The Dissertation Committee for Emrah Zarifoglu Certifies that this is the approved version of the following dissertation:

PLANNING AND SCHEDULING IN SEMICONDUCTOR MANUFACTURING

Committee:

Erhan Kutanoglu, Co-Supervisor

John J. Hasenbein, Co-Supervisor

David P. Morton

Elmira Popova

Stephen M. Gilbert

PLANNING AND SCHEDULING IN SEMICONDUCTOR MANUFACTURING

by

Emrah Zarifoglu, B.S.; M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August, 2010

Dedication

To my family

Acknowledgements

I'd like to give my special thanks to Dr. Erhan Kutanoglu and Dr. John Hasenbein for their patience, understanding and full support in supervising my dissertation and completing my degree. I'd like to also thank to my dissertation committee members Dr. David Morton, Dr. Elmira Popova and Dr. Steve Gilbert for their time to review and evaluate my dissertation. I'd like to thank AMD for their support to my studies, especially Peng Qu and Shekar Krishnaswamy for introducing me to the semiconductor manufacturing research. I' like to thank Robert Wright of ISMI, who supported me during most of my studies and have provided me necessary means to complete my research. I also would like to thank Alexander Grosser of ISMI and Globalfoundries for his help in finding me new research ideas which contributed to my dissertation.

Planning and Scheduling in Semiconductor Manufacturing

Publication No._____

Emrah Zarifoglu, Ph.D.

The University of Texas at Austin, 2010

Co-Supervisor: Erhan Kutanoglu

Co-Supervisor: John J. Hasenbein

Semiconductor manufacturing is one of the most complex existing manufacturing systems. It requires constant improvement to meet demands and expectations. This dissertation studies semiconductor manufacturing under three main topics, preventive maintenance scheduling, lot size management and AMHS scheduling. We first provide an optimization based decomposition algorithm and a heuristic algorithm to solve preventive maintenance scheduling problem along with direct optimization. Then, we develop an analytic tool to investigate and find optimal lot sizes to run in a manufacturing environment to minimize cycle time. Finally, we propose an optimization based AMHS scheduling algorithm and compare its performance to a myopic algorithm.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The first phase of industrial revolution started with textile industry. The textile industry was naturally the most common sector around the world. The second phase of the industrial revolution witnessed iron founding and metal manufacturing in general. Each phase has brought more powerful and more technological tools to various industrial sectors. With these changes, the modern manufacturing era made a quick start. As necessities bore inventions, inventions also made themselves new necessities. These manufactured products have been used widely in our daily lives. Although we have not recently witnessed the drastic changes in manufacturing technologies of 19th century, we have been not only witnesses but also actors of another important phase of the industrial revolution which put a signature on the post-modern era, semiconductor manufacturing. Semiconductor devices dominate all our lives and they have brought a great degree of dependency for sustaining life in our enormously populated and globalized world in the 21st century.

Semiconductor manufacturing applications are everywhere in life from our computers to cars, mobile phones to microwave ovens. We depend on it for our communication, education, entertainment and business. Semiconductor manufacturing is probably the most complex and advanced type of manufacturing that currently exists. Theoretically, there is not a boundary on its improvement. It requires the utmost degree of excellence in manufacturing and continuous improvement in technology to satisfy consumer demand and needs.

In its execution, the complexity of semiconductor manufacturing and the scale of production engender countless problems. While it has conventionally been an area of expertise for chemical and electrical engineers, increased complexity over time has

required new set of capabilities. Therefore, semiconductor manufacturing has been one of the major areas in which the expertise of industrial engineers and operations researchers is needed. The applications have been so diverse that they range from integer programming implementations to queueing theory analysis.

This dissertation focuses on planning and scheduling in a context which spans a wide set of implementation areas on three main topics under the umbrella of semiconductor manufacturing. Chapter 2 studies preventive maintenance scheduling in semiconductor manufacturing, at the machine level. Chapter 3 incorporates production lots into the picture and turns its focus to lot size management, critically analyzing small lot size manufacturing efforts. A third dimension, the movement of lots, is added in the Chapter 4 as automated material handling (AMHS) scheduling in semiconductor manufacturing. Our wide spanning approach to the planning and scheduling problem is a result of complex and constantly evolving nature of semiconductor manufacturing. Each problem has been the main focus in the industry during the time frame they have been studied. Starting the research with preventive maintenance scheduling has helped learn equipment and manufacturing environment and has been a very good introductory to the industry suppliers and manufacturers for us. While the research on lot size management has provided us an extensive knowledge on factory dynamics, material handling research has helped us close open ends within separate levels of operations in the production environment.

Semiconductor manufacturing has its own unique terminology. This dissertation only deals with wafer fabrication, which is front end of semiconductor manufacturing. Therefore, the terminology contains wafer fabrication related definitions. A manufacturing plant is usually called a *fab*, a machine is called a *tool*, and a production

unit is called a *wafer*. A group of wafers, called a *lot*, is moved in a carrier called FOUP (Front Opening Unified Pod). The size of these lots is fixed and it usually varies from 1 to 25. Each specific operation on a wafer is called step. A route or flow specifies the pre-defined sequence of tool visits taken by a lot. A route consists of 500 steps on average. Each step is a combination of a number of sub-operations. Tools are grouped in tool groups according to their capabilities. There are 50 tool groups on average in a fab and a typical fab consists of about 700 tools. An average 300-mm processor chip fab is capable of processing 30000 wafers in a month. Each tool group is visited multiple times. Therefore, a route consists of reentrant flows.

The costs of the tools are very high in semiconductor manufacturing. Depending on its capabilities, current tool price can be up to $100 million. Therefore, tools need to be under a strict maintenance regime to keep them running a healthy and long processing life. Also, since breakdowns of tools interrupt production unexpectedly, scheduled (preventive) maintenance tasks are employed to limit the occurrence of unscheduled down times. From a planner's point of view, a preventive maintenance task acts like a production lot and uses a part of the capacity on a tool except that there is no wafer output from this use. Employing a preventive maintenance task at a certain time on a certain tool means that a part of the tool's capacity is going to be unavailable for actual wafer processing. The capacity loss due to scheduling of preventive maintenance tasks on a tool group causes fluctuations on the available capacity of the tool group. For planning purposes, a smooth capacity profile over time is preferred. An effective way to provide a smooth capacity profile is controlling the maximum capacity loss due to preventive maintenance scheduling.

In Chapter 2, we propose a mixed integer programming formulation to model preventive maintenance scheduling on a tool group, which aims to find a feasible schedule for preventive maintenance tasks within a certain planning period while minimizing the maximum capacity loss on the tool group. We develop three different algorithms to solve the problem. One way is that we try to solve this problem with direct optimization. In another algorithm, we decompose the problem such that the problem reduces to solving a preventive maintenance scheduling problem for each tool, then the tool schedules are combined iteratively to minimize the maximum capacity loss. Our third algorithm is a computationally efficient heuristic based on the idea of scheduling task with largest capacity loss to the time period the least accumulated capacity loss within the planning horizon. Which of these three algorithms works best depends on the particular planning environment. Having a daily or weekly planning horizon, different levels of variation on capacity loss, and varying processing times of the preventive maintenance task all affect the performance of these algorithms. Thus, in Chapter 2, we also undertake a detailed experimental study.

The manufacturing environment in the semiconductor industry is open to continuous improvement efforts. Usually the ultimate aim is to minimize cycle time in a fab. A large portion of cycle time goes to activities that are not related to actual processing, such as waiting and handling. Since recent lean manufacturing efforts in semiconductor manufacturing require the elimination of waste, the waiting time of wafers is targeted for reduction. Since wafers are carried in lots, some portion of delay is caused by wafers waiting for the entire lot to complete processing after each step. The current convention of the 25 wafer lot size serves as a base to start with for further improvement efforts using lot size management. While some in the industry prefer lot size reduction in

order to reduce cycle time, others believe that running large lot sizes helps control system variation. In fact, the system parameters have a nuanced effect on the performance of a various lot sizes.

In Chapter 3, we analyze the effect of lot size on cycle time. Although there have been a number of studies regarding lot size reduction in the literature, these studies are mostly simulation based. We introduce an analytic model which uses queueing theory, and depends on alternative lot sizes and technology levels, to represent a slice of the manufacturing flow. We use some approximation techniques to provide a closed form analytic formulation for cycle time. A simulation study confirms the validity of the approximation. We give a detailed analysis of the behavior of cycle time under varying conditions. For several special cases, we derive optimal lot sizes.

Moving lots from tools to tools and between storage areas and tools after every step creates a challenging problem. Improvements and changes in fab layouts force material handling methods to transform drastically over time. While manual material handling was common in the early days of semiconductor manufacturing, increased wafer sizes make it difficult for human operators to carry the lots. Most up-to-date systems run fully automated material handling systems. The layout of a material handling system can vary greatly, depending on the fab layout and production requirements. The spine layout in Figure 1.1 is a very common one in which major tracks surround the receiving ends of tool groups in the inner loop. Smaller tracks are built which serve the tools in a tool group. These tracks constitute outer loops. Each tool has some limited space for storage which is called tool buffer. The common storage place for a tool group is called stocker, which is placed by the tracks of inner loop. Stockers have a large capacity, and in most cases they can be assumed practically uncapacitated. Lots are carried in FOUPS

traveling via various types of vehicles. Automated guided vehicles, over head transports and conveyor belts are examples of some of these vehicle types. AMHS design in semiconductor manufacturing is a very large study area by itself.

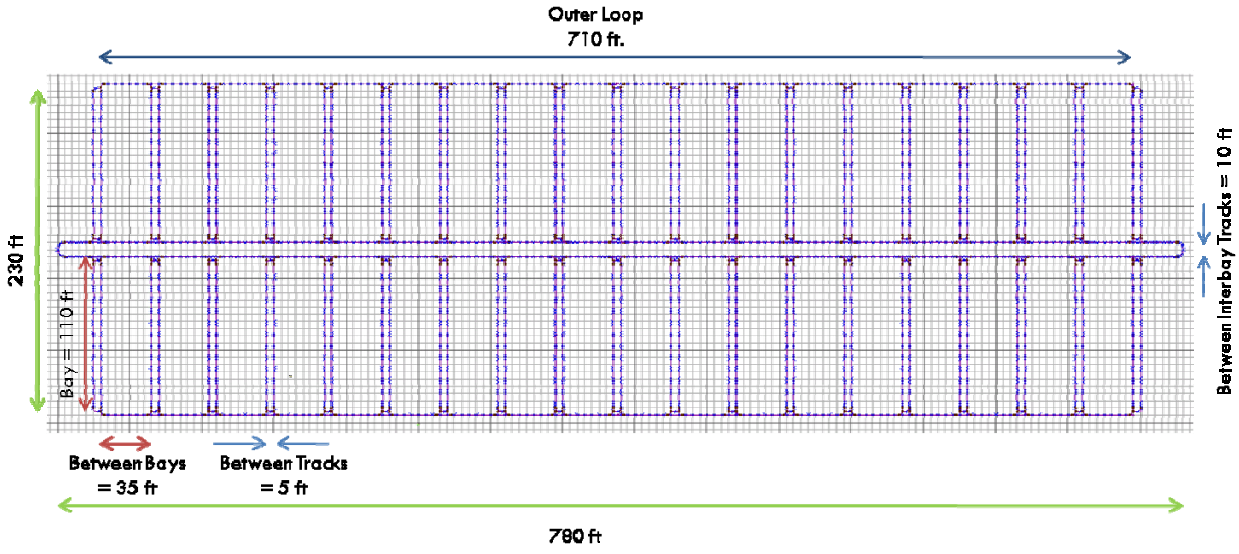


Figure 1.1 Sample AMHS layout in a fab

Managing the movement of vehicles and lots is an important problem. Improving the technology and scale of manufacturing adds to the complexity of lot and vehicle movements. There are numerous applications of lot and vehicle dispatching in the literature, and many of the proposed dispatching algorithms aim to minimize conflicts in vehicle traffic and congestion in the system. A significant portion of these studies are in parallel with AMHS design. Although industrial algorithms perform very well, they have a major drawback in the limited scope of the solutions they provide. Despite varying methodology, most of these algorithms can be classified as myopic since they have limited information availability due to the information structure of the fab shop floor. Conventionally, the information infrastructures of production and AMHS are surprisingly separate, meaning that their corresponding systems operate independently practically all

the time. Information sharing occurs between the two separate systems only when an entity of one system becomes available to the other. For example, the AMHS system sees a lot only when it becomes available to be picked up (e.g., at a tool). On the other hand, a vehicle becomes visible to the production system only when it is available for pickup. However, there have been recent efforts among the suppliers of the industry to increase the degree of information sharing between the two systems. The main effort is to provide some visibility and information availability ahead of time. Although the window of this visibility is measured only in minutes, it brings new opportunities for higher quality solutions. In Chapter 4, we propose an optimization based algorithm which solves the lot-vehicle assignment problem for each visibility period in real time. Since the algorithm contains some consideration of future decisions, it does not suffer from the myopia of most conventional algorithms. In short, we present this optimization based look-ahead scheduling approach to the problem rather than relying solely on a one lot, one vehicle at a time dispatching approach. We evaluate the look-ahead approach using a version of the myopic algorithm as a benchmark. We run an extensive experimental design to analyze, evaluate and compare the performance of the look-ahead and myopic algorithms.

Since the content of each chapter is essentially independent from the others, each chapter is largely self-contained and hence can be read separately. After next three chapters, the dissertation is closed with a novel conclusion.

Chapter 2: Preventive Maintenance Scheduling

Planning and scheduling preventive maintenance tasks in semiconductor manufacturing wafer fabrication facilities (*fabs* for short) is a very important activity due to high equipment costs and the risk of losing expensive capacity due to potential unexpected down times. PM activities and their schedule (when, and in what combination they are carried out) directly affect the fab performance, especially throughput, cycle time and Work-In-Process (WIP) inventory levels. Moreover, fabs usually operate with the goal of achieving weekly or daily production targets. The way that the PM tasks are scheduled might determine whether the fab can achieve these targets or not. Hence, the preventive maintenance planning and scheduling problem addressed in this paper is an important problem in semiconductor manufacturing. As a highly complex process due to its size, intricate interactions among elements of the system, machine/tool dedication and reentrant flow, semiconductor manufacturing planning and scheduling decisions are typically handled in a hierarchical manner. We envision a similar framework here by modeling, solving, and analyzing the overall PM planning and scheduling problem in a hierarchy of two levels, in which higher level feeding its output (PM planning) to the lower level to produce a detailed PM schedule (PM scheduling). In this paper, we focus on the lower level of the hierarchy, detailed PM scheduling.

The semiconductor manufacturing environment consists of tool (or machine) groups in which individual tools of similar production capabilities are grouped together. Different types of wafers (or product types in general) are processed through the tools by visiting a tool in each tool group. One essential aspect of semiconductor manufacturing is its *reentrant* (or recirculation) nature: Wafers come back to a tool group many times for additional processing (to complete additional operational steps to make the photo layers

on the wafer one after another) before they leave the fab. This is called *loop*, which consists of a series of operations starting and ending at the same tool. The main reason for "looping" is that expensive tools are not dedicated to the layers, as this would require enormous amount of capital. There are other complexities of semiconductor manufacturing essential for any reasonable modeling and analysis. Approaches developed for semiconductor manufacturing planning and scheduling try to capture these issues in one of the levels of decision making hierarchy. We follow a similar route by taking these issues into account on the lower level of the PM decision making hierarchy, tool group PM scheduling problem (For more in-depth discussion on higher level PM planning, see Yao et al. (2001) and Yao et al. (2004)).

The tool group PM scheduling model deals with scheduling given PM tasks (given in the output of a higher-level PM plan) on individual tools within a specific tool group during a relatively short planning horizon. The idea is to run the model at the beginning of every planning horizon (e.g., everyday or every week) for each tool group using the assignments dictated by the higher level PM planning model solution. The main output of the tool group PM scheduling model is a detailed PM schedule that shows the best times to perform the PM tasks on each tool during the planning horizon. In this model, multi-tasking (i.e., carrying out multiple PM activities at the same time) is allowed. Objectives may vary in the model depending on the operational issues to be addressed. In our study, we aim to control the pattern of capacity loss due to PM allocation on the tool group. Therefore, in our mathematical model, we use minimization of the maximum capacity loss throughout the planning horizon as an objective. This objective function helps control production capacity fluctuation in the tool group and limits variation in the capacity over planning horizon.

Next, we present a compilation of literature on preventive maintenance scheduling in semiconductor manufacturing. Afterwards, we provide more details on the tool group PM scheduling model with a mixed integer programming (MIP) approach. Then, we present a decomposition based heuristic solution approach for the problem. For larger problems, we also propose a heuristic.

2.1 LITERATURE REVIEW

PM scheduling in the semiconductor manufacturing literature is relatively new topic. Since the main cost factor is equipment and any unexpected delay in the production flow due to equipment failures may affect the bottom line due to increased cycle times and decreased customer satisfaction, PM planning and scheduling has become increasingly more important, attracting new research.

Charles et al. (2003) demonstrate the importance of scheduling considerations in the optimization of PM strategies in a semiconductor manufacturing environment. They use an object oriented simulator. While they mainly focus on preventive and corrective maintenance, they also take residual breakdowns into account since they still take place despite the scheduled preventive maintenance tasks.

Performing a PM task can be either calendar based (e.g., every 30 days) or event based (e.g., between every 100 wafers processed on a tool). A calendar-based method is typically preferred to an event based one because the former is relatively easy to interpret and take action. Chih-Hong and Shih-Chao (2002), and Ramirez-Hernandez and Fernandez-Gaucherand (2003) study conversions from an event-based system (which may be needed in the fab due to PM task specifications) to calendar-based conversions and forecasting methods (which may be used in the PM modeling and scheduling system). Chih-Hong and Shih-Chao (2002) propose a new forecasting method based on weighted evaluation of “long period average daily movement” and “short period average

daily movement" data extracted from the real time wafer movement data coming from automated databases. Ramirez-Hernandez and Fernandez-Gaucherand (2003) propose a mathematical model and an algorithm to convert event-based PM tasks to calendar-based ones.

Markov decision processes are also used in similar contexts as a modeling tool for PM scheduling in semiconductor manufacturing. Sloan and Shantikumar (2000) examine production and maintenance models together with the situation in which the equipment condition affects different products in different ways. They develop a Markov Decision Process (MDP) model for a single machine, multi product environment. They solve a linear programming (LP) model to find the combined product and maintenance schedule. Their results show that the improvement provided by the combined schedule becomes more significant as the products become more diverse (more product types) and that the most of the increase is due to changing the production schedule rather than the maintenance schedule. Yao et al. (2005) study the problem of jointly optimizing PM and production policies with non-negligible and stochastic maintenance times. They also formulate the problem as a MDP model. Their results show that when inventory is below a certain point, the optimal policy is either to perform a PM task or to produce at the maximal production rate; when inventory is above a certain point, the optimal action is either to carry out the PM task or not to produce at all. In between these inventory levels, they analyze different settings and propose corresponding optimal control policies.

A different approach to PM scheduling is adding work-in-process (WIP) inventory control to the model with appropriate objective function. Important contributions in this regard are due to Yao et al. (2001, 2004 and 2005). They propose a structure that separates the overall PM problem into two levels, one being higher level,

and the other being lower level. While the fab-wide higher level problem schedules PM tasks over a longer time horizon, the lower level problem considers scheduling PM tasks across a group of cluster tools over a shorter horizon. They mainly focus on the lower level problem, which they model as a MIP model that takes WIP levels into account. They run a simulation model to compare the performance of the output of the MIP model with the actual fab PM schedules, and observe that the MIP model output outperforms the actual PM schedules in performance measures such as the average number of wafers completed on each tool and the average WIP on each tool. Our study follows their path and focuses on the lower level problem using a more detailed and flexible model.

2.2 TOOL GROUP PM SCHEDULING PROBLEM

The tool group PM scheduling model takes the planning horizon, PM tasks, and tool group assignments, and assigns all PM tasks on individual tools in each tool group over the planning horizon.

The goal of the tool group PM scheduling model is to control the capacity loss due to PM tasks. We formulate this as an objective function that minimizes the maximum capacity loss across all time periods within the tool group. This helps control fluctuations in the capacity levels over the planning horizon, which in turn helps achieve smoother WIP levels over time.

Each PM task is associated with a tool. When a task is performed on a tool, it decreases the production capacity of the tool by a certain percentage. The main reason that the tool may be still available for production during PM (though at less than 100% capacity) is that the most wafer processing tools have multiple chambers and while PM is carried on one or more of the chambers, the others may continue to process wafers. The new percentage of the tool capacity incurring by performing a task on that tool is called

PM task availability. The percentage decrease in production capacity due to a task allocation is called *PM task unavailability*.

More than one task is allowed to be performed at the same time on a tool. Also, more than one tool can be performing PM tasks at the same time in a tool group. (For this model, we assume that we have sufficient PM resources that would carry out multiple PM tasks in parallel). The total capacity loss on a tool in a time period is calculated by adding up the *unavailabilities* caused by the PM tasks performed on that tool at that time period. This method leads to linear calculation of the capacity loss in the tool group, which results from direct effect of unavailability of each PM task scheduled. As an example from the fab, the linear calculation method follows from the strategic decision that requires one chamber to be dedicated for one PM task. While that chamber stops production, the rest can continue to use their overall capacity, or other PM tasks can be performed while a PM task being implemented in one of the chambers.

We assume an *initial capacity profile* of the tool group, which is the collection of capacity levels over time periods of the planning horizon. All the capacity loss due to PM task allocation is subtracted from this initial capacity profile to form a *final capacity profile*. The initial capacity profile is assumed given (typically 100% in all periods), and we seek to minimize the maximum capacity loss from this profile while obtaining the final capacity profile.

The mathematical model introduced in the next section explains components of the PM task scheduling problem in more detail. We present a sample PM task assignment problem after explaining the mathematical model to depict how the system works.

2.3 MATHEMATICAL MODEL – THE MAIN PROBLEM

We now present a mixed integer programming (MIP) formulation for the optimal assignments of PM tasks on the tools of a tool group. We assume that the tasks are to be scheduled over a short term horizon, such as a day or a week. We assume that the time horizon is divided into T time periods of equal length (say, hourly time periods). Each tool i in the tool group has n_i PM tasks to be scheduled during the horizon. Tool i's k^{th} task is denoted by pair (i,k). We denote the set of tool i's tasks by K_i. Task (i,k) has a length of p_{ik} units (in the same units as the time periods, say in hours). Each tool has a production capacity s_i (in units/hour), representing at most how many wafers tool i can process in an hour when the tool is 100% available. Task (i,k) has availability f_{ik}, which shows the portion of number of wafers processed by tool i in an hour if task (i,k) is scheduled in that hour. That is, task (i,k) reduces the tool's full (100%) availability to $100f_{ik}$% causing a capacity loss.

The main decision variable X_{ikt} (defined for all tool-task pairs (i,k), and time periods t) is 1 if task (i,k) is scheduled to start at time period t; 0 otherwise. Another binary decision variable, Y_{ikt}, is a transformation of X_{ikt}, and it is 1 if task (i,k) is being performed in time period t, 0 otherwise. The conversion between X_{ikt} and Y_{ikt} is handled through the following constraints:

$$Y_{ikt} = \sum_{\tau=t-p_{ik}}^{t} X_{ik\tau}, \forall i,k \in K_i, t. \tag{2.1}$$

The following assignment constraints make sure that a task can start at only one time period, and that a task has to be finished within the horizon:

$$\begin{aligned} \sum_{t=1}^{T-p_{ik}+1} X_{ikt} &= 1, \forall i,k \in K_i, \\ \sum_{t=T-p_{ik}+2}^{T} X_{ikt} &= 0, \forall i,k \in K_i. \end{aligned} \tag{2.2}$$

The decrease in the capacity level (capacity loss) due to scheduled PM tasks is calculated in the following constraints, which keep track of the maximum capacity loss across time periods in the decision variable, W:

$$W \geq \sum_{i \in I} \left[\sum_{k \in K_i} (1 - f_{ik}) Y_{ikt} \right] s_i \,, \forall t \in T. \tag{2.3}$$

There is a physical limit on how many tasks can be assigned in a time period on a tool. The summation of unavailabities of all PM tasks assigned in a time period on a tool should not exceed 100% as captured in the following constraints:

$$\sum_{k \in K_i} (1 - f_{ik}) Y_{ikt} \leq 1, \forall i, t \in T. \tag{2.4}$$

Necessary binary and non-negativity constraints are given as follows:

$$\begin{aligned} &X_{ikt} = 0 \text{ or } 1, \forall i, k \in K_i, t \in T, \\ &Y_{ikt} = 0 \text{ or } 1, \forall i, k \in K_i, t \in T, \\ &W \geq 0. \end{aligned} \tag{2.5}$$

However, we can replace constraints (2.5) by the following since Y_{ikt} values are directly related to the values of X_{ikt}.

$$\begin{aligned} &X_{ikt} = 0 \text{ or } 1, \forall i, k \in K_i, t \in T, \\ &Y_{ikt} \geq 0, \forall i, k \in K_i, t \in T, \\ &W \geq 0. \end{aligned} \tag{2.6}$$

The objective function of the MIP model is to minimize the maximum capacity loss over the planning horizon:

$$\min W \,. \tag{2.7}$$

Then, the optimization problem is to minimize (2.7) subject to constraints (2.1-2.4) and (2.6).

2.3.1 Numerical Example

We use a hypothetical example with 5 tools and 10 PM tasks. Table 2.1 shows the data associated with the example. This data is also used later in the paper to illustrate the heuristic algorithms developed for this problem.

Table 2.1 Task data of sample problem instance

		Process Time	Availability (%)
Tool 1	Task 1	3	60
	Task 2	2	80
	Task 3	2	40
Tool 2	Task 1	1	0
	Task 2	2	80
Tool 3	Task 1	3	60
Tool 4	Task 1	1	20
	Task 2	2	40
Tool 5	Task 1	2	80
	Task 2	1	20

Tool capacities are same and equal to 100 units (say wafers) per time period. We solve this problem over a 12-hour planning horizon, representing a shift. Figure 2.1 depicts the corresponding Gantt chart for PM tasks and lists the capacity losses after PM tasks are scheduled according to the optimal MIP solution.

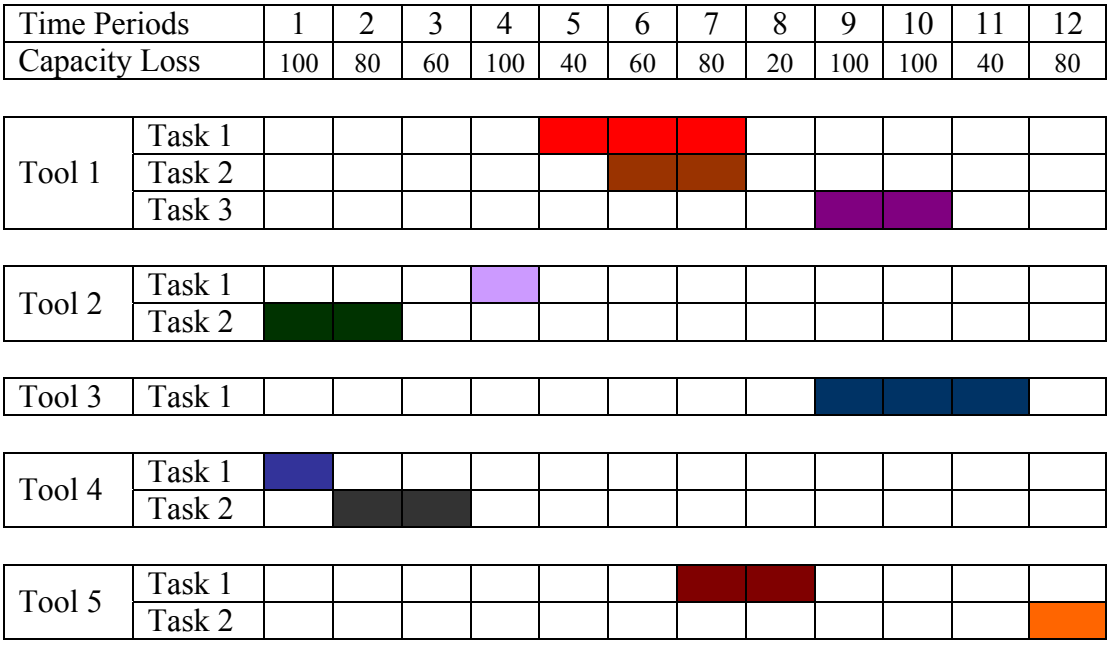


Figure 2.1 Gantt chart showing the optimal PM schedule

The maximum capacity loss is 100 wafers. While tasks are usually distributed along the planning horizon, in some periods there are multiple tasks being processed on the same tool, as in time periods 5 and 6 on Tool 1, or across tools as it is most of the time.

By Theorem 1, we show that the PM scheduling problem represented with the MIP mathematical model is computationally intractable even when the problem is simplified with unit-length tasks and one tool in the tool group.

Theorem 1: The PM scheduling problem is NP-hard.

Proof: Consider the setting where we seek to schedule multiple PM tasks each with unit processing time (e.g., 1 hour) on a single tool. Assume that we have n tasks ($n > 2$) and the planning horizon is 3 (hourly) time periods. We now show that minimizing maximum capacity loss in this setting while scheduling PM tasks on a single tool over 3 time periods is equivalent to minimizing makespan while scheduling n jobs on 3 parallel

machines ($P3||C_{max}$). Analogy between the two problems can be observed by showing the equivalent parameters in each problem. Time periods in the PM scheduling problem are analogous to the parallel machines in the makespan problem. PM tasks are analogous to jobs in the makespan problem in which each job has a processing time equivalent to the corresponding task's unavailability. Capacity losses in the PM problem denote the completion times of jobs in the makespan problem. Therefore, the objective of minimizing maximum capacity loss becomes equivalent to minimizing makespan in $P3||C_{max}$. If we rotate the capacity loss chart of a PM task schedule 90 degrees clockwise, we obtain a picture analogous to the Gantt chart of the parallel machines in the makespan problem. Since $P3||C_{max}$ is an NP-hard problem (Pinedo, 2008) and it corresponds to a simple version of the PM scheduling problem, the more general PM scheduling problem is NP-hard. □

In the next section we propose a tool-based decomposition method to solve the same problem. Then, we propose a greedy heuristic algorithm that aims to find a high-quality schedule quickly in large problems. After introducing alternative solution methods, we compare performances of all three methods with an extensive experimental study.

2.4 TOOL DECOMPOSITION PROBLEM

The direct optimization of the MIP problem takes excessive amount of solution time to prove optimality for large problems. Therefore, we propose a decomposition based heuristic algorithm to solve the PM scheduling problem. We decompose the problem with respect to tools, solving a tool-level task assignment problem for each tool independently. In the MIP formulation, the only set of constraints that combine the tool capacity loss contributions is (1.3), maximum capacity loss calculation. By keeping the

constraints for each tool the same and revising (1.3), we obtain the task assignment problem for each tool.

The tool-level problem assigns PM tasks of a tool on a *limited* planning horizon, which is assumed to represent only a portion of the overall planning horizon of the original problem. Then these independent tool loads are combined over the whole planning horizon. The motivation behind this is that more uniformly distributed unavailabilities of task assignments over time likely result in more uniform capacity losses which will keep the maximum capacity loss under control. To this end, the proposed tool based decomposition scheme solves the problem first by spreading unavailability on a tool over a pre-defined (and shorter) planning horizon at the tool level, and then combining these tool-level schedules together over the whole planning horizon. Limiting the horizon at the tool level helps balance unavailability and lower the overall maximum capacity loss as explained below.

The main difference of the tool problem from the overall MIP formulation is that the horizon T_i of the tool problem is the allocated portion of (hence shorter than) the overall planning horizon, T. The tool-level horizon is allocated proportional to the tool's unavailability contribution, $U_i = \sum_{k \in K_i} (1 - f_{ik}) p_{ik}$, with respect to the total unavailability, $U = \sum_{i \in I} \sum_{k \in K_i} (1 - f_{ik}) p_{ik}$, as below,

$$T_i = \max\left\{ \max_{k \in K_i} [p_{ik}], \left\lceil \frac{U_i}{U} T \right\rceil \right\}. \tag{2.8}$$

The unavailability created by a PM task on a tool has a significant role in constructing the decomposition based algorithm. There is an analogy between the unavailability of a PM task and a brick in the sense that the latter is used to build a wall

as the PM task unavailabilities result in capacity losses in the "wall." The length (horizontal side) of a brick corresponds to the processing time of a PM task (p_{ik}) and the height (vertical side) of a brick is analogous to unavailability caused by PM task per unit time. These bricks are put on top of or next to each other to build a wall, whose maximum height represents the maximum capacity loss. The total horizontal length in the decomposition problem is limited by the tool-level planning horizon, T_i , defined for each tool. We conjecture that if every tool gets a planning horizon (horizontal length) that is proportional to its bricks' total side area (i.e., the total unavailability caused by its tasks), then the walls (capacity losses) independently built by these tools will have similar heights and this uniformity will produce a near-optimal solution for the original problem when these pieces are combined horizontally (side by side) to have a capacity loss "wall" for the tool group.

The mixed integer programming formulation of the decomposition problem follows from the main problem. T_i is added as a new parameter to this problem as defined above. Our new variables are B_{it} , capacity loss on tool i at time t, and B_i , maximum capacity loss on tool i .

Assignment constraints of the tool decomposition problem are obtained by rewriting assignment constraints of the main problem for tool i ,

$$\begin{aligned}
&\sum_{t=1}^{T_i-p_{ik}+1} X_{ikt} = 1, \forall k \in K_i, \\
&\sum_{t=T_i-p_{ik}+1}^{T_i} X_{ikt} = 0, \forall k \in K_i, \\
&\sum_{\tau=t-p_{ik}}^{t} X_{ik\tau} = Y_{ikt}, \forall k \in K_i, t \in T_i.
\end{aligned} \tag{2.9}$$

Capacity loss for each time period and maximum capacity loss calculation constraints are similar to the maximum capacity loss calculation constraints of the main problem,

$$\begin{aligned} &B_{it} = s_i \sum_{k \in K_i} (1 - f_{ik}) Y_{ikt}, \forall t \in T_i, \\ &B_i \geq B_{it}, \forall t \in T_i. \end{aligned} \tag{2.10}$$

The feasibility constraint for tool *i* is as follows,

$$\sum_{k \in K_i} (1 - f_{ik}) Y_{ikt} \leq 1, \forall t \in T_i. \tag{2.11}$$

Binary and non-negativity constraints are as before,

$$\begin{aligned} &X_{ikt} = 0 \text{ or } 1, \forall k \in K_i, t \in T_i, \\ &Y_{ikt} \geq 0, \forall k \in K_i, t \in T_i, \\ &B_{it} \geq 0, \forall t \in T_i, \\ &B_i \geq 0. \end{aligned} \tag{2.13}$$

The objective function is the minimization of maximum capacity loss,

$$\min \ B_i . \tag{2.14}$$

2.5 MASTER PROBLEM

We need a master problem to combine the schedules of tool level problems by minimizing the maximum capacity loss. B_{it} values from the tool decomposition problems are given as input to the master problem. $Z_{ilt}, i \in I, l \in T_i, t \in T$ is the new set of binary variables each denoting whether *block* l of tool *i* is put on time period *t* or not. A *block* is a load of (potentially multiple) PM tasks assigned together in a time period in the solution of a tool problem. There are T_i blocks resulting from a tool problem. Therefore, l ranges from 1 to T_i .

PM blocks of tools are assigned with the following constraints,

$$\begin{aligned} \sum_{t\in T} Z_{ilt} &= 1, \forall i \in I, v_i \in T_i, \\ \sum_{l\in T_i}\sum_{t\in T} Z_{ilt} &= T_i, \forall i \in I, \\ \sum_{l\in T_i} Z_{ilt} &\le 1, \forall i \in I, \\ Z_{ilt} &\le Z_{i,l+1,t+1}, \forall i \in I, l \in T_i, t \in T, l < |T_i|. \end{aligned} \tag{2.15}$$

Capacity loss calculation is made over the tool-level capacity losses,

$$\begin{aligned} W_t &= d_t + \sum_{i\in I}\sum_{l\in T_i} B_{il} Z_{ilt}, \forall t \in T, \\ W &\ge W_t, \forall t \in T. \end{aligned} \tag{1.16}$$

Binary and non-negativity constraints are as follows,

$$\begin{aligned} Z_{ilt} &= 0,1, \forall i \in I, l \in T_i, t \in T, \\ W_t &\ge 0, \forall t \in T, \\ W &\ge 0. \end{aligned} \tag{2.17}$$

Objective function is to minimize the maximum capacity loss,

$$\min W. \tag{2.18}$$

The overall decomposition algorithm works by solving the tool level problems first, then solving the master problem at the end. The maximum of the objective function values of the tool problems is a lower bound to the master problem:

1. Initialize T_i for all *i* according to (2.8).
2. For each tool i, solve the corresponding tool-level decomposition problem (modeled by equations (2.9-2.11,2.13,2.14)). If problem *i* reports no feasible solution, increase its time horizon allocation (i.e., assign $T_i \leftarrow \min\left(\sum_{i\in I}\sum_{k\in K_i} p_{ik}, T\right)$) and solve the problem again. If the problem is still infeasible, STOP, and report that the overall problem is infeasible.

3. Solve the master problem (modeled by equations (2.15-2.18)), and report the solution.

One of the two reasons why infeasibility occurs in the decomposition algorithm is the limited planning horizon in the tool problem. When the horizon for tool *i* is revised with $T_i \leftarrow \min\left(\sum_{i\in I}\sum_{k\in K_i} p_{ik}, T\right)$, the only reason left for infeasibility would be the tool's available capacity constraints (2.11) of the tool problem. Considering a longer horizon (as long as the original problem's horizon as in constraint set (2.4)) gives the tool problem the opportunity to detect if the original problem is infeasible.

2.5.1 Numerical Example

We use the same numerical example introduced earlier to show an alternative solution obtained by the decomposition algorithm. Each tool solves its own PM task allocation problem in its limited planning horizon first. We obtain the Gantt charts in Figure 2.2 (note the differences in horizons allocated to each tool). Total time horizons allocated to each tool is 15 which is greater than 12. This is because of the granularity. Since we cannot allocate a fractional time period to a tool, all fractions are rounded to its floor. The capacity losses due to these PM task assignments over the planning horizons of tools are also shown in Figure 2.2.

Time Periods		1	2	3	4
Capacity Loss		100	100	60	20
Tool 1	Task 1				
	Task 2				
	Task 3				

Capacity Loss		20	20	60
Tool 2	Task 1			
	Task 2			

Capacity Loss		40	40	40
Tool 3	Task 1			

Capacity Loss		60	60	80
Tool 4	Task 1			
	Task 2			

Capacity Loss		20	100
Tool 5	Task 1		
	Task 2		

Figure 2.2 Gantt charts of the optimal tool level solutions

Taking the capacity losses obtained from the tool level solutions as "blocks," the master problem combines them without breaking their pattern over time and squeezes them into 12 time periods. The resulting capacity loss profile and Gantt Chart is in Figure 2.3 (DA stands for "decomposition algorithm").

The maximum capacity loss is 100, which is the same as the value of the MIP solution. The DA capacity loss profile has more variance than the capacity loss profile of MIP solution.

Time Periods		1	2	3	4	5	6	7	8	9	10	11	12
Capacity Loss		20	20	100	0	100	100	60	60	100	100	100	100
Tool 1	Task 1												
	Task 2												
	Task 3												
Tool 2	Task 1												
	Task 2												
Tool 3	Task 1												
Tool 4	Task 1												
	Task 2												
Tool 5	Task 1												
	Task 2												

Figure 2.3 Gantt chart of DA Solution

2.6 HEURISTIC ALGORITHM

As the problem is NP-hard, it is no surprise that solving the MIP model for an optimal PM schedule takes an excessive amount of time, especially as the number of tasks increases, and even more so as each tool has increasing number of tasks to be scheduled. This results in a challenge in a production environment where scheduling and planning decisions change dynamically and manufacturing needs prompt solutions. Although the decomposition algorithm is an attempt to resolve this computational problem, we cannot guarantee that it can work efficiently on every type of problem, as its success also inherently depends on solving integer programming problems (both tool and master problems) efficiently. Therefore, we focus on developing a fast heuristic algorithm. The heuristic is based on the idea of filling deep "cavities" in the capacity loss profile by first scheduling the tasks that cause larger capacity losses in those cavities. That way, the tasks that have large effects on the capacity loss profile are assigned first.

This idea is based on the Longest Processing Time (LPT) rule logic used in the analogous parallel machine makespan problem (Graham, 1969).

Suppose PM tasks are sorted in non-increasing order of their capacity loss contributing to the common capacity loss profile. The tasks in this order are re-indexed with $v = 1,..., \sum_{i \in I} N_i$; the smaller the index, the bigger the capacity loss. The capacity loss is calculated as $(1 - f_{ik}) s_i, \forall i, k \in K_i$. Let p_v denote the processing time, and let f_v denote the availability of the v^{th} task. Also, let $i(v)$ denote the tool which v^{th} task is assigned. We use variable W_t as the capacity loss level in time period t, which is initialized to be 0 before any PM tasks are scheduled. The steps of the heuristic are as follows.

1. Initialize the task to be scheduled: $v \leftarrow 1$

   Initialize $W_t = 0, \forall t$ $W_t = 0, \forall t$

   Initialize unavailabilities of tools for each time period: $u(i,t) = 0, \forall t, i \in I$.

2. Update the unavailabilities of the tool for each time period: $u(i(v),t) \leftarrow u(i(v),t) + (1 - f_v), t < T - p_v + 1$.

3. Calculate the accumulated capacity loss over the task processing time,

   $W_t = \sum_{\tau=t}^{t+p_v-1} W_\tau, \forall \mathrm{t} \in \{\delta : u(i(v),\alpha) \le 1, \alpha = \delta,..., \delta + p_v - 1\},$

   $W_t = \infty$, o.w.

4. Find the time period in which task v is to start, $t_{\min} = \arg\min_{t \in T}(W_t)$.

   If $\min(W_t) < \infty$ Update the capacity loss profile:

   $W_t \leftarrow W_t + (1 - f_v) s_{i(v)}, t = t_{\min},..., t_{\min} + p_v - 1$. Schedule task v to tool $i(v)$ as starting at time period $t_{\min}$.

   Else, stop and report that no feasible solution is found.

5. If $v = \sum_{i \in I} N_i$, stop and report the schedule. Otherwise, set $v \leftarrow v + 1$, and go to Step 2.

2.6.1 Numerical Example

The same sample data used in numerical examples of the MIP model is utilized for the heuristic algorithm. Resulting capacity losses and Gannt Chart from the solution of heuristic algorithm (we use the acronym "HA" for "heuristic algorithm") is in Figure 2.4.

Although the maximum capacity loss is the same as that of the MIP solution, the heuristic algorithm has a different solution (alternative optimal solution). This solution has less variance of capacity losses over time periods mainly due to the structure of the heuristic algorithm (since tasks are allocated starting from the largest capacity lost task to the smallest capacity lost task, deviation of capacity losses have a tendency to be lower than another method).

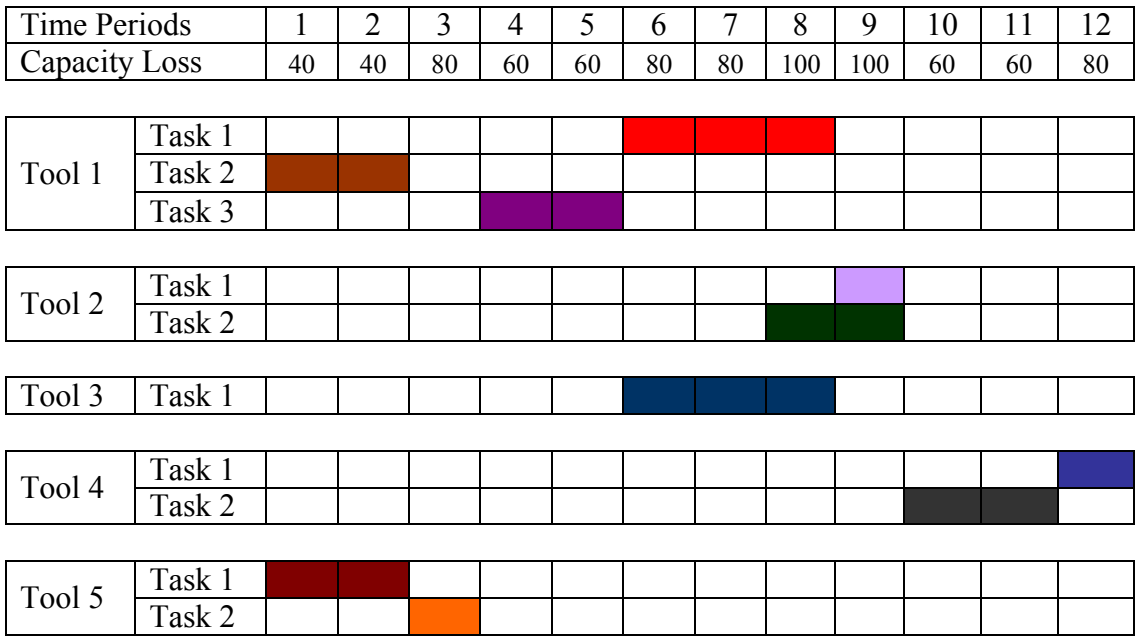


Figure 2.4 The Gantt chart of the HA solution

2.7 THEORETICAL INSIGHTS ON DECOMPOSITION APPROACH

We now build several theorems to find lower and upper bounds to the optimal maximum capacity loss using the decomposition approach for special cases of the problem. We first list three assumptions to specify these cases.

- A1 denotes the assumption of unit task times, $p_{ik} = 1, \forall i \in I, k \in K_i$.
- A2 is the assumption for the same availability per unit time value for all PM tasks, $f_{ik} = f, \forall i \in I, k \in K_i$.
- A3 is the integer value assumption for the tool planning horizons, i.e., $T_i = \frac{U_i}{U} T \in Z^+, \forall i \in I$.

For all the theorems and their corollaries, we further assume that tools have the same production capacity per unit time, $s_i = s, \forall i \in I$. Also, we assume that the problem data is consistent to find feasible solutions (such as task processing times are not greater

than the planning time horizon, total capacity of a tool is enough to handle all PM tasks allocated to give a feasible PM task allocation, etc.).

All the theorems in this section utilize the tool decomposition structure. While they do not have direct implications or relations to the decomposition algorithm, they provide analytical insights to the decomposition structure and provide analytical bounds for corresponding special cases.

Theorem 2 gives the simplest representation of the wall building analogy.

Theorem 2: Assume A1, A2, and A3 hold. The decomposition structure gives an optimal solution to the original problem. The optimal maximum capacity loss, W^{u*}, is

$$W^{u*} = \left\lceil \frac{\sum_{i \in I} |K_i|}{T} \right\rceil (1-f)s \,. \tag{2.19}$$

Proof: A1 and A2 are related to the problem input and we can find an optimal solution to the problem by considering the problem over the full planning horizon, T. The most straight forward solution to this problem is to assign individual tasks over periods row by row (starting from the first period, filling a row of tasks until all time periods have one task, and then assigning the remaining tasks, if any, to the first period again in the next row until all tasks are assigned). The maximum capacity loss value for this solution is given in (1.19). (The theorem claims that the tool based decomposition approach would reach the same solution.) A similar logic can be applied to each tool independently to find a solution for the tool level problem. On each tool i, we assign the tool's tasks as mentioned previously, but this time over T_i periods. The maximum capacity loss on the tool is $W_i^{u*} = \left\lceil \frac{|K_i|}{T_i} \right\rceil (1-f)s$. If we replace T_i with its value, $\frac{|K_i|}{\sum_{i \in I} |K_i|} T$ (under the unit

processing time and same availability assumptions), then the maximum capacity loss on tool i, W_i^{u*}, equals to $\left\lceil \frac{\sum_{i\in I}|K_i|}{T} \right\rceil (1-f)s$. This value is the same for all tools since it does not depend on i. Since $\sum_{i\in I} T_i = T$, we can put independent tool loads consecutively one after another in any order on the combined planning horizon, T, without any overlap. Then the maximum capacity loss value for the overall combined schedule would be the same as one tool's, $\left\lceil \frac{\sum_{i\in I}|K_i|}{T} \right\rceil (1-f)s$, which is equal to the optimal solution value found earlier. □

If we relax the same availability assumption, A2, then we cannot prove optimality. However, this approach leads to a lower bound. Then, we can build a feasible solution from this lower bound solution and produce an optimality gap for the feasible solution:

Theorem 3: Assume A1 and A3 hold and maximum unavailability among all tasks is less than the average unavailability of all tasks over time, $\max_{i\in I, k\in K_i} (1-f_{ik}) \le \frac{U}{T}$. The decomposition structure produces the following lower bound to the MIP problem,

$$W^* \ge \frac{sU}{T}. \tag{2.20}$$

Proof: We first find a direct combinatorial lower bound to the problem. The logic would be to distribute the total unavailability uniformly among the time periods in the horizon. Adding the capacity losses, we obtain $\frac{sU}{T}$. The same logic applies to each tool level problem independently. At tool i, we distribute the total unavailability over T_i periods.

The lower bound on the maximum capacity loss on the tool is $W_i^{u*} \geq \frac{sU_i}{T_i}$. If we replace T_i with its value, $\frac{U_i}{U}T$, then the lower bound equals to $\frac{sU}{T}$. This value is the same for all the tools i. Since $\sum_{i \in I} T_i = T$, we can put independent tool loads one after another on the combined planning horizon, T, in any order without any overlap. Then the lower bound on the maximum capacity loss for the overall combined schedule would be the same as one tool's, $\frac{sU}{T}$, which is the same as the combinatorial bound calculated earlier. We exclude the maximum unavailability being higher than the average unavailability to reduce ambiguity in the lower bound with adding an extra term for excluded cases. Without this exclusion, lower bound would be $W^* \geq \max\left[\max_{i \in I, k \in K_i}\left(1 - f_{ik}\right), \frac{sU_i}{T}\right]$. In the computational experiments, only around 25% of the experimental instances have the possibility to have maximum unavailability greater than the average unavailability over time.□

Corollary 1: If all the assumptions in Theorem 3 hold, then we obtain the following upper bound from the lower bound found in Theorem 3:

$$W^* \leq \frac{sU}{T} + \left(1 - \frac{U}{TU_i}\right) \max_{i \in I, k \in K_i} \left[s\left(1 - fi_k\right)\right]. \qquad (2.21)$$

Proof: We construct an upper bound using the lower bound in Theorem 3. At tool i, we take the task with the maximum unavailability out and find a looser lower bound, $W_i^* \geq \frac{sU_i - \max_{k \in K_i}\left[s\left(1 - f_{ik}\right)\right]}{T_i}$, which is equal to the average unavailability of remaining tasks over tool's planning horizon, T_i. A feasible PM task schedule can be obtained

assigning the largest unavailability PM task first to the time period with the least capacity loss (we do not assign the task with the largest availability since we also excluded it in the new lower bound). The deviations of the capacity loss values for each time period from the new lower bound obtained cannot be greater than the unavailability created by the largest unavailability PM task. Therefore, if we add the capacity loss due to the unavailability of this task, we obtain an upper bound. Then the upper bound is

$$W_i^* \le \frac{sU_i - \max_{k\in K_i}\left[s\left(1-f_{ik}\right)\right]}{T_i} + \max_{k\in K_i}\left[s\left(1-f_{ik}\right)\right].$$

The rest is mathematically rearranging the inequality so that we can obtain (2.21), as we illustrate below:

$$W_i^* \le \frac{sU_i}{T_i} - \frac{\max_{k\in K_i}\left[s\left(1-f_{ik}\right)\right]}{T_i} + \max_{k\in K_i}\left[s\left(1-f_{ik}\right)\right], \text{ and}$$

$$W_i^* \le \frac{sU_i}{T_i} + \left(1-\frac{1}{T_i}\right)\max_{k\in K_i}\left[s\left(1-f_{ik}\right)\right].$$

Replacing T_i with its value, $\frac{U_i}{U}T$, we obtain

$$W_i^* \le \frac{U}{T} + \left(1-\frac{U}{TU_i}\right)\max_{k\in K_i}\left[s\left(1-fi_k\right)\right].$$

The maximum of the tool lower bounds represents an upper bound for the overall problem, $W^* \le \max_{i\in I}\left(W_i^*\right)$. Hence, we obtain (2.21) as follows:

$$W^* \le \max_{i\in I}\left\{\frac{sU}{T} + \left(1-\frac{U}{TU_i}\right)\max_{k\in K_i}\left[s\left(1-fi_k\right)\right]\right\},$$

$$W^* \le \frac{sU}{T} + \left(1-\frac{U}{TU_i}\right)\max_{i\in I}\left\{\max_{k\in K_i}\left[s\left(1-fi_k\right)\right]\right\},$$

$$W^* \le \frac{sU}{T} + \left(1 - \frac{U}{TU_i}\right) \max_{i \in I, k \in K_i} \left[s(1 - fi_k)\right]. \square$$

Corollary 2: If all the assumptions in Theorem 3 hold, then the upper bound's relative optimality gap is as follows:

$$\frac{\left(1 - \frac{U}{TU_i}\right) \max_{i \in I, k \in K_i} \left[s(1 - fi_k)\right]}{\frac{U}{T}}. \tag{2.22}$$

Proof: We obtain the gap by subtracting the lower bound found in Theorem 3, (2.20), from the upper bound calculated in Corollary 1, (2.21), and divide this value to the lower bound, (2.20), to find the relative optimality gap in (2.22). □

Theorem 4: Assume A2 and A3 hold and $\max_{k \in K_i}(p_{ik}) \le T_i, \forall i \in I$. The decomposition structure produces the following lower bound,

$$W^{u*} \ge \frac{\sum_{i \in I} \sum_{k \in K_i} s(1-f) p_{ik}}{T}. \tag{2.23}$$

Proof: We first find a direct combinatorial lower bound to the problem. The logic would be to distribute the total unavailability uniformly among the time periods in the horizon. Adding the capacity losses, we obtain $\frac{\sum_{i \in I} \sum_{k \in K_i} s(1-f) p_{ik}}{T}$. The same logic applies to each tool level problem independently. At tool i, we distribute the total unavailability over T_i periods. The lower bound on the maximum capacity loss level on the tool is $W_i^{u*} \ge \frac{\sum_{k \in K_i} s(1-f) p_{ik}}{T_i}$. If we replace T_i with its value, $\frac{\sum_{k \in K_i} p_{ik}}{\sum_{i \in I} \sum_{k \in K_i} p_{ik}} T$, then the lower bound

equals to $\frac{\sum_{i\in I}\sum_{k\in K_i} s(1-f)p_{ik}}{T}$. This value is the same for all the tools. Since $\sum_{i\in I} T_i = T$, we can put independent tool loads one after another on the combined planning horizon, T, in any order without any overlap. Then the lower bound on the maximum capacity loss for the overall combined schedule would be the same as one tool's, $\frac{\sum_{i\in I}\sum_{k\in K_i} s(1-f)p_{ik}}{T}$, which is the same as the combinatorial bound calculated earlier. □

Corollary 3: If all the assumptions in Theorem 4 hold, then we obtain the following upper bound:

$$W^{u*} \le \left(\left\lfloor \frac{\sum_{i\in I}\sum_{k\in K_i} p_{ik}}{T} \right\rfloor + 1 \right)(1-f)s \,. \tag{2.24}$$

Proof: We construct a feasible solution by assigning the longest task to the smallest indexed time period with the lowest capacity loss first on a tool (similar logic as we obtain an optimal solution in Theorem 2). Then the upper bound on tool i becomes $W_i^{u*} \le \left(\left\lfloor \frac{\sum_{k\in K_i} p_{ik}}{T_i} \right\rfloor + 1 \right)(1-f)s$. When we replace T_i with its value, $\frac{\sum_{k\in K_i} p_{ik}}{\sum_{i\in I}\sum_{k\in K_i} p_{ik}} T$, then the upper bound equals to $W_i^{u*} \le \left(\left\lfloor \frac{\sum_{i\in I}\sum_{k\in K_i} p_{ik}}{T} \right\rfloor + 1 \right)(1-f)s$. As in Theorem 4, since

$\sum_{i\in I} T_i = T$, $W^{u*} \le \left(\left\lfloor \frac{\sum_{i\in I}\sum_{k\in K_i} p_{ik}}{T} \right\rfloor + 1 \right)(1-f)s$ becomes upper bound for the capacity loss of the tool group.□

Corollary 4: If all the assumptions in Theorem 4 hold, then the upper bound's relative optimality gap is as follows:

$$\frac{\left\lfloor \frac{\sum_{i\in I}\sum_{k\in K_i} p_{ik}}{T} \right\rfloor + 1 - \frac{\sum_{i\in I}\sum_{k\in K_i} p_{ik}}{T}}{\frac{\sum_{i\in I}\sum_{k\in K_i} p_{ik}}{T}}. \qquad (2.25)$$

Proof: We obtain the gap by subtracting the lower bound found in Theorem 4, (2.23), from the upper bound calculated in Corollary 3, (2.24), and divide this value to the lower bound, (2.23), to find the relative optimality gap in (2.25).□

If we further relax the uniform availability assumption, then we get a very loose lower bound. We can still obtain an upper bound and a loose optimality gap out of this lower bound.

Theorem 5: Assume A3 holds and maximum unavailability among all tasks is less than the average unavailability of all tasks over time, $\max_{i\in I, k\in K_i}(1-f_{ik}) \le \frac{\sum_{i\in I}\sum_{k\in K_i} p_{ik}(1-f_{ik})}{T}$. Also, assume that $\max_{k\in K_i}(p_{ik}) \le T_i, \forall i \in I$. The decomposition structure gives a lower bound to the MIP problem.

$$W^{u*} \geq \frac{\sum_{i \in I} \sum_{k \in K_i} p_{ik}(1 - f_{ik})s}{T}. \qquad (2.26)$$

Proof: We follow a similar logic used in the previous theorems. The combinatorial lower bound is $\frac{\sum_{i \in I} \sum_{k \in K_i} p_{ik}(1 - f_{ik})s}{T}$, found by distributing the total unavailability over the planning horizon, T. In the decomposition structure, we can similarly distribute the load which comes from unavailability caused by PM tasks over planning horizon, T_i and find a lower bound for tool decomposition problem, that is $W_i^{u*} \geq \frac{\sum_{k \in K_i} p_{ik}(1 - f_{ik})s}{T_i}$. Replacing T_i with its value, $\frac{\sum_{k \in K_i} p_{ik}(1 - f_{ik})}{\sum_{i \in I} \sum_{k \in K_i} p_{ik}(1 - f_{ik})} T$, we obtain a lower bound on maximum capacity loss on tool i, $\frac{\sum_{i \in I} \sum_{k \in K_i} p_{ik}(1 - f_{ik})s}{T}$. Since $\sum_{i \in I} T_i = T$, we can put independent tool loads one after another on the combined planning horizon, T, in any order without any overlap. Then the lower bound on the maximum capacity loss for the overall combined schedule is that same as any one of the tools', $\frac{\sum_{i \in I} \sum_{k \in K_i} p_{ik}(1 - f_{ik})s}{T}$, which leads to the same bound as the combinatorial bound.□

The upper bound and optimality gaps are found in the following corollaries.

Corollary 5: If all assumptions for Theorem 5 hold, then we get following upper bound:

$$W^{u*} \le \frac{\sum_{i\in I}\sum_{k\in K_i} s(1-f_{ik})}{T} + \left[1 - \frac{p_{\arg\max_{i\in I,k\in K_i}(1-f_{ik})}\sum_{i\in I}\sum_{k\in K_i} p_{ik}(1-f_{ik})}{T\sum_{k\in K_i} p_{ik}(1-f_{ik})}\right] \max\left[(1-ft_k)s\right]. \quad (2.27)$$

Proof: The construction of an upper bound is similar to Corollary 2. Exclude the task with highest unavailability per time period. Form a new looser lower bound. Add excluded task on top and obtain an upper bound for the optimal value. □

Corollary 6: If all assumptions for Theorem 5 hold, then we get following relative optimality gap:

$$\frac{\left[1 - \frac{p_{\arg\max_{i\in I,k\in K_i}(1-f_{ik})}\sum_{i\in I}\sum_{k\in K_i} p_{ik}(1-f_{ik})}{T\sum_{k\in K_i} p_{ik}(1-f_{ik})}\right] \max\left[(1-f_{ik})s\right]}{\frac{sU}{T}}. \quad (2.28)$$

Proof: Similar to Corollary 2's proof. □

For more general cases of the problem, while upper bounds are not valid any more, lower bounds become very loose. However, it is worth to further investigate the theoretical contribution of tool decomposition approach to the PM assignment problem.

In the next section, we present a heuristic algorithm to solve large scale PM task problems within relatively short computational time since both MIP and decomposition algorithm approaches may take excessive amounts of time for large scale problems as they are both based on solving the integer programming models.

2.8 COMPUTATIONAL EXPERIMENTS

We have designed a set of experiments to analyze the behavior of alternative formulations and solution methods. We use 5 factors each with 2 levels. Table 2.2 lists these factors and their levels.

Table 2.2 Experimental factors and their levels

Experimental Factors	Levels
Planning Horizon	- Daily (24 hours) - Weekly (168 hours)
Number of Tools	- 10 - 30
Task/Tool Ratio	- 2 (20 tasks for 10 tools and 60 tasks for 30 tools) - 4 (40 tasks for 10 tools and 120 tasks for 30 tools)
Processing Times	- Discrete Uniform [1,3] (hours) - Discrete Uniform [1,7] (hours)
Availabilities	- One of 60%, 80% - One of 0%, 20%, 40%, 60%, 80%

Experiments are mainly set to understand the behavior of the three proposed methods, (1) direct optimization of the MIP model, (2) the decomposition-based algorithm, and (3) the heuristic algorithm, for different settings in a fab. Therefore data sets represent fairly realistic settings of a semiconductor fab. Daily and weekly PMs are the most common types of time-based PMs. Tool families may contain from a handful of tools to 50 or more tools. Therefore our proposed values represent a variety of realistic fab settings. Availability data is based on the number of chambers a tool has. Having five chambers is a typical situation for a tool in a semiconductor fab. Therefore availability levels in the experimental design are proportional to the number of chambers shut down at the same time for maintenance. We create 5 instances for each experimental point (combination of levels of experimental factors) and solve each instance with all solution methods. There are 160 problem instances in total solved across 32 experimental points. We formulate the MIP models in GAMS (Rosenthal 2007) and solve them using CPLEX 9.0 (ILOG 2003). We set a 1 hour limit for solution time. Heuristic algorithm is coded in Java (Sun Microsystems 2009).

If the decomposition algorithm gives the same or better objective function value as the overall MIP within the same solution time or faster in majority of the cases for an experimental point, then we can conclude that the decomposition algorithm works better as compared to the direct optimization of the overall MIP model for that specific experimental point. Similar arguments are valid for comparison of solutions of the heuristic algorithm and the direct optimization of the overall MIP.

We first compare the decomposition algorithm and the direct optimization of the MIP model for all experimental points. Since we run the heuristic algorithm only for weekly instances, we discuss its comparison with the optimization based methods separately after the comparison of the direct optimization and the decomposition algorithm.

The decomposition algorithm reports the same objective function value as the direct optimization's value and works faster than or as fast as MIP in 75 instances out of all 160 instances. Table 2.3 summarizes percentage gaps between the capacity loss values of direct optimization and decomposition averaged over individual instances on each experimental point (percentage gap = 100x(capacity loss of decomposition-capacity loss of direct optimization)/capacity loss of direct optimization). Table 2.4 compares the average solution times between the direct optimization and the decomposition algorithm.

Table 2.3 Average percentage gaps between the objectives of the direct optimization and decomposition algorithm

		Process Time ~ Discrete Uniform [1,3]		Process Time ~ Discrete Uniform [1,7]	
Planning Horizon	Number of Tools/Number of Tasks	Availability ~ 60% or 80%	Availability ~ 0%, 20%, 40%, 60% or 80%	Availability ~ 60% or 80%	Availability ~ 0%, 20%, 40%, 60% or 80%
1 Day	10/20	13.33	3.33	7.33	6.49
	10/40	19.33	9.30	5.89	13.02
1 Week	10/20	0	0	0	0
	10/40	0	0	16.67	0
1 Day	30/60	0	1.25	0	2.00
	30/120	2.50	1.25	1.96	5.12
1 Week	30/60	0	0	37.50	N/A
	30/120	33.33	N/A	N/A	N/A

Table 2.4 Average solution times of the direct optimization and decomposition algorithm in seconds (MIP: the direct optimization, DA: the decomposition algorithm)

		Process Time ~ Discrete Uniform [1,3]				Process Time ~ Discrete Uniform [1,7]			
		Availability ~ 60% or 80%		Availability ~ 0%, 20%, 40%, 60% or 80%		Availability ~ 60% or 80%		Availability ~ 0%, 20%, 40%, 60% or 80%	
Planning Horizon	Number of Tools/Number of Tasks	MIP	DA	MIP	DA	MIP	DA	MIP	DA
1 Day	10/20	3206.74	720.55	2160.38	755.81	3600.00	900.00	3600.00	545.37
	10/40	2160.11	1128.45	2881.13	1582.54	3600.00	454.29	3600.00	90.81
1 Week	10/20	1017.55	13.54	1723.55	10.03	635.99	56.74	3142.73	29.75
	10/40	1040.48	214.19	3600.00	39.77	973.04	2437.55	2853.04	1309.94
1 Day	30/60	2880.60	2885.40	3600.00	3600.00	2880.14	2977.90	3600.00	3600.00
	30/120	3600.00	3600.00	2882.67	3600.00	3600.00	3600.00	2896.37	3600.00
1 Week	30/60	1142.57	126.26	3600.00	39.69	2719.13	206.63	2708.86	3600.00
	30/120	3600.00	2906.25	2836.70	3600.00	3600.00	3600.00	3600.00	3600.00

Since in some of the instances either the direct optimization or the decomposition algorithm cannot find a feasible solution in 3600 seconds, we use the averages of the remaining instances while preparing Table 2.3. However solution times listed in Table 2.4 are averages of all instances whether or not any solution method could find a feasible solution. For the experimental points where the direct optimization or the decomposition algorithm cannot find a feasible solution in an hour of solution time, we enter "N/A."

Table 2.3 lists many cases (75 out of 160) where the gap between the direct optimization's objective value and the decomposition algorithm's value is 0%. Direct optimization proves optimality in 38 out of these instances. Therefore, the decomposition algorithm also reaches the optimal value in the same instances where it works faster than the direct optimization. In such cases, decomposition algorithm is almost 60% faster than the direct optimization (1052 seconds vs. 430 seconds).

Planning Horizon: While the direct optimization works better for daily PM scheduling, the decomposition algorithm shows a faster performance for weekly problems. The decomposition algorithm gives same result as the direct optimization's objective function value in 45 instances of 80 weekly instances, of which 42 of them shows that the decomposition algorithm is working faster. Overall, relative gap between the methods is 7.29% in weekly planning for the cases where the direct optimization and the decomposition algorithm can find feasible solution in an hour. However, in considerable amount of cases, the decomposition algorithm cannot find an optimal solution for weekly planning instances. MIP solves 85% of such cases.

Number of Tools: The decomposition algorithm dominates 10 tool instances. In 46 of 51 instances where relative gap is 0%, the decomposition algorithm works faster. While the direct optimization solves 10 tool instances in 2487 seconds on average, the decomposition algorithm spends 643 seconds on average. Relative gap for 10 tool instances is 5.92%. The direct optimization of MIP and the decomposition algorithm spend very close amount of time on average for 30 tool instances. However, the decomposition algorithm has problem to find feasible solution in limited amount of time. Therefore, the direct optimization is preferable for 30 tool instances.

Task/Tool Ratio: The decomposition algorithm shows good performance in majority of the instances of 10 tool 20 task instances and 30 tool 60 task instances. However, as the number of tasks per tool increases, direct optimization of the original MIP model starts to show better performance. Among 80 instances of 10 tool, 40 task, and 30 tool, 120 task instances, the decomposition algorithm gives the same objective function value as that of the direct optimization of the MIP model, and only in 20 of them it works faster than the direct optimization. The decomposition algorithm gives close results to the direct optimization's value (relative gap is 4.75%) for lower task/tool ratio in a shorter amount of time (1254 seconds vs. 2639 seconds).

Processing Times: The decomposition algorithm works faster for both high and low variability instances. However, it works better in low variability case because relative gap is smaller (5.58%) and it faces infeasibility problem in limited amount of solution time in high variability instances.

Availabilities: The decomposition algorithm is working better in higher variability cases while the direct optimization of the MIP is more reliable for lower variability cases.

Among all the experimental points, the decomposition algorithm works best when there is weekly planning horizon with 10 tools and 20 tasks. The decomposition algorithm can be replaced with the direct optimization of the MIP, in most of the instances of weekly planning with 30 tools and 120 tasks.

Figure 2.5 depicts effects of dual experimental factor interactions. There are low and high levels of each of five factors given in Table 2.2. For every interaction of each level of each factor, we list on which solution method (the direct optimization of the MIP or the decomposition algorithm) works better. Every quarter square in the table

corresponds to 60 instances. If at least half of these instances result for the favor of the decomposition algorithm (it means the relative bound between the results of the decomposition algorithm and the direct optimization of the MIP is 0% but the decomposition algorithm ends faster than the direct optimization), then we conclude that the decomposition algorithm works for that interaction. Otherwise, we state that the direct optimization of the MIP works better.

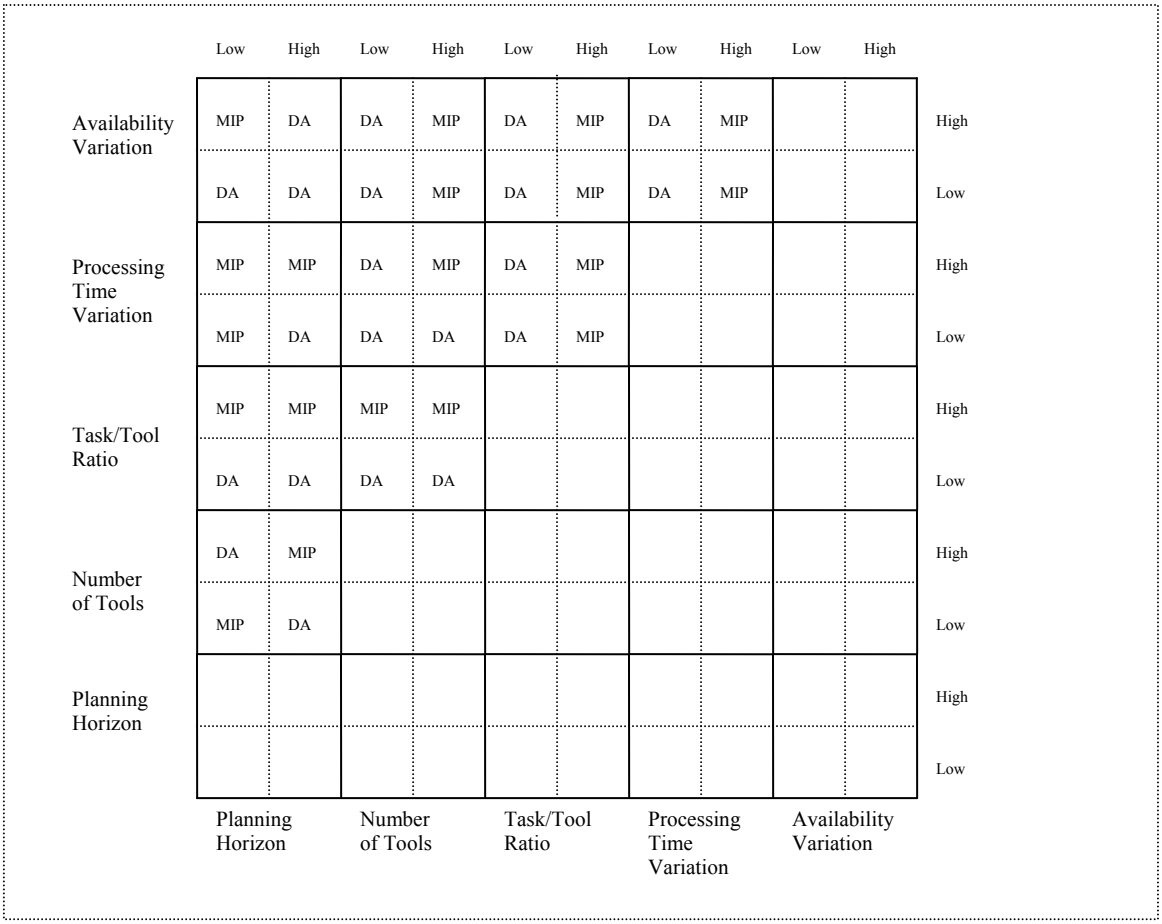


Figure 2.5 Interactions of experimental factors

The direct optimization works best when there is high task/tool ratio and tool count is large. Also, cases with daily planning in a high task/tool ratio are the ones where MIP works better.

The decomposition algorithm shows its best performance with the interaction of low task/tool ratio and low processing time variation. It also works significantly better when there is low task/tool ratio and small number of tools or low availability variation.

Since the weekly instances are usually bigger in scale, the optimization based methods ((both the overall model and the tool–based decomposition models) take excessive amount of solution time. Daily scheduling instances are taken care with both optimization based methods. Therefore we ran experiments for the heuristic algorithm, for the cases of weekly schedules. The heuristic algorithm overcomes solution time problem efficiently although it does not guarantee high quality results in all instances. Heuristic algorithm is coded in Java programming language (Sun Microsystems 2009).

The heuristic algorithm works more efficient than MIP in 65 instances of 80 weekly scheduling instances (instances with 168 hours). It means that the heuristic algorithm reaches the same result as the direct optimization's value in a shorter amount of time. In 5 of these cases, CPLEX (ILOG 2003) cannot even find a feasible solution in an hour of computation time to the original MIP model, but the heuristic algorithm reports its best solution within 40 seconds in the worst case. The heuristic usually finds a feasible solution less than a second. While Table 2.5 shows percentage gaps between the heuristic algorithm objective and the MIP objective along with the gaps between the decomposition algorithm and the heuristic algorithm, each averaged over instances of each experimental point, Table 2.6 shows the average solution times among these three solution methods.

Table 2.5 Percentage gaps of averages of upper bound values of weekly planning horizon instances of the direct optimization of the MIP, the decomposition algