C_{\max} = (U-bK)\rho_1 \frac{c^{L-b+1}-1}{c^{L-b}-1} + bK\rho_1 .$$

From now on, in this chapter, we mainly discuss on the case of $c > 1$. The solution of the case of $c < 1$ is simply the reverse of that for the case of $c > 1$.

4.4 Lots with different processing times

As described in Chapter 2, different lots may belong to different product types and have different processing times. When they are placed in a carrier, the time required to process that carrier is the sum of the processing times for different product types. We have the following property for this case.

Proposition 4.4: There exits an optimal solution in which sublot sizes are consistent.

Proof: Our proof of this proposition follows the line of arguments used by Potts and Baker (1989) to show a similar result. Let x_{ilk} indicate the number of wafers of product type *i* belonging to sublot *l* on machine *m*, sl_{lk} be the size of sublot *l* on machine *k*, $sl_{lk} = \sum_{i=1}^{I} x_{ilk}$, X_{lk} be the cumulative size of the first *l* sublots on machine *k*, $X_{lk} = \sum_{j=1}^{l}\sum_{i=1}^{I} x_{ijk}$. By flow shop machine capacity constraints, we have

$$C_{lk} \geq C_{l-1,k} + \sum_{i=1}^{I} p_{il} x_{ilk} \text{, for } i=1, 2, \ldots, I, k=1, 2, \ldots, M; l=1, 2, \ldots, L \qquad \text{(i)}$$

By flow shop production constraints, we have

$$C_{lk} \geq C_{h(l,k),l-1} + \sum_{i=1}^{I} p_{il} x_{ilk} \text{, for } i=1, 2, \ldots, I, k=1, 2, \ldots, M; l=1, 2, \ldots, L \qquad \text{(ii)}$$

where $h(l,k)$ is the last sublot on machine *k*-1 containing wafers included in sublot *l* on machine *k*.

We want to show that there exists an optimal schedule in which $x_{il1} = x_{il2}$ and $x_{il,M-1} = x_{ilM}$ for *i*=1, 2, …, *I*, *l*=1, 2, …, *L*. Consider any optimal schedule defined by sublots sl_{lk} that give completion time C_{lk}. We first show that an alternative optimal schedule is given by sublots $sl_{lk}^{'}$, producing completion times C'_{lk}, where $x_{ilk} = x_{ilk}^{'}$ for i=1, 2, .., I, l=1, 2, …, L, k=1, 2, …, M-1, and $x'_{ilM} = x_{il,M-1}$, for i=1, 2, …, I, l=1, 2, .., L.

From (i) and (ii), we may assume that

$$C_{Lk}^{'} = C_{u',k-1}^{'} + \sum_{l=u'}^{L}\sum_{i=1}^{I} p_{ik} x_{ilk}^{'} \tag{iii}$$

for some u', where $x_{iu',k-1}^{'} = x_{iu'k}^{'} > 0$. We now construct a lower bound on the maximum completion time of the original schedule. For this original schedule, let u denote the first sublot on machine M, which contains wafers from sublot u' on machine $M-1$, i.e., sublot u is chosen so that

$$X_{u-1,M} \le X_{u'-1,M-1} < X_{uM}. \tag{iv}$$

Thus, for k=M and l=u in (ii), we have $h(u,M) \ge u'$. In addition, the middle term satisfies

$$X_{u'-1,M-1} = X_{u'-1,M-1}^{'} = X_{u'-1,M}^{'}. \tag{v}$$

Applying (i) and (ii), we obtain

$$C_{LM} \ge C_{h(u,M),M-1} + \sum_{l=u}^{L}\sum_{i=1}^{I} p_{iM} x_{ilM} \ge C_{u',M-1} + \sum_{l=u}^{L}\sum_{i=1}^{I} p_{iM} x_{ilM}. \tag{vi}$$

Since sublots on the first M-1 machines are the same in both the original and alternative schedules, we have $C_{u',M-1}^{'} = C_{u',M-1}$. Therefore, we deduce from (iii) and (vi) that

$$\begin{aligned} C_{LM} - C_{LM}^{'} &\ge p_{iM}\left(\sum_{l=u}^{L}\sum_{i=1}^{I} x_{ilM} - \sum_{l=u}^{L}\sum_{i=1}^{I} x'_{ilM}\right) \\ &= p_{iM}[(1 - X_{u-1,M}) - (1 - X_{u'-1,M}^{'})] \\ &\ge p_{iM}(X_{u'-1,M}^{'} - X_{u-1,M}). \end{aligned}$$

From (iv) and (v), we know that the term in parentheses must be nonnegative; hence it follows that $C_{LM} \ge C_{LM}^{'}$. This shows that our alternative schedule with consistent sublots on the last pair of machines is optimal. Thus, for two machine flow shop problem, sublot sizes are consistent. ◘

Proposition 4.5: The no idling restriction does not impact the optimal makespan value.

Proof: This result follows by the fact that the first machine always processes sublots without intermittent idling. By the reversibility property of flow shop, the same is true for the second machine as well. ◘

4.4.1 Same ratio of processing time per wafer on machine 2 to that on machine 1 $(c_i = c)$

4.4.1.1 Uncapacitated optimal sublot sizes

Proposition 4.6: In the optimal solution, all the sublots are critical.

Proof: We show this result by contradiction. Suppose there exist optimal sublot sizes, sl_1, $sl_2, \ldots sl_L$, in which sublot *j* is non-critical. We designate this set of sublot sizes by *S*, and its makespan by $M^*(S)$. Then, by the definition of a critical sublot, we have

$$M^*(S) - \sum_{k=1}^{j}\sum_{i=1}^{I} sl_{ik}\rho_{i1} + \sum_{k=j}^{L}\sum_{i=1}^{I} sl_{ik}\rho_{i2} = \Delta \text{, for some } \Delta > 0 .$$

Construct another set of sublot sizes sl_1', $sl_2', \ldots$ sl_L', designated by S', as follows:

$$sl_j' = \sum_{i=1}^{I}(sl_{ij}(1-\theta) + U_i * \theta),\ 0 \le \theta \le 1$$

$$sl_k' = \sum_{i=1}^{I} sl_{ik}(1-\theta) \text{, for all } k \ne j ,$$

We can find at least one critical sublot, and indicate it by γ'. Let $M^*(S')$ designate the makespan for S', then,

$$M^*(S') = \sum_{k=1}^{\gamma'}\sum_{i=1}^{I} sl'_{ik}\,\rho_{i1} + \sum_{k=\gamma'}^{L}\sum_{i=1}^{I} sl'_{ik}\,\rho_{i2}\ ,$$

If $\gamma' \neq j$,

$$M^*(S') = \sum_{k=1}^{\gamma'}\sum_{i=1}^{I} sl_{ik}\rho_{i1}(1-\theta) + \sum_{k=\gamma'}^{L}\sum_{i=1}^{I} sl_{ik}\rho_{i2}(1-\theta) + \sum_{i=1}^{I} U_i\theta\rho_{i2}\ \text{, if } \gamma' < j\text{,}$$

$$M^*(S') = \sum_{k=1}^{\gamma'}\sum_{i=1}^{I} sl_{ik}\rho_{i1}(1-\theta) + \sum_{k=\gamma'}^{L}\sum_{i=1}^{I} sl_{ik}\rho_{i2}(1-\theta) + \sum_{i=1}^{I} U_i\theta\rho_{i1}\ \text{, if } \gamma' > j\ .$$

In other words,

$$M^*(S') \le (1-\theta)[\sum_{k=1}^{\gamma'}\sum_{i=1}^{I} sl_{ik}\rho_{i1} + \sum_{k=\gamma'}^{L}\sum_{i=1}^{I} sl_{ik}\rho_{i2}] + \theta\max\{\sum_{i=1}^{I} U_i\rho_{i1}, \sum_{i=1}^{I} U_i\rho_{i2}\}\ .$$

Since $M^*(S) \ge \sum_{k=1}^{\gamma'}\sum_{i=1}^{I} sl_{ik}\rho_{i1} + \sum_{k=\gamma'}^{L}\sum_{i=1}^{I} sl_{ik}\rho_{i2}$ and $M^*(S) > \max\{\sum_{i=1}^{I} U_i\rho_{i1}, \sum_{i=1}^{I} U_i\rho_{i2}\}$, we have,

$$M^*(S') < (1-\theta) * M^*(S) + \theta * M^*(S) = M^*(S)\ ,$$

which contradicts the fact that $M^*(S)$ is optimal.

If $\gamma' = j$, then

$$M^*(S') = \sum_{k=1}^{\gamma'}\sum_{i=1}^{I} sl_{ik}\rho_{i1}(1-\theta) + \sum_{k=\gamma'}^{L}\sum_{i=1}^{I} sl_{ik}\rho_{i2}(1-\theta) + \sum_{i=1}^{I}(U_i\theta\rho_{i1} + U_i\theta\rho_{i2})$$

$$= (1-\theta)(M^*(S)-\Delta) + \theta\sum_{i=1}^{I}U_i(\rho_{i1}+\rho_{i2})$$

$$= (1-\theta)M^*(S) - (1-\theta)\Delta + \theta\sum_{i=1}^{I}U_i(\rho_{i1}+\rho_{i2}) + \theta * M^*(S) - \theta * M^*(S)$$

$$< (1-\theta)M^*(S) + \theta * M^*(S) < M^*(S),$$

for $0 < \theta < \dfrac{\Delta}{\sum_{i=1}^{I}U_i(\rho_{i1}+\rho_{i2}) + \Delta - M^*(S)} \le 1$, which contradicts the fact that $M^*(S)$ is optimal. Thus, our supposition is wrong, and sublot *j* must be critical. ◘

Corollary 4.1: The processing times of the sublots follow the compact block structure.

Proof: Since all the sublots are critical, we must have $\sum_{i=1}^{I} sl_{i,l+1}\rho_{i1} = \sum_{i=1}^{I} s_{il}\rho_{i2}$, for *l*=1,2, …*L*-1. ◘

Next, we identify some properties to help determine an optimal solution.

Proposition 4.7: The processing time for the first sublot on machine 1 and the last sublot on machine 2, i.e., T_{11} and T_{L2}, are unique.

Proof: Since the optimal solution follows a compact block structure, we have

$$C_{\max} = \sum_{l=1}^{L}\ \sum_{i=1}^{I} sl_{il}\rho_{i1} + \sum_{i=1}^{I} sl_{iL}\rho_{i2} = \sum_{i=1}^{I}U_i\rho_{i1} + T_{L2}, \text{ or}$$

$$C_{\max} = \sum_{i=1}^{I} sl_{i1}\rho_{i1} + \sum_{l=1}^{L}\ \sum_{i=1}^{I} sl_{il}\rho_{i2} = \sum_{i=1}^{I}U_i\rho_{i2} + T_{11}.$$

The optimal $C_{\max}$ value is unique, and $\sum_{i=1}^{I}U_i\rho_{i1}$ and $\sum_{i=1}^{I}U_i\rho_{i2}$ are constants. Thus T_{11} and T_{L2} are also unique. ◘

Proposition 4.8: The processing times of the sublots on machines 1 and 2, besides those of T_{11} and T_{L2}, are geometric in nature (see Figure 4.7).

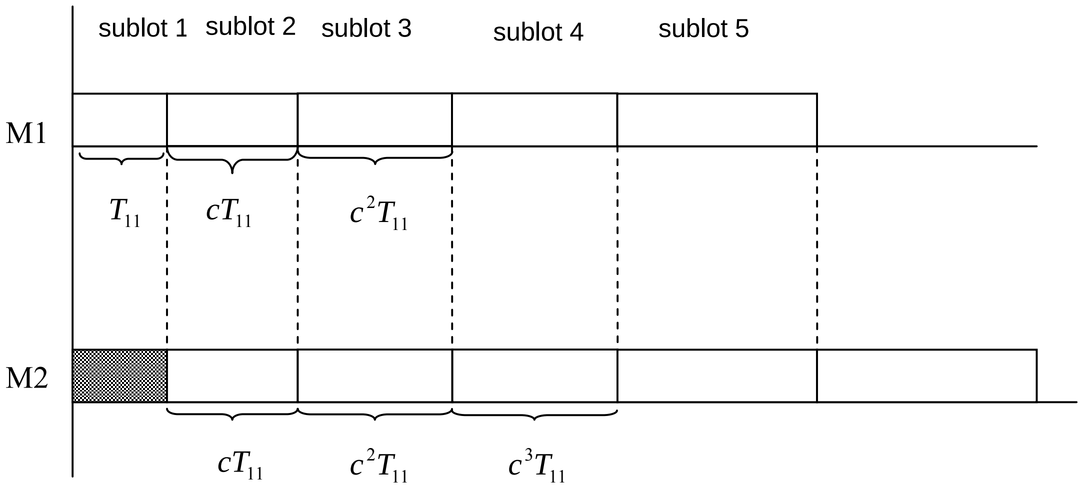

Figure 4.7: The geometric distribution of the processing times

Proof: Suppose we have an optimal solution with sublot sizes $\{sl_1, sl_2, ... sl_L\}$. We have $T_{11} = \sum_{i=1}^{I} sl_{i1}\rho_{i1}$, and $T_{12} = \sum_{i=1}^{I} sl_{i1}\rho_{i2} = \sum_{i=1}^{I} sl_{i1}(c\rho_{i1}) = c\sum_{i=1}^{I} sl_{i1}\rho_{i1} = cT_{11}$. By compact block structure, $T_{21} = T_{12} = cT_{11}$. Similarly, $T_{31} = c^2T_{11}$, …, $T_{L1} = c^{L-1}T_{11}$. Thus, the processing times of the sublots except for T_{11} and T_{L2} are geometric in nature. ◘

Since, $T_{11}(1 + c^1 + c^2 + .. + c^{L-1}) = \sum_{i=1}^{I} U_i\rho_{i1}$, we have, for l=1, 2, …, L,

$$T_{l1} = (\sum_{i=1}^{I} U_i\rho_{i1})\frac{c^{l-1}(c-1)}{c^L - 1},$$

and the optimal makespan,

$$C_{\max} = \sum_{i=1}^{I} U_i\rho_{i2} + (\sum_{i=1}^{I} U_i\rho_{i1})\frac{c-1}{c^L - 1} = (\sum_{i=1}^{I} U_i\rho_{i1})(c + \frac{c-1}{c^L - 1}) = (\sum_{i=1}^{I} U_i\rho_{i1})\frac{c^{L+1} - 1}{c^L - 1}.$$

It should be noted that the optimal solution exhibits a geometric pattern for sublot processing times, but not necessarily for sublot sizes. Actually, the sublot sizes can be consistently increasing, consistently decreasing, or a combination of the two. The only case for which the optimal sublot sizes would also follow a geometric distribution is when we find the optimal solution by individually applying the geometric-size patterns for each product type, i.e., we solve I two-machine lot streaming problems independently. In this way, $sl_{il} = U_i \frac{c^{i-1}(c-1)}{c^L - 1}$, for i=1, 2, …, I, and l=1, 2, …, L.

One procedure (**Sam-Uncap**) to construct an optimal solution is as follows.

<u>Algorithm Sam-Uncap:</u>

<u>Step 1</u>: Determine values of T_{11}, T_{21}, … T_{L1} by applying Proposition 4.8

<u>Step 2</u>: Arrange the products in non-decreasing order of ρ_{i1} values.

<u>Step 3</u>: Starting from l=1, pick sufficient amount of products from the sequence determined in step 2 to ensure that sum of the processing times for that amount is equal to the calculated value T_{l1}.

<u>Step 4</u>: Let $l = l+1$. if $l \le L-1$, repeat step 3; otherwise, go to step 5.

<u>Step 5</u>: Stop. The leftover amount will automatically be equivalent to T_{L1}.

Note that in the above procedure, if the products are arranged in arbitrary order, it will also lead to an optimal solution.

4.4.1.2 Capacitated optimal sublot sizes

In this section, we determine optimal sublot sizes in the presence of the capacity constraint of K for each sublot. With this capacity constraint, the properties for optimal solution will be different from that for the uncapacitated case. But some basic properties are still the same. For instance, there exists one optimal solution in which sublot sizes are consistent. The reverse property also holds.

Proposition 4.9: In an uncapacitated solution, if for two consecutive sublots *a* and *b* with *a* preceding *b*, $sl_a = \sum_{i=1}^{I} sl_{ia} > K$, $sl_b = \sum_{i=1}^{I} sl_{ib} \leq K$, $\sum_{i=1}^{I} sl_{ia}\rho_{i1} = T_{a1}$, $\sum_{i=1}^{I} sl_{ib}\rho_{i1} = T_{b1}$, $T_{b1} > T_{a1}$, and $\overline{\rho}_{a1}$ and $\overline{\rho}_{b1}$ are average processing times on machine 1 for sublots *a* and *b* respectively, with

$$\frac{T_{a1}}{\sum_{i=1}^{I} sl_{ia}} = \overline{\rho}_{a1} \leq \overline{\rho}_{b1} = \frac{T_{b1}}{\sum_{i=1}^{I} sl_{ib}}. \qquad \text{(Eq. 3-1)}$$

then, we can transform this solution to an equivalent solution consisting of sublots a' and b' such that $\sum_{i=1}^{I} sl_{ia'} = K$, $\sum_{i=1}^{I} sl_{ia'}\rho_{i1} = T_{a1}$, and $\sum_{i=1}^{I} sl_{ib}'\rho_{i1} = T_{b1}$. (Note that there is no restriction on the size of sl_b after the transformation).

Proof: First, we sequence the products in each sublot in non-decreasing value of ρ_{i1}. Then, we exchange a part of each sublot which has the processing time of γ on machine 1, $0 \leq \gamma \leq T_{a1}$ (see Figure 4.8). Correspondingly, on machine 2, the exchanged parts have processing time equivalent to $c\gamma$. For sublot a, γ starts from the smallest ρ_{i1} and it is the reverse for sublot b. By performing this exchange, the makespan remains unaffected, but sublot size sl_a decreases.

Also, since each sublot is exchanged with proportions of equivalent processing times, after the exchange, the processing time for each sublot is not affected. We first show that after the exchange, the sublot size $sl_a' = sl_a'(\gamma)$ is a continuous monotonic decreasing function of γ.

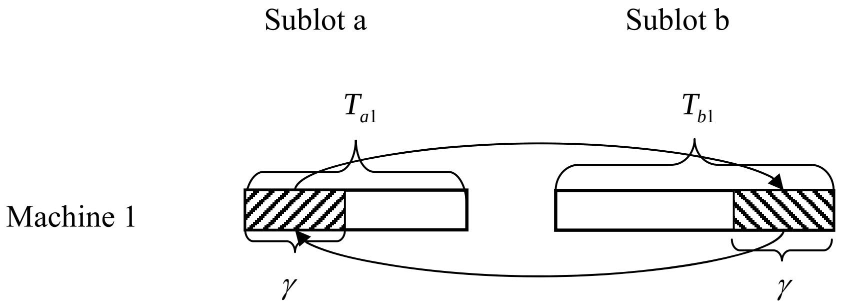

Figure 4.8: Exchange of parts from sublots *a* and *b* with processing time of γ

(1) Let $sl_{a-\gamma}$ and $sl_{b-\gamma}$ indicate the number of products forming the part corresponding to γ in sublots a and b respectively, and $\overline{\rho}_{a-\gamma,1}$ and $\overline{\rho}_{b-\gamma,1}$ indicate their average processing times on machine 1. By (Eq. 3-1) and the fact that the products in each sublot are arranged in non-decreasing order of ρ_{i1}, we have

$$\overline{\rho}_{a-\gamma,1} \le \overline{\rho}_{a1} \le \overline{\rho}_{b1} \le \overline{\rho}_{b-\gamma,1}.$$

Thus, $sl_{a-\gamma} = \frac{\gamma}{\overline{\rho}_{a-\gamma,1}} > \frac{\gamma}{\overline{\rho}_{b-\gamma,1}} = sl_{b-\gamma}$. So, $sl_a' = sl_a - sl_{a-\gamma} + sl_{b-\gamma} < sl_a$. Therefore, $sl_a'(\gamma)$ is a monotonic decreasing function of γ.

(2) Suppose, we increase γ by ε_1, $\varepsilon_1 \to 0$. Recall that ε_1 is contributed by products from sublots a and b. We indicate them by μ and θ respectively. Now, the sublot size $sl_a'' = sl_a - (\frac{\gamma}{\overline{\rho}_{a-\gamma,1}} + \frac{\varepsilon}{\rho_{\mu 1}}) + (\frac{\gamma}{\overline{\rho}_{b-\gamma,1}} + \frac{\varepsilon}{\rho_{\theta 1}}) = sl_a' + \varepsilon \frac{\rho_{\mu 1} - \rho_{\theta 1}}{\rho_{\mu 1}\rho_{\theta 1}}$. In other words, $sl_a'' - sl_a' \le \varepsilon \left| \frac{\rho_{\mu 1} - \rho_{\theta 1}}{\rho_{\mu 1}\rho_{\theta 1}} \right|$. Let $\varepsilon = \frac{\varepsilon'}{\left| \frac{\rho_{\mu 1} - \rho_{\theta 1}}{\rho_{\mu 1}\rho_{\theta 1}} \right|} > 0$, $\varepsilon' \to 0$, and we have $sl_a'' - sl_a' \le \varepsilon'$.

This implies $\lim_{\gamma+\varepsilon \to \gamma} sl_a''(\gamma) = sl_a'(\gamma)$. Thus, $sl_a'(\gamma)$ is a continuous function of γ.

From above, as γ increases from 0 to T_{a1}, sl_a decreases to sl_a'. Since $T_{a1} \le T_{b1}$, if $\gamma = T_{a1}$, then $sl_a^{'} \le sl_b \le K$. Hence, we can determine an appropriate value of γ for which $sl_a' = K$.

Similarly, sl'_b is also a continuous, monotonic increasing function of γ. ◘

Proposition 4.10 There exists at least one uncapacitated optimal solution which is also a capacitated optimal solution if and only if $\sum_{l=k}^{L} sl_l \le (L-l+1)K$ and $\sum_{l=k}^{L}\sum_{i=1}^{I} sl_{il}\rho_{i1} = \sum_{l=k}^{L} T_{l1}$, $\forall k = L, L-1, ..., 1$ with products appearing in the solution in non-decreasing order of ρ_{i1} values.

Proof: First, consider the forward part of the proposition, that is, if there exists a capacitated optimal solution, then $\sum_{l=k}^{L} sl_l \le (L-l+1)K$ for $\sum_{l=k}^{L}\sum_{i=1}^{I} sl_{il}\rho_{i1} = \sum_{l=k}^{L} T_{l1}$, $\forall k = L, L-1, ...1$, with products appearing in the solution in non-decreasing order of ρ_{i1} values.

We prove this by contradiction. Suppose in an capacitated optimal solution, there exists a $k = l^*$ such that $\sum_{l=l^*}^{L} sl_l > (L-l+1)K$ for $\sum_{l=l^*}^{L}\sum_{i=1}^{I} sl_{il}\rho_{i1} = \sum_{l=l^*}^{L} T_{l1}$. Since the products appear in the non-decreasing order of ρ_{i1}, there does not exist another solution $\sum_{l=l^*}^{L} sl_l^{'} = (L-l+1)K$ for $\sum_{l=k}^{L}\sum_{i=1}^{I} sl_{il}^{'}\rho_{i1} = \sum_{l=k}^{L} T_{l1}$. This means that there does not exist an optimal solution which does not violate the capacity constraint. Therefore, our assumption is incorrect.

To prove the second part, we we start from $k = L$. If $s_L \le K$ for $\sum_{i=1}^{I} sl_{iL}\rho_{i1} = T_{L1}$, then we find a sublot *L*, which does not violate capacity constraint. For $k = L-1$, if

$\sum_{l=L-1}^{L} sl_l \leq (L-l+1)K$ for $\sum_{l=L-1}^{L}\sum_{i=1}^{I} sl_{il}\rho_{i1} = \sum_{l=k}^{L} T_{l1}$, there are two possible scenarios. If $s_{L-1} \leq K$, then we have a sublot *L*-1 feasible to the constraint; otherwise, based on Proposition 4.9, we can always find a solution with $sl'_{L-1} = K$. Since $sl'_L + sl'_{L-1} \leq 2K$, $sl_L' \leq K$, both sublots *L*-1 and *L* will be feasible. Continuing this argument for other sublots, when we reach at *k*=1, we find feasible sublots 1, 2, …, *L*. In this way, we can find an uncapacitated optimal solution which is also feasible to capacity constraint. ◘

Based on Proposition 4.10, we can develop a procedure for checking whether there exists an uncapacitated optimal solution, which is also feasible for the capacitated case. We indicate this algorithm of checking feasibility as **CF**, and it is presented next.

Algorithm CF:

Step 1: Apply Sam-Uncap algorithm to solve the uncapacitated problem

Step 2: Let *k*=*L*.

Step 3: Apply Proposition 4.10, if $\sum_{l=k}^{L} sl_l \leq (L-l+1)K$ and $\sum_{l=k}^{L}\sum_{i=1}^{I} sl_{il}\rho_{i1} = \sum_{l=k}^{L} T_{l1}$ is violated, go to step 6; otherwise, go to step 4

Step 4: Let *k*=*k*-1. If k>0, repeat step 3; otherwise, go to step 5.

Step 5: Stop. This solution is a capacitated optimal solution.

Step6: Stop. This solution is not a capacitated optimal solution.

Proposition 4.11 If in the solution obtained using Algorithm **CF,** sublot *L* is infeasible (i.e., $sl_L > K$), there exists a capacitated optimal solution, in which, $sl_L = K$, and $SL_L \subset \{K \text{ wafers out of } U \text{ with the largest processing time on machine 1}\}$.

Proof: Suppose there does not exist a capacitated optimal solution in which the *L*th sublot consists of *K* wafers having the largest ρ_{i1} values. There are the following two possible cases for sublot *L*:

(1) $sl_L < K$.

We can move some products from the remaining L-1 sublots and make $sl_L = K$. Since the remaining L-1 sublots follow a compact block structure for an optimal solution, this does not result in idling of machine 2 but reduces T_{11} and the makespan, which is $C_{\max} = T_{11} + \sum_{i=1}^{I} U_i \rho_{i1}$. Thus, our assumption is incorrect.

(2) Sublot L does not have the longest ρ_{i1} values.

We can substitute sublot L with K products with the largest ρ_{i1} values. As above, this will not result in the idling of machine 2 but will reduce T_{11}. Thus, our assumption is incorrect. ◘

Proposition 4.12 If (i) the product types are arranged in the non-decreasing order of ρ_{i1}, and (ii) $\sum_{l=k}^{L} sl_l \le (L-l+1)K$ for $\sum_{l=k}^{L}\sum_{i=1}^{I} sl_{il}\rho_{i1} = \sum_{l=k}^{L} T_{l1}$, k=L, L-1, ……, l^*+1, while for $k = l^*$, $\sum_{l=l^*}^{L} sl_l > (L-l+1)K$ for $\sum_{l=l^*}^{L}\sum_{i=1}^{I} sl_{il}\rho_{i1} = \sum_{l=l^*}^{L} T_{l1}$, then there exists a capacitated optimal solution in which $sl_l = K$ and,

$$\{\bigcup_{l=l^*}^{L} SL_l\} \subset \{(L-l^*+1)K \text{ wafers with the largest processing time on machine 1}\}.$$

Proof: By definition of l^*, we have $sl_l \le K$ for $\sum_{i=1}^{I} sl_{il}\rho_{i1} = T_{l1}$, $\forall l = l^*+1, l^*+2, \ldots L$, and $sl_{l^*} > K$ for $\sum_{i=1}^{I} sl_{il^*}\rho_{i1} = T_{l^*1}$. Suppose there does not exist a capacitated optimal solution S in which $sl_l = K$ and

$$\{\bigcup_{l=l^*}^{L} SL_l\} \subset \{(L-l^*+1)K \text{ wafers with the largest processing time on machine 1}\}.$$

There are two possible cases:

(i) $\exists l \geq l^*$ such that $sl'_l < K$

Construct another solution S' by keeping the last $(L - l^* + 1)K$ products for sublots l^* to L. Note that, in the process, $(L - l^* + 1)K - \sum_{l=l^*+1}^{L} sl_l$ products from sublot l^* will be kept in sublot l^* for solution S' and the leftover products $\psi = sl_{l^*} - [(L - l^* + 1)K - \sum_{l=l^*+1}^{L} sl_l]$ from sublot l^* will be added to previous $l^* - 1$ sublots (see Figure 4.9). As $(L - l^* + 1)K - \sum_{l=l^*+1}^{L} sl_l \geq K$, by applying the results in Proposition 4.9, we can find a solution of $sl'_{l^*} = K$ and $sl'_{l^*+1} \geq K$. We can repeat the process of applying Proposition 4.9 until $sl'_{L-1} = K$ and $sl'_L = K$. In the process, there will be no idling created on machine 2. Moreover, $T_{11} = (U - \sum_{l=l^*}^{L} sl_l)\frac{c-1}{c^{L-l^*} - 1}$ is reduced in new schedule S', and so is the makespan, $C_{\max} = T_{11} + \sum_{i=1}^{I} U_i \rho_{i2}$. This implies that our assumption is incorrect.

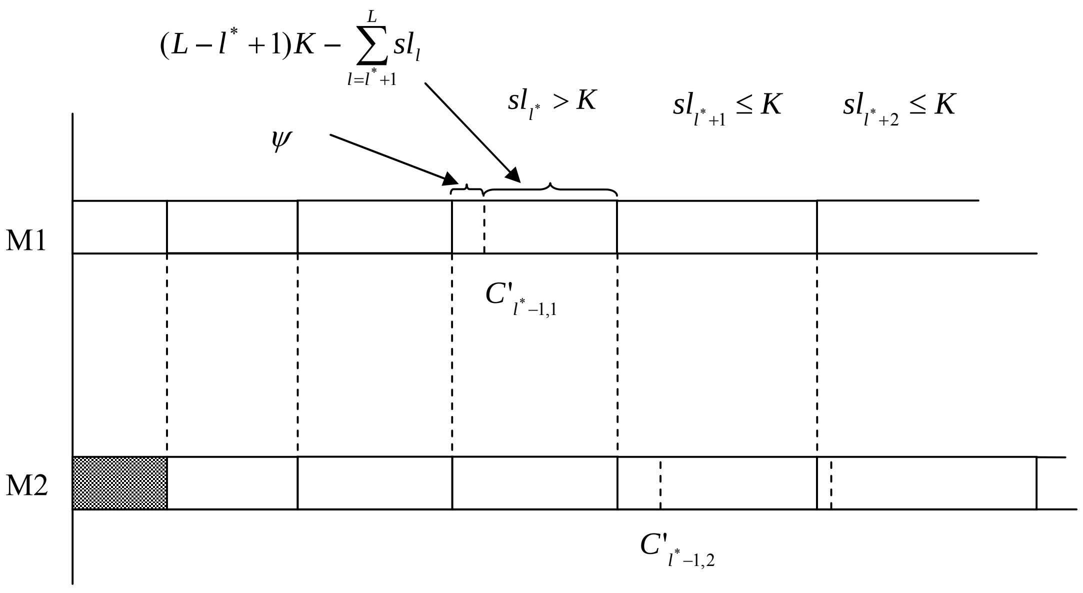

Figure 4.9: Construction of another solution *S'*

(ii) $\{SL_l\} \not\subset \{(L-l^*+1)K \text{ wafers out of } U \text{ with the largest processing time on machine } 1\}$. Substitute those sublots with $(L-l^*+1)K$ products with the largest ρ_{i1} values. As above, this will not result in idling on machine 2 but will reduce T_{11}. Thus, our assumption is incorrect. ◘

Combining the results of the propositions above, we can develop an algorithm, designated Algorithm **Sam-Cap** to find the capacitated optimal solution. This is presented next.

Algorithm Sam-Cap

Step 1: Solve the uncapacitated problem using Algorithm **Sam-Uncap** and find the value T_{l1}, for *l*=1, 2, .., *L*.

Step 2: Apply Algorithm **CF** to check whether there exists an optimal solution, which is capacity feasible. If yes, go to step 7; otherwise go to step 3.

Step 3: Let $l_{end} = L$

Step 4: Starting from $k = l_{end}$, check whether $\sum_{l=k}^{l_{end}} sl_l \le (l_{end} - l + 1)K$ for $\sum_{l=k}^{l_{end}} \sum_{i=1}^{I} sl_{il}\rho_{i1} = \sum_{l=k}^{l_{end}} T_{l1}$, until we find the first $k = \gamma$ such that $\sum_{l=\gamma}^{l_{end}} sl_l > (l_{end} - l + 1)K$ for $\sum_{l=\gamma}^{l_{end}} \sum_{i=1}^{I} sl_{il}\rho_{i1} = \sum_{l=\gamma}^{l_{end}} T_{l1}$.

Step 5: Pick $(l_{end} - \gamma + 1) * K$ number of products from the end of the sequence, and apply Proposition 4.9 to construct a solution with $sl_l = K$, for $l = \gamma, \gamma+1, \ldots l_{end}$.

Step 6: If $\gamma = 1$, go to step 7, otherwise, let $l_{end} = \gamma - 1$ go to step 4.

Step 7: Stop. We have identified a capacitated optimal solution.

4.4.2 Different ratios of processing times of products on machine 2 to those on machine 1 ($c_i \neq c_j$)

Note that we assume c_i to be such that either $c_i > 1$ or $c_i < 1$, $\forall i$ =1, 2, …, *I*. We do not consider the mix case in which $c_i > 1$ for some *i* and $c_i < 1$ for others. This is a reasonable assumption due to the physics of wafer fabrication. For example, processing time at a stepper is consistently longer than that at an etching tool for a product. However, as it can be seen later in this section, some results that we present can actually be applied to the case of mixed $c_i > 1$ and $c_i < 1$ values as well.

4.4.2.1 Uncapacitated optimal sublot sizes

Proposition 4.13: The compact-block-structure property is a necessary but not a sufficient condition for optimality of sublot sizes.

Proof: The proof of the necessity of compact block structure for optimal sublot sizes follows by Proposition 4.6 and Corollary 4.1.

We show by a counter example that the compact-block-structure property is not sufficient for optimality of sublot sizes. We use the following procedure to construct such an example.

Step 1: Apply Proposition 4.2 to product type *i* individually, *i*=1, 2, …, *I*. This gives sublot sizes sl_{il}, *i*=1, …, I .

Step 2: Combine the sublots of different product types in each sublot index *l*, i.e., $sl_l = \sum_{i=1}^{I} sl_{il}$, for *l*=1, 2, …, *L*.

The resulting solution follows the compact block structure, however, it need not be optimal.

Example 4-1:

The data for this example is presented in Table 4.1.

Table 4.1: Data for Example 4-1

Lot index		1	2	3	4	5	6	7	8
Lot size (s_i)		1	2	8	7	9	5	1	10
Processing time (ρ_{ij})	ρ_{i1}	2.1	2.5	3.1	3.5	1.4	2.7	3.8	2.4
	ρ_{i2}	4.6	3.9	6.4	5.8	2.9	3.1	6.8	5.3

Table4.2: Schedule for Example 4-1 obtained by applying the above procedure

Completion times (C_{lj})	Sublot 1	Sublot 2	Sublot 3	Sublot 4	Sublot 5
Machine 1	5.83	15.23	31.18	59.54	111.9
Machine 2	15.23	3.18	59.54	111.9	211.438

The schedule obtained by applying the above procedure is shown in Table 4.2. Note that, the completion times of the sublots follow the compact block structure, and the makespan is 211.438. however, the optimal makespan is 208.349.

Proposition 4.14: There exists an optimal solution in which the product types appear in the non-increasing order of c_i values.

Proof: Suppose there exists an optimal solution *S*, with neighboring sublots *i* and *i*+1 containing product type *a* in sublot *i* and product type *b* in sublot *i*+1, such that $c_a < c_b$. Let T_{a1}, T_{b1}, and T_{a2}, T_{b2} represent the processing times of product types *a* and *b* on machines 1 and 2, respectively (see Figure 4.10).

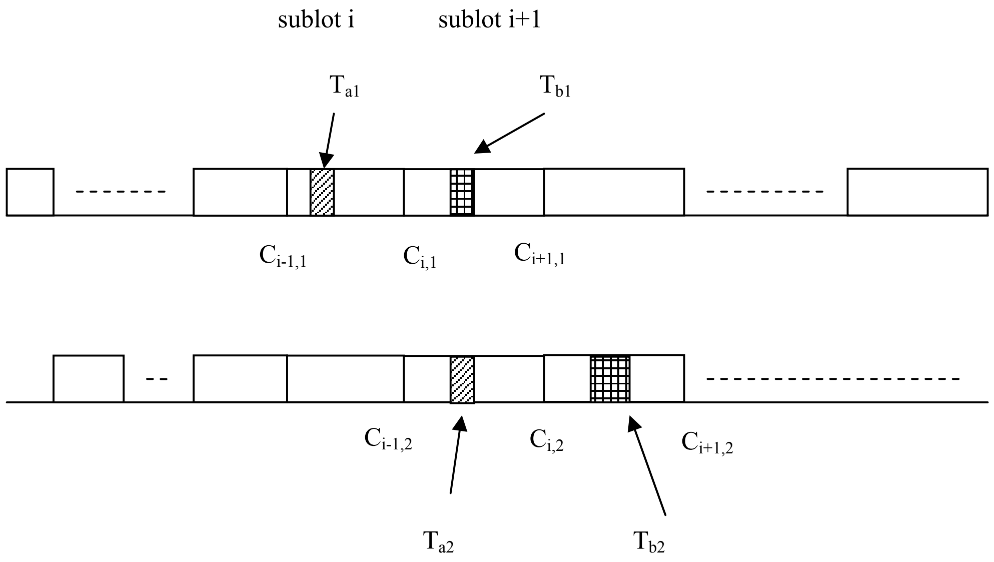

Figure 4.10: Exchange of products between sublots *a* and *b*

We have the following two cases:

(1) $T_{b1} \geq T_{a1}$

Let $T_{b1} = T_{a1} + \eta$. We can exchange product a with a part of product b that is equivalent to the processing time of T_{a1}.

After this exchange, $C_{i+1,1}$ remains unchanged, but

$C'_{i,2} = C_{i,2} - T_{a2} + T_{a1} * c_b = C_{i,2} - T_{a1} * c_a + T_{a1} * c_b = C_{i,2} + T_{a1} * (c_b - c_a) > C_{i,2}$.

Thus, there is no idle time created between sublot *i* and sublot *i*+1 as a result of this exchange, while $C_{i+1,2}$ remains the same. Thus, after the exchange, the solution remains optimal. Therefore, by putting a product with a higher c_i value earlier in the sequence does not worsen the solution.

(2) $T_{b1} \le T_{a1}$

Let $T_{a1} = T_{b1} + \eta$. We can exchange product b with a part of product *a* that is equivalent to the processing time of T_{b1}.

After the exchange, $C_{i+1,1}$ remains unchanged, but

$$\begin{aligned} C'_{i,2} &= C_{i,2} - T_{a2} + (T_{b1} * c_b + \eta * c_a) = C_{i,2} - (T_{a1} * c_a) + (T_{b1} * c_b + \eta * c_a) \\ &= C_{i,2} + \eta * (c_b - c_a) > C_{i,2}. \end{aligned}$$

Thus, there is no idle time created between sublot *i* and sublot *i*+1 as a result of the exchange, while $C_{i+1,2}$ remains the same. Thus, after the exchange, the solution remains optimal.

This exchange process can be repeated until the desired condition is achieved without worsening the solution. ◘

But, how to divide these product types ordered by non-increasing c_i values into *L* parts remains a key question.

First, note that, since the products are arranged in the non-increasing sequence of c_i values, once the first sublot sl_1 is chosen, there exists a unique schedule if the compact block structure is enforced, except for the last sublot. We can illustrate this as follows.

.

If we choose a small sl_1 and apply the compact block structure, the resulting schedule is shown in Figure 4.11.

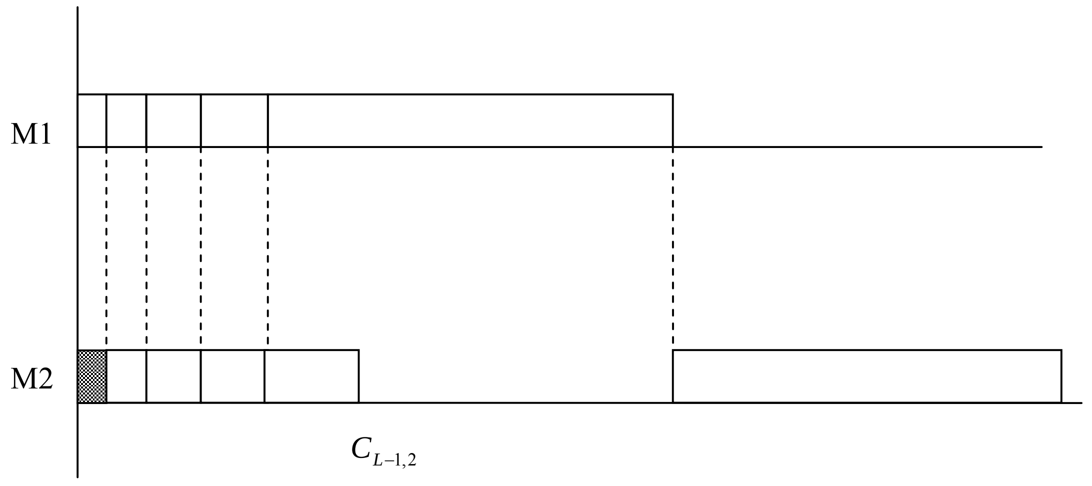

Figure 4.11: A compact schedule if the first sublot is small

On the other hand, if we choose a larger sl_1, and apply the compact block structure, the resulting schedule may be as shown in Figure 4.12.

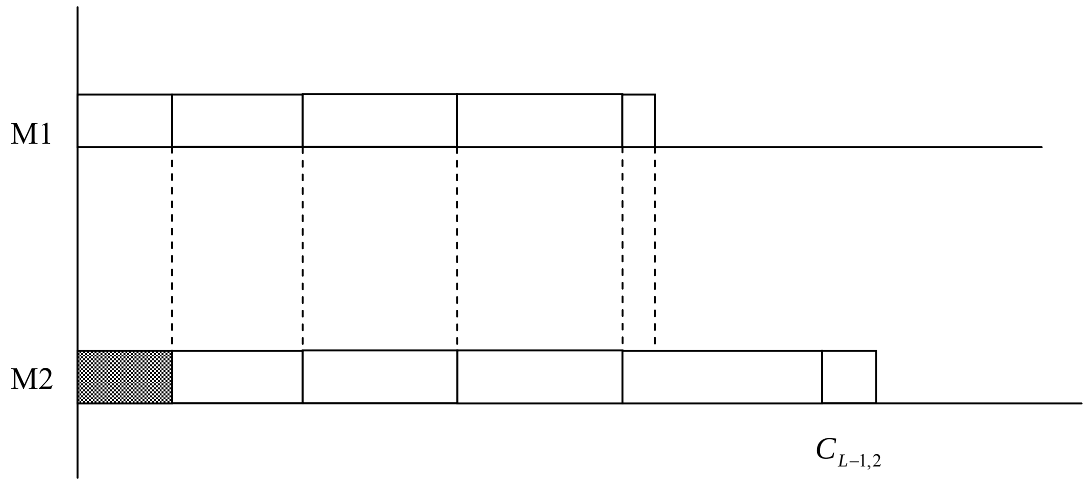

Figure 4.12: A compact schedule if the first sublot size is large

Therefore, depending on the value of sl_1 chosen, sublot L may not follow the compact block structure. However, if sl_1 is chosen correctly, then the last sublot will also follow the compact block structure. We show this next.

But, first, we show that $C_{L-1,2}$ is a continuous monotone increasing function of sl_1.

Proposition 4.15: $C_{L-1,2} = C_{L-1,2}(sl_1)$ is a continuous monotone increasing function of sl_1 if the compact block structure is enforced for intervening sublots.

Proof: (1) Since $c_i > 1$, the sublots are increasing in size starting from sublot 1, and as the size of first sublot sl_1 increases, sl_{L-1} will increase correspondingly, $C_{L-1,2} = \sum_{l=1}^{L-1} T_{l2} + T_{11}$ also increases. Thus, $C_{L-1,2} = C_{L-1,2}(sl_1)$ is a monotone, increasing function of sl_1.

(2) Suppose we increase sl_1 by ε_1, $\varepsilon_1 \to 0$ as shown in Figure 4.13. This ε_1 is contributed by certain product type, indicated by μ_1. Thus, T_{12} increases by $\Delta T_{12} = \varepsilon_1 * \rho_{\mu_1 2}$. Therefore, we have,

$T_{11}' = T_{11} + \varepsilon_1 \rho_{\mu_1 1}$,

$T_{12}' = T_{12} + \varepsilon_1 \rho_{\mu_1 2}$.

$$\begin{aligned} C_{12}' &= T_{11}' + T_{12}' = (T_{11} + T_{12}) + \varepsilon_1(\rho_{\mu_1 1} + \rho_{\mu_1 2}) \\ &= C_{12} + \varepsilon_1 \rho_{\mu_1 1} + \varepsilon_1 \rho_{\mu_1 2}, \end{aligned}$$

$C_{21}' = C_{12}' = T_{11} + T_{21} + \varepsilon_2 \rho_{\mu_2 1}$, where $\varepsilon_2 = \dfrac{\varepsilon_1(\rho_{\mu_1 1} + \rho_{\mu_1 2})}{\rho_{\mu_2 1}}$.

Similarly, we have,

$$\begin{aligned} C_{22}' &= T_{11}' + T_{12} + T_{22} + \varepsilon_2 \rho_{\mu_2 2} \\ &= C_{22} + \varepsilon_1 \rho_{\mu_1 1} + \varepsilon_2 \rho_{\mu_2 2}, \end{aligned}$$

$C_{31}' = C_{22}' = T_{11} + T_{21} + T_{31} + \varepsilon_3 \rho_{\mu_3 1}$, where $\varepsilon_3 = \dfrac{\varepsilon_1 \rho_{\mu_1 1} + \varepsilon_2 \rho_{\mu_2 2}}{\rho_{\mu_3 1}}$.

and

$$\begin{aligned} C_{32}' &= T_{11}' + T_{12} + T_{22} + T_{32} + \varepsilon_3 \rho_{\mu_3 2} \\ &= C_{32} + \varepsilon_1 \rho_{\mu_1 1} + \varepsilon_3 \rho_{\mu_3 2}. \end{aligned}$$

$C_{41}' = T_{11} + T_{21} + T_{31} + T_{41} + \varepsilon_4 \rho_{\mu_4 1}$, where $\varepsilon_4 = \dfrac{\varepsilon_1 \rho_{\mu_1 1} + \varepsilon_3 \rho_{\mu_3 2}}{\rho_{\mu_4 1}}$.

Next, we want to use induction to show $\varepsilon_l = \dfrac{\varepsilon_1 \rho_{\mu_1 1} + \varepsilon_{l-1} \rho_{\mu_{l-1} 2}}{\rho_{\mu_l 1}}$, for l=1, 2, …, L-1.

Suppose when l=k, $\varepsilon_k = \dfrac{\varepsilon_1 \rho_{\mu_1 1} + \varepsilon_{k-1} \rho_{\mu_{k-1} 2}}{\rho_{\mu_k 1}}$. We have,

$C_{k,2}' = T_{11}' + T_{12} + T_{22} + T_{32} + \ldots + T_{k,2} + \varepsilon_k \rho_{\mu_k 2}$, and

$C_{k+1,1}' = T_{11} + T_{21} + T_{31} + \ldots + T_{k+1,1} + \varepsilon_{k+1} \rho_{\mu_{k+1} 1}$, where $\varepsilon_{k+1} = \dfrac{\varepsilon_1 \rho_{\mu_1 1} + \varepsilon_k \rho_{\mu_k 2}}{\rho_{\mu_{k+1} 1}}$.

Hence, we have $\varepsilon_{L-1} = \dfrac{\varepsilon_1 \rho_{\mu_1 1} + \varepsilon_{L-2} \rho_{\mu_{L-2} 2}}{\rho_{\mu_{L-1} 1}}$.

Based on the above result, we can also write a closed-form representation containing only ε_1 for ε_l, l=2, 3, …, L-1, as follows,

$$\varepsilon_{L-1} = \frac{\varepsilon_1 \rho_{\mu_1 1}}{\rho_{\mu_{L-1} 1}} (1 + c_{L-2} + c_{L-2}c_{L-3} + c_{L-2}c_{L-3}c_{L-4} + \ldots + c_{L-2}c_{L-3} \ldots . c_2 c_1),$$

$$C_{L-1,2}' = C_{L-1,2} + \varepsilon_1 \rho_{\mu_1 2} + \varepsilon_{L-1} \rho_{\mu_{L-1} 2} = C_{L-1,2} + \varepsilon_1 \rho_{\mu_1 1}(c_1 + c_{L-1} + c_{L-1}c_{L-2} + c_{L-1}c_{L-2}c_{L-3}$$
$$+ \ldots .. + c_{L-1}c_{L-2} \ldots c_1).$$

Let $c_{\max} = \max\{c_1, c_2, \ldots, c_L\}$, we have $C_{L-1,2}' - C_{L-1,2} \le \varepsilon_1 (\rho_{\mu_1 1}(c_{\max} + \dfrac{c_{\max}^L - 1}{c_{\max} - 1}))$.

Let $\varepsilon_1 = \dfrac{\varepsilon_1'}{\rho_{\mu_1 1}(c_{\max} + \dfrac{c_{\max}^L - 1}{c_{\max} - 1})} > 0$. As $\varepsilon_1' \to 0$, we have $C_{L-1,2}' - C_{L-1,2} \le \varepsilon_1'$. This implies that $\lim_{sl_1' \to sl_1} C_{L-1,2}(sl_1') = C_{L-1,2}(sl_1)$. Thus, $C_{L-1,2}(sl_1)$ is a continuous function of sl_1. ◘

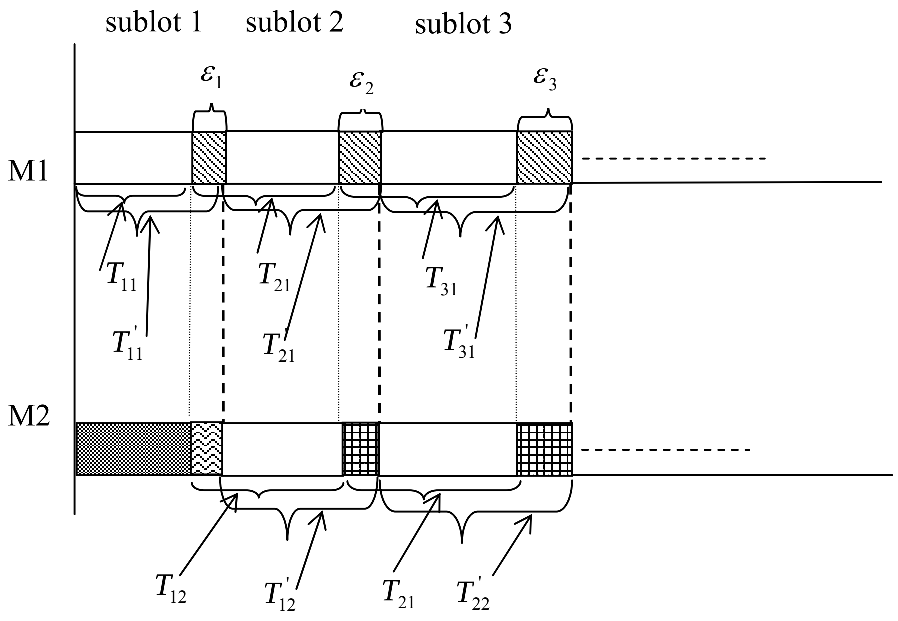

Figure 4.13: The change in the compact block structure when sl_1 increases

Proposition 4.16: If the product types are arranged in the non-increasing order of c_i values, all the sublots are critical in an optimal solution.

Proof: Due to the non-increasing order of c_i values, we cannot simply employ the method that was used in Proposition 4.6 to prove this proposition. For instance, if $sl_{ij}' = sl_{ij}(1-\theta)+U_i\theta$ and $sl_{ij}=0$, the solution, sl_{ij}' will not follow the desired non-increasing sequence of c_i values.

Starting from the first sublot, let sublot *j* be the first sublot which is not critical. We have the following two cases:

(1) $C_{j-1,2} > C_{j1}$

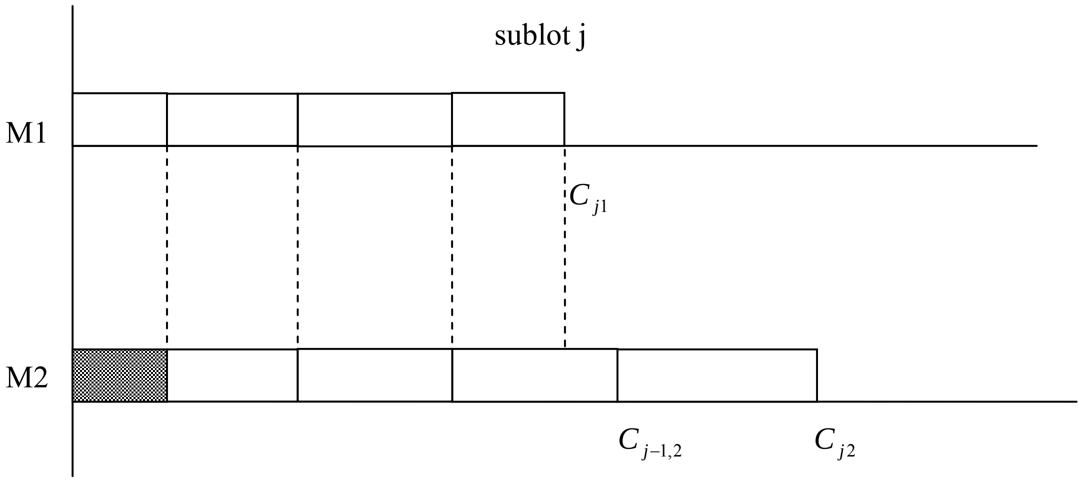

Figure 4.14: The schedule when $C_{j-1,2} > C_{j1}$

For this case, the completion time for the first *j* sublots is $C_{j2} = T_{11} + \sum_{i=1}^{j} T_{i2}$ (see Figure 4.14). In Proposition 4.15, we have shown $C_{j-1,2}(sl_1)$ to be a continuous, monotone increasing function of sl_1 if compact block structure is followed for the intervening sublots, which is the case here. We can reduce sl_1 and construct a compact block structure for sublots 1, 2, …, *j*. As a result, we have new completion time of sublot *j* on machine 2 $C_{j2}' < C_{j2}$.

(2) $C_{j-1,2} < C_{j1}$

For this case, the completion time for the first *j* sublots on machine 2, $C_{j2} = T_{j2} + \sum_{i=1}^{j} T_{i1}$ (see Figure 4.15). As before, by increasing sl_1, we can construct a compact block structure for sublots 1, 2, …, *j*. We have new completion time of sublot *j* on machine 2, $C_{j2}' = T_{j2}' + \sum_{i=1}^{j} T_{i1}$. Note that, as a result of this adjustment, the size of the last sublot, sl_j, decreases for a given lot size since the sizes of the other sublots increase. Consequently, we have $sl_j' < sl_j$ and $C_{j2}' < C_{j2}$.

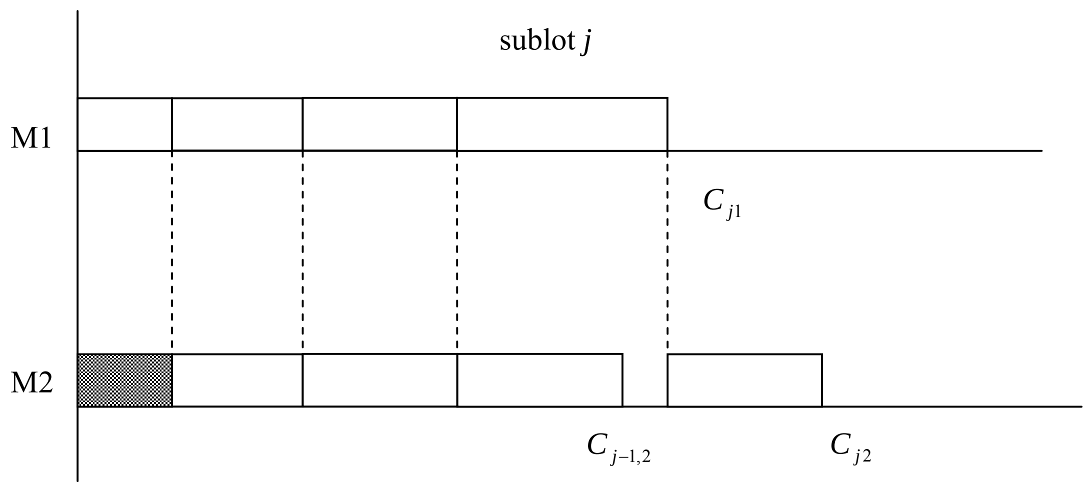

Figure 4.15: The schedule when $C_{j-1,2} < C_{j1}$

If $j = L$, we stop, and have proved the result. Otherwise, we compare $C_{j+1,1}$ and C'_{j2}. There are three possible cases: (1) $C_{j+1,1} > C_{j2}'$. In this case, we can, once again, construct the compact block structure for the first *j*+1 sublots as described above, and obtain $C_{j+1,2}' < C_{j+1,2}$. (2) $C_{j+1,1} = C_{j2}'$. There will be idling on machine 2 between the completion of sublot *j*, $C_{j2}^{'}$ and the beginning of sublot *j*+1 because $C_{j2}^{'} < C_{j2}$ and C_{j2} is the starting time of sublot *j*+1 on machine 2. We can left-shift sublot *j*+1 on machine 2 by $C_{j2} - C_{j2}'$ and obtain $C_{j+1,2}' < C_{j+1,2}$. This way, we have the compact block structure for the first *j*+1 sublots. (3) $C_{j+1,1} < C_{j,2}'$. Once again, as described above, we can construct the compact block structure for the first *j*+1 sublots, and obtain $C_{j+1,2}' < C_{j+1,2}$. We can repeat this process until *j*=*L*, and finally, obtain $C_{\max}' = C_{L2}' < C_{L2} = C_{\max}$. The resulting solution follows the compact block structure. ◘

Note that the processing times of sublots is increasing since $c_i > 1$. As for the sublot sizes, it really depends on the value of c_i as well as ρ_{i1} and ρ_{i2}, *i*=1, 2, …, *I*. Let $\overline{c_l}$, $\overline{\rho_{l1}}$ and $\overline{\rho_{l2}}$ indicate the average ratio, and average processing times on machine 1 and machine 2, respectively, for sublot *l*. They can be calculated as follows: $\overline{c_l} = \frac{\sum_{i=1}^{I} sl_{il}\rho_{i2}}{\sum_{i=1}^{I} sl_{il}\rho_{i1}} = \frac{\overline{\rho_{l2}}}{\overline{\rho_{l1}}}$,

$\overline{\rho_{l1}} = \frac{\sum_{i=1}^{I} sl_{il}\rho_{i1}}{sl_l}$, and $\overline{\rho_{l2}} = \frac{\sum_{i=1}^{I} sl_{il}\rho_{i2}}{sl_l}$. For compact block structure, we have $\overline{\rho_{l2}} sl_l = \overline{\rho_{l+1,1}} sl_{l+1}$. Therefore, if $\overline{c_l} > \frac{\overline{\rho_{l+1,1}}}{\overline{\rho_{l1}}}$, the sublot sizes will be increasing, i.e., $sl_{l+1} > sl_l$, *l*=1, …, *L*, and vice versa. Consequently, it is possible to have fluctuating sublot sizes.

An upper bound on the processing time of the first sublot, T_{11-LB}, can be found by using the average of the processing times of all sublots on machine 1, $T_{11-UB} = \frac{\sum_{i=1}^{I} U_i \rho_{i1}}{L}$, since $T_{l1} \geq T_{11}$, *l*=2, 3, .., *L*.

Remark 4.1: An ω-optimal solution can be found by the interval bi-section search algorithm, where $\omega > 0$ and ω is a small value.

The one dimensional interval bi-section search algorithm (**Dif-Uncap**) can be described as follows.

Algorithm Dif-Uncap:

Step 1: Arrange the products in the non-increasing order of c_i values.

Step 2: Let $T_{11-LB} = \omega$ and calculate $T_{11-UB} = \frac{\sum_{i=1}^{I} U_i \rho_{i1}}{L}$.

Step 3: Pick the first sl_1 products such that $T_{11} = \frac{T_{11-UB} + T_{11-LB}}{2}$. These products form sublot 1.

Step 4: Determine the compact block solution of the first L-1 sublots. The left-over products form sublot L.

Step 5: Compare $C_{L-1,2}$ and C_{L1}. If $\left|C_{L-1,2} - C_{L1}\right| > \omega$, there are two cases: (i) $C_{L-1,2} > C_{L1}$, new upper bound $T'_{11-UB} = T_{11}$, and go to step 3; (ii) If $C_{L-1,2} < C_{L1}$, new lower bound $T'_{11-LB} = T_{11}$, and go to step 3. Otherwise, go to step 6.

Step 6: Optimal solution is obtained. Stop.

4.4.2.2 Optimal sublot sizes for the capacitated case

We mainly consider a special case for this problem in which the processing times on the first machine are the same for different product types, i.e., $\rho_{i1} = \rho_1$.

Proposition 4.18 If $\rho_{i1} = \rho_1$, then there exists an optimal solution in which the products appear in the non-increasing order of c_i.

Proof: We use the method employed for the proof of Proposition 4.14. Since $\rho_{i1} = \rho_1$, any exchange of products between neighborhood sublots will not change the sublot sizes. Therefore, Proposition 4.14 remains valid for the capacitated case as well. ◘

Proposition 4.19: If $\rho_{i1} = \rho_1$, and $c_i > 1$, then there exists an optimal capacitated solution in which the ending sublots are of size K and the remaining sublots follow the compact block structure. Furthermore, all products follow the non-increasing order of c_i values.

Proof: Since $c_i > 1$, we have $T_{11} < T_{12} = T_{21}$. Thus, the processing times of the sublots keep increasing and so do sublot sizes since $\rho_{i1} = \rho_1$. The ending sublots must be those violating capacity constraint.

Suppose this is not the optimal solution. Then, there are two possible cases.

(i) The remaining sublots do not follow the compact block structure

But, this will not be an optimal solution by Proposition 4.16.

(ii) There exists an l^*, l^* belonging to the ending sublot, such that $s_{l*} < K$.

The makespan for this case is $C_{\max} = T_{11} + \sum_{i=1}^{N} U_i \rho_{i2}$ We can transfer some items from the remaining sublots and make $s'_{l*} = K$. Then, makespan, $C'_{\max} = T_{11}' + \sum_{i=1}^{N} U_i \rho_{i2}$. Since for both cases, the remaining sulots are in compact block structure, $T_{11}' < T_{11}$, which leads to a contradiction. ◘

The algorithm to find capacitated optimal solution (**Dif-Cap**) can be described as follows.

<u>Algorithm Dif-Cap:</u>

<u>Step 1</u>: Arrange the products in the non-increasing order of c_i values.

<u>Step 2</u>: Let $l_{end} = L$.

<u>Step 3</u>: Apply **Dif-Uncap** algorithm

<u>Step 4</u>: Check capacity constraint. If there is no violation, then we have found an optimal solution, go to step 7. Otherwise, go to step 5.

<u>Step 5</u>: Find the largest sublot l^* such that $sl_{l^*} > K$, i.e., $l^* = \arg\max_{l, l \le l_{end}} \{sl_l > K\}$. Let $sl_l = K$ for l=l^*, $l^* + 1$, …, l_{end}.

Step 6: Update l_{end} with $l_{end} = l^* - 1$. If $l_{end} > 0$, go to step 3. Otherwise, go to step 7.

Step 7: Sop. We have an optimal solution.

4.5 Adjustment heuristics

Recall from Section 4.2.2 that the capacitated optimal schedules obtained from Algorithms **Sam-Cap** and **Dif-Cap** may not be feasible to MLCSP3 as wafers belonging to a lot may be split among different sublots (carriers). But, they can be considered as "target" solutions. Heuristic procedures are used to fine-tune the target solutions, and make them feasible to MLCSP3 while keeping minimum deviation from the target.

4.5.1 Lots with the same ratio of processing times ($c_i = c$)

As described in section 4.4.1, in the capacitated optimal solution, the processing time for each sublot is unique but sublots have high variety of products and sizes. So, the target should be the processing time for each sublot T_{l1} rather than the sublot sizes sl_l as in Laub et al. (2007). Since c_i is the same, once processing time at machine 1 is used as target, the processing time at machine 2 will be automatically "targeted".

We use the following four heuristic procedures.

(i) First Fit for T_{l1} with Increasing T_{l1} (Sam-FFI).

Step 1: Arrange lots in the decreasing order of processing times on machine 1, $s_i\rho_{i1}$, for *i*=1, 2, …, *N*.

Step 2: Arrange T_{l1} in the increasing order, for *l*=1, 2, …, *L*.

Step 3: Let $T_{l1}^{'}$ indicate the sum of the processing times of the lots on machine 1 assigned to carrier *l*, and $s_l^{'}$ indicate the sum of the sizes of the lots assigned to carrier *l*, for *l*=1, …, *L*.

Step 4: Start with i=1.

Step 5: Assign lot i to the first available carrier l such that $T_{l1} \geq T_{l1}^{'} + s_i * \rho_{i1}$ and $s_l^{'} + s_i \leq K$, go to step 7. If such a carrier l cannot be found, go to step 6.

Step 6: Assign lot i to carrier l^* with $l^* = \max_l \{T_{l1} - T_{l1}^{'}\}$.

Step 7: if i=L, go to step 8; otherwise, let i=i+1, and go to step 5.

Step 8: Stop.

(ii) First Fit for T_{l1} with Decreasing T_{l1} (Sam-FFD).

The same as (i) except that T_{l1} is arranged in decreasing order.

(iii) Best Fit for T_{l1} with Increasing T_{l1} (Sam-BFI).

The same as (i) except that the best fit policy is used, which is to assign a lot to the fullest possible carrier, i.e., lot i is assigned to carrier l^* with $l^* = \max_l \{T_{l1}^{'} + s_i \rho_{i1}\}$ such that $T_{l1} \geq T_{l1}^{'} + s_i * \rho_{i1}$ and $s_l^{'} + s_i \leq K$.

(iv) Best Fit for T_{l1} with Decreasing T_{l1} (Sam-BFD).

The same as (iii) except that T_{l1} is arranged in decreasing order.

4.5.2 Lots with different ratios of processing times ($c_i \neq c_j$)

In this case, consideration of processing time on machine 1 or that on machine 2 as a target is not enough. Due to different ratios, when one is targeted, the other one will not be automatically targeted. Thus, the target should be processing times for each sublot on both machine 1, T_{l1}, and machine 2, T_{l2}.

We use the following four heuristic procedures.

(i) First Fit for T_{l1} with Decreasing $s_i\rho_{i1}$ while considering T_{l2} (Dif-FF1).

Step 1: Arrange lots in the decreasing order of processing times on machine 1, $s_i\rho_{i1}$, for i=1, 2, …, N.

Step 2: Keep the original sequence of T_{l1}, for l=1, 2, …, L.

Step 3: Let $T_{l1}^{'}$, $T_{l2}^{'}$ indicate the sum of the processing times of the lots assigned to carrier l on machines 1 and 2, respectively, and $s_l^{'}$ indicate the sum of the sizes of the lots assigned to carrier l, for l=1, …, L.

Step 4: Start with i=1.

Step 5: Assign lot i to the first available carrier l such that $T_{l1} \ge T_{l1}^{'} + s_i * \rho_{i1}$, $T_{l2} \ge T_{l2}^{'} + s_i * \rho_{i2}$ and $s_l^{'} + s_i \le K$, go to step 7. If such a carrier l cannot be found, go to step 6.

Step 6: Assign lot i to carrier l^* with $l^* = \max_l \{(T_{l1} - T_{l1}^{'} - s_i\rho_{i1}) + (T_{l2} - T_{l2}^{'} - s_i\rho_{i2})\}$.

Step 7: if i=L, go to step 8; otherwise, let i=i+1, go to step 5.

Step 8: Stop.

(ii) First Fit for T_{l2} with Decreasing $s_i\rho_{i2}$ while considering T_{l1} (Dif-FF2).

The same as (i) except that T_{l1} is exchanged with T_{l2}.

(iii) Best Fit for T_{l1} with Decreasing $s_i\rho_{i1}$ while considering T_{l2} (Dif-BF1).

The same as (i) except that best fit policy is used.

(iv) Best Fit for T_{l2} with Decreasing $s_i\rho_{i2}$ while considering T_{l1} (Dif-BF2).

The same as (ii) except that best fit policy is used.

4.6 Numerical experimentation

In this section, we test the performance of the heuristic procedures (**Sam-FFI, Sam-FFD, Sam-BFI, Sam-BFD, Dif-FF1, Dif-FF2, Dif-BF1, Dif-BF2**) with the direct solution of the problem using the AMPL CPLEX Solver (version 10.1). The performance measure is the gap between the heuristic solutions and optimal solutions. The summary of the test data is contained in Table 4.3. We test 3 cases with 15, 20, and 25 lots. For each case, three *L* values are used, namely, small, medium and large, which result in schedules with high, medium and low densities. Thus, there are a total of $3 \times 3 = 9$ data sets. For each data set, we generate 20 problem instances with randomly generated lot sizes.

Table4.3: Data used in numerical experimentation for problem MLCSP3

Number of lots (*N*)		15, 20, 25
Number of carriers available (*L*)	High density	$\left\lceil \frac{N}{K/\bar{s}} \times 100\% \right\rceil$
	Medium density	$\max\{\left\lceil \frac{N}{K/\bar{s}} \times 100\% \right\rceil + 1, \left\lceil \frac{N}{K/\bar{s}} \times 140\% \right\rceil\}$
	Low density	$\max\{\left\lceil \frac{N}{K/\bar{s}} \times 100\% \right\rceil + 2, \left\lceil \frac{N}{K/\bar{s}} \times 180\% \right\rceil\}$
Lot size (s_i)		Uniform distribution [1,10] †1
Average lot size ($\bar{s}$)		5.5
Processing time per wafer ρ_{ij} (minutes)		Uniform distribution [0.6,1.5] †2
Ratio of processing times (c_i)		Uniform distribution [1,5] †3
Carrier capacity (*K*)		25 wafers

(*†1 see ITRS 2006 Factory Integration, †2 , †3 from real fab data)*

We coded the heuristic procedures using Excel VBA (version 2003). All numerical tests were done on a Dell computer with Pentium 4 processor (2.8GHz).

Table 4.4: The experimental results for the solution of MLCSP3 when the lots have the same ratio of processing times[†]

Data set	Number of Lots (N)	Number of Carriers (L)	LB	Sam-FFI	Sam-FFD	Sam-BFI	Sam-BFD
1	15	4	99.20%	101.20%	100.72%	100.72%	101.20%
2	15	5	99.11%	100.89%	101.23%	100.89%	101.20%
3	15	6	99.67%	100.33%	100.92%	101.87%	100.33%
4	20	5	98.73%	101.27%	100.16%	101.27%	100.16%
5	20	7	99.39%	100.61%	100.16%	100.61%	100.16%
6	20	8	99.55%	100.45%	101.23%	100.83%	100.72%
7	25	6	99.81%	100.19%	100.10%	100.19%	100.10%
8	25	8	99.63%	100.37%	100.16%	100.37%	100.16%
9	25	10	99.91%	100.09%	100.94%	101.03%	101.21%

(The base value is $C^*_{\max}$)

Table 4.5: The experimental results for the solution of MLCSP3 when the lots have different ratios of processing times[†]

Data set	Number of Lots (N)	Number of Carriers (L)	LB	Dif-FFI	Dif-FFD	Dif-BFI	Dif-BFD
1	15	4	97.58%	102.83%	100.96%	102.93%	100.66%
2	15	5	98.15%	102.17%	102.15%	101.17%	102.15%
3	15	6	99.10%	101.94%	100.56%	101.89%	100.56%
4	20	5	98.58%	102.33%	100.66%	102.69%	101.56%
5	20	7	97.15%	101.17%	102.15%	101.17%	102.15%
6	20	8	99.40%	102.90%	100.97%	101.93%	100.97%
7	25	6	99.66%	101.08%	101.36%	101.09%	101.08%
8	25	8	99.58%	101.33%	101.40%	102.45%	103.20%
9	25	10	99.59%	102.93%	100.92%	102.98%	101.41%

(The base value is $C^*_{\max}$)

The experimental results of heuristic procedures are presented in Tables 4.4 and 4.5 for the solution of MLCSP3 when the lots have the same ratio of processing times and different ratios of processing times, respectively. The lower bounds (LB) are the solutions obtained from algorithms **Sam-Cap** and **Dif-Cap**. Note that the gap between LB and optimal solution $C^*_{\max}$ is less than 2% for MLCSP3 when the lots have the same ratio of processing times and less than 3% for MLCSP3 when the lots have different ratios of processing times. Thus, the targets we use in our heuristic procedures are very good. It can be seen that all the eight heuristic procedures perform very well and the solutions obtained are consistently within 3% of the optimum for MLCSP3 when the lots have same ratio of processing times and 4% for MLCSP3 when the lots have different ratios of processing times. This is because our heuristics are based on First Fit and Best Fit algorithms for the bin packing problem. Those algorithms have been shown to generate solutions close to optimality (Coffman et al. (1996)). In particular, the first four procedures (**Sam-FFI, Sam-FFD, Sam-BFI and Sam-BFD**) generate solutions closer to optimum than the remaining four (**Dif-FF1, Dif-FF2, Dif-BF1 and Dif-BF2**) procedures. This follows because the latter target the processing times on both machine 1 and machine 2. In addition, note that, the performance of our heuristic procedures is robust in terms of problem size (number of lots N), carrier densities, and processing times. This follows by the fact that the algorithms for the bin packing problem are not affected by these factors.

For the number of lots greater than 25, the CPLEX Solver requires more than 3600 seconds to solve the problem. Instead, our heuristic procedures are able to generate solutions in seconds. Since the optimal solution $C^*_{\max}$ is not available for larger-size problems, another performance measure that we use is the gap between the LBs and the solutions generated by our heuristic procedures. Additional tests were run for larger problems, and the results are presented in Tables 4.6 and 4.7. It can be seen that the gaps between the LBs and solutions generated by our heuristic procedures are less than 4% and 6% for MLCSP3 when the lots have the same ratio of processing times and different ratios of processing times, respectively, which indicate that the solutions generated by the heuristic procedures are within 2% and 4%, respectively, for the cases with the same and different ratios of processing times if a gap of 2% still holds between the LBs and optimal solutions.

Table 4.6: The experimental results for the solution of MLCSP3 when the lots have the same ratio of processing times†

Data set	Number of Lots (N)	Number of Carriers (L)	Sam-FFI	Sam-FFD	Sam-BFI	Sam-BFD
1	25	6	101.39%	101.15%	101.59%	101.15%
2	25	8	101.46%	101.37%	101.35%	101.26%
3	25	10	101.04%	102.25%	102.93%	102.62%
4	30	8	102.28%	101.42%	101.27%	102.09%
5	30	11	102.59%	103.56%	101.69%	103.20%
6	30	13	101.33%	102.63%	102.83%	101.53%
7	50	14	103.62%	101.46%	102.57%	101.65%
8	50	17	101.64%	101.39%	101.81%	102.36%
9	50	22	101.55%	102.49%	102.49%	103.37%

(The base value is LB)

Table 4.7: The experimental results for the solution of MLCSP3 when the lots have different ratios of processing times†

Data set	Number of Lots (N)	Number of Carriers (L)	Dif -FFI	Dif-FFD	Dif -BFI	Dif-BFD
1	25	6	103.80%	102.11%	103.80%	102.11%
2	25	8	104.14%	105.15%	104.14%	105.15%
3	25	10	103.52%	101.58%	102.55%	103.58%
4	30	8	101.42%	101.71%	101.44%	101.42%
5	30	11	101.76%	101.83%	102.88%	103.24%
6	30	13	103.35%	101.33%	103.40%	102.83%
7	50	14	102.42%	102.22%	102.62%	101.91%
8	50	17	101.94%	102.54%	103.38%	105.28%
9	50	22	102.89%	104.38%	102.47%	104.51%

(The base value is LB)

4.7 Conclusions

In this chapter, we have addressed a two-machine flow shop, multiple-lots-per-carrier, single-wafer-processing-technology scheduling problem for the objective of minimizing the makespan (MLCSP3). We have first formulated this problem as an integer programming model. Due to the difficulty of solving this model directly, we consider a relaxation of the problem by permitting unlimited carrier capacity and allowing splitting of the lots in different carriers, and transform the original problem to a two-machine flow shop lot streaming problem. For the lot streaming problem with lots having the same ratio of processing times, we propose an algorithm (designated **Sam-Cap** algorithm) to find the optimal capacitated sublot sizes. For lots with different ratios of processing times, the lot streaming problem is more complex. We, first, develop a bi-section algorithm (called **Dif-Uncap**) to find the optimal uncapacitated solutions for the general case. Then, for the case of lots with the same processing times on machine 1, we propose an algorithm (called **Dif-Cap**) to find the optimal capacitated sublot sizes.

Since the optimal solutions obtained from the lot streaming problem may not be feasible to the MLCSP3, we develop heuristic procedures based on the heuristic algorithms for the bin packing problem, including four heuristic procedures **Sam-FFI, Sam-FFD, Sam-BFI**, and **Sam-BFD** for lots with the same ratio of processing times, and another four algorithms **Dif-FF1, Dif-FF2, Dif-BF1**, and **Dif-BF2** for lots with different ratios of processing times.

Our numerical tests indicate that the proposed heuristic procedures generate solutions that are close to optimum. The gap is less than 3% for **Sam-FFI, Sam-FFD, Sam-BFI**, and **Sam-BFD**, and less than 4% for **Dif-FF1, Dif-FF2, Dif-BF1**, and **Dif-BF2.** In addition, the heuristic procedures find solutions in few seconds while the CPLEX Solver cannot solve the problems with number of lots larger than 25 within the allowable time of 3600 seconds. For large-size problems, the performance measure that we use is the gap between the LB and the solution values generated by our heuristic procedures. The results indicate that this gap is within 5% when the lots have the same ratio of processing times and within 6% when the lots have different ratios of processing times.

Chapter 5: Minimization of Makespan for an Integrated AMHS and Lot Scheduling Problem (IMHLSP) with Infinite Vehicle Capacity

5.1 Introduction

The semiconductor manufacturing is being transformed from the processing of 200mm to 300 mm wafers. The dual promises of more chips per wafer and economies of scale have attracted the leading semiconductor manufacturers toward the development of the 300 mm fabs. In a 300 mm fab, wafers with diameters of 300 mm (12 inches) are carried in a 25-wafer box, called a front opening unified pod (FOUP) (see Figure 5.1). With a weight of 18 lbs, the FOUP is too heavy to be carried around the factory and handled manually on a repeated basis. Additionally, it will not be efficient to use manual carts in a high volume manufacturing environment. Thus, the implementation of an automated material handling system (AMHS) has never been so important in such an environment.

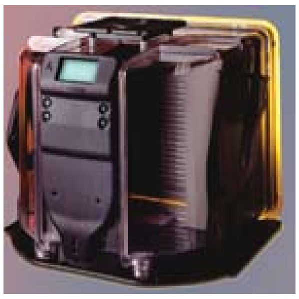

Figure5.1: A front opening unified pod (FOUP)

A typical AMHS consists of three components: tracks built under the ceiling, overhead vehicles (OHV), and stockers (see Figure 5.2). In general, there are two types of AMHSs that are used in a wafer fab. The first is the interbay system, which transports boxes of wafers between process bays. The other is the intrabay system, which transports boxes of wafers within a process bay. Unlike a 200 mm fab where, usually, only an interbay system is widely used, a 300 mm fab utilizes both interbay and intrabay systems. Figure 5.3 shows one simplified AMHS layout, which contains a single loop of the interbay system, eight intrabay systems, and 16 stockers. The tools shown in Figure 5.3 are categorized into 6 areas: diffusion, etching, implant, lithography, thin-film and inspection.

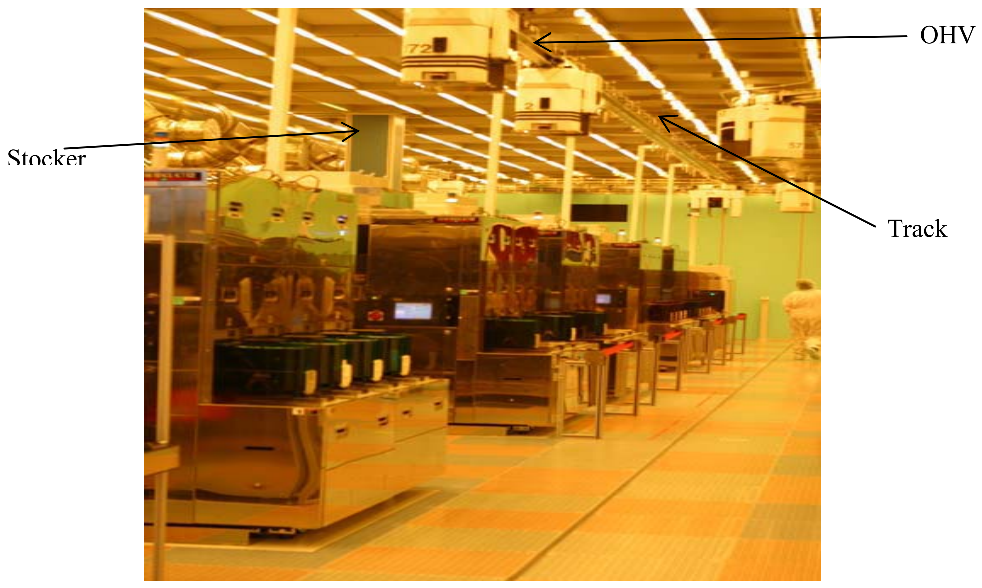

Figure5.2: A snapshot of AMHS in a 300mm fab

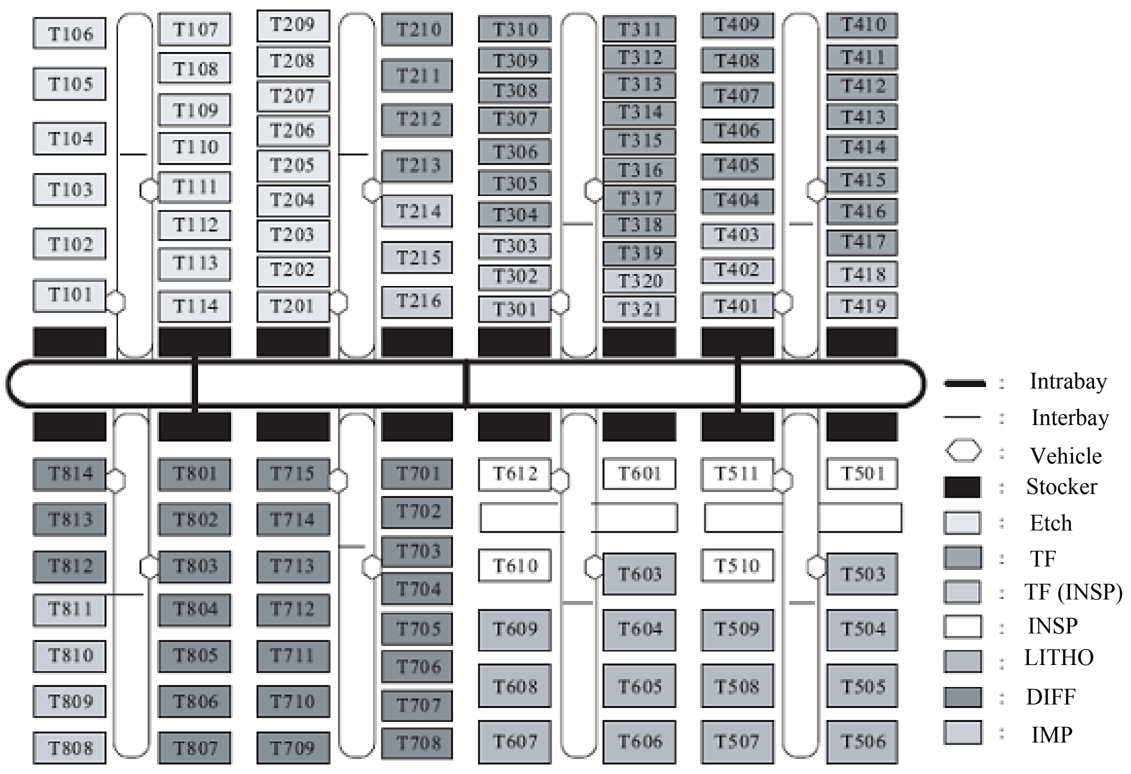

Figure 5.3: A simplified AMHS layout

5.1.1 Important control issues of an AMHS

Vehicles allocation and routing

In a 300 mm fab, overhead vehicles can be assigned to different parking places, or process bays during different shifts or periods. Also, due to the complexity of the AMHS layout, vehicles may follow different paths while delivering a lot from an origin to a destination.

Vehicles scheduling

One key requirement of an AMHS is to deliver the right lot at the right time to the right place.

Delivery time estimation

Delivery time is an important parameter for developing vehicle dispatching heuristics of the AMHS. Delivery time includes load/unload time, traveling time with empty load, loaded-traveling time, and waiting time due to traffic congestion. Besides, different routes will require different travel times.

Node balancing

Nodes are defined as the load/unload places in the AMHS network. One of the observed problems with the AMHS is that vehicles tend to cluster around one busy node at certain production times. This is a direct result of lot batching and frequent move-requests at one specific node. Hence, it is necessary to balance the flow of traffic at the busy nodes in the AMHS.

Deadlock prevention

Most tools in a 300 mm fab have very limited buffer size, usually 2 or 3 FOUPs. Deadlock happens when one lot finishes processing at the tool and requests to be transported, but the buffer space at the next tool is full.

Response to interruption

Both an AMHS and a 300 mm fab are complex systems. A breakdown of the AMHS, tool failure, scheduled maintenance, and traffic congestion are encountered inevitably in the daily operation of these systems. Fast response strategies to these interruptions or unplanned events are critical in order to maintain their continuous operation.

5.1.2 Lot scheduling

Since the development of VLSI technology about four decades ago, semiconductor manufacturing has evolved into one of the most complex manufacturing systems. Typically, one such system requires over 400 processing steps on 100 or more different tools, with processing routes consisting of re-entrant flows (revisiting the same sequence of machines for each masking layer). Furthermore, the 300 mm fabs are as large as three football fields.

Extensive studies have been reported that address the traditional scheduling problems involved in the 200 mm fabs. These pertain to the study of qualification lots, integrated metrology, send-ahead wafers, lot sizes, priority lots, reticle scheduling, among others. These issues will continue to exist in a 300 mm fab as well. In addition, new scheduling related issues are expected to arise because of the new features of the 300mm fab.

5.1.3 Integration of lot delivery and scheduling

In the control framework of a 300 mm fab, the AMHS control is connected with production scheduler/dispatcher via a message bus (see Figure 5.4). The factory control system (FCS) contains information about process route of each lot, WIP information, and process recipe required at each operation. The material control system (MCS) is the "traffic controller" of the AMHS. Its function is performed by three subsystems: vehicle controller, track sensor and stocker controller. The vehicle controller tracks the status and positions of the vehicles, and assigns a vehicle to perform a specific move. The stocker controller manages the stocker robot, and keeps track of storage locations of the lots. The vehicle traffic is monitored by track sensors.

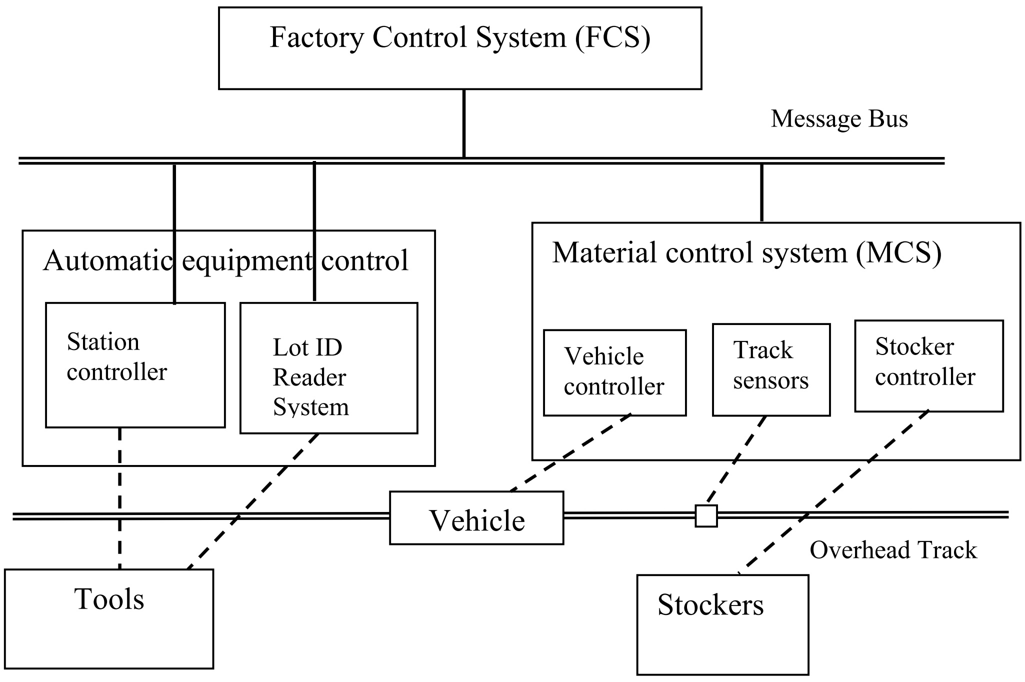

Figure 5.4: The FCS and MCS control framework

The FCS determines scheduling and dispatching of the lots and the reticles. It informs the MCS where to move a lot after the completion of a process step, and which lot to deliver to a tool that has just become available. Once the MCS receives the information from FCS, it determines whether the lot specified by the FCS is in a stocker or in transit. If the next tool is available, it will assign one vehicle to transport the lot to the tool. Otherwise, it will direct the lot to a stocker.

While a significant amount of research effort has been devoted to the operational control of an AMHS and production scheduling individually, the two problems are, however, tightly inter-connected. Clearly, production scheduling provides delivery requests and, in turn, impacts operational control of the AMHS, especially its vehicle scheduling, and provides the release time for lot operations on downstream machines. It is possible to conceive of a situation where a "good" production schedule, determined independently, provides a very "poor" input for vehicle scheduling, and vice versa, making the overall schedule unacceptable. An overall schedule determined by taking into consideration the lot delivery issues and scheduling of production will perform better with regard to overall fab performance metrics, e.g., throughput rate, and cycle time, among others.

International SEMATECH has specified the following requirements to that end for the new dispatcher/scheduler to effectively address the integration problems in a 300 mm fab (International SEMATCH (2000)).

Coordinated delivery for batch processing

This pertains to scheduling coordinated delivery of all the materials included in a batch process operation. The scheduler/dispatcher needs to coordinate the delivery of the materials so that the required materials arrive at the destination equipment within a specified time window for the batch operation.

Determining next destination

This involves determination of the next equipment for a lot after having finished its current processing. The scheduler/dispatcher must be able to determine the next destination for the material to allow for the scheduling of the delivery resources required to transport it to that destination.

Delivery to optimal location

The scheduler/dispatcher may need to schedule the movement of material in order to optimize future delivery times to the next equipment. This situation might be caused by selection of an alternative storage location due to limited storage capacity in the first choice of a stocker. Such an interim delivery would position the material such that the transport time is minimized when the delivery of that material to a production equipment is scheduled.

5.1.4 Problem statement

The integrated AMHS and lot scheduling problem (IMHLSP), or in short, the integrated problem, can be formally stated as follows. G*iven the fab manufacturing environment, including AMHS layout, numbers and types of vehicles, numbers and types of machine tools, layout of machine tools, among others, determine the optimal schedule (including production schedule and vehicle schedule) in order to achieve the best performance metrics such as throughput rate, cycle time and bottleneck utilization in a 300 mm fab,.*

In most cases, an AMHS has limited (finite) vehicle capacity, implying availability of a limited number of vehicles. However, in special cases, e.g., at the ramp-up state, a fab may have low-volume production, and thus, vehicles will not be a constraint. According to M. Kidambi (2006), in such a scenario, the AMHS will not encounter the vehicle delivery problem. We can call this case as the AMHS with infinite vehicle capacity. We study the case of an AMHS with infinite vehicle capacity in this chapter. The case of finite vehicle capacity will be studied in Chapter 6.

The rest of this chapter is organized as follows. In Section 5.2, we present a literature review of the integrated problem. A classification of the integrated problem and relevant mathematical models will be presented in Section 5.3. Section 5.4 contains an analysis and present methodologies for solving this problem. Results of our numerical experimentation are included in Section 5.5. We extend our analysis to fab-wide AMHS in Section 5.6, and, finally conclude our study in Section 5.7.

5.2 Literature review

In the literature, studies pertaining to the IMHLSP can be found within and outside the domain of semiconductor manufacturing. Very few studies have been reported in this regard for the 200 mm and 300 mm fabs. As regards the other domains, some studies on this problem can be found in flexible manufacturing systems (FMS), automated flow shops, job shops and assembly lines.

5.2.1 The IMHLSP in 200 mm fabs

The semiconductor manufacturers introduced automation of 200mm tools to increase product yield, and interbay AMHS was used to improve tool utilization, in the 1990s. Interbay AMHSs are deployed in almost every 200 mm fab throughout the world, which provides an effective means of logistically controlling wafer flow. Cardarelli et al. (1995, 1996) have evaluated the performance measurements of an interbay AMHS by simulation, which includes the effects of design choices, production planning and scheduling, system management, and operator behaviors. Their results have highlighted the importance of storage capacity distribution in the wafer fab. Both papers have also shown the importance of production planning and scheduling, and operator behaviors. Thus, the integration of interbay AMHS system and production scheduling is necessary. But, these papers did not mention how to achieve this integration.

Heinrich and Pyke (1995) have simulated the first large-scale application of conveyor-based AMHS system in 200 mm fabs. Their results have shown that elimination of transport batching reduces factory cycle time, and further reduction in cycle time is possible with dynamic routing of WIP in the AMHS system. Again, this paper emphasized the necessity of integration but did not indicate how to solve the integrated problem.

5.2.2 The IMHLSP in 300 mm fabs

The necessities of integration

The 300mm wafer processing was in vogue by the end of 1990s. The semiconductor manufacturers recognized that fab-wide AMHS (including interbay and intrabay system) is the enabling technology needed to optimize overall fab productivity. This was motivated by economics of equipment utilization and handling of large-size wafers. Based on their observation of the PRI Automation 300mm test line, the first 300mm fab AMHS in the world built by PRI Automation Inc., Chrisos and Patt (1998) suggested that scheduling of WIP production must address the fab-wide wafer transport capability that links one process tool to another whether those tools are across an isle from each other or are at opposite ends of the fab. They also pointed out three types of integration risks: the increased amount of AMHS delivery throughput; the expansive software control needed for WIP planning and scheduling; and the need for service and support.

Besides, NSF/SRC/SEMATECH partnership (National Science Foundation, Semiconductor Research Corporation (SERC) Factory Science Program, and International SEMATECH) have listed several possible research topics in their specification proposal (SERC 2004). The list included: *performance improvements for simulation models for full factory with and without AMHS (interbay, intraby and direct transport system) for both wafer and reticle delivery in fabs, and improving AMHS system throughput for interbay and intrabay.* Similarly, International Technology Roadmap for Semiconductor (ITRS, Factory Integration Section) has consistently listed such topics in their documents in the past years [9].

Integration models and solution methodologies

(1) Industry experience

For semiconductor manufacturers, integration of lot delivery and scheduling, firstly, means the control architecture and software implementation. Reveliotis (1999) proposed the control architecture for integration of lot delivery and scheduling. The requirements for automation software in 300 mm fabs for IBM and Intel are as follows (Carrlker 2004):

1. basic interbay/intrabay transport, load and unload
2. batching, consolidation, queueing, command center
3. factory scheduling integration with in-bay and Flexible Manufacturing Systems (FMS) software

Based on practical experience, Carrlker (2004) introduced some insights for integration: proactively staging vehicles at pickup points to avoid waiting for empty vehicles, optimizing transport requests to minimize the time required to swap lots at a load port, and allowing priority lots to reschedule vehicles previously allocated to other lots. One new challenge arising from scheduling in 300 mm fab is multiple-lots in one carrier. Due to requirement of high carrier utilization or tight customer due date, carrier exchanges are required. Wang et al. (2002) have demonstrated ways for executing automatic carrier exchange, and for determining dynamic routes for splitting/merging of wafers.

(2) Mathematical models

Apart from the publications by the semiconductor manufacturers, which, usually, provide “good” practical insights, some researchers tried to build analytical models in order to derive optimal solutions. Liao et al. (2004) adopted Petri nets to model the dynamics among transport jobs and over-head transport (OHT) vehicles in an intrabay AMHS system. The great advantage of Petri net modeling is its ability to capture the congestion phenomenon among the OHT vehicles. The problem was, then, formulated as an integer programming problem for the objective of minimizing average cycle time of jobs. A small case study consisting of 4 vehicles and 6 jobs was presented. The results showed that the optimal solutions achieved 25.6% improvement over the commonly used Nearest Job First (NJF) rule.

(3) Heuristics

Due to the intractability of large problems, most researchers resort to the use of heuristics or dispatching rules for the solution. Tyan et al. (2004) have presented an integrated tool and vehicle dispatching (ITV) strategy. Interactions of various vehicle dispatching rules and tool dispatching rules were evaluated using simulation models. Due to a large number of combinations of rules, a 2^k factorial experimental design was applied. Based on the simulation results, with the help of a multiple response optimization technique, ITV dispatching strategies were identified. Except for the vehicle and tool dispatching rules, Lin et al. (2006) considered the push/pull control logic in order to improve the system performance and proposed a hybrid push/pull (PP) dispatching rule. The simulation results showed that WIP and cycle time are reduced as a consequence of implementing a PP dispatching rule. Similar results can also be found in Wakabayashi et al. (2004).

Instead of using simple and fast dispatching rules, some papers have introduced more advanced and complex dispatching rules. Li et al. (2005) described an intelligent integrated delivery (IDD) rule by assigning a priority value to each lot. The priority value k_i is given by

$$k_i = \sum_{j=1}^{m} f(k_i^1, k_i^2, \ldots, k_i^m) \bullet \varsigma_j \bullet k_i^j \ ,$$

where k_i^j denotes the lot delivery priority for lot i from source j; ς_j denotes the weighting factor for k_i^j ; and $f(k_i^1, k_i^2, \ldots, k_i^m)$ denotes the correlation function among influencing factors k_i^j , j=1, 2, …, m. The actual delivery time was shown to improve by 18% after deploying IID in one of Intel’s 300 mm factories.

Some research has employed artificial intelligence (AI) techniques to improve the performance of dispatching rules. Min and Yih (2003) have developed a real-time scheduler for selection of dispatching rules for both machines and AMHS in order to obtain desired performance measures at the end of a short production interval. A simulation experiment was conducted to collect data, and then, a competitive neural network was applied to gather scheduling knowledge for future use. The results of their study indicated that this dispatching methodology was effective in considering the complexity of the semiconductor wafer fabrication. Kuo and Huang (2005) proposed a fuzzy-logic-based multimission-oriented OHT dispatcher. The dispatcher is adjusted to accommodate the high-risk lots so that most of the production strategies could be satisfied. Simulation results showed the proposed dispatcher to perform better than other published dispatching rules.

(4) Delivery time estimator/predictor

As is evident from the section above, lot delivery time is an important input parameter for the development of good dispatching rules for the integrated problem. Therefore, some studies have been conducted to predict/estimate the lot delivery time. Liao and Wang (2004) adopted a neural network approach to estimate the delivery times for both priority and regular lots. Numerical experiments have indicated that this neural network approach is sound and effective for the prediction of average delivery times (see Figure 5.5). Besides, Mackulak and Savory (2001) have demonstrated how to use regression model to approximate the lot delivery time. The main parameters in the regression model include vehicle speed and number of tools in the bay.

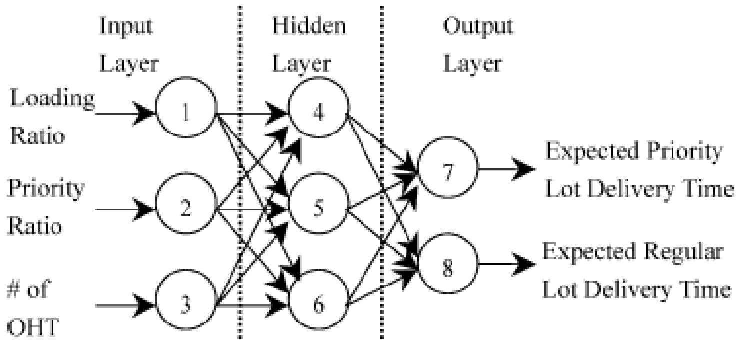

Figure 5.5: A three-layer neural-network model

5.2.3 The IMHLSP in other domains

Some fabs employ Automated Guided Vehicles (AGVs) instead of OHT vehicles for their interbay AMHS system. AGVs have been widely applied in other manufacturing systems over the last several decades. Therefore, this integrated problem can also be found in other manufacturing systems where AGVs are used. These manufacturing systems are flexible manufacturing systems (FMS), automated flow shops, job shops, and assembly lines, among others. For a detailed review of the design and control issues of AGVs, please refer to Ganesharajah et al (1998) and Tuan and Koster (2006).

The integration problem in FMS

Sabuncuoglu and Hommertzheim (1992) have proposed an on-line dispatching algorithm, which uses relevant information concerning the load of the system and the status of the jobs. It consisted of two parts: a logic associated with the scheduling of jobs on the AGVs and a logic associated with the scheduling of jobs on the machine. The first part consisted of four hierarchical levels:

- push logic (checking the critical stations),
- buffer logic (checking the parts in the central buffer area),
- pull logic (checking the idle stations),
- push-pull logic (identification of the most appropriate workstation and part to be serviced).

The second part mainly computed the priority index for each candidate job waiting in the workstation queue as follows:

$$\text{priority index} = \text{weight}_1 * \text{operation time} + \text{weight}_2 * \text{expected waiting time on the next operation}$$

Performance of the proposed algorithm was compared with several machine and AGV scheduling rules by using mean flow time and mean tardiness criteria. Simulation results indicated that the proposed algorithm produced significant improvements over existing scheduling rules for a variety of experimental conditions.

A similar idea can be found in Fataneh (1997). He presented a vehicle dispatching rule, called vehicle assignment by load utility evidence (VALUE), to select the next assignment for an available vehicle. The criticality index for output queue at workstation k, x_k , is calculated as follows:

$$x_k = S_k / Q_k ,$$

where, Q_k = buffer capacity of output queue at workstation k, and S_k = current length of output queue at workstation k.

And, the value added for each unit-load at output queue of work station k, v_{ij} , is:

$$v_{ij} = p_{ij} / n_{ij} ,$$

where, p_{ij} = total number of operations completed for the jth unit-load waiting in output buffer of workstation k, and n_{ij} = total number of operations required for the jth unit-load waiting in the output buffer of workstation k.

The VALUE rule selected the unit-load with smallest n_{ij} value from the output queue of the workstation which has the biggest x_k value. Simulation experiments showed that the VALUE rule outperformed some of the best dispatching rules reported in the literature in terms of throughput and flow time.

Instead of applying on-line scheduling heuristics to solve this integration problem, some researchers have tried to find the optimal schedule. Bilge and Ulusoy (1995) formulated the integrated problem as a mixed integer problem (MIP). The formulated model can be decomposed into two subproblems: a machine scheduling problem and a generic vehicle scheduling problem (VSP). These two subproblems interact with each other. Since both subproblems are NP-hard, an iterative solution procedure was proposed. At each iteration, a new machine schedule was generated by a heuristic procedure. The operation completion times obtained from this schedule are used to construct time windows for the trips, and a feasible solution is searched for the second subproblem, which is handled as a sliding time-window problem. Numerical experiments were conducted on 90 example problems and the iterative solution procedure achieved 7% improvement in makespan on the average.

For the large-size problems, a genetic algorithm (GA) was proposed by the same authors in Ulusoy et al (1997). In the GA algorithm, chromosomes represented both operation sequencing and AGV assignment dimensions of the search space. A third dimension, time, was implicitly given by the ordering of operations of the chromosomes. A special uniform crossover operator was developed which produced one offspring from two parent chromosomes. One hundred and eighty test problems were solved. An easily computable lower bound was introduced and compared with the results of GA. In 60% of the problems, the proposed GA reached the lower bound, thereby indicating obtainment of optimal solutions.

A similar MIP model was proposed by Liu and MacCarthy (1997) with special constraints about the central storage and storage buffer for each machine. Due to computational complexity, two heuristic procedures were developed based on the analysis of the MIP model. One of these procedures is a “loading then sequencing” procedure and the other is a global heuristic procedure. Computational results showed that the global heuristic procedure performed better than the widely used “loading then sequencing” procedure in FMS.

The IMHLSP in automated assembly lines

Anwar and. Nagi (1998) have considered the simultaneous scheduling of AGVs and manufacturing equipment in the production of a complex assembled product. It is named the “transportation integrated scheduling problem (TISP)”. The objective is to minimize the makespan. A heuristic procedure, called transportation integrated problem scheduling algorithm (TIPSA) is proposed for this problem. The proposed heuristic regarded the transportation process as an *"operation*" and included it in the operation network of the assembly line (see Figures 5.6 and 5.7). In Figure 5.6, each node represents an operation for the assembly line. New nodes, designated with name starting with "T", in Figure 5.7, are the transportation tasks needed between relevant operations. The transportation schedule was, thus, obtained by exploiting the critical path of the operation network. Experimentation on a large number of examples showed that the schedules obtained by TIPSA result in considerable improvement in makespan over other scheduling methods reported in the literature.

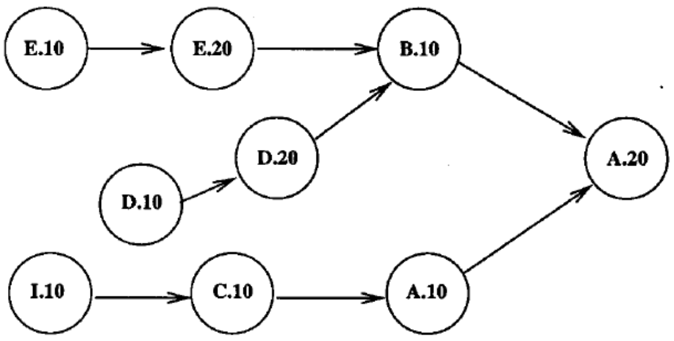

Figure 5.6: The operational network for a complex product

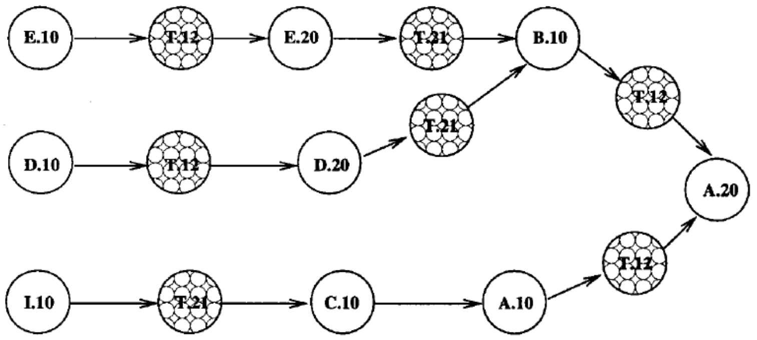

Figure 5.7: The operational network for a complex product including the transportation operation

Similar studies on the integrated problems arising in automated flow shop and job shops can be found in Han and Mcginnis (1989), Kise et al. (1991), Hall et al (2001), Kim et al (1999), and Smith et al (1999).

5.2.4 Conclusions

In this section, we have summarized solution methodologies for the IMHLSPs encountered in the 200mm and 300mm fabs and other domains. The differences among the IMHLSPs encountered in these domains are also indicated.

Most 200 mm fabs implement an interbay AMHS system, but intrabay AMHS system is rarely employed in the 200 mm fab. Operators act as integrating units between an interbay AMHS system and production scheduling. Hence, there is usually no fully automated system in 200 mm fabs, and the integration problem in 200 mm fab is not critical. Very few papers have addressed this issue.

Some work has been reported on the integrated problem in 300 mm fabs. But, most of the studies use dispatching rules, or dispatching rules with AI techniques embedded in order to improve their performance. There is only one reported study, which formulated a mathematical programming model for a variation of the integrated problem that applied Petri net modeling technology, and solved it optimally. The lack of reported work in this regard is because of the difficulty in solving large-size problems or real industry-size problems as the integrated problem is NP-hard.

There have been more extensive studies reported on the integrated problems in other domains where AGVs are widely used. This is because those industries have a longer history than that of the semiconductor industry. Some research papers have proposed mathematical models for FMS, automated flow shop and assembly lines. These models provide useful information for building mathematical model for the integrated problem in 300 mm fabs. When dealing with large-size problems, the heuristic algorithms are used, such as GA, and priority index assignment.

However, it should be noted that the manufacturing environments in the domains outside of semiconductor manufacturing are different from 300 mm fabs. The 300 mm fab employs more vehicles and much more complex AMHS routes than those involved in other domains. There are also a greater number of jobs and machine tools involved in the 300 mm fabs. Therefore, the problem of integrating AMHS and lot scheduling in 300mm fabs is a challenging one, but at the same time, it can be quite rewarding.

5.3 Problem classification and mathematical models

5.3.1 AMHS Operating Model

Different models have been used for operating an AMHS, depending upon the operating logic used. We present these next.

Segregate Model

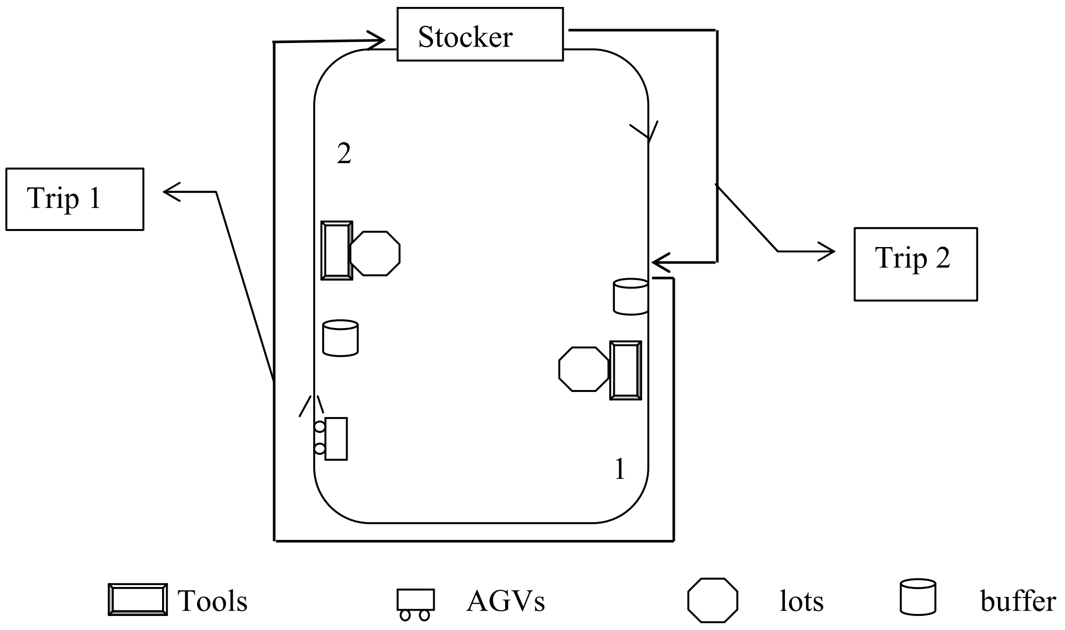

Figure 5.8: A segregate model

Under this logic of operating an AMHS, once a job is finished, it is always transported back to the central stocker first even if there is an available buffer space at its next destination tool. It will, then, be moved to its next destination tool (see Figure 5.8). Thus, to move a lot from a tool to the next tool requires two trips. The segregate model is easy to control and it does not need complicated material control system (MCS). But, it increases the demand for lot movements. It has been widely used at the ramp-up stage for a 300mm fab.

Direct model

Under this operating logic, once a job is finished, it is directly transported to its next destination tool if there is an empty buffer space available at the destination tool (see Figure 5.9). Otherwise, it stays at the current tool, thereby, blocking it. This model reduces the transportation tasks of the AMHS. But, tool blocking will likely happen, and it will reduce tool utilization.

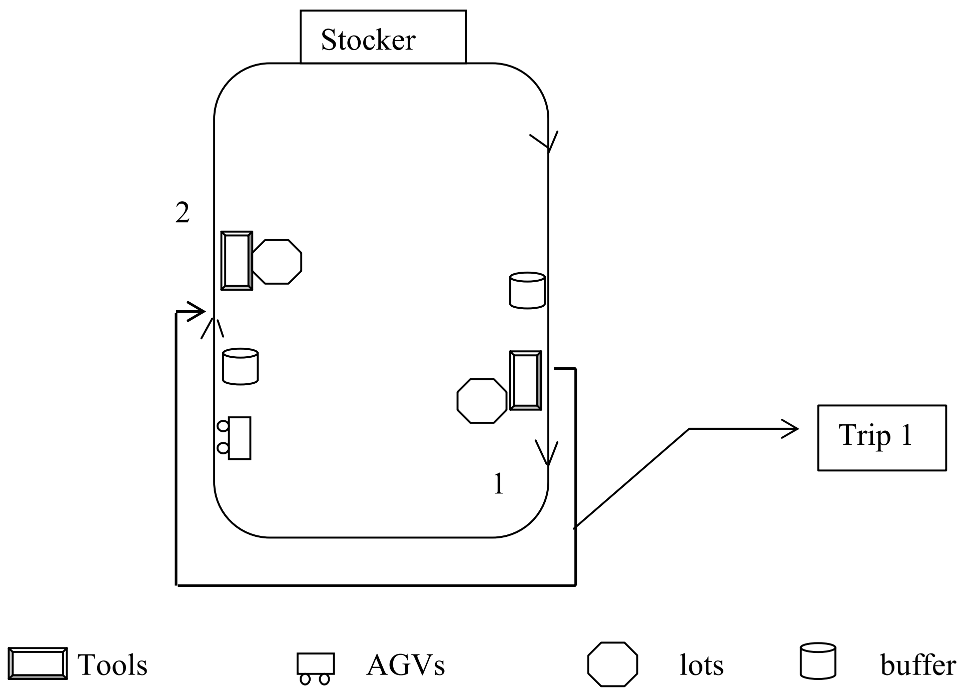

Figure 5.9: A direct model

Weak hybrid model

This operating logic is a hybrid of the above two. Once a job finishes processing on machine 1, it is immediately moved to machine 2 provided there is an empty buffer available on machine 2. Otherwise, it is transported immediately to the stocker. This is like a “push”. This system tends to maximize the equipment utilization, which is critical to the memory chip manufacturers.

Strong hybrid model

This operating logic is a further refinement of the weak hybrid model. Once a job finishes processing on machine 1, to schedule its next operation, we choose the best between the segregate and direct models. It has advantages of both of these models. But, it is a difficult model to implement, and thus, it is not popular in practice. However, we will further explore this model in our study.

Note that, the above figures only indicate the intrabay scenarios. Actually, we can extend these to the entire AMHS including interbay and intrabay (see Figure 5.10). The only difference will be the transportation length and time.

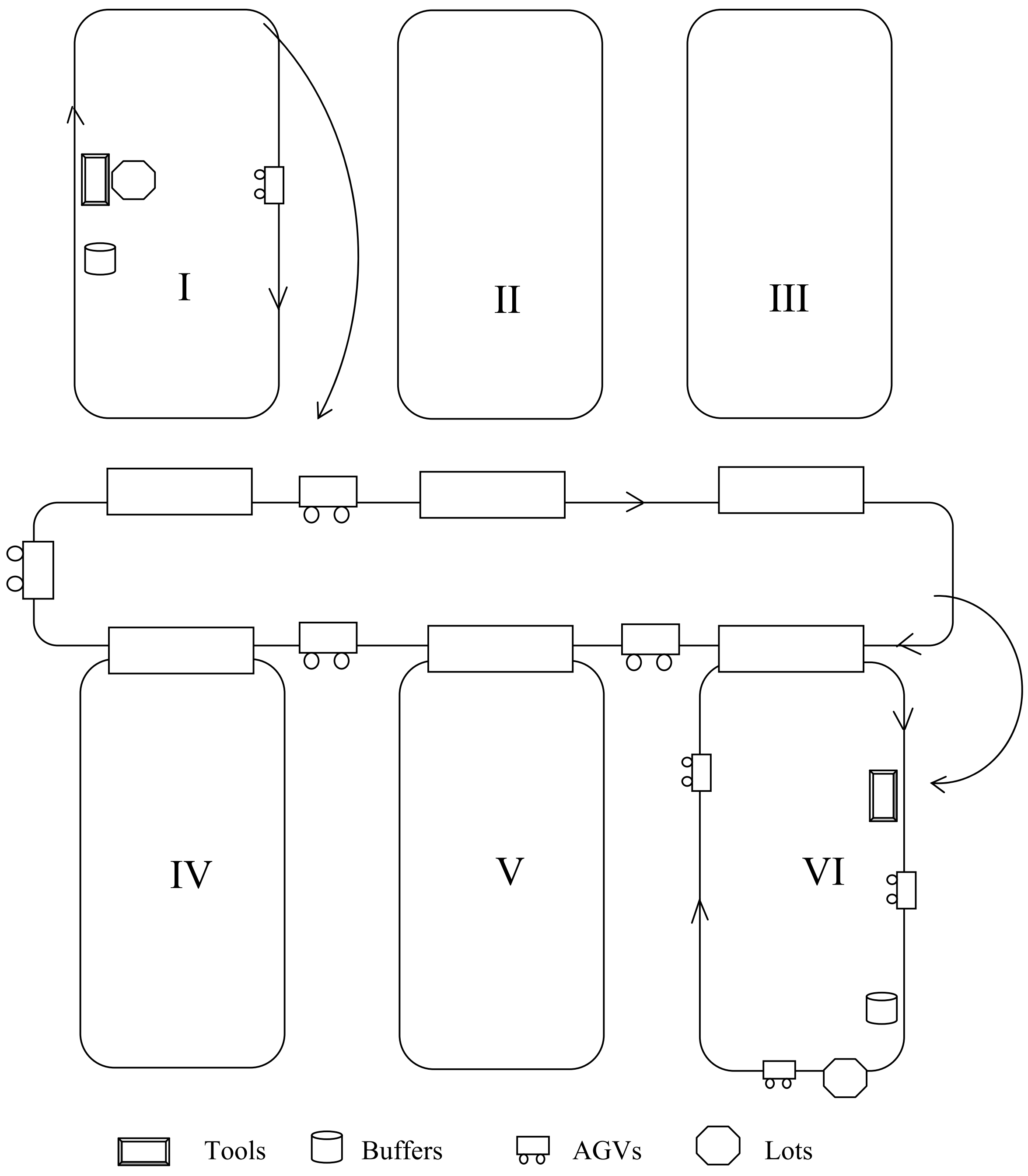

Figure 5.10: A model for the entire AMHS

5.3.2 Mathematical models

Parameters:

N : number of jobs,

J : number of machines (equivalent to operations)

p_{ij} : processing time for operation *j* of job *i*

lt_j : loading travel time from stocker to machine *j* (we call movement of a job from the stocker to a machine as "loading travel")

ut_j : unloading travel time from machine *j* to stocker (we call movement of a job from a machine to the stocker as "unloading travel")

$tt_{j,j+1}$: travel time from machine *j* to machine *j*+1

b_j : buffer size on machine *j*

H : a big positive number

TT : travel time of one intrabay loop (the sum of times from the stocker to machine 1, from machine 1 to machine 2, and from machine 2 to the stocker)

Decision variables:

Z : makespan

T_{kj}: completion time of operation *j* of the job scheduled in position *k*

S_{kj}: starting time on machine *j* for the job scheduled in position *k*

$$X_{ik} = \begin{cases} 1, & \text{If job } i \text{ is scheduled in position } k \\ 0, & \text{Otherwise} \end{cases}$$

$$Seg_{kj} = \begin{cases} 1, & \text{If the segregate model is implemented for the lot scheduled in position } k \text{ on machine } j \\ 0, & \text{Otherwise (which implies use of the direct model)} \end{cases}$$

Segregate model

Minimize Z

Subject to

$$Z \geq T_{NJ} + ut_J \tag{5.1}$$

$$\sum_{i=1}^{N} X_{ik} = 1, k=1, 2, .., N \tag{5.2}$$

$$\sum_{k=1}^{N} X_{ik} = 1, i=1, 2, \ldots, N \tag{5.3}$$

$$T_{kj} = S_{kj} + \sum_{i=1}^{N} X_{ik} p_{ij}, k=1, 2, \ldots, N, j=1, 2, \ldots, J \tag{5.4}$$

$$S_{kj} \geq T_{k,j-1} + TT + tt_{j-1,j}, k=2,\ldots, N, j=2,\ldots, J \tag{5.5}$$

$$S_{kj} \geq T_{k-1,j}, k=2,\ldots, N, j=1,2, \ldots, J \tag{5.6}$$

$$S_{kj} \geq lt_j, k=1, 2, \ldots, N, j=1, 2, \ldots, J \tag{5.7}$$

$$X_{ik} \text{ binary}, i=1, 2, \ldots, N, k=1, 2, \ldots, N; S_{kj}, T_{kj} \geq 0, k=1, \ldots, N, j=1, \ldots, J \tag{5.8}$$

The makespan is the completion time of the last job *N* plus the unloading travel time of ut_J. This is captured by constraint set (5.1). Constraint sets (5.2) and (5.3) ensure that only one job is assigned to one position in the sequence, and also, one position contains only one job. Constraint (5.4) states that completion time of a job is equal to the start time plus its processing time. Starting from the second operation, since the segregate model is used, the start time of a job scheduled at position *k* at operation *j* ($j > 1$) is no earlier than the time it finishes its previous operation (*j*-1) and the time required to transport it to operation *j,* with transportation time of $TT + tt_{j-1,j}$. This is represented by constraint set (5.5). Constraint set (5.6) asserts that the job scheduled at position *k* can only start after the job scheduled at position *k*-1 on that machine has been completed. And, constraint set (5.7) enforces that the start time for a job must be no earlier than the time it is transported to the machine.

Direct model

Minimize Z

Subject to

$Z \geq T_{NJ} + ut_J$ (5.9)

$\sum_i X_{ik} = 1$, k=1, 2, …, N (5.10)

$\sum_k X_{ik} = 1$, i=1, 2, …, N (5.11)

$T_{kj} = S_{kj} + \sum_i X_{ik} p_{ij}$, k=1, 2, …., N, j=1, 2, …, J (5.12)

$S_{kj} \geq T_{k,j-1} + tt_{j-1,j}$, k=1, 2,…, N, j=2, … , J (5.13)

$S_{kj} \geq T_{k-1,j}$, k=2,…, N, j=1, 2, …, J (5.14)

$S_{kj} \geq S_{k-b_{j+1}-1,j+1} - tt_{j,j+1}$, $k = b_{j+1}$+2, …, N, j=1,2, …, J-1 (5.15)

$S_{kj} \geq lt_j$, k=1, 2, …, N, j=1, 2, …, J (5.16)

X_{ik} binary, i=1, 2, …, N, k=1, 2, …, N , S_{kj} , $T_{kj} \geq 0$, k=1, …, N, j=1, …, J (5.17)

Constraint sets (5.9), (5.10), (5.11), (5.12), (5.14) and (5.16) are identical to constraint sets (5.1), (5.2), (5.3), (5.4), (5.6) and (5.7), respectively. Since the direct model is used, the transportation time from operation j-1 to j is $tt_{j-1,j}$. This is captured in constraint (5.14). Constraint set (5.15) is the constraint for limited buffer size b_{j+1} on machine j+1. This means that the job scheduled at position k will not start processing on machine j unless the job at position (k-b-1) has already started processing on machine j+1.

Strong Hybrid model

Minimize Z

Subject to

$Z \geq T_{NJ} + ut_J$ (5.18)

$\sum_i X_{ik} = 1$, k=1, 2, …, N (5.19)

$$\sum_k X_{ik} = 1,\ i{=}1, 2, \ldots, N \tag{5.20}$$

$$T_{kj} = S_{kj} + \sum_i X_{ik} p_{ij},\ k{=}1, 2, \ldots, N, j{=}1, 2, \ldots, J \tag{5.21}$$

$$S_{kj} \geq T_{k,j-1} + TT * Seg_{kj} + tt_{j-1,j},\ k{=}1, 2,\ldots, N, j{=}2, \ldots, J \tag{5.22}$$

$$S_{kj} \geq T_{k-1,j},\ k{=}2, \ldots, N,\ j{=}1, 2,\ldots, J \tag{5.23}$$

$$S_{kj} + H * Seg_{k-1,j+1} \geq S_{k-b_{j+1}-1,j+1} - (1 - Seg_{k-1,j+1}) * tt_{j,j+1},\ k = b{+}2,\ldots, N, j{=}1, 2,\ldots, J{-}1 \tag{5.24}$$

$$S_{kj} \geq lt_j,\ k{=}1, 2, \ldots, N, j{=}1, 2, \ldots, J \tag{5.25}$$

X_{ik} binary, i=1, 2, …, N, k=1, 2, …, N, Seg_{kj} binary, i=1, 2, …, N, k=1, 2, …, N; S_{kj},

$T_{kj} \geq 0$, k=1, …, N, j=1, …, J (5.26)

The above constraint sets are basically a combination of segregate and direct models. Decision variable Seg_{kj} is used to determine whether the segregate model is used for position k at machine j or not (in which case the direct model is chosen). If segregate model is selected, constraint set (5.22) will enforce the transportation delay of $TT + tt_{j-1,j}$ and the constraint about limited buffer size (5.24) will be relaxed. On the contrary, if the direct model is chosen, constraint (5.24) will be enforced and constraint set (5.22) will impose a transportation delay of $tt_{j,j+1}$. Constraint sets (5.18), (5.19), (5.20), (5.21), (5.23) and (5.25) are identical to constraint sets (5.1), (5.2), (5.3), (5.4), (5.6) and (5.7), respectively. H must be larger than the maximum starting time for job on machine J-1. This can be easily obtained by generating a feasible solution, and then, using its makespan as this value.

5.4 Solution methodologies

In the section, we will analyze each of the above models as well as compare their performances. There are two factors, which make this problem hard to solve. One is the limited buffer size, and the other is a choice between segregate and direct models. There have not been many studies reported in the literature on scheduling problems in the presence of limited buffer sizes between the machines. Furthermore, most of the studies that are conducted propose the use of simple heuristics (see Leisten (1990), Hall and Sriskandarajah (1996)).

We will start by considering the cases with infinite or zero buffer size due to the following reasons.

(1) For a two-machine case, for certain values of TT, $tt_{1,2}$ p_{k1}, and p_{k2}, machine 2 can be approximated to have infinite buffer.

Proposition 5.1: Given p_{k1}, p_{k2}, if $\max\{p_{k2}\} < \min\{p_{k1}\}$, for *k*=1, 2, …, *N*, then, this is equivalent to saying that machine 2 has infinite buffer.

Proof: Given a sequence π, we can find the job completion time on machine 1 and 2 as follows.

First job finishes processing on machine 1 at $lt_1 + p_{11}$.

The nth job finishes processing on machine 1 at $lt_1 + \sum_{k=1}^{n} p_{k1}$.

First job finishes on machine 2 at $(lt_1 + p_{11} + tt_{1,2}) + p_{12}$.

Second job finishes on machine 2 at

$$\max\{(lt_1 + p_{11} + p_{21} + tt_{1,2}), (lt_1 + p_{11} + tt_{1,2} + p_{12})\} + p_{22}$$
$$= lt_1 + p_{11} + tt_{1,2} + \max\{p_{21}, p_{12}\} + p_{22}.$$

Note that the second job does not wait on machine 2 since $p_{21} \geq p_{12}$. This is true for other jobs as well. ◘

Remark 5.1: We can extend the result of Proposition 5.1 to *J* machines provided $\min_k\{p_{kj}\} \geq \max_k\{p_{k,j+1}\}$ for *j*=1, 2, …, *J*-1.

(2) Similarly, if $\min\{p_{k2}\} > \max\{p_{k1}\}$, the jobs are going to wait on machine 2. This is equivalent to saying that machine 2 has zero buffer.

(3) There is a current trend in 300mm wafer fabs for building "zero footprint buffers" close to equipments (Kidambi 2006). The buffers are built under the ceiling or close to the overhead track. This leads to having approximately infinite buffers for some important equipment, if not all.

(4) The zero buffer capacity solution often acts as a good starting point for determining a solution for the case of limited buffer size.

5.4.1 Infinite buffers

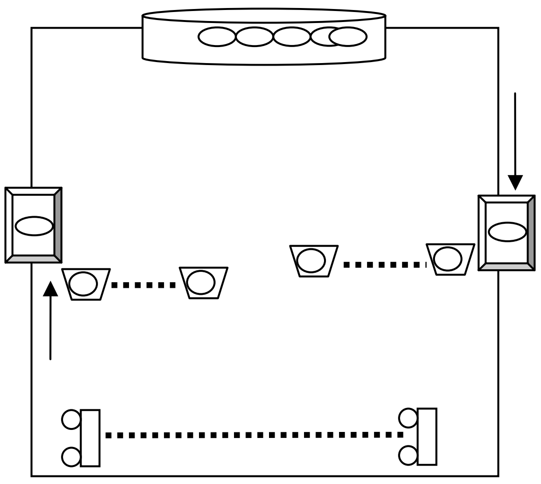

Figure 5.11: An intrabay AMHS with infinite buffers

Proposition 5.2 For the direct model, Johnson's permutation schedule is optimal.

Proof: The situation on hand is shown in Figure 5.11. For the case where there are enough vehicles, i.e., once a job finishes processing, it is immediately moved by a vehicle, the scenario reduces to that of a flow shop with arbitrary time-lags, t_j, between the operations of job *j* on successive machines. For the 2-machine flow shop problem with time lags, Mitten (1959) has proved that an optimal schedule can be obtained in polynomial time by Johnson's algorithm (1954) with $p'_{1j} = p_{1j} + t_j$ and $p'_{2j} = p_{2j} + t_j$. In our case, since $tt_{1,2}$ is the same for all jobs, actually Johnson's algorithm can be directly applied without using p'_{1j} , p'_{2j} . ◘

Let C_1 indicate the makespan obtained from Johnson's permutation schedule. With arbitrary time lags $tt_{1,2}$, the makespan is $C_1' = C_1 + tt_{1,2}$. For the intrabay AMHS, we need to consider the loading time from the stocker to machine 1 and the unloading time from machine 2 to stocker (see Figure 5.12). So the makespan is $C_{\max} = C_1 + tt_{1,2} + lt_1 + ut_2$.

For the segregate model, the optimal makespan obtained using Johnson's algorithm, $C_{\max} = C_1 + TT + tt_{1,2} + lt_1 + ut_2$.

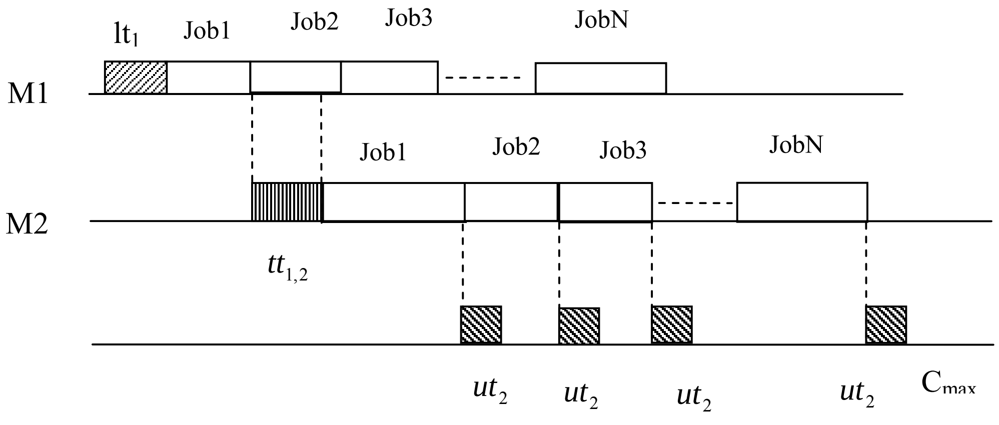

Figure 5.12: The optimal makespan for direct model with infinite buffers

Corollary 5.1 The optimal makespan value for the direct model is smaller than that for the segregate model by an amount of TT .

This is the reason that the 300mm fabs desire to build more "zero footprint" buffers and apply the direct model. The optimal makespan for this case can be a lower bound of makespan for the case of zero buffer or limited buffer. Note that, hybrid models (both weak and strong) do not exist for this case due to infinite buffer spaces.

5.4.2 Zero buffer

5.4.2.1 Segregate model

Illustrations of segregate model for the zero-buffer case are given in Figures 5.13 and 5.14.

The stocker is assumed to have unlimited buffer capacity. Also, each finished job waits in the stocker before it is transferred to machine 2, when it becomes available, under segregate model. Therefore, the fact that we have zero buffer at machine 2 is irrelevant for this system.

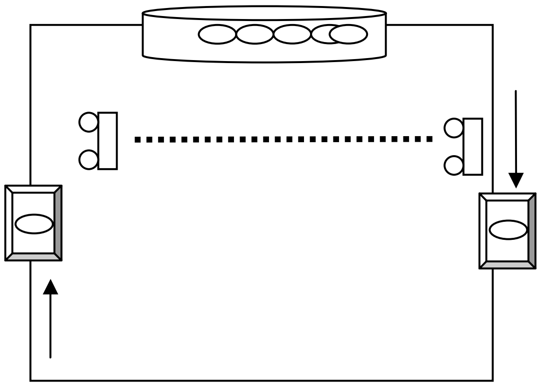

Figure 5.13: An intrabay AMHS for zero buffers

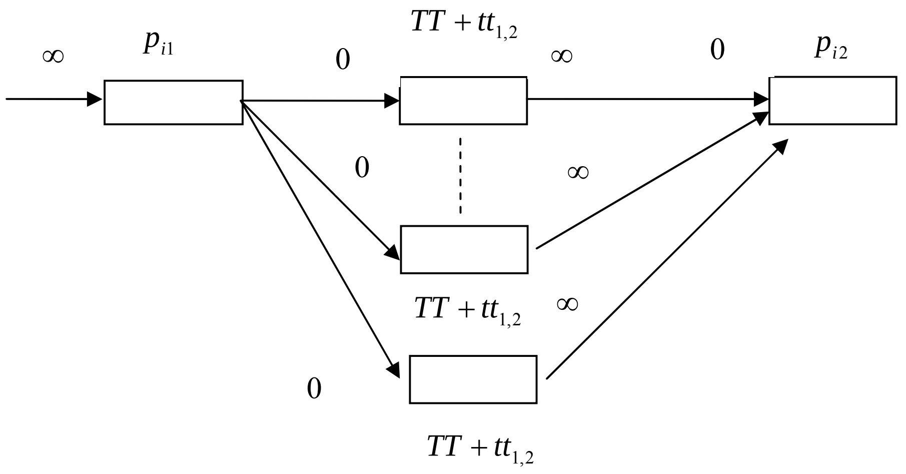

Figure 5.14: A illustration of segregate model for zero buffer at machine 2

Alternatively, we can show this by using the closed-form expression of the makespan value for this case. Let C_{ij} indicate completion time of operation *j* of job *i*, operation *j*, We have

$$C_{11} = lt_1 + p_{11}.$$

$$C_{21} = lt_1 + p_{11} + p_{21}.$$

$$C_{N1} = lt_1 + \sum_{i=1}^{N} p_{i1}.$$

$$C_{12} = C_{11} + ut_1 + lt_2 + p_{12}.$$

$$\begin{aligned} C_{22} &= \max\{C_{12}, C_{21} + ut_1 + lt_2\} + p_{22} \\ &= \max\{C_{11} + ut_1 + lt_2 + p_{12} + p_{22}, C_{21} + ut_1 + lt_2 + p_{22}\}. \end{aligned}$$

$$\begin{aligned} C_{32} &= \max\{C_{22}, C_{31} + ut_1 + lt_2\} + p_{32} \\ &= \max\{C_{11} + ut_1 + lt_2 + p_{12} + p_{22} + p_{32}, C_{21} + ut_1 + lt_2 + p_{22} + p_{32}, C_{31} + ut_1 + lt_2 + p_{32}\}. \end{aligned}$$

By simple induction, we have

$$\begin{aligned} C_{N2} &= \max\{C_{N-1,2}, C_{N1} + ut_1 + lt_2\} + p_{N2} \\ &= \max\{C_{11} + ut_1 + lt_2 + p_{12} + p_{22} + p_{32} + \ldots + P_{N2}, C_{21} + ut_1 + lt_2 + p_{22} + p_{32} + \ldots + P_{N2}, \\ &\qquad C_{31} + ut_1 + lt_2 + p_{32} + p_{42} + .. + P_{N2}, \ldots\ldots, C_{N1} + ut_1 + lt_2 + P_{N2}\}. \end{aligned}$$

After substituting for C_{i1}, i=1, 2, ..., N, from above, we have

$$C_{N2} = lt_1 + ut_1 + lt_2 + \max\{\{\sum_{i=1}^{K} p_{i1} + \sum_{i=K}^{N} p_{i2}\}\},\ K=1, 2, \ldots, N.$$

This is equivalent to finding the critical path (longest path) for the classical 2-machine flow shop problem. Using Johnson's algorithm, the optimal makespan is determined as $C_{\max} = C_1 + TT + tt_{1,2} + lt_1 + ut_2$.

Note that at the beginning, machine 2 is idle, it is more efficient to move the first job directly to machine 2 instead of routing it to stocker first. Thus, from now on, we apply direct model for the first job in segregate model for zero-buffer case.

Proposition 5.3 With the first job using the direct model, the optimal makespan value is obtained by applying Johnson's algorithm to the remaining N-1 jobs.

Proof: A closed-form expression for makespan is as follows:

$$\begin{aligned} C_{N2} &= \max\{C_{N-1,2}, C_{N1} + ut_1 + lt_2\} + p_{N2} \\ &= \max\{C_{11} + tt_{1,2} + p_{12} + p_{22} + p_{32} + \ldots + P_{N2}, C_{21} + ut_1 + lt_2 + p_{22} + p_{32} + \ldots + P_{N2}, \\ &\qquad C_{31} + ut_1 + lt_2 + p_{32} + p_{42} + .. + P_{N2}, \ldots\ldots, C_{N1} + ut_1 + lt_2 + P_{N2}\}. \end{aligned}$$

The above expression can be re-arranged as follows:

$$\begin{aligned} C_{N2} &= \max\{(C_{11} + tt_{1,2} + p_{12} + p_{22} + p_{32} + \ldots + P_{N2}), \\ &\qquad C_{11} + \max\{p_{21} + ut_1 + lt_2 + p_{22} + p_{32} + \ldots + P_{N2}, p_{21} + p_{31} + ut_1 + lt_2 + p_{32} + p_{42} \\ &\qquad + .. + P_{N2}, \ldots\ldots, p_{21} + p_{31} + \ldots + p_{N1} + ut_1 + lt_2 + P_{N2}\}\}. \end{aligned}$$

The second part represents determination of the critical path for the remaining N-1 jobs for a 2-machine flow shop.

Thus, an algorithm for the segregate model (**Improve-Seg-1**) is as follows.

Improve-Seg1 Algorithm:

Step 1: Arrange the N jobs in an arbitrary sequence. Select the first job of the sequence; set k=1.

Step 2: Schedule job k as the first job using the direct model, and for the remaining N-1 jobs, apply the segregate model. Determine optimal sequence using Johnson's algorithm.

Step 3: Let $k = k$+1. If $k > N$, go to step 4; otherwise, go to Step 2

Step 4: Compare the N schedules and find the optimal schedule with minimum makespan.

5.4.2.2 Direct model

An illustration of this model is given in Figure 5.15.

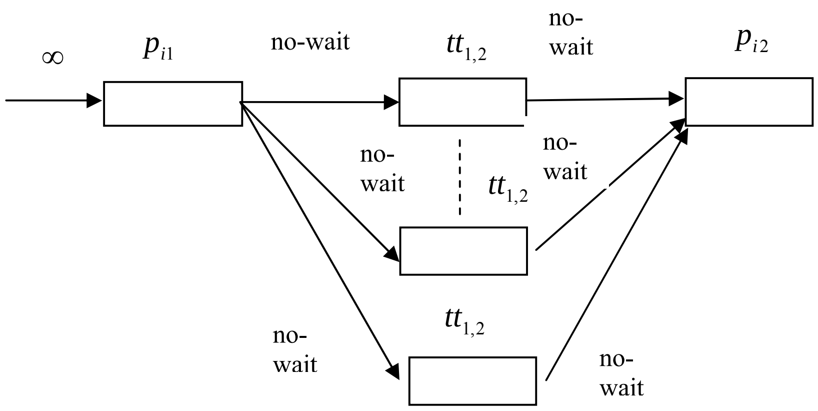

Figure 5.15: An illustration of the intrabay direct AMHS model with zero buffer

Proposition 5.4: This problem is solvable in polynomial time by the algorithm proposed by Gilmore-Gomory (1964).

Proof: Due to zero buffer at machine 2, a job, once having completed processing on machine 1, is transported to machine 2 for its operation on that machine with no wait. Since $tt_{1,2}$ is a fixed value for every job i, this situation is equivalent to a 2-machine, no-wait flow shop with all jobs on machine 1 right-shifted by $tt_{1,2}$ (see Figure 5.16). For a 2-machine no-wait flow shop, the algorithm by Gilmore-Gomory (1964) generates an optimal solution in polynomial time. ◘

Alternatively, we can write closed-form expression for the makespan. There may be blocking time on machine 1 for each job. We denote it by Δ_{i1} for job i, $i \geq 2$. This is the time that job i is delayed for processing on machine 1 because job i-1 is still waiting on machine 1 due to the unavailability of machine 2.

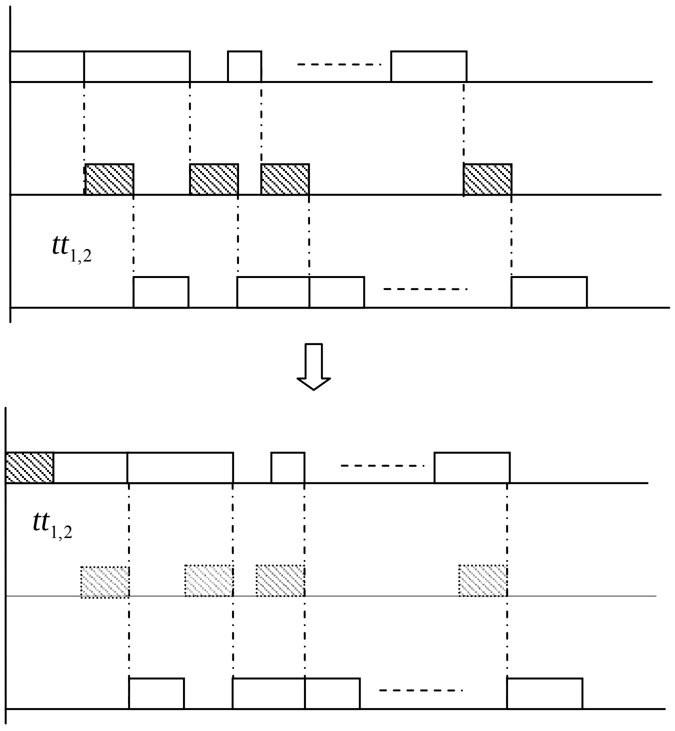

Figure 5.16: Transformation of the direct model to a no-wait 2-machine flow shop

$C_{11} = lt_1 + p_{11}$.

$C_{21} = lt_1 + p_{11} + \Delta_{21} + p_{21}$.

$C_{N1} = lt_1 + \sum_{i=1}^{N} p_{i1} + \sum_{i=2}^{N} \Delta_{i1}$.

$C_{12} = lt_1 + p_{11} + tt_{1,2} + p_{12}$.

$$\begin{aligned}C_{22} &= \max\{C_{12}, C_{21} + tt_{1,2}\} + p_{22} \\ &= C_{21} + tt_{1,2} + p_{22}.\end{aligned}$$

$$\begin{aligned}\Delta_{21} &= \max\{0, C_{12} - (C_{11} + p_{21} + tt_{1,2})\} \\ &= \max\{0, (C_{11} + p_{12} + tt_{1,2}) - (C_{11} + p_{21} + tt_{1,2})\} \\ &= \max\{0, (p_{12} - p_{21})\}.\end{aligned}$$

$$\begin{aligned}C_{32} &= \max\{C_{22}, C_{31} + tt_{1,2}\} + p_{22} \\ &= C_{31} + tt_{1,2} + p_{32}.\end{aligned}$$

$$\begin{aligned}\Delta_{31} &= \max\{0, C_{22} - (C_{21} + p_{31} + tt_{1,2})\} \\ &= \max\{0, (C_{21} + p_{22} + tt_{1,2}) - (C_{21} + p_{31} + tt_{1,2})\} \\ &= \max\{0, (p_{22} - p_{31})\}.\end{aligned}$$

By simple induction, we have

$$\begin{aligned}\Delta_{N1} &= \max\{0, C_{N-1,2} - (C_{N-1,1} + p_{N1} + tt_{1,2})\} \\ &= \max\{0, (C_{N-1,1} + p_{N-1,2} + tt_{1,2}) - (C_{N-1,1} + p_{N1} + tt_{1,2})\} \\ &= \max\{0, (p_{N-1,2} - p_{N1})\}.\end{aligned}$$

$$\begin{aligned}C_{N2} &= C_{N1} + tt_{1,2} + p_{N2} \\ &= C_{N-1,1} + \Delta N1 + p_{N1} + tt_{1,2} + p_{N2} \\ &= lt_1 + tt_{1,2} + \sum_{i=2}^{N} \max\{0, (p_{i-1,2} - p_{i1})\} + \sum_{i=1}^{N} p_{i1} + p_{N2}.\end{aligned}$$

Note that C_{N2} is exactly the expression for a no-wait 2-machine flowshop.

Corollary 5.2: With zero buffer, the direct model is not necessarily better than the segregate model for a given sequence π.

Proof: Direct model tends to generate idle time on machine 1 to compensate for the longer transportation time of going back to the stocker. On the other hand, the segregate model never generates idle time on machine 1, while it incurs the cost of longer transportation back to the stocker. Thus, the dominance of one method over the other depends on the values of p_{i1}, p_{i2}, $tt_{1,2}$ and TT. ◘

Consider the following examples.

Example 1:

Data for this example is given in Table 5.1. $TT = 3$ and $tt_{1,2} = 1$. For the sequence {1, 2, 3}, makespan for the segregate model is 19 while that for the direct model is 17.

Table 5.1: Data for Example 1

Operations	Job 1	Job 2	Job 3
1	3	3	3
2	5	2	5
$TT = 3$			
$tt_{1,2} = 1$			

Example 2:

Table 5.2: Data for Example 2

Operations	Job 1	Job 2	Job 3
1	3	3	5
2	7	2	3
$TT = 2$			
$tt_{1,2} = 1$			

Data for this example is shown in Table 5.2. For the sequence {1, 2, 3}, makespan for the segregate model is 18 while that for the direct model is 21.

5.4.2.3 Weak hybrid model

In this system, machine 1 is never idle. We have,

$C_{11} = lt_1 + p_{11}$.

$C_{21} = lt_1 + p_{11} + p_{21}$.

$C_{N1} = lt_1 + \sum_{i=1}^{N} p_{i1}.$

$C_{12} = C_{11} + ut_1 + lt_2 + p_{12}.$

For $C_{22,}$ we have 2 cases:

Case 1: $C_{21} + tt_{1,2} \geq C_{12}$

We use direct model, $C_{22} = C_{21} + tt_{1,2} + p_{22}.$

Case 2: $C_{21} + tt_{1,2} < C_{12}$

We use segregate model, $C_{22} = \max\{C_{12}, C_{21} + ut_1 + lt_2) + p_{22}..$

For a given sequence π, the closed-form makespan expression for the weak hybrid model is as follows:

$$C_{N2} = \max\{(C_{K1} + tt_{1,2} + \sum_{i=K}^{N} p_{i2}), \max_{h=K+1,\ldots,N}\{C_{K1} + ut_1 + lt_2 + \sum_{i=K+1}^{h} p_{i1} + \sum_{i=h}^{N} p_{i2}\}\}.$$

where *K* is the job which uses direct model and the remaining (*N*-*K*) jobs use segregate model.

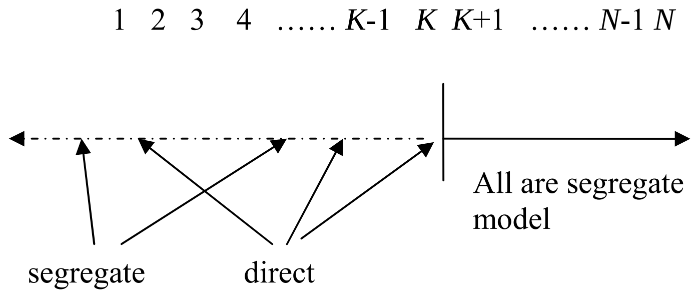

Figure 5.17: An illustration of the weak hybrid model

Case 1: If job *N* is scheduled via direct model, then

$C_{N2} = C_{N1} + tt_{1,2} + p_{N2}.$

Case 2: Otherwise, let K be the index number for which the jobs from K+1 to N, are scheduled using segregate model (as a result, the previous K-1 jobs do not affect makespan, see Figure 5.17). Then,

$$C_{K2} = C_{K1} + tt_{1,2} + p_{K2}.$$

$$\begin{aligned} C_{K+1,2} &= \max\{C_{K2}, C_{K+1,1} + ut_1 + lt_2\} + p_{K+1,2} \\ &= \max\{C_{K1} + tt_{1,2} + p_{K2} + p_{K+1,2}, C_K + p_{K+1,1} + ut_1 + lt_2 + p_{K+1,2}\}. \end{aligned}$$

By simple induction,

$$C_{N2} = \max\{(C_{K1} + tt_{1,2} + \sum_{i=K}^{N} p_{i2}), \max_{h=K+1,...N}\{C_{K1} + ut_1 + lt_2 + \sum_{i=K+1}^{h} p_{i1} + \sum_{i=h}^{N} p_{i2}\}\}.$$

Remark 5.2: For a given sequence π, the weak hybrid model is no worse than the segregate model.

This essentially follows from the very nature of the weak hybrid model. It will use direct model to save transportation time when there is an empty buffer space available on machine 2.

Remark 5.3: The weak hybrid model is not necessary better than direct model for a given sequence π.

This is due to the "push" feature of the weak hybrid model.

5.4.2.4 Strong hybrid model

Under the strong hybrid model, for each job to be scheduled, we have the choice of choosing either segregate model (S) or direct model (D). Thus, in addition to the job sequencing as in traditional flow shop, we also have to determine an AMHS operating model sequence (S-D sequence). Thus, an example of this combined sequence for 8 jobs is as follows: 3(S)-4(D)-1(D)-7(D)-6(D)-8(S)-5(D)-2(S).

Proposition 5.5: Let σ indicate a scheduled set of jobs, and job i be the next job scheduled. (1) If $C_{\sigma 1}+p_{i1}+tt_{1,2}\geq C_{\sigma 2}$, then D dominates S; (2) If $C_{\sigma 1}+p_{i1}+TT+tt_{1,2}\leq C_{\sigma 2}$, then S dominates D; (3) Otherwise, one does not dominate the other.

Proof: Let C_{i1}^{S}, C_{i2}^{S} denote the finished processing times for job i on machines 1 and 2, respectively, when job i is associated with S, and C_{i1}^{D}, C_{i2}^{D} be the finished processing times when assigned with D. We have

Case (1): if $C_{\sigma 1}+p_{i1}+tt_{1,2}\geq C_{\sigma 2}$

$C_{i1}^{S}=C_{\sigma 1}+p_{i1}$,

$C_{i2}^{S}=C_{\sigma 1}+p_{i1}+TT+tt_{1,2}+p_{i2}$.

$C_{i1}^{D}=C_{\sigma 2}+p_{i1}$,

$C_{i2}^{D}=C_{\sigma 1}+p_{i1}+tt_{1,2}+p_{i2}$.

Note that $C_{i1}^{S}=C_{i1}^{D}$ and $C_{i2}^{S}>C_{i2}^{D}$, which implies that D dominates S.

Case (2): if $C_{\sigma 1}+p_{i1}+TT+tt_{1,2}\leq C_{\sigma 2}$

$C_{i1}^{S}=C_{\sigma 1}+p_{i1}$,

$C_{i2}^{S}=C_{\sigma 2}+p_{i2}$.

$C_{i1}^{D}=C_{\sigma 2}-tt_{1,2}$,

$C_{i2}^{D}=C_{\sigma 2}+p_{i2}$.

Note that $C_{i1}^{S}<C_{i1}^{D}$ and $C_{i2}^{S}=C_{i2}^{D}$, which implies that S dominates D.

Case (3): if $C_{\sigma 1}+p_{i1}+tt_{1,2}<C_{\sigma 2}$ and $C_{\sigma 1}+p_{i1}+TT+tt_{1,2}>C_{\sigma 2}$

$C_{i1}^{S}=C_{\sigma 1}+p_{i1}$,

$C_{i2}^{S}=C_{\sigma 1}+p_{i1}+TT+tt_{1,2}+p_{i2}$.

$C_{i1}^{D} = C_{\sigma 2} - tt_{1,2}$,

$C_{i2}^{D} = C_{\sigma 2} + p_{i2}$.

Note that $C_{i1}^{S} < C_{i1}^{D}$ but $C_{i2}^{S} > C_{i2}^{D}$, thus one does not dominate the other. ◘

.

The weak hybrid model is simple, efficient, and has been widely applied in wafer fabs. However, strong hybrid model is better in performance than the weak hybrid model. Thus, we have included this model in our study. The proof of NP-hardness of the strong hybrid model with zero buffer remains an open question.

5.4.3 Limited buffer of size b

5.4.3.1 Segregate model

Similar to the case of zero-buffer, machine 2 is idle at the beginning, thus it is reasonable to use direct model for the first (*b*+1) jobs.

Proposition 5.6 When the first (*b*+1) jobs are processed using the direct model and the remaining (*N*-*b*-1) jobs are processed using the segregate model, the optimal solution is obtained by appending the schedule of the first (*b*+1) jobs and the remaining (*N*-*b*-1) jobs, both using Johnson's algorithm.

Proof: Similar to the proof for Proposition 5.3, we can derive the closed-form expression for the makespan as follows

$$\begin{aligned} C_{N2} &= \max\{C_{N-1,2}, C_{N1} + ut_1 + lt_2\} + p_{N2} \\ &= \max\{C_{11} + tt_{1,2} + p_{12} + p_{22} + p_{32} + \ldots + P_{N2}, \ldots\ldots\ldots, C_{b+1,1} + tt_{1,2} + p_{b+1,2} + p_{b+2,2} + \ldots \\ &\qquad + P_{N2}, C_{b+2,1} + ut_1 + lt_2 + p_{b+2,2} + p_{b+3,2} + \ldots + P_{N2}, \ldots\ldots, C_{N1} + ut_1 + lt_2 + P_{N2}\}. \end{aligned}$$

By re-arranging,

$$C_{N2} = \max\{[\max(C_{11} + tt_{1,2} + p_{12} + p_{22} + p_{32} + ... + P_{b+1,2}, ..., C_{b+1,1} + tt_{1,2} + p_{b+1,2}) + p_{b+2,2} + ... + P_{N2}], [C_{b+1,1} + \max(p_{b+2,1} + p_{b+3,1} + ut_1 + lt_2 + p_{b+3,2} + .. + P_{N2}, ..., p_{b+2,1} + ... + p_{N1} + ut_1 + lt_2 + P_{N2})]\}.$$

The first part of the above expression gives the critical path for the first (*b*+1) jobs for a 2-machine flow shop. And, the second part represents determination of the critical path for the remaining (*N*-*b*-1) jobs for a 2-machine flow shop. The item $C_{b+1,1}$ in the second part appends the second part to the first part. ◘

Thus, we have the following **Improve-Seg-b** algorithm for this case, which is similar to the **Improve-Seg-1** algorithm.

Algorithm Improve-Seg-b:

Step 1: Randomly select (*b*+1) jobs to be scheduled.
Step 2: Use direct model with transportation time delay of $tt_{1,2}$ and infinite buffer. Then, for the remaining (*N*-*b*-1) jobs, apply the segregate model with transportation time delay of $TT + tt_{1,2}$. Append both schedules together.
Step 3: Enumerate all possible C_N^{b+1} combinations of the first (*b*+1) jobs, and repeat step 2.
Step 4: Compare all schedules, and find the optimal schedule with minimum makespan.

5.4.3.2 Direct model

Proposition 5.7: The IMHLSP for the case of limited buffer size for the direct model is NP-hard.

Proof: Let $lt_1 = 0$, $ut_2 = 0$, and $tt_{1,2} = 0$. The resulting problem becomes a classical 2-machine flow shop problem with limited buffer size, which is a well-known NP-hard problem (Papadimitrious and Kanellakis, 1980). ◘

Papadimitrious and Kanellakis (1980) have proposed an efficient heuristic for this problem. The worst case performance of this heuristic is $\frac{2b+1}{b+1}$ (of the optimal makespan).

5.4.3.3 Strong hybrid model

Proposition 5.8: The IMHLSP for the case of limited buffer size for the strong hybrid model is NP-hard.

Proof: Constructing a special case where $TT \geq \sum_{i=1}^{N} P_{2i}$, segregate model will be dominated by direct model, which is NP-hard by the result of Proposition 5.7. ◘

Remark 5.4: The reversibility property, which is common for many flow shop scheduling problems, is not valid for the strong hybrid model.

Proof: Consider a flow shop with 2 machines, depicted as machines 1 and 2, and 2 jobs, namely, jobs m and n. Let the schedule be $\{m, n\}$ on machine1 followed by that on machine 2, and C_{ij} be the completion time for the ith job on the jth machine in the schedule.

$C_{11} = p_{m1}$.

$C_{21} = p_{m1} + p_{n1}$.

$C_{12} = p_{m1} + tt_{1,2} + p_{m2}$.

For the completion time on machine 2 for job 2, we have 3 cases:

Case 1: $p_{n1} \geq p_{m2}$

We have, $p_{m1} + p_{n1} + tt_{1,2} = C_{21} + tt_{1,2} \geq p_{m1} + tt_{1,2} + p_{m2} = C_{12}$.

Thus, $C_{22} = C_{21} + tt_{1,2} + p_{n2} = p_{m1} + p_{n1} + tt_{1,2} + p_{n2}$.

Case 2: $p_{n1} < p_{m2}$, and $p_{n1} + TT \geq p_{m2}$

We have, $p_{m1} + p_{n1} + tt_{1,2} = C_{21} + tt_{1,2} < p_{m1} + tt_{1,2} + p_{m2} = C_{12}$, and

$p_{m1} + p_{n1} + tt_{1,2} + TT = C_{21} + tt_{1,2} + TT \geq p_{m1} + tt_{1,2} + p_{m2} = C_{12}$.

Thus, $C_{22} = C_{21} + TT + tt_{1,2} + p_{n2} = p_{m1} + p_{n1} + TT + tt_{1,2} + p_{n2}$.

Case 3: $p_{n1} + TT < p_{m2}$

We have, $p_{m1} + p_{n1} + tt_{1,2} + TT = C_{21} + tt_{1,2} + TT < p_{m1} + tt_{1,2} + p_{m2} = C_{12}$.

Thus, $C_{22} = C_{12} + p_{n2} = p_{m1} + p_{m2} + tt_{1,2} + p_{n2}$.

Now, consider the reverse schedule {n, m} on machine 2 followed by that on machine 1.

$C_{11} = p_{n2}$.

$C_{21} = p_{n2} + p_{m2}$.

$C_{12} = p_{n2} + tt_{1,2} + p_{n1}$.

For the completion time on machine 2 for job 2, we have 3 cases:

Case 1: $p_{m2} \geq p_{n1}$

We have, $p_{n2} + p_{m2} + tt_{2,1} = C_{21} + tt_{2,1} \geq p_{n2} + tt_{2,1} + p_{n1} = C_{12}$.

Thus, $C_{22} = C_{21} + tt_{2,1} + p_{m1} = p_{n2} + p_{m2} + tt_{2,1} + p_{m1}$.

Case 2: $p_{m2} < p_{n1}$, and $p_{m2} + TT \geq p_{n1}$

We have, $p_{n2} + p_{m2} + tt_{2,1} + TT = C_{21} + tt_{2,1} + TT \geq p_{n2} + tt_{2,1} + p_{n1} = C_{12}$ and

$C_{21} + tt_{1,2} < C_{12}$.

Thus, $C_{22} = C_{21} + TT + tt_{2,1} + p_{m1} = p_{n2} + p_{m2} + TT + tt_{2,1} + p_{m1}$

Case 3: $p_{m2} + TT \leq p_{n1}$

We have, $p_{n2} + p_{m2} + tt_{2,1} + TT = C_{21} + tt_{2,1} + TT \le p_{n2} + tt_{2,1} + p_{n1} = C_{12}$.

Thus, $C_{22} = C_{12} + p_{m1} = p_{n2} + p_{n1} + tt_{2,1} + p_{m1}$.

Note that the expressions for C_{22} are not identical. ◘

We propose a branch-and-bound algorithm (designated as **Strong-B&B**) for this problem. Let σ, σ' be the scheduled and unscheduled sets of jobs. First, we show that the optimal makespan can be obtained by appending the optimal schedule of the unscheduled set σ' to the scheduled set σ. This result will also be used to derive lower and upper bounds.

In a 2-machine problem, if, for the unscheduled set σ', machine 2 has infinite buffer, then the optimal schedule is obtained by appending optimal schedule for the unscheduled set σ' to scheduled set σ.

Consider Figure 5.18, which shows scheduled set σ and unscheduled set σ'. The makespan expression is as follows.

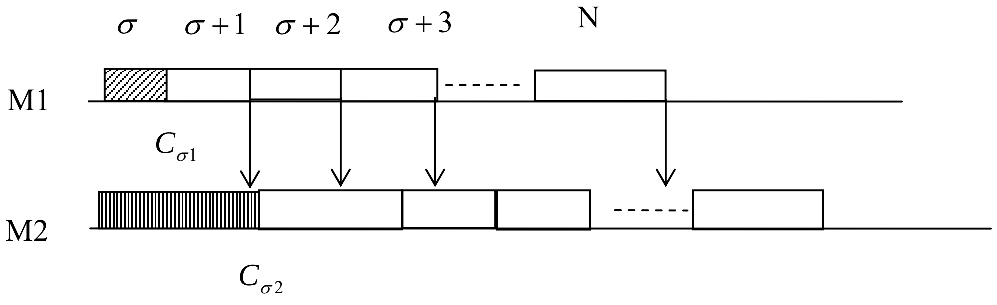

Figure 5.18: An illustration of the critical path for the unscheduled set

$$C_{N2} = \max\{(C_{\sigma 2} + p_{\sigma+1,2} + p_{\sigma+2,2} + \ldots + p_{N2}), C_{\sigma 1} + [\max(p_{\sigma+1,1} + p_{\sigma+2,,1} + p_{\sigma+3,1} + \ldots$$
$$+ p_{N1} + p_{N2}, p_{\sigma+1,1} + p_{\sigma+2,,1} + p_{\sigma+3,1} + \ldots + p_{N-1,1} + p_{N-1,2} + p_{N2}, \ldots.., p_{\sigma+1,1} +$$
$$p_{\sigma+1,,2} + p_{\sigma+2,2} + \ldots + p_{N-1,2} + p_{N2})]\}.$$

The second part is the determination of the critical path for the unscheduled set σ'. The item $C_{\sigma 1}$ stands for appending the unscheduled set σ' to scheduled set σ.

Corollary 5.3: By applying Johnson's algorithm with transportation delay of $tt_{1,2}$ for jobs σ', and attaching it to the schedule of σ, we obtain a lower bound of the makespan.

Proof: This follows by the fact that the optimal schedule for jobs σ', when machines have limited buffer, cannot be better than that for the case when they have infinite buffer, and, then apply the Proposition 5.2.

Corollary 5.4: Let b denote the current empty buffer space at machine 2. Apply Algorithm **Improve-Seg-b** with transportation delay of $TT + tt_{1,2}$ for σ' if $b > 0$. Attach the schedule to scheduled set σ. The makespan obtained is an upper bound (denoted by **UB1**).

Corollary 5.5: Apply the algorithm due to Papadimitrious and Kanellakis (1980) to the unscheduled set σ'. And then, attach the schedule obtained to the scheduled set σ. The makespan obtained is an upper bound (denoted by **UB2**).

Enumeration scheme

As shown in Figure 5.19, there are two possibilities at each node: *S* and *D*. This increases the number of nodes by an exponential factor (2^N) over those encountered in the classical flow shop problem.

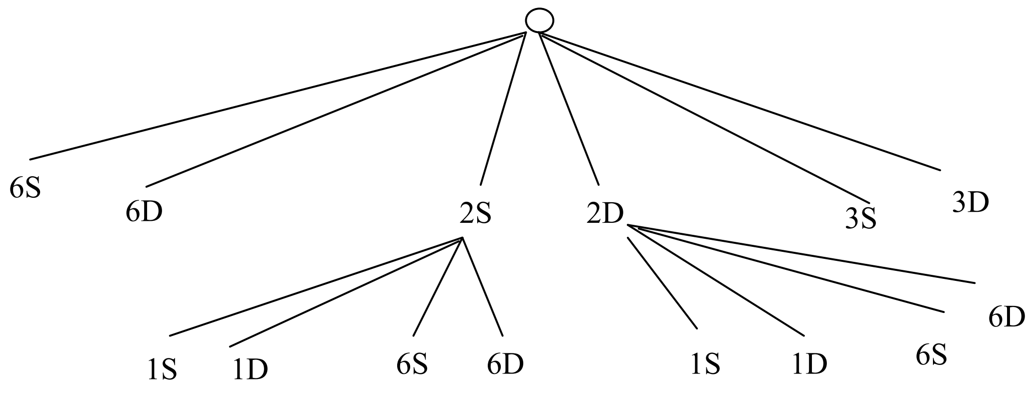

Figure 5.19: An illustration of node generation for the proposed Strong-B&B algorithm

Fathoming strategies

Dominance Property 1: For the scheduled set σ, but different sequences γ and θ, if $C_{\sigma 1}^{\gamma} \le C_{\sigma 1}^{\theta}$ and $C_{\sigma 2}^{\gamma} \le C_{\sigma 2}^{\theta}$, then sequence γ dominates sequence θ.

Dominance Property 2: The application of Proposition 5.5 at each new node constitutes another dominance property.

Dominance Property 3: If at node k, the number of empty buffers is *b* on machine 2, then for jobs *k*, …, *k*+(*b*-1), the sequence obtained by Johnson's algorithm with transportation delay of $tt_{1,2}$ is a dominance sequence.

Dominance Property 4 If both nodes *IS* and *ID* are not fathomed, and for their immediate child nodes *IS-JD* dominates *IS-JS* and *ID-JD* dominates *ID-JS*, then node *ID-JD* will be dominated by node *IS-JD* (see Figure 5.20).

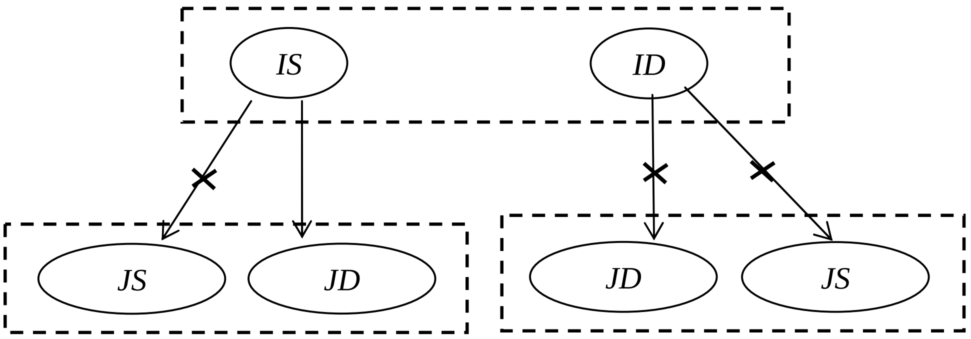

Figure 5.20: The case in which node ID-JD is fathomed

Proof: Let *D* and *S* represent direct model and segregate model respectively, *q* be either direct or segregate model, the scheduled set ahead of node *IS* and *ID* be σ, $C_{ij}^{Sq,1}$ and $C_{ij}^{Dq,2}$ represent the completion times for job *i* on machine *j* with models *S* and model *q* used for operations on machines *j*-1 and *j*, and with models *D* and model *q* used for operations on machines *j*-1 and *j*, respectively. By Proposition 5.5, if both *IS* and *ID* are not fathomed, then $C_{\sigma 1} + p_{i1} + tt_{1,2} < C_{\sigma 2}$ and $C_{\sigma 1} + p_{i1} + TT + tt_{1,2} > C_{\sigma 2}$. We have the following 3 cases:

Case (1): $p_{j1} \geq TT + p_{i2}$

Since at node *IS*, direct model dominates segregate model, we have

$C_{J1}^{SD,1} = C_{i1} + p_{j1} = C_{\sigma 1} + p_{i1} + p_{j1}$.

$C_{J2}^{SD,1} = C_{j1} + tt_{1,2} + p_{j2} = C_{\sigma 1} + p_{i1} + p_{j1} + tt_{1,2} + p_{j2}$.

Since at node *ID*, direct model dominates segregate model, we have

$C_{J1}^{DD,1} = C_{i1} + p_{j1} = C_{\sigma 2} - tt_{1,2} + p_{j1}$.

$C_{J2}^{DD,1} = C_{J1}^{DD} + tt_{1,2} + p_{j2} = C_{\sigma 2} + p_{j1} + p_{j2}$.

Thus, we know

$C_{J1}^{SD,1} = C_{\sigma 1} + p_{i1} + p_{j1} < C_{\sigma 2} - tt_{1,2} + p_{j1} = C_{J1}^{DD,2}$.

$C_{J2}^{SD,1} = C_{\sigma 1} + p_{i1} + p_{j1} + tt_{1,2} + p_{j2} < C_{\sigma 2} + p_{j1} + p_{j2} = C_{J2}^{DD,2}$.

Therefore, *IS-JD* node dominates *ID-JD* node.

Case (2): $p_{j1} > p_{i2}$ and $p_{j1} < p_{i2} + TT$

Case (3): $p_{j1} \leq p_{i2}$

We can try comparisons similar to those for cases (2) and (3) leading to no dominating node. ◘

Dominance property 5: If for a subset ϕ, we obtain an optimal schedule S^{ϕ} by applying direct model to ϕ and no job in ϕ waits on machine 1 after completing its processing, then the partial sequences which contain S^{ϕ} are not fathomed.

Proof: This follows by the fact that if direct model does not cause delay on machine 1, then it is optimal.

Dominance property 6: : If for a subset ϕ, we obtain an optimal schedule S^{ϕ} by applying the segregate model for ϕ and machine 2 does not wait due to unavailability of jobs, then the partial sequences, which contain S^{ϕ}, are not fathomed.

5.5 Numerical experimentation

In this section, we test the performance of **Strong-B&B** with the direct solution of the problem using the AMPL CPLEX Solver (version 10.1). The performance measure is cpu time required to find an optimal solution. The summary of the test data is contained in Table 5.3. We test eight problem sets with different number of lots, i.e., 10, 20, 30, 40, 50, 60, 70, and 80 lots. For each problem, the transportation times can be two possible values. The larger one is for interbay movements and the smaller one is for intrabay movements. Thus, there are a total of $8 \times 2 = 16$ data sets. For each data set, we generate 10 problem instances with randomly generated data.

Table 5.3: Data used in numerical experimentation[†]

Number of lots (*N*)		10, 20, 30, 40, 50, 60, 70, 80
Lot processing time ρ_{i1}, ρ_{i2} (mins/lot)		Uniform distribution [5,10]
Buffer size (*b*)		2
Transportation time *TT*	Interbay movements	10
	Intrabay movements	4
Transportation time tt_{12}	Interbay movements	5
	Intrabay movements	1
H		Sum of the processing times on machine 1 and 2 of all lots, $\sum_{i=1}^{N}(\rho_{i1} + \rho_{i2})$

(*† courtesy of a memory fab)*

We coded **Strong-B&B** using Excel VBA (version 2003). All numerical tests were done on a Dell computer with Pentium 4 processor (2.8GHz).

Table 5.4: The average cpu times required to find optimal solutions by Strong-B&B and the AMPL CPLEX 10.1 solver for strong hybrid model with intrabay movements

Data set	Number of Lots (N)	Average cpu time (seconds)	
		Strong-B&B	CPLEX 10.1 Solver
1	10	0.68	0.32
2	20	1.33	0.61
3	30	1.57	1.21
4	40	7.61	8.3
5	50	40.89	65.4
6	60	247.32	536.67
7	70	1418.63	3109.98
8	80	2923.57	>3600

The average cpu times required to find the optimal solutions for our algorithm and the AMPL CPLEX 10.1 Solver for the strong hybrid model with intrabay movements are presented in Table 5.4. It can be seen that for small-size problems, both **Strong-B&B** algorithm and CPLEX Solver can solve the problem optimally in a short time. When problem size increases, our algorithm is able to find the optimal solution faster than the CPLEX Solver. In fact, the CPLEX Solver cannot solve a problem involving the number of lots of more than 70 within the allowable cpu time of 3600 seconds. This is due to two reasons. The first one is the use of a large positive number H in the integer programming (IP) model for strong hybrid model, which impacts the tightness of the model. The second is that, when the number of lots increases, the number of binary variables X_{ik} and Seg_{kj} increases as well, which requires longer cpu time needed for the solver to find an optimal solution.

On the other hand, **Strong-B&B** algorithm uses modified Johnson's algorithm to determine a lower bound. It has been shown that for a two machine flow shop, the makespan derived by Johnson's algorithm is close to the optimal solution even for the case with limited buffer size. In addition, one of the upper bound is also determined based on Johnson's algorithm but with transportation time of $TT + tt_{12}$. Thus the maximum gap between the lower bound and the upper bound is TT, which is a small value compared to the processing times. Experimental results indicate that **Strong-B&B** can solve the problem with number of lots up to 80 within the allowable cpu time of 3600 seconds.

Table 5.5: The average cpu times required to find optimal solutions by Strong-B&B and the AMPL CPLEX 10.1 solver for strong hybrid model with interbay movements

Data set	Number of Lots (N)	Average cpu time (seconds)	
		Strong-B&B	CPLEX 10.1 Solver
1	10	0.73	0.39
2	20	5.17	3.72
3	30	22.5	19.4
4	40	387.61	448.92
5	50	1238.58	1834.7
6	60	2138.65	3308.74
7	70	3419.26	>3600
8	80	>3600	>3600

For interbay movements, i.e., a lot moved from a machine in a bay to its destination machine located in another bay across the central isle, the transportation time TT is much larger than that for the movement inside the same bay. Usually the transportation time TT can be as large as 10 minutes or more. Similarly, the value tt_{12} is longer since the machines are now located in different bays. The average cpu times required to find the optimal solutions for our algorithm and the AMPL CPLEX 10.1 Solver for the strong hybrid model with interbay movements are presented in Table 5.5. It can be seen that for the same problem size, to solve the strong hybrid model with interbay movement, the cpu times required by both **Strong-B&B** algorithm and CPLEX Solver are larger than those for intrabay movement. This is because when the values of TT and tt_{12} become larger, the related constraints become looser. In addition, the maximum gap between the lower bound and the upper bound also increases. In general, **Strong-B&B** solves the lager-size problems faster than the CPLEX Solver. The former can solve the problem with 70 lots within the allowable cpu time of 3600 seconds while the latter is only able to solve problems up to 60 lots within the time limit.

5.6 Extension to multiple-machine intrabay and the entire interbay/intrabay AMHS

When extended to multiple-machine (more than 2 machines) intrabay system, the total transportation delay for segregate model and direct models will be $(m-1)TT + \sum_{i=1}^{m-1} tt_{i,i+1}$ and $\sum_{i=1}^{m-1} tt_{i,i+1}$, respectively (see Figure 5.21). The advantages of direct model over segregate model will become more substantial.

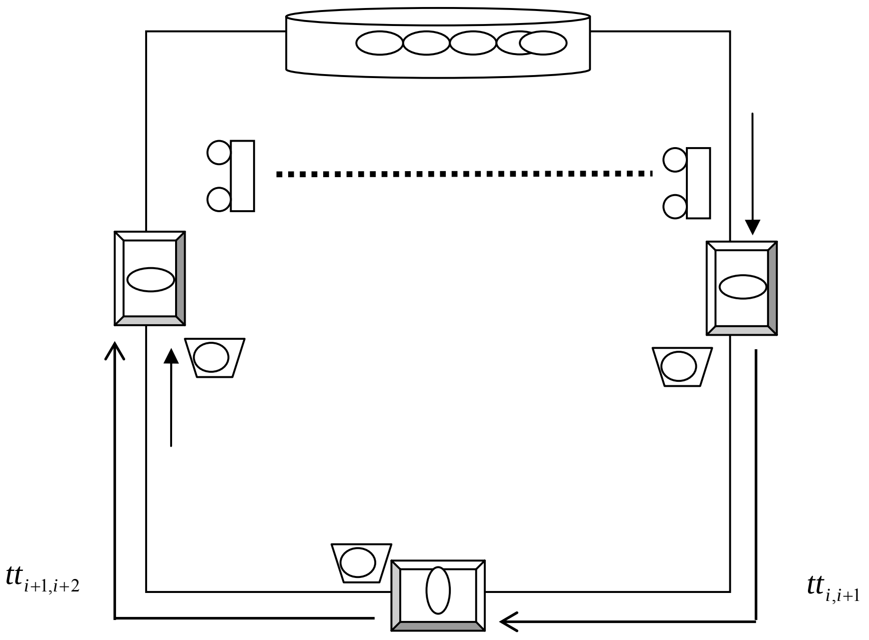

Figure 5.21: An illustration of the multiple-machine intrabay AMHS

Since we assume unlimited number of vehicles, the interbay transportation ends up adding time for transportation across bays. Thus, the entire interbay/intrabay AMHS can be assumed to be equivalent to a multiple-machine intrabay system with longer transportation delays. Note that, this is not the case for limited number of vehicles, which will be studied in the next chapter.

5.7 Conclusions

In this chapter, we have addressed the IMHLSP in the presence of infinite vehicle capacity. We, first, classified the AMHS operation models into four different types: the segregate model, the direct model, the weak hybrid model, and the strong hybrid model. We have proposed, for the first time, the strong hybrid model as a combination of the segregate and direct models is expected to lead to a global optimal schedule. The decisions involved in the strong hybrid model are the sequence in which to process the lots as well as the selection between segregate/direct model for each lot. We, then, formulate these models as integer programming models.

One difficult factor involved in this problem is the limited buffer size on the machines. Only limited studies have been presented in the literature on scheduling problems involving limited buffer size. We are able to show that under certain conditions about the processing times, the problem can be approximated by the case of either infinite buffer size or zero-buffer size. Hence, we consider all three cases of the IMHLSP in this chapter: infinite buffer, zero-buffer, and limited buffer size.

For the case with infinite buffer size, modified Johnson's algorithm is optimal for both the segregate model and the direct model. The hybrid models (both weak and strong) do not exist for this case. Since, for the segregate model, a lot having finished processing on a machine goes through the center buffer, which has unlimited storage size, the cases with zero-buffer and limited-buffer size are identical to that for infinite buffer. For the direct model, when the buffer size is zero, the problem can be solved by using Gilmore-Gomory's algorithm. However, when the buffer size is limited, the heuristic procedure due to Papadimitrious and Kanellakis (1987) is an efficient one to use.

For the strong hybrid model with limited buffer size (which is also applicable for the case with zero-buffer), we propose a branch-and-bound algorithm, designated **Strong-B&B**, which uses modified Johnson's algorithm to determine a lower bound. Two upper bounds are also determined, one using the procedure due to Papadimitrious and Kanellakis (1987) and the other using a modified Johnson's algorithm. Several dominance properties are also developed to fathom nodes that speed up the convergence of our algorithm.

Numerical tests indicate that **Strong-B&B** finds optimal solutions faster than the direct solution of the IMHLSP model using the AMPL CPLEX 10.1 Solver. The CPLEX Solver cannot solve the IMHLSP for problems containing number of lots greater than 70 within the allowable cpu time of 3600 seconds with intraby movements, and also, problems containing number of lots greater than 60 when interbay movements are considered. Experimental results also indicate that for the same problem size, the cpu times required to solve the IMHLSP model with interbay movements are larger than those with intrabay movements.

Chapter 6: Minimization of Makespan for Integrated Automated Material Handling and Lot Scheduling Problem (IMHLSP) with Finite Vehicle Capacity

6.1. Introduction

In this chapter, we study the integrated automated material handling and lot scheduling problem (IMHLSP) for 300mm fabs in the presence of vehicles with finite capacities. This problem is typically encountered in high volume 300mm fabs, especially those memory fabs where equipment utilization needs to be maximized. In the presence of limited number of vehicles, whenever a lot is finished, there may not be a vehicle immediately available to transport that lot. Ideally, one would want to put as many vehicles as possible in the fab. However, this will cause difficulties in space planning as well as layout and AMHS control. Thus, realistically, there are only a limited number of vehicles in a fab. When a lot is ready to move, it is possible that all vehicles are busy. Moreover, due to traffic problems, the vehicle assigned to transport a lot may not be able to arrive at the assigned location in time. Also, it is possible that the transportation demand for the AMHS may be underestimated at the design phase.

Unlike the IMHLSP with infinite vehicle capacity, this problem involves two types of decisions: permutation scheduling of jobs on the machines and sequencing of vehicle movements. However, models for handling the movements of jobs over the machines are still the same, namely segregate, direct, and hybrid (either weak hybrid or strong hybrid) models. The hybrid model becomes too complicated in the presence of vehicles with finite capacity. We, thus, consider only segregate and direct models.

The rest of this chapter is organized as follows. In Section 6.2, we formulate the IMHLSP as an integer programming model. The most commonly used heuristic procedures in 300mm fabs, also called RTD (real time dispatching) rules, are summarized in Section

6.3. In view of the complexity of the problem on hand, we study its one and two-machine instances in Sections 6.4 and 6.5, respectively. The interbay system is, then, considered in Section 6.6. Finally, we conclude our study in Section 6.7.

6.2. Mathematical Models for the IMHLSP with finite vehicles

We define makespan as the time at which the last lot is transported back to central stocker after having finished all operations. We capture this by defining a dummy operation as the last operation for each lot, which is to be performed at the central stocker. Its processing time is zero but it requires lot movement from the last machine to the stocker.

Parameters:

i : lot index, i=1, 2, …, N

j : operation index for each lot, j = 1, 2, …, J (note that operation J is a dummy operation)

l : intrabay vehicle index, l =1, 2, …, L

p_{ij} : processing time of job i to perform operation j

H : a large positive number,

$tt_{j_1 j_2}$: travel time between operations j_1 and j_2, j_1=0, 1, 2, …, J-1, j_2=0, 1, 2, …., J

k : position in a schedule of lots, k=1,2, …, N

Decision Variables:

Z : makespan,

S_{kj} : start time of operation j for the lot scheduled in position k,

T_{kj} : completion time of operation j for the lot scheduled in position k,

LT_{kj} : trip completion time for loading operation j of the lot scheduled in position k from its previous operation j-1

$$X_{ik} = \begin{cases} 1, & \text{if lot } i \text{ is scheduled in position } k \\ 0, & \text{otherwise} \end{cases}$$

$$W_{kjpq} = \begin{cases} 1, & \text{if a vehicle is assigned for loading operation } q \text{ of the lot scheduled in position } p \\ & \text{after having loaded operation } j \text{ of the lot scheduled in position } k \\ 0, & \text{othereise} \end{cases}$$

$$FW_{kj} = \begin{cases} 1, & \text{if a vehicle starts to load operaiton } j \text{ of the lot scheduled in position } k \text{ as} \\ & \text{its first assignment} \\ 0, & \text{othereise} \end{cases}$$

$$LW_{kj} = \begin{cases} 1, & \text{if a vehicle returns back to stocker after loading operaiton } j \text{ of the lot} \\ & \text{scheduled in position } k \text{ as its last assignment} \\ 0, & \text{othereise} \end{cases}$$

Direct model

Minimize Z

Subject to

$$Z \geq T_{IJ} \tag{6.1}$$

$$\sum_{i=1}^{N} X_{ik} = 1 \text{ for } k\text{=1, 2, .., } N \tag{6.2}$$

$$\sum_{k=1}^{N} X_{ik} = 1 \text{ for } i\text{=1, 2, …, } N \tag{6.3}$$

$$T_{kj} = S_{kj} + \sum_{i=1}^{N} X_{ik} p_{ij} \text{ for } k\text{=1, 2, …, } N, j\text{=1, 2, …, } J \tag{6.4}$$

$$S_{kj} \geq T_{k-1,j}, \text{ for } j\text{=1, 2, …, } J, k\text{=2, …, } N \tag{6.5}$$

$$FW_{kj} + \sum_{p=1}^{N}\sum_{q=1}^{J} W_{pqkj} = 1, \text{ for } p\text{=1, 2, …} N, q\text{=1, 2, …, } J, \text{ if } p \neq k, \text{ or for } q < j, \text{ if } p = k \tag{6.6}$$

$LW_{kj} + \sum_{p=1}^{N}\sum_{q=1}^{J} W_{kjpq} = 1$, for p=1, 2, …N, q=1, 2, …, J, if $p \neq k$, or for $q > j$, if $p = k$

(6.7)

$$\sum_{k=1}^{N}\sum_{j=1}^{J} FW_{kj} \leq L \tag{6.8}$$

$$\sum_{k=1}^{N}\sum_{j=1}^{J} FW_{kj} - \sum_{k=1}^{N}\sum_{j=1}^{J} LW_{kj} = 0 \tag{6.9}$$

$LT_{kj} - tt_{j-1,j} \geq FW_{kj} tt_{0,j-1} + \sum_{p=1}^{N}\sum_{q=1}^{J} W_{pqkj}(LT_{pq} + tt_{q,j-1})$, for p=1, 2, …N, q=1, 2, …, J,

if $p \neq k$, or for $q < j$, if $p = k$ (6.10)

$S_{kj} \geq LT_{kj}$, for k=1, 2, …, N, j=1, 2, …, J (6.11)

$LT_{kj} - tt_{j-1,j} \geq T_{k,j-1}$ for k=1, 2, …, N, j=1, 2, …, J (6.12)

$LT_{kj} \geq S_{k-b_{j+1}-1,j+1} - tt_{j,j+1}$, for $k > b_{j+1} + 1$, j=1, 2, …, J-2 (6.13)

X_{ik} binary, for k=1, 2, …, N, j=1, 2, …, N, $S_{kj}, LT_{kj}, T_{kj} \geq 0$, for k=1, 2, …, N, j=1, 2, .., J

(6.14)

Constraint sets (6.2) to (6.5) constitute the machine scheduling aspects of the problem. Constraint sets (6.2) and (6.3) are assignment constraints, and ensure that a lot is assigned to only one position of a schedule, and a position can accommodate only one lot. Constraint set (6.4) captures the fact that the completion time of an operation *j* scheduled in position *k* is equal to its start plus processing times. Similarly, constraint set (6.5) asserts that an operation *j* for the lot scheduled in position *k* cannot begin its processing unless operation *j* of the lot scheduled in position *k*-1 has been completed.

On the other hand, constraint sets (6.6) to (6.10) represent a generic vehicle scheduling problem (VSP). Constraint set (6.6) ensures that an operation *j* of a lot scheduled in position *k* is transported by a vehicle, which either starts with transporting that lot to operation *j* as its first assignment, or just finishes transporting a lot scheduled in position *p* to its operation *q*. Similarly, constraint set (6.7) implies that for a vehicle, after having finished transporting an operation *j* of a lot scheduled in position *k*, either returns back to the stocker (thus, this transportation is its last assignment), or it continues to transport a lot scheduled in position *p* for its operation *q*. Consequently, each operation of each job is transported by a vehicle only once. Constraint set (6.8) limits the number of vehicles to enter the system (a maximum value of *L*), and constraint set (6.9) enforces that the number of vehicles in the system do not change. Constraint set (6.10) states that the transportation start time by a vehicle for loading operation *j* of the lot scheduled in position *k* must be no earlier than the time that vehicle finishes its previous loading operation and moves to operation *j*-1 with an empty load in it.

The machine and vehicle scheduling aspects of the problem interact through constraint sets (6.11) to (6.13). Constraint (6.11) asserts that each operation can not start earlier than the completion time of its loading trip. Constraint set (6.12) ensures the start time of the loading trip for operation *j* of a lot scheduled in position *k* to be no earlier than completion time of operation *j*-1 of that lot. Limited buffer constraint is captured by constraint set (6.13). And finally, constraint set (6.1) determines the makespan.

Note that constraint set (6.10) is non-linear. We can transform it into linear constraints as follows.

$$LT_{kj} - tt_{j-1,j} \geq FW_{kj} tt_{0\,j-1} \tag{6.15}$$

$$LT_{kj} - tt_{j-1,j} - (LT_{pq} + t_{qj-1}) + H \geq H(W_{pqkj}) \tag{6.16}$$

Segregate model

The difference of this model from the direct model is that every time an operation is finished, the lot will be transported back to the stocker first before it is dispatched to its next destination tool. We can model this situation by adding one additional “operation” after each processing step. The “tool” for this “operation” is the central stocker. This “operation” consumes zero processing time, and the “tool” has unlimited buffer space. With these modifications, the integer programming model presented for the direct model above can be adapted for this case.

6.3 RTD (Real Time Dispatching) rules

The optimal makespan value can be found by solving a mathematical programming model for the situation in consideration. However, the presence of integer variables and a large positive number H, makes the model, even for a small size problem difficult to solve. On the other hand, RTD rules have been derived from daily fab experiences, are easy to use, and are, generally, quite effective. We give a brief overview of these rules next.

Table 6.1: The most common RTD rules used in 300mm fabs

Groups	Rules
MID	• FIFO (first in first out) • SRPT (shortest remaining processing time first) • CR (critical ratio) • EDD (earliest due date)
VID	• FEFS - first encountered first served • Lot with longest waiting time • Shortest travel distance to the lot • Longest travel distance to the lot • Sweep – the vehicle performs all possible deliveries in the current looping cycle • MaxWIP – first serving the job whose destination queue has the largest total number of jobs currently waiting • MinWIP – first serving the job whose destination queue has the least total number of jobs currently waiting
LID	• NV- nearest empty vehicle • Farthest vehicle • Longest idle vehicle • Least utilized vehicle

RTD rules, which are fast and easy to implement, have been widely used in executing daily fab operations (for a complete review, see Sarin et al. 2008). They can be classified into three categories: machine initiated dispatching (MID), vehicle initiated dispatching (VID) rules, and lot initiated dispatching (LID) rules. A MID rule is used to decide which lot to move from a stocker or a machine when there is an empty buffer available on a tool. A VID rule deals with the situation in which a vehicle has the choice of selecting among multiple FOUPs that are waiting for pickup at different locations. A LID rule deals with the problem of matching a task to a vehicle when multiple empty vehicles are idle and waiting for a task assignment. The most common RTD rules from literature are summarized in Table 6.1.

Usually, in the presence of heavy material flow rate, a VID rule becomes more significant because whenever a vehicle completes its transportation mission, it has to seek the next mission immediately. For a fab scheduling system, which does not include the AMHS, the RTD essentially implies MID. In the literature, there are extensive studies on MID, very few studies on VID, and no study focusing on LID. Furthermore, there has not been any study devoted to developing RTD rules based on the IMHLSP.

In this chapter, instead of solving the integer model or use RTD rules directly, we study a small-size intrabay system consisting of one or two machines and one or two vehicles, and present polynomial-time algorithms for their solution. Then, based on our results, we derive some RTD rules, which are expected to be promising for realistic-size systems.

The methodology that we use is as follows.

Step 1: Use an arbitrary lot sequence π.

Step 2: Check if an optimal vehicle movement sequence can be determined for a given lot sequence π. If not, consider all possible vehicle movement sequences.

Step 3: Derive the closed-form expression for the makespan for each vehicle movement sequence.

Step 4: Solve the closed-form expression, and determine the optimal lot sequence π^*.

Step 5: If there is only one vehicle movement sequence, then go to step 6. Otherwise, compare the makespan values for different vehicle movement sequences and find the optimal vehicle movement sequence and the corresponding lot sequence, π^*.

Step 6: Stop.

6.3 One-machine IMHLSP problem

Let lt indicate the travel time required to move one lot from the stocker to the machine, ut indicate the travel time required to move one lot from the machine to the stocker, T be the time needed for a vehicle to finish one loop of traveling, where $T = ut + lt$, p_i be the processing time for the lot scheduled in position *i*, T_i be the completion time of the operation scheduled in position *i*, and C_i be the time the lot scheduled in position *i* is transported back to central stocker.

We make the following assumptions: (1) The loaded travel speed of a vehicle is the same as its empty travel speed, i.e., the travel time is only related to the travel distance. (2) No-wait (NW) policy is employed, which means that a vehicle is not allowed to stop while in travel except at the central stocker, which acts as a parking space for vehicles.

Since we are considering only one machine, there is no difference between the direct and segregate models.

6.3.1 One-vehicle zero-buffer problem

Given the lot sequence, π, we can find the optimal vehicle movement sequence, indicated by sequence 1-1. A vehicle loads the first lot on the machine, and goes back to the stocker. A vehicle leaves the stocker to pick up the first job for the machine, carry it and unload it at the stocker. This cycle repeats until all the lots are finished.

For the first lot, we have

$T_1 = lt + \max(ut + lt, p_1)$.

$C_1 = lt + \max(ut + lt, p_1) + ut$.

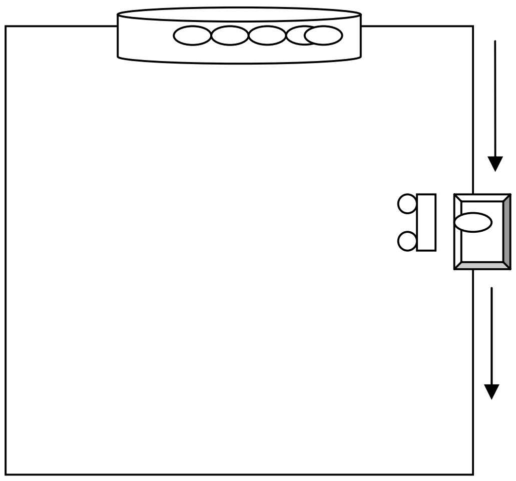

Figure 6.1: One-machine, one-vehicle zero-buffer problem

By repeating this N times for N lots, we have the makespan value for sequence 1-1, indicated by Z_1, as follows

$$Z_1 = C_N = N * (lt + ut) + \sum_{i=1}^{N} \max(ut + lt, p_i).$$

Therefore, any lot sequence π is optimal.

6.3.2 One-vehicle one-buffer problem

There are two possible vehicle movement sequences for this system. In one vehicle movement sequence, we do not fill the buffer, and it is the same as sequence 1-1. The second vehicle movement sequence, indicated by sequence 1-2, is to fill the buffer. For the given lot sequence, π, the vehicle loads the first lot at the machine and goes back to the stocker. Then, the vehicle transfers the second lot from the stocker to the buffer of the machine. Upon completion, the vehicle transfers the first lot from the machine to the stocker. Then, vehicle leaves the stocker and loads the third lot to the buffer at the machine. The vehicle, then, transfers the second lot to the stocker. This cycle repeats until all the lots are completed.

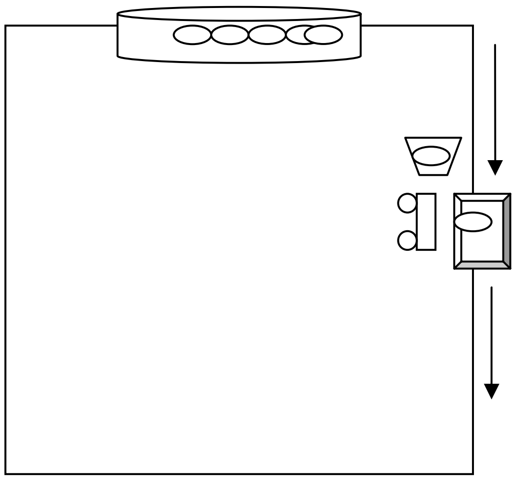

Figure 6.2: One-machine, one-vehicle one-buffer problem

We have,

$T_1 = lt + p_1$.

$C_1 = lt + \max(ut + lt, p_1) + ut$.

$T_2 = lt + \max(ut + lt, p_1) + p_2$.

$C_2 = lt + \max(ut + lt, p_1) + \max(ut + lt, p_2) + ut$.

$T_3 = lt + \max(ut + lt, p_1) + \max(ut + lt, p_2) + p_3$.

$C_3 = lt + \max(ut + lt, p_1) + \max(ut + lt, p_2) + \max(ut + lt, p_3) + ut$.

By induction, we have

$C_N = (lt + ut) + \sum_{i=1}^{N} \max(ut + lt, p_i)$.

Thus, the makespan for sequence 1-2, $Z_2 = (lt + ut) + \sum_{i=1}^{N} \max(ut + lt, p_i)$. This can also be written as $Z_2 = lt + \sum_{i \in K} p_i + \sum_{i \notin K} (ut + lt) + ut$, where $K = \{i : p_i \geq (ut + lt)\}$.

Note that $Z_1 - Z_2 = (N-1) * (ut + lt) > 0$. Therefore, the vehicle movement sequence 1-2 is optimal, and in sequence 1-2, lot sequence, π, can be arbitrary.

Proposition 6.1: The schedule, which fills one buffer is better than the one that does not fill the buffer.

This result is quite intuitive. With a buffer space, the machine can start to process another lot immediately after having moved a previous lot, thereby enhancing throughput rate.

Consequently, we can propose two VID rules based on this result.

- **EBF** (empty buffer first). A vehicle delivers a lot whose destination tool has an empty buffer space.
- **SQF** (shortest queue first). A vehicle delivers a lot whose destination tool has shortest queue at the buffer space.

6.3.3 One-vehicle two-buffer problem

There are three possible vehicle movement sequences for this system. Besides sequences 1-1 and 1-2, sequence 1-3 fills two buffers. It can be described as follows. For sequence 1-3, vehicle transfers the first lot to the machine and goes back to the stocker. It, then, transfers the second lot to the buffer of the machine, and goes back to the stocker and transfers the third lot to the buffer of the machine. Upon completion of the first lot, the vehicle transfers the first lot to the stocker. Then, the vehicle transfers the fourth lot from the stocker to the buffer of the machine. The vehicle transfers the second finished lot to the stocker. This cycle repeats until all the lots are finished.

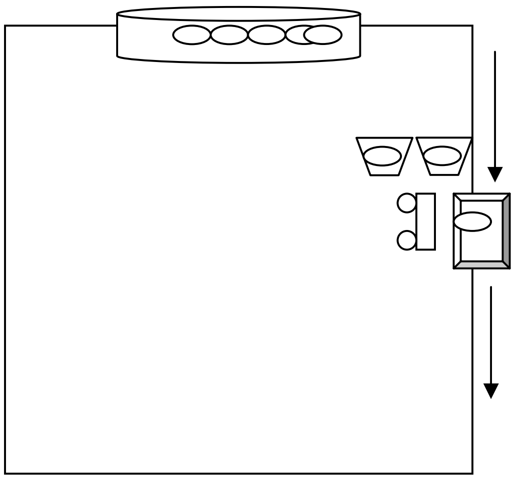

Figure 6.3: One-machine, one-vehicle two-buffer problem

We have,

$$C_1 = lt + \max(2(ut + lt), p_1) + ut.$$

$$C_2 = lt + \max(2(ut + lt), p_1) + \max(ut + lt, p_2) + ut.$$

$$C_3 = lt + \max(2(ut + lt), p_1) + \max(ut + lt, p_2) + \max(ut + lt, p_3) + ut.$$

$$C_4 = lt + \max(2(ut + lt), p_1) + \max(ut + lt, p_2) + \max(ut + lt, p_3) + \max(ut + lt, p_4) + ut.$$

By induction, we have makespan for sequence 1-3 indicated by Z_3, as follows

$$Z_3 = lt + \max(2(ut + lt), p_1) + \sum_{i=2}^{N} (\max(ut + lt, p_i)) + ut\,.$$

Note that only the first lot affects the makespan. The optimal lot sequence, π^*, can be found by enumerating the first lot in $O(N)$ time.

If we compare vehicle sequence 1-2 and 1-3,

$$Z_3 - Z_2 = \max(2(ut + lt), p_1) - \max(ut + lt, p_1) \geq 0\,.$$

Therefore, sequence 1-2 is optimal for this system.

Proposition 6.2: The filling of two buffers is not better than that of filling one buffer only.

This follows by the fact that only one vehicle is available to transport the finished jobs. The vehicle may become the bottleneck. If a lot is finished but vehicle has not arrived to unload it, the machine cannot start to process another lot in the buffer. Therefore, the extra buffer capacity cannot be utilized.

Thus, we can propose one LID rule.

- **NV** (nearest vehicle first). When a lot finishes processing at a machine, the nearest vehicle is assigned to move it out of the machine.

In fact, NV has turned out to be the most common LID rule implemented in 300mm fabs in literature (see Sarin et al. 2008)

6.3.4 Two-vehicle zero-buffer problem

In this system, given lot sequence π, the optimal vehicle movement sequence is for the first vehicle to transfer the first lot to the machine, and for the other vehicle to transfer the first lot to the stocker. This process repeats until all the lots are completed.

We have,

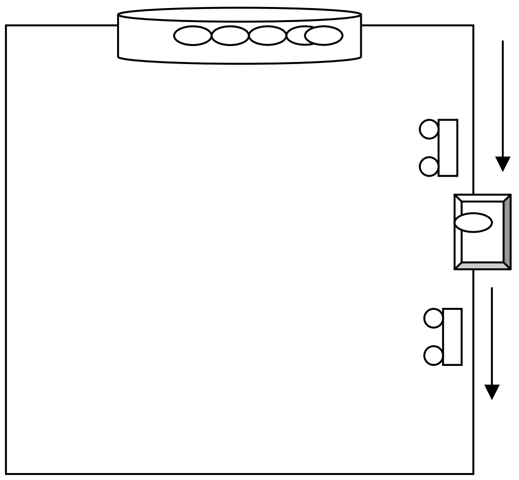

Figure 6.4: One-machine, two-vehicle zero-buffer problem

$T_1 = lt + p_1$.

$C_1 = T_1 + ut$.

$T_2 = lt + \max(ut + lt, p_1) + p_2$.

$C_2 = \max(C_1 + lt, T_2) + ut = \max(lt + p_1 + T, lt + T + p_2, lt + p_1 + p_2) + ut$

$T_3 = lt + \max(ut + lt, p_1) + \max(ut + lt, p_2) + p_3$.

$$\begin{aligned} C_3 &= \max(C_2 + lt, T_3) + ut \\ &= \max(3TT + p_1, 3T + p_2, p_1 + p_2 + 2T, \max(T, p_1) + \max(T, p_2) + p_3 + T). \end{aligned}$$

$T_4 = lt + \max(ut + lt, p_1) + \max(ut + lt, p_2) + \max(ut + lt, p_3) + p_4$.

$$\begin{aligned} C_4 = {} & \max(4T + p_1, 4T + p_2, p_1 + p_2 + 3T, \max(T, p_1) + \max(T, p_2) + p_3 + 2T, \\ & \max(T, p_1) + \max(T, p_2) + \max(T, p_3) + p_4 + T). \end{aligned}$$

$T_5 = lt + \max(ut + lt, p_1) + \max(ut + lt, p_2) + \max(ut + lt, p_3) + \max(ut + lt, p_4) + p_5$.

$C_5 = \max\{T_5, C_4 + lt\} + ut$.

By induction, we have

$$T_N = lt + \sum_{i=1}^{N-1} \max(T, p_i) + p_N.$$

$$C_N = \max\{T_N, C_{N-1} + lt\} + ut.$$

Thus, we can write the expression for the makespan, Z_4, as follows.

$$Z_4 = C_N = \max(N*T + p_1, N*T + p_2, p_1 + p_2 + (N-1)T, \\ \max_{K=2,3,\ldots,N-1}\{\sum_{i=1}^{K}\max(T, p_i) + p_{K+1} + (N-K)T\}).$$

Since $C_N = \max\{T_N, C_{N-1} + lt\} + ut \geq T_N + ut = T + \sum_{i=1}^{N-1}\max(T, p_i) + p_N$, we have a lower bound for Z_4 among all possible lot sequences π, as follows

$$LB_{Z_4} = \arg\max_{\mathrm{i}}\{T + p_j + \sum_{i \neq j}\max(T, p_i)\}.$$

Also, since $C_{N-1} \leq T + \sum_{i=1}^{N-1}\max(ut + lt, p_i)$, we have

$$Z_4 = C_N \leq \max\{T_N + ut, lt + \sum_{i=1}^{N-1}\max(T, p_i) + ut + T\} \\ = \max\{lt + \sum_{i=1}^{N-1}\max(T, p_i) + p_N + ut, \sum_{i=1}^{N-1}\max(T, p_i) + 2T\} = T + \sum_{i=1}^{N}\max\{T, p_i\}.$$

Thus, we find an upper bound $UB_{Z_4} = T + \sum_{i=1}^{N}\max\{T, p_i\}$. Note that

$$UB_{Z_4} - LB_{Z_4} = \arg\min_{i=1,\ldots,N}\{\max\{0, T - p_i\}\}.$$

Proposition 6.3: It is always better to use all vehicles, including additional vehicles.

Proof: This follows by the fact that $UB_{Z_4} = Z_2$, as the first vehicle transfers a new lot and performs the functions of a "mobile" buffer. ◘

Note that having additional vehicles enhances "mobile" buffer capacity, and hence, the system performance. But, in reality, it is impossible to put too many vehicles in the system as they may block the traffic and cause congestion and delays.

Based on the above, we can propose the following LID rule.

- **LUVF** (least utilized vehicle first). Select the least utilized vehicle for the lot movement.

6.3.5 Two-vehicle one-buffer problem

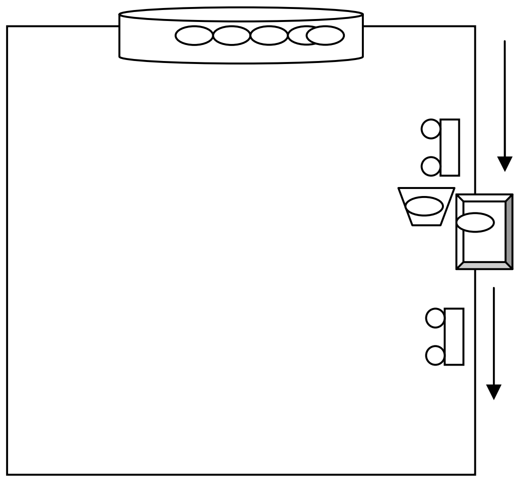

Figure 6.5: One-machine, two-vehicle one-buffer problem

Given a lot sequence, π, the optimal vehicle sequence is for the first vehicle to transfer lot 1 from the stocker to the machine, and for the other vehicle to transfer lot 2 from the stocker to the machine, and then, to transfer lot 1 from the machine to the stocker. Subsequently, the first vehicle transfers lot 3 from the stocker to the machine, and then, transfers lot 2 from the machine to the stocker. This cycle is repeated.

We have,

$T_1 = lt + p_1$.

$C_1 = T_1 + ut$.

$T_2 = lt + p_1 + p_2$.

$C_2 = lt + \max(T, p_1 + p_2) + ut$.

$T_3 = lt + \max(T, p_1 + p_2) + p_3 = T_2 - ut + p_3$.

$C_3 = \max(T_1 + lt, C_3) = \max(T_1 + lt, T_2 - ut + p_3) + ut$. $T_4 = T_3 - ut + p_4$.

$C_4 = \max(T_2 + lt, T_3 - ut + p_4) + ut$. $T_5 = T_4 - ut + p_5$.

$C_5 = \max(C_3 + lt, C_4 - ut + p_5) + ut$.

By induction, we have the makespan for this system, indicated by Z_5,

$$Z_5 = C_N = \max\{T_{N-2} + lt, T_{N-1} - ut + p_N) + ut.$$

For this problem, it is not possible to write a closed-form expression for the makespan. However, note that $Z_5 \le Z_4$, which indicates the same result as that in sub-section 6.3.3.

6.5 Two-machine integration problem

For the case with more than one machine, we can use two possible dispatching policies for the vehicle:

(1) **ES** (earliest start) policy. The vehicle is dispatched at the earliest available time if one movement is required.
(2) **LS** (latest start) policy. The vehicle is dispatched at the latest possible time to perform more than one movement. One example of this policy is the load-unload (L-U) movement, that is, transfer one lot to machine 1, and then, transfer another finished lot from machine 2 to the stocker during the same loop (see Figure 6.6).

Next, we analyze the case of two machines and one vehicle.

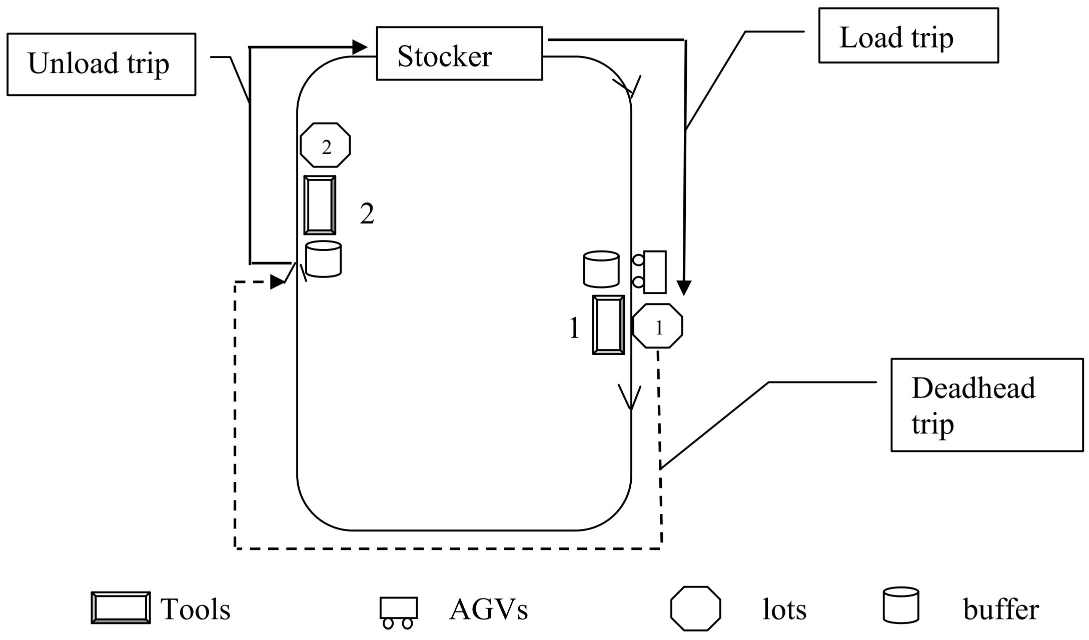

Figure 6.6: Illustration of the LS policy

Let lt_1 and lt_2 indicate the travel times from the stocker to machine 1 and machine 2, respectively, ut_1 and ut_2 be travel times from machine 1 and machine 2 to the stocker, respectively, tt represent the travel time from machine 1 to machine 2, T_{i1} and T_{i2} be the operation completion times on machines 1 and 2, respectively, for the lot scheduled in position *i*, and C_{i1} and C_{i2} be the times at which lot *i* is transported back to stocker after having finished operation at machines 1 and 2, respectively.

6.5.1 Zero-buffer problem

6.5.1.1 Segregate model

Given the lot sequence, π, the optimal vehicle movement sequence is not obvious, since, now, we have more than one vehicle. We need to consider different sequences and compare them.

Sequence 6-1: Finish all operations at machine 1 first, and then all operations at machine 2. That is, transfer a lot from the stocker to machine 1, and then transfer it back to the stocker after its completion. This is repeated until all the lots are processed at machine 1. Then, we do the same for processing the lots at machine 2.

The makespan indicted by $Z_{6,1}$ is as follows.

$$Z_{6,1} = \sum_{i=1}^{N}(lt_1 + \max(ut_1 + lt_1, p_{i1}) + ut_1) + \sum_{i=1}^{N}(lt_2 + \max(ut_2 + lt_2, p_{i2}) + ut_2).$$

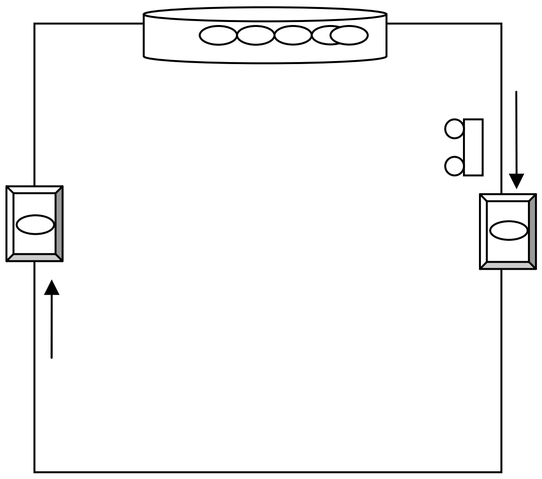

Figure 6.7: Two-machine, one-vehicle zero-buffer problem

Sequence ES-1: In this case, the vehicle loads job 11 (i.e. transfers lot 1 from the stocker to machine 1), goes back to the stocker, waits and unloads job 11. Then, it loads job 12, goes back to the stocker, and loads job 21, and then, goes back to the stocker. It waits and unloads job 12, and then, goes back to the stocker, waits and unloads job 21. This process repeats until all jobs are done.

Note that $T = ut_1 + lt_1 = ut_2 + lt_2$. We have

$T_{11} = lt_1 + p_{11}$.

$C_{11} = lt_1 + \max(ut_1 + lt_1, p_{11}) + ut_1$.

$T_{12} = T_{11} + lt_2 + p_{12}$.

$C_{12} = C_{11} + \max(lt_2 + ut_2 + lt_1 + ut_1 + lt_2, lt_2 + p_{12}) + ut_2$.

$T_{21} = C_{11} + T + lt_1 + p_{21}$.

$$\begin{aligned} C_{21} &= \max(C_{12} + lt_1, T_{21}) + ut_1 \\ &= \max(C_{11} + (T + lt_1 + p_{21} + ut_1), C_{11} + \max(2T + lt_2 + ut_2 + lt_1 + ut_1, lt_2 + p_{12} + ut_2 + lt_1 + ut_1)) \\ &= C_{11} + \max(2T + p_{21}, 4T, 2T + p_{12}). \end{aligned}$$

By induction, we have

$$C_{N1} = C_{11} + \sum_{i=2}^{N} \max(2T + p_{i1}, 4T, 2T + p_{i-1,2}).$$

$$C_{N2} = C_{N1} + \max(T + lt_2, lt_2 + p_{N2}) + ut_2.$$

Thus, the makespan for sequence ES-1, indicated by $Z_{6,ES-1}$, is as flows.

$$Z_{6,ES-1} = C_{N2} = 4T + \max\{T, p_{11}\} + \sum_{i=2}^{N} \max\{2T, p_{i1}, p_{i-1,2}\} + \max\{T, p_{N2}\}.$$

Proposition 6.5 Sequence 6-1 is worse than sequence *ES-1*.

Proof:

$$\begin{aligned} Z_{6,1} &= \sum_{i=1}^{N} (lt + \max(ut + lt, P_{i1}) + ut) + \sum_{i=1}^{N} (lt + \max(ut + lt, P_{i2}) + ut) \\ &= 2NT + \sum_{i=2}^{N} [\max\{T, p_{i1}\} + \max\{T, p_{i-1,2}\}] + \max\{T, p_{11}\} + \max\{T, p_{N2}\} \\ &= 2NT + \sum_{i=2}^{N} \max\{T + T, p_{i1} + T, p_{i-1,2} + T, p_{i1} + p_{i-1,2}\} + \max\{T, p_{11}\} + \max\{T, p_{N2}\} \\ &> 4T + \max\{T, p_{11}\} + \max\{T, p_{N2}\} + \sum_{i=2}^{N} \max\{2T, p_{i1}, p_{i-1,2}\} = Z_{6,ES-1}. \end{aligned}$$

We can propose a VID rule based on this result as follows.

- **LB** (line balancing). A vehicle delivers the lot in order to balance the line.

The closed-from expression for $Z_{6,ES-1}$ is given above for a lot sequence, π. Next, we want to find the optimal lot sequence π^*.

Proposition 6.6: For ES-1 policy, the optimal lot sequence π^* can be found in polynomial time.

Proof: First, we want to show that the two-machine zero-buffer IMHLSP with ES-1 policy is equivalent to a Traveling Salesman Problem (TSP) with $N+1$ cities.

Let the distance from city *i* to *j* be equal to

$d_{0i} = \max\{T, p_{i1}\}$.

$d_{j0} = \max\{T, p_{j2}\}$.

$d_{ij} = \max\{A_j, B_i\}$, for $i \neq 0$ and $j \neq 0$, where $A_j = \max\{2T, p_{j1}\}$ and $B_i = p_{i2}$.

We can rewrite the closed-from expression for $Z_{6,ES-1}$ as

$$Z_{6,ES-1} = 4T + \max\{T, p_{11}\} + \sum_{j=2}^{N} \max\{A_j, B_{j-1}\} + \max\{T, p_{N2}\}.$$

This TSP can be solved in $O(N^2)$ time by the algorithm proposed by Gilmore and Gomory (Gilmore and Gomory 1964). ◘

Sequence ES-2: This sequence differs from sequence ES-1 by the sequence of the following two movements: transferring of a lot at machine 1 to stocker and transferring of the next lot from the stocker to machine 1. The vehicle loads job 11, goes to the stocker, waits and unloads job 11. Then, it loads job 21, goes back to the stocker, and loads job 12, and goes back to the stocker. It waits and unloads job 21, and then, goes back stocker. It waits and unloads job 12. This process is repeated until all the jobs are processed.

We have

$T_{11} = lt_1 + p_{11}$.

$C_{11} = lt_1 + \max(ut + lt, p_{11}) + ut_1$.

$T_{21} = C_{11} + lt_1 + p_{21}$.

$C_{21} = \max(C_{11} + lt_1 + p_{21}, C_{11} + 2T + lt_1) + ut_1$.

$T_{12} = C_{11} + T + lt_2 + p_{12}$.

$$\begin{aligned} C_{12} &= \max(C_{11} + T + lt_2 + p_{12}, C_{21} + lt_2) + ut_2 \\ &= \max\{C_{11} + 2T + p_{12}, C_{11} + 2T + p_{21}, C_{11} + 4T). \end{aligned}$$

$T_{31} = C_{12} + lt_1 + p_{31}$.

$C_{31} = \max(C_{12} + lt_1 + p_{31}, C_{12} + 2T + lt_1) + ut_1$.

$T_{22} = C_{12} + T + lt_2 + p_{22}$.

$$\begin{aligned} C_{22} &= \max(C_{12} + T + lt_2 + p_{12}, C_{31} + lt_2) + ut_2 \\ &= \max\{C_{12} + 2T + p_{22}, C_{12} + 2T + p_{31}, C_{12} + 4T). \end{aligned}$$

By induction, we have

$$C_{N-1,2} = C_{11} + \sum_{i=2}^{N} \max(2T + p_{i-1,2}, 2T + p_{i1}, 4T).$$

$$C_{N2} = C_{N-1,2} + lt_2 + \max(ut_2 + lt_2, p_{N2}) + ut_2.$$

Note that $Z_{6,ES-2} = Z_{6,ES-1}$.

Sequence LS: This sequence differs from sequences ES-1 and ES-2 in the way that a vehicle does two transfers during travel in one loop. The vehicle loads job 11, goes back to the stocker, waits and unloads job 11. Then, it loads job 12, goes back to the stocker. It waits and loads job 21, and in the same cycle, unloads job 12, and then, goes back to the stocker. It waits and unloads job 21. This process is repeated until all the jobs are completed.

We have,

$T_{11} = lt_1 + p_{11}$.

$C_{11} = lt_1 + \max(ut + lt, p_{11}) + ut_1$.

$T_{12} = C_{11} + lt_2 + p_{12}$.

$$\begin{aligned} C_{12} &= C_{11} + \max(T + \Delta_{21} + lt_2, lt_2 + p_{12}) + ut_2 = C_{11} + \max(2T + \max(0, p_{12} - T), T + p_{12}) \\ &= C_{11} + \max(2T, T + p_{12}). \end{aligned}$$

$T_{21} = C_{11} + T + \Delta_{21} + lt_1 + p_{21}$ where $\Delta_{21} = \max(0, p_{12} - T)$.

$$\begin{aligned} C_{21} &= \max(C_{12} + lt_1, T_{21}) + ut_1 \\ &= \max(C_{11} + \max(2T, T + p_{12}) + lt_1, C_{11} + t + \max(0, p_{12} - T) + lt_1 + p_{21}) + ut_1 \\ &= C_{11} + \max(2T + T, 2T + p_{12}, 2T + p_{21}, p_{12} + T + p_{21}). \end{aligned}$$

By induction, we have

$$C_{N1} = C_{11} + \sum_{i=2}^{N} \max(3T, 2T + p_{i1}, T + p_{i-1,2} + p_{i1}, 2T + p_{i-1,2}).$$

$$C_{N2} = C_{N1} + \max(T + lt_2, lt_2 + p_{N2}) + ut_2.$$

The makespan indicated by $Z_{6,LS}$ is expressed as follows.

$$Z_{6,LS} = 3T + \max\{T, p_{11}\} + \sum_{i=2}^{N} \max\{2T, T + p_{i1}, T + p_{i-1,2}, p_{i1} + p_{i-1,2}\} + \max\{T, p_{N2}\}.$$

Note that

$$\begin{aligned} Z_{6,LS} &= 3T + \max\{T, p_{11}\} + \sum_{i=2}^{N} [\max\{T, p_{i1}\} + \max\{T, p_{i-1,2}\}] + \max\{T, p_{I2}\} \\ &= 3T + \sum_{i=1}^{N} \max\{T, p_{i1}\} + \sum_{i=1}^{N} \max\{T, p_{i2}\}. \end{aligned}$$

Thus, any lot sequence, π, is optimal for sequence LS.

Corollary 6.1: The two-machine zero-buffer IMHLSP for the segregate model can be solved in polynomial time.

Proof: The optimal solutions for the ES-1 and ES-2 policies can be determined in polynomial time by Proposition 6.6. And, for LS policy, the optimal lot sequence is arbitrary. Therefore, we can compare the optimal makespan values for the ES and LS policies and find the optimal solution in polynomial time. ◘

Corollary 6.2: For the same sequence, π, the LS policy is not necessarily better than the ES policy.

Based on the results above, we can propose two VID rules:

- **Sweep**. A vehicle tries to perform as many deliveries as possible in the same loop.
- **Touch**. A vehicle tries to perform as few deliveries as possible in the same loop.

6.5.1.2 Direct model

Next, we consider the direct model.

Sequence ES: The vehicle loads job 11, and goes back to the stocker. It waits, and then unloads 11, and in the same circle loads 12, goes back to the stocker. It waits, loads 21, goes back to the stocker, and waits. It unloads 12, and goes back to the stocker. It waits, unloads 21 and in the same circle loads 22, and goes back to the stocker. It waits and loads job 31, and goes back to the stocker and unloads 22. This process is repeated until all the lots are completed.

We have,

$T_{11} = lt_1 + p_{11}$.

$C_{11} = lt_1 + \max(ut_1 + lt_1, p_{11}) + tt_{1,2}$.

$T_{12} = C_{11} + p_{12}$.

$C_{12} = \max(C_{11} + 2T, C_{11} + p_{12}) + ut_2$.

$T_{21} = C_{11} + ut_2 + lt_1 + p_{21}$.

$$\begin{aligned} C_{21} &= \max(C_{12} + lt_1, T_{21}) + tt_{1,2} \\ &= \max(C_{11} + 2T + ut_2 + lt_1 + tt_{1,2}, C_{11} + p_{12} + ut_2 + lt_1 + tt_{1,2}, C_{11} + ut_2 + lt_1 + p_{21} + tt_{1,2}) \\ &= C_{11} + \max(3T, T + p_{12}, T + p_{21}). \end{aligned}$$

$T_{22} = C_{21} + p_{22}$.

$C_{22} = \max(C_{21} + 2T, C_{21} + p_{22}) + ut_2$.

$T_{31} = C_{21} + ut_2 + lt_1 + p_{31}$.

$$\begin{aligned} C_{31} &= \max(C_{22} + lt_1, T_{31}) + tt_{1,2} \\ &= \max(C_{21} + 2T + ut_2 + lt_1 + tt_{1,2}, C_{21} + p_{22} + ut_2 + lt_1 + tt_{1,2}, C_{21} + ut_2 + lt_1 + p_{31} + tt_{1,2}) \\ &= C_{21} + \max(3T, T + p_{22}, T + p_{31}). \end{aligned}$$

By induction, we have

$$C_{N1} = C_{11} + \sum_{i=2}^{N} \max(3T, T + p_{i1}, T + p_{i-1,2}).$$

$$C_{N2} = C_{N1} + \max(T, p_{N2}) + ut_2.$$

The makespan indicated by $Z_{7,ES}$ is expressed as follows:

$$Z_{7,ES} = C_{N2} = 2T + \max\{T, p_{11}\} + \sum_{i=2}^{N} \max\{2T, p_{i1}, p_{i-1,2}\} + \max\{T, p_{N2}\} + ut_2 .$$

Compared with $Z_{6,ES-1} = 4T + \max\{T, p_{11}\} + \sum_{i=2}^{N} \max\{2T, p_{i1}, p_{i-1,2}\} + \max\{T, p_{N2}\}$, we can see that $Z_{7,ES}$ has improved makespan in the amount of $2T - ut_2$. And, the optimal lot sequence, π^*, is the same as that for sequence ES-1.

Sequence LS: This sequence differs from sequence ES because of two movements in the same circle. The vehicle loads job 11, goes back to the stocker and waits, and then, unloads 11, and in the same circle loads 12. It goes back to the stocker and waits, and loads 21, and in the same circle unloads 12. It goes back to the stocker, waits, and unloads 21, and in the same circle loads 22. It goes back to the stocker, waits and loads 31, and in the same circle unloads 22. This process is repeated until all the jobs are completed.

We have,

$$T_{11} = lt_1 + p_{11}.$$

$$C_{11} = lt_1 + \max(ut + lt, p_{11}) + tt_{1,2}.$$

$$T_{12} = C_{11} + p_{12}.$$

$$\begin{aligned}C_{12} &= C_{11} + ut_2 + \Delta_{21} + lt_2 + ut_2 = C_{11} + T + \max(0, (p_{12} - T) + ut_2 \\ &= C_{11} + \max(T, p_{12}) + ut_2.\end{aligned}$$

$T_{21} = C_{11} + ut_2 + \Delta_{21} + lt_1 + p_{21}$, where $\Delta_{21} = \max(0, (p_{12} - (ut_2 + lt_1 + tt_{1,2})))$.

$$\begin{aligned}C_{21} &= \max(C_{12} + lt_1, T_{21}) + tt_{1,2} \\ &= \max(C_{11} + T + ut_2 + lt_1 + tt_{1,2}, C_{11} + p_{12} + ut_2 + lt_1 + tt_{1,2}, C_{11} + ut_2 + \Delta_{21} + lt_1 + p_{21} + tt_{1,2}) \\ &= C_{11} + \max(2T, T + p_{12}, \max(T, p_{12}) + p_{21} \\ &= C_{11} + \max(2T, T + p_{12}, T + p_{21}, p_{12} + p_{21}).\end{aligned}$$

By induction, we have

$$C_{N1} = C_{11} + \sum_{i=2}^{N} \max(2T, T + p_{i1}, T + p_{i-1,2}, p_{i-1,2} + p_{i1}).$$

$$C_{N2} = C_{N1} + \max(T, p_{N2}) + ut_2.$$

The makespan is

$$\begin{aligned}Z_{7,LS} &= T + \max\{T, p_{11}\} + \sum_{i=2}^{N} \max\{2T, T + p_{i1}, T + p_{i-1,2}, p_{i1} + p_{i-1,2}\} + \max\{T, p_{N2}\} + ut_2 \\ &= T + \sum_{i=1}^{N} \max\{T, p_{i1}\} + \sum_{i=1}^{N} \max\{T, p_{i2}\} + ut_2.\end{aligned}$$

Note that, any lot sequence, π, is optimal.

Compared with $Z_{6,LS} = 3T + \sum_{i=1}^{N} \max\{T, p_{i1}\} + \sum_{i=1}^{N} \max\{T, p_{i2}\}$, note that sequence LS has improved makespan value of $2T - ut_2$.

Corollary 6.3: The two-machine, zero-buffer IMHLSP for the direct model can be solved in polynomial time.

Corollary 6.4: For the two-machine zero-buffer IMHLSP, the direct model is better than the segregate model.

Proof: This follows by comparing the values of $Z_{6,ES-1}$, $Z_{7,ES}$, $Z_{6,LS}$, and $Z_{7,LS}$. ◘

This explains why SEMATCH has been consistently encouraging 300mm fabs to implement tool to tool delivery.

6.5.2 One-buffer problem

Since the segregate model is not expected to dominate the direct model, we will mainly focus on the direct model.

Proposition 6.7: The IMHLSP with two-machine and one-buffer is NP-hard.

Proof: One special case of this problem is when all types of travel times (lt_1, lt_2, $tt_{1,2}$, ut_1, ut_2) are zeroes. This special case is the two-machine flow shop scheduling problem with limited buffers, which is strongly NP-hard (Papadimitriou and Kanellakis 1980). ◘

In the presence of buffers in front of the machines, there are too many possible movement sequences for the vehicle to transport lots between the machines and the stocker. Thus, the method that we used before, i.e., deriving a closed-form expression of the optimal makespan for a given lot sequence, π, cannot be applied here. We just list a promising sequence that is based on the LS policy. It can serve as a heuristic, and the makespan value obtained from it can be considered as an upper bound

LS sequence: The vehicle loads job 11, and goes back to the stocker. It waits and loads job 21, unloads job 11 and loads job 12 in the same circle and goes back to the stocker. It waits and loads job 31, unloads job 21, loads job 22, and unloads job 12 in the same circle, and then goes back to the stocker. This is repeated until all the jobs are completed.

We have,

$T_{11} = lt_1 + p_{11}$.

$C_{11} = lt_1 + \max(T, p_{11}) + tt_{1,2}$.

$T_{21} = lt_1 + \max(T, p_{11}) + p_{21}$.

$C_{21} = C_{11} + ut_2 + \Delta_{21} + lt_2 + ut_2$ where $\Delta_{21} = \max(\max(0,(p_{21} - T), \max(0,(p_{12} - T))$.

$T_{21} = T_{11} + \max(T, p_{21}, p_{12}) + ut_2$.

$T_{12} = C_{11} + p_{12}$.

$C_{12} = C_{11} + ut_2 + \Delta_{21} + lt_2 + ut_2$.

$T_{31} = C_{21} - tt_{1,2} + p_{31}$.

$C_{31} = C_{21} + ut_2 + \Delta_{31} + lt_1 + tt_{1,2}$ where $\Delta_{31} = \max(\max(0,(p_{31} - T), \max(0,(p_{22} - T))$.

$C_{31} = C_{21} + \max(T, p_{31}, p_{22}) + ut_2$.

$T_{22} = C_{21} + p_{22}$.

$C_{22} = C_{21} + ut_2 + \Delta + lt_2 + ut_2$.

$T_{41} = C_{31} - tt_2 + p_{41}$.

By induction, we have

$$C_{N1} = C_{11} + \sum_{i=2}^{N} \max(T, p_{i1}, p_{i-1,2}) + ut_2 .$$

$$C_{N2} = C_{N1} + \max(T, p_{N2}) + ut_2 .$$

The makespan for sequence LS, indicated by $Z_{8,LS}$, is as follows.

$$Z_{8,LS} = T + \max\{T, p_{11}\} + \sum_{i=2}^{N} \max\{T, p_{i1}, p_{i-1,2}\} + \max\{T, p_{N2}\} + ut_2 .$$

If we let $A_j = \max\{T, p_{j1}\}$, we can transform the problem of determining the optimal lot sequence, π^*, for sequence LS to a special type of the TSP, which can be solved by the method of Gilmore and Gomory (1965) in polynomial time to obtain optimal π^*.

By comparing $Z_{7,ES}$, $Z_{7,LS}$ and $Z_{8,LS}$, note that filling of one buffer is better than filling of no buffer.

6.6 General AMHS system

This pertains to the entire fab involving multiple bays and vehicles.

6.6.1 Direct model

The implementation of this model for general AMHS is identical to that for an intrabay system except for the transportation of a lot from a machine of a bay to a machine of another bay. As a result, the lot is transported between stockers in addition to its movement within bays, and it results in longer transportation times.

6.6.2 Segregate model

In the interbay system, vehicles loop through the stockers and move lots from one stocker to another stocker. A simple interbay AMHS is shown in Figure 6.8. There are four stockers which are located at the head of four bays: Etch, Litho, CMP, and Wets. The travel times between any two stockers are also indicated.

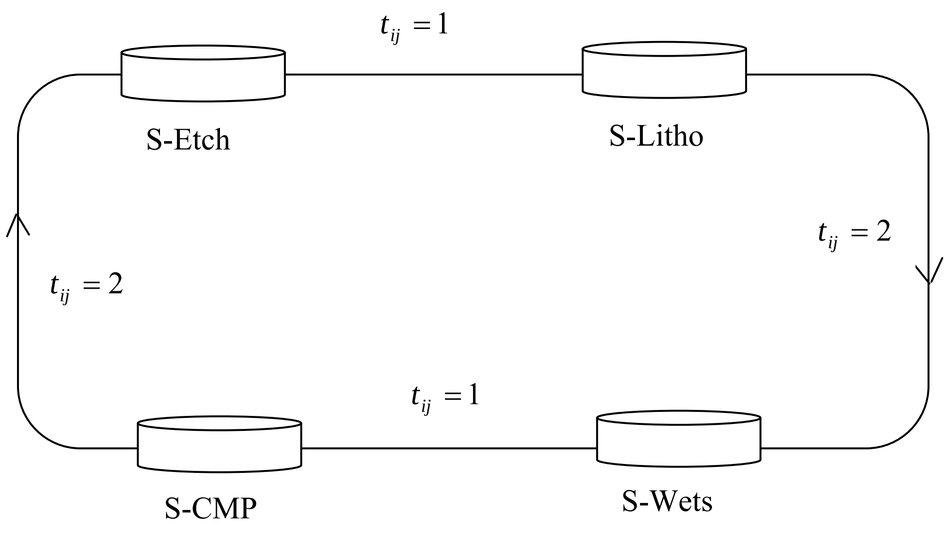

Figure 6.8: A simple interbay AMHS system

The travel time t_{ij} matrix is listed in Table 6.2.

Table 6.2: The travel time t_{ij} matrix

	S-Etch	S-Litho	S-CMP	S-Wets
S-Etch	NA	1	4	3
S-Litho	5	NA	3	2
S-CMP	2	3	NA	5
S-Wets	3	4	1	NA

Remark 6.1: The interbay movement part of the IMHLSP problem is a single-stage scheduling problem with parallel machines (corresponding to vehicles) and sequence-dependent setup times (corresponding to the deadhead travel times). It is also equivalent to the asymmetric traveling salesman problem (ATSP) with multiple salesmen.

Each vehicle is equivalent to a machine. Each lot movement can be considered as a job and the deadhead trip time is setup time. For example (refering to Figure 6.8), a vehicle moves one lot from S-Etch to S-CMP (job 1), and then, moves one lot from S-Litho to S-CMP (job 2). In the equivalent single-stage scheduling problem, Job 1 processing time is 4 and job 2 processing time is 3. The setup time between job 1 and job 2 is 1.5. However, if the sequence of jobs is reversed, the setup time becomes 1.

The presence of multiple vehicles makes it a multiple asymmetric traveling salesman problem.

The multiple ATSP is a very difficult problem to solve. However, there are some heuristic procedures, which have proved to be effective for the solution of this problem. The followings are some VID rules that we have developed based on these ideas.

- **Shortest Total Travel Time (STTT).** Each vehicle starts from its current location, selects the next lot with the smallest sum of loaded travel time and deadhead trip time. If the same lot is selected by two vehicles, the tie is broken based on the sum of loaded travel times and empty travel times.

- **Shortest Total Travel Time with Look-ahead (STTT with Look-ahead).** A vehicle starts from its current location, finds the next 2 consecutive lots which have smallest sum of loaded travel times and empty travel times to finish these two lot movements. Both lots are selected by this vehicle. If the same lot is selected by two vehicles, the tie is broken by the sum of loaded travel times and empty travel times.

Currently, in 300mm fabs, one of the most widely used VID rule is the shortest distance to a lot. In interbay movement, note that STTT rule is a good heuristic, which selects the lot with the smallest sum of deadhead trip time and loaded travel time. The shortest distance to lot rule, which selects a lot with shortest deadhead trip time, may not perform well.

A summary of our recommended RTD rules and the RTD rules used in 300mm fabs is given in Table 6.3.

Table 6.3: Our recommended RTD rules and the RTD rules used in 300mm fabs

RTD rules	Our recommended rules	Existing rules in 300mm fabs
FEFS (first encountered first served)		√
Lot with longest waiting time		√
Shortest distance to a lot		√
Longest distance to a lot		√
MaxWIP		√
MinWIP	√	√
EBF	√	
Sweep	√	√
Touch	√	
LB	√	
NV- nearest empty vehicle	√	√
Farthest vehicle		√
Longest idle vehicle		√
Least utilized vehicle	√	√
STTT	√	
STTT with look-ahead	√	

6.7 Conclusions

In this chapter, we have addressed the IMHLSP in the presence of limited number of vehicles. We, first, formulate a linear integer model for this problem. Due to the complexity of the underlying problem, we do not solve the model directly. Instead, we analyze small-size versions of this problem and develop algorithms for their solutions. These pertains to systems with one to two machines and one to two vehicles. For some of these problems, we can find optimal solutions in polynomial time. The two-machine one-buffer problem has been shown to be NP-hard. However, the optimal solution for the two-machine zero-buffer system can be used as a good heuristic for this problem.

On the other hand, in real-life fabs, RTD (real time dispatching) rules have been widely implemented. RTD rules can be classified into three categories: MID (machine initialized dispatching), VID (vehicle initialized dispatching) and LID (lot initialized dispatching) rules. Based on our analysis on small-size systems, we have shown that some RTD rules used in real fabs are expected to perform well while not the others. In addition, we also propose some new promising RTD rules based on our study.

Furthermore, we also show that the direct model is expected to dominate the segregate model. This result reinforces the recommendations made by SEMATECH and other organizations.

Chapter 7: Conclusions and Future Research

7.1 Conclusions

In this dissertation, we have addressed two types of new scheduling problems encountered in the latest semiconductor manufacturing 300mm fabs: multiple-lots-per-carrier scheduling problem (MLCSP) and integrated automated material handling and lot scheduling problem (IMHLSP). We have further classified the MLCSP into four different categories depending on the number of machines used, the processing technology of the machines, and the type of objective functions used. For IMHLSP, we study two cases, one with infinite number of vehicles and the other with finite number of vehicles.

In Chapter 2, we have addressed a single-machine, multiple-lot-per-carrier with single-wafer-processing-technology scheduling problem for the objective of minimizing the total completion time (MLCSP1). We have first formulated this problem as an integer programming model. Due to the difficulty of solving this model directly by using an optimization software, we analyze variations of this problem in order to determine some inherent structural properties. To that end, we, first, study the problem by relaxing the carrier (FOUP) capacity. For this problem, when all the lots are of the same size, the optimal solution is easy to obtain. However, for the case of different lot sizes, we propose a dynamic programming-based algorithm, **RelaxFOUP-DP.** For the problem with limited FOUP capacity, when all the lots are of the same size, we show that the optimal solution is the same as that for its relaxed version. When the lots are of different sizes, a branch-and-bound algorithm, **MLCSP1-B&B**, is developed that relies on the **RelaxFOUP-DP** algorithm**.** Numerical tests indicate that **MLCSP1-B&B** finds optimal solutions much faster than the direct solution of the MLCSP1 model by the AMPL CPLEX 10.1 Solver. The CPLEX Solver cannot solve the problems involving 15 and higher number of lots within the allowable cpu time of 14,400 seconds. In particular, our computational results show that, with an increment in the number of carriers, *L*, the cpu time required by the CPLEX Solver increases sharply while that required by **MLCSP1-B&B** for these instances falls dramatically. In fact, for the medium and low density problems, the **RelaxFOUP-DP** algorithm finds the optimal solutions at the starting node (node zero) itself, whereas the AMPL CPLEX Solver is unable to solve these problem instances within the allowable cpu time of 14,400 seconds.

Next, in Chapter 3, we consider a single-machine, multiple-lots-per-carrier with single-carrier-processing-technology scheduling problem for the objective of minimizing total completion time (MLCSP2). We have, first, formulated this problem as a generalized assignment problem (GAP) instead of an integer programming model using a big number *M* as shown in the literature. We, then, analyze the problem and determine some inherent properties. Based on our results, the optimal solution for the case in which all the lots are of the same size can be obtained easily. For the case of different lot sizes, we determine a lower bound and an upper bound for the problem. It is shown that the worst-case of the lower bound is 0.5 of the optimal solution and that of the upper bound is 2 times the optimal solution.

In Chapter 4, another MLCSP, a two-machine flow shop, multiple-lots-per-carrier with single-wafer-processing-technology scheduling problem for the objective of minimizing the makespan (designated as MLCSP3) is addressed. We, first, consider a relaxation of this problem, and transform the original problem to a two-machine flow shop lot streaming problem. For the lot streaming problem for lots with the same ratio of processing times on the machines, we propose the **Sam-Cap** algorithm to find the optimal capacitated sublot sizes. When lots involve different ratios of processing times, the **Dif-Cap** algorithm is developed to find optimal capacitated sublot sizes. Since the optimal solutions obtained for the lot streaming problem may not be feasible to the MLCSP3, we develop four heuristic methods based on the heuristic procedures for the bin packing problem, namely, **Sam-FFI, Sam-FFD, Sam-BFI**, and **Sam-BFD** for lots with the same ratio of processing times, and another four procedures, namely, **Dif-FF1, Dif-FF2, Dif-BF1**, and **Dif-BF2** for lots with different ratios of processing times. Numerical tests indicate that our heuristic procedures are able to generate solutions closer to optimum. The gap is less than 4% for **Sam-FFI, Sam-FFD, Sam-BFI**, and **Sam-BFD** and less than 5% for **Dif-FF1, Dif-FF2, Dif-BF1**, and **Dif-BF2.** In addition, the heuristic procedures find solutions in few seconds while the CPLEX Solver cannot solve the problems consisting of lots greater than 25 within 3600 seconds of cpu time. For problems of larger sizes, the performance measure that we use is the gap between the LB and the solutions generated by our heuristic procedures. The results indicate this gap to be within 5% for lots with the same ratio of processing times and within 6% for lots with different ratios of processing times.

In Chapter 5, we have addressed the IMHLSP in the presence of infinite number of vehicles. We have, first, classified the AMHS operation model into four different types: the segregate model, the direct model, the weak hybrid model, and the strong hybrid model. The strong hybrid model is new and being a combination of the segregate and direct models, is expected to lead to a global optimal schedule. The decisions involved in the strong hybrid model are the sequence of the lots as well as a selection between the use of segregate or direct models for each lot, whichever optimizes system performance. One difficult factor involved in this problem is the limited buffer size on the machines. There are only limited research results available in the literature on scheduling problem with a limited buffer size. We show that, under certain conditions about the processing times, the problem can be approximated by the case of either infinite buffer size or zero-buffer size. Hence, we consider all three cases of the IMHLSP in this chapter: infinite buffer, zero-buffer, and limited buffer size. For the case of infinite buffer size, modified Johnson's algorithm is optimal for both the segregate model and the direct model. The hybrid models (both weak and strong) do not exist for this case. Since, for the segregate model, a lot, after having finished processing on a machine, goes through the center buffer, which has unlimited storage size, the cases with zero-buffer and limited buffer size are identical to that for infinite buffer. For the direct model, when the buffer size is zero, the algorithm by Gilmore-Gomory (1958) solves the problem easily. However, when the buffer size is limited, a heuristic procedure by Papadimitrious and Kanellakis (1987) algorithm is an effective one for this problem.

For the strong hybrid model with limited buffer size (which is also applicable to the case of zero-buffer), we propose a branch-and-bound algorithm, **Strong-B&B**, which uses modified Johnson's algorithm to determine a lower bound. Two upper bounds for **Strong-B&B** are also determined, one using the procedure of Papadimitrious and Kanellakis (1987) and the other using the modified Johnson's algorithm. Numerical tests indicate that **Strong-B&B** finds optimal solutions faster than the direct solution of the IMHLSP model by the AMPL CPLEX 10.1 Solver. The CPLEX Solver cannot solve problems containing number of lots of 70 and higher within the allowable cpu time of 3600 seconds for the IMHLSP model with intraby movements, and problems containing number of lots of 60 and higher when interbay movements are considered. Experimental results also indicate that for the same problem size, the cpu times required to solve the IMHLSP model with interbay movements are larger than those with intrabay movements.

In Chapter 6, we have addressed the IMHLSP in the presence of limited number of vehicles. We, first, formulate a linear integer model for this problem. This method is difficult to solve to optimality by using an optimization solver. Therefore, we analyze small-size versions of this problem and develop algorithms for their solutions. These pertains to systems with one to two machines and one to two vehicles. For some of these problems, we can find optimal solutions in polynomial time. The two-machine one-buffer problem has been shown to be NP-hard. However, the optimal solution for the two-machine zero-buffer system can be used as a good heuristic for this problem. On the other hand, in real fabs, RTD (real time dispatching) rules have been widely implemented. RTD rules can be classified into three categories: MID (machine initialized dispatching), VID (vehicle initialized dispatching) and LID (lot initialized dispatching) rules. Based on our analysis on small-size systems, we have shown that some RTD rules used in real fabs are expected to perform well while not the others. In addition, we also propose some new and promising RTD rules based on our study.

7.2 Future research

In reality, different lots may have different priorities. Thus, one future research direction is to study the MLCSP1 with objective of minimizing total weighted completion time. Note that we have covered a part of this topic in Chapter 2 when we determine the third lower bound by using a relaxed problem. However, for the relaxed problem, all the lots have the same size, which means that it is a special case of the problem. Similarly, the objective of minimizing the total weighted completion time can also be applied to problems MLCSP2, MLCSP3 and MLCSP4.

The optimal solution of the direct model chooses either segregate model or direct model when a lot finishes processing on a machine. This is similar to the scheduling problem with alternative routes, which is a NP-hard problem (Baker 1972). It is conjectured that the direct model with zero-buffer is also a NP-hard problem. The proof remains open in our study.

It appears that the most difficult factor involved in the IMHLSP, both for the cases of infinite and finite number of vehicles, is the limited buffer size. In this study, our methodologies include a branch-and-bound algorithm and derivation of closed-form expressions for small-size problems. They are efficient for small-size problems (mostly two-machine flow shops). When problem size becomes larger (containing multiple machines or multiple vehicles), the strategy needs to be changed. Hence, for future research, we recommend investigation of different mathematical programming techniques for the IMHLSP.

For the MLCSP, in this study, we consider a small-size version of this problem (involving one to two machines). In real-life, a fab has hundreds of machines. The algorithms proposed in this research can be applied to large-size problems with some necessary modifications. Case studies involving the applications of proposed methodologies in industry will further enhance the value of this research.

References

1. M. F. Anwar and R. Nagi. 1998. Integrated scheduling of material handling and manufacturing activities for just-in-time production of complex assemblies. *International Journal of Production Research,* vol.36, no.3, pp.653-681.
2. M. Azizoglu and S. Webster. 2001. Scheduling a batch processing machine with incompatible job families. *Computers & Industrial Engineering*, vol.39, pp.325-335.
3. M. Azizoglu and S. Webster. 2000. Scheduling a batch processing machine with non-identical job sizes. *International Journal of Production Research*, vol.38, no.10, pp.2173-2184.
4. K. R. Baker. 1974. *Introduction to Sequencing and Scheduling*. John Wiley & Sons.
5. U. Bilge and G. Ulusoy. 1995. A time window approach to simultaneous scheduling of machines and material handling system in an FMS. *Operations Research,* vol.43, no.6, pp.1058-1070.
6. G. Cardarelli and P. M. Pelagagge, 1995. Simulation tool for design and management optimization of automated interbay material handling and storage systems for large wafer fab. *IEEE Transactions on Semiconductor Manufacturing*, vol.8, no.1, pp.44-49.
7. G. Cardarelli, P. M. Pelagagge and A. Granito, 1996. Performance analysis of automated interbay material handling and storage systems for large wafer fab. *Robotics & Computer-Integrated Manufacturing*, vol.12, no.3, pp.227-234.
8. W. Carrlker. 2004. Intel 300 mm fab layout and material handling automation integration learning. *2004 IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, pp.257-261.
9. V. D. Carvalho. 1999. Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, vol.86, pp.629–659.
10. J. Chrisos and J. N. Patt. 1998. Integraiton risks in 300 mm wfer fab automation. *Solid State Technology*, vol.41, no.7, pp.193-196.
11. C. M. Christensen, S. King, M. Verlinden, and W. Yang. 2008. The New Economics of Semiconductor Manufacturing. *IEEE Spectrum*, pp.25-29.

12. E. G. Coffman, M. R. Garey, and D. S. Johnson. 1996. Approximation Algorithms for Bin Packing: A survey. *Approximation algorithms for NP-hard problems*, pp.46-93. PWS Publishing Co., Boston, MA, USA.
13. P. Damodaran, P. K. Manjeshwar, and K. Srihari. 2006. Minimizing makespan on a batch-processing machine with non-identical jobs sizes using genetic algorithms. *International Journal of Production Economics*, vol.103, pp.882-891.
14. G. Dobson and R. S. Nambimadom. 2001. The batch loading and scheduling problem. *Operations Research*, vol.49, no.1, pp.52-65.
15. V. Erramilli and S. J. Mason. 2006. Multiple Orders per Job Compatible Batch Scheduling. *IEEE Transactions on Electronics Packaging Manufacturing*, vol.29, no.4, pp.285 – 296.
16. International SEMATCH. 2007. *Factory Integration, International Technology Roadmap for Semiconductors: 2006 Update*. SEMATECH Inc.
17. E. Falkenauer. 1996. A hybrid grouping Genetic Algorithm for bin packing. *Journal of Heuristics*, vol.2, pp.5–30.
18. T. D. Fataneh. 1997. A value-added approach for automated guided vehicle task assignment. *Journal of Manufacturing Systems*, vol.16, no.1, pp.24-34.
19. M. L. Fisher, R. Jaikumar and L. N. V. Wassenhove. 1986. A multiplier adjustment method for the generalized assignment problem. *Management Science*, vol.9, no.32, pp.1095-2104.
20. E. Fu. Private communications 2007. Micron Technology Inc.
21. T. Ganesharajah, N. G. Hall and C. Sriskandarajah. 1998. Design and operational issues in AGV-served manufacturing systems. *Annals of Operations Research*, vol. 76, pp.109-154.
22. M. R. Garey and D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco.
23. F. J. Ghazvini and L. Dupont. 1998. Minimizing mean flow times criteria on a single batch processing machine with non-identical job sizes. *International Journal of Production Economics*, vol.55, pp.273-280.

24. P. C. Gilmore and R. E. Gomory. 1964. Sequencing a one state-variable machine: a solvable case of the traveling salesman problem. *Operations Research*, vol.12, pp. 655-679.
25. N. G. Hall and C. Sriskandarajah. 1996. A survey of machines scheduling problems with blocking and no-wait in process. *Operations Research*, vol.44, no.3, pp.510-525.
26. N. G. Hall, C. Sriskandarajah and T. Ganesharajah. 2001. Operational decisions in AGV-served flowshop loops: scheduling. *Annals of Operations Research*, vol.107, pp.161-188.
27. M. H. Han and L. F. Mcginnis. 1989. Control of material handling transporter in automated manufacturing. *IIE Transactions*, vol.21, no.2, pp.184-190.
28. R. Hubscher and F. Glover. 1994. Applying Tabu Search with influential diversification to multiprocessor scheduling. *Computers and Operations Research*, vol.21, pp.877–884.
29. H. Heinrich and A. Pyke.1995. The impact of conveyor transports on factory performance at Infineon's (Siemens) 200 mm fab. *Semiconductor Fabtech Journal*.
30. International SEMATECH. 2000. *Scheduler/Dispatcher User Requirements*. International SEMATECH Inc.
31. International SEMATECH. 2003 and 2004. *International technology roadmap for semiconductors: Factory Integration*. International SEMATECH Inc.
32. S. M. Johnson. 1954. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, vol.1, no.1, pp.61-68.
33. M. Kidambi. Private communications. 2006. Qimonda Inc.
34. C. W. Kim, J. M. A. Tanchoco and P. H. Koo. 1999. AGV dispatching based on workload balancing. *International Journal of Production Research*, vol.37, no.17, pp.4053-4066.
35. H. Kise, T. Shioyama and T. Ibaraki. 1991. Automated two-machine flowshop scheduling: a solvable case. *IIE Transactions*, vol.23, no.1, pp.10-16.
36. C. H. Kuo and C. S. Huang. 2005. Dispatching of overhead hoist vehicles in a fab inrabay using a multimission-oriented controller. *International Journal of Advanced Manufacturing Technology.*

37. E. Kutanoglu, F. Tanrisever, and S. J. Mason. 2004. Forming and Scheduling Jobs from Multiple Orders with Different Attributes: a Computational Study of Special Cases. *Proceedings of the 2004 IIE Annual Conference and Exhibition*.
38. J. D. Laub, J. W. Fowler, and A. B. Keha. 2007. Minimizing makespan with multiple-orders-per-job in a two-machine flowshop. *European Journal of Operational Research*, vol.182, pp.63-79.
39. R. Leisten. 1990. Flowshop sequencing problems with limited buffer storage. *International Journal of Production Research*, vol.28, no.11, pp.2085-2100.
40. B. Li, J. Wu, W. Carriker and R. Giddings. 2005. Factory throughput improvement through intelligent integrated delivery in semiconductor fabrication facilities. *IEEE Transactions on Semiconductor Manufacturing*, vol.18, no.1, pp.222-231.
41. D. Y. Liao and C. N. Wang. 2004. Neural-network based delivery time estimates for prioritized 300mm automatic material handling operations. *IEEE Transactions on Semiconductor Manufacturing*, vol.17, no.3, pp.324-332.
42. D. Y. Liao, M. Jeng and M. Zhou. 2004. Petri net modeling and Lagrangian relaxation approach to vehicle scheduling in 300mm semiconductor manufacturing. *Proceedings of the 2004 IEEE Conference on Robotics and Automation*, New Orleans, LA, pp.5301- 5306.
43. D. Y. Liao and H. S. Fu. 2004. Speedy delivery: dynamic OHT allocation and dispatching in large-scale, 300 mm AMHS management. *IEEE Robotics & Automation Magazine*.
44. J. T. Lin, F. K. Wang and Y. M. Chang. 2006. A hybrid push/pull dispatching rule for a photobay in a 300mm wafer fab. *Robotics and Computer-Integrated Manufacturing*, vol.22, pp.47-55.
45. J. Liu and B. L. MacCarthy. 1997. A global MILP model for FMS modeling. *European Journal of Operational Research*, vol.100, pp.441-453
46. G. T. Mackulak and P. Savory. 2001. A simulation-based experiment for comparing AMHS performance in a semiconductor fabrication facility. *IEEE Transactions on Semiconductor Manufacturing*, vol.14, no.3, pp.273-280.

47. S. Melouk, P. Damodara, P. Y. Chang. 2004. Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. *International Journal of Production Economics*, vol.87, pp.141-147.

48. H. S. Min and Y. Yih. 2003. Selection of dispatching rules on multiple dispatching decision points in real-time scheduling of a semiconductor wafer fabrication system. *International Journal of Production Research*, vol.41, no.16, pp.3921-3941.

49. L. G. Mitten. 1959. Sequencing n jobs on two machines with arbitrary time lags. *Management Science*, vol.5, no.3, pp.293-298.

50. C. Papadimitriou and P. Kanellakis. 1980. Flow shop scheduling with limited temporary storage. *Journal of Associate Computing Machinery*, vol.27, pp.533-549.

51. M. Pinedo. 2005. *Scheduling: Theory, Algorithms and Systems*. Prentice Hall, Upper Saddle River.

52. J. D. Plummer, M. D. Deal and P. B. Griffin. 2000. *Silicon VLSI Technology: Fundamentals, Practice, and Modeling*. Prentice Hall, Upper Saddle River.

53. C. N. Potts and M. Y. Kovalyov. 2000. Scheduling with batching: a review. *European Journal of Operational Research*, vol.120, pp.228-249.

54. P. Qu. 2004. The Single Machine Multiple Orders per Job Scheduling Problem. Ph.D thesis, Department of Industrial Engineering, University of Arkansas.

55. P. Qu and S. J. Mason. 2004. Using Tabu Search on the Single Machine Multi-Orders per Job Scheduling Problem. *Proceedings of the 2004 IIE Annual Conference and Exhibition*.

56. P. Qu, S. J. Mason, E. Kutanoglu, and J. W. Fowler. 2004. A Polynomial Time Heuristic for Scheduling Multiple-Order Jobs on a Single Machine. *Proceedings of the 2004 IIE Annual Conference and Exhibition*.

57. P. Qu and S. J. Mason. 2005. Metaheuristic Scheduling of 300mm Lots Containing Multiple Orders. *IEEE Transactions on Semiconductor Manufacturing*, vol.18, no.4, pp.633-643.

58. S. Reveliotis. 1999. Real-time control of flexibly automated production systems. *AutoSimulations Symposium '99 – Driving the golden spike*.

59. I. Sabuncuoglu and D. L. Hommertzheim. 1992. Dynamic dispatching algorithm for scheduling machines and automated guided vehicles in a flexible manufacturing system. *International Journal of Production Research*, vol.30, no.5, pp.1059-1079.
60. S. C. Sarin and P. Jaiprakah. 2007. *Flow Shop Lot Streaming*. Springer.
61. S. C. Sarin, A. Varadarajan and L. Wang. 2008. A review on dispatching rules for operational control in wafer fabrication. *International Journal of Production Planning and Control: A Special Issue on "Novel Approaches in Semiconductor Manufacturing"*, to appear.
62. R. Scholl, R. Klein, and C. Jurgens. 1997. BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem." *Computers and Operations Research*, vol.24, pp.627–645.
63. P. Schwerin and G.W. Ascher. 1999. A new lower bound for the bin-packing problem and its integration into MTP." *Pesquisa Operacional*, vol.19, pp.111–129.
64. D. Scott. 2000. How do material and information flows impact fab performance. *The Ninth International Symposium on Semiconductor Manufacturing*, pp 233-236.
65. J. S. Smith, B. A. Peters and A. Srinivasan. 1999. Job shop scheduling considering material handling. *International Journal of Production Research*, vol.37, no.7, pp.1541-1560.
66. D. Simchi-Levi. 1994. New worst-case results for the bin-packing problem. *Naval Research Logistics*, vol.41, pp.579-585.
67. S. Spear and K. Bowen. 1999. Decoding the DNA of the Toyota Production System. *Harvard Business Review*, pp.96-106.
68. 300 mm integrated vision for semiconductor factories. 1999. *International 300 mm Initiative and Japan 300 mm Semiconductor Technology Conference*. Version 3.0.
69. L. Tuan and M. D. Koster. 2006. A review of design and control of automated guided vehicle systems. *European Journal of Operational Research*, vol.171, pp.1-23.
70. J. C. Tyan, T. C. Du, J. C. Chen and I. H. Chang. 2004. Multiple response optimization in a fully automated fab: an integrated tool and vehicle dispatching strategy. *Computers & Industrial Engineering*, vol.46, pp.121-139.

71. G. Ulusoy and S. S. Funda and U. Bilge. 1997. A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles. *Computers Operations Research*, vol.24, no.4, pp.335-351.
72. United Microelectronics Manufacturing (UMC). *World Class Manufacturing*. United Microelectronics Manufacturing Website, 2008
73. R. Uzsoy. 1994. Scheduling a single batch processing machine with non-identical job sizes. *International Journal of Production Research*, vol.32, no.7, pp.1615-1635.
74. F. Vanderbeck. 1999. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, vol.86, pp.565–594.
75. T. Wakabayashi, S. Watanabe, Y. Kobayashi, T. Okabe and A. Koike. 2004. High-speed AMHS and its operation method for 300mm QTAT fab. *IEEE Transactions on Semiconductor Manufacturing*, vol.17, no.3, pp.317-323.
76. Y. C. Wang, C. S. Wu and L. Jann. 2002. Sorter automatic operations in a 300mm fab. *2002 Semiconductor Manufacturing Technology Workshop*, pp.119-122.
77. S. Webster and K. R. Baker. 1995. Scheduling groups of jobs on a single machine. *Operation Research*, vol.43, no.4, pp.692-703.
78. R. Wilson. 2007. Intel grabs market share back from AMD. *Electronic Weekly Magazine Online*.
79. S. Martello and P. Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley.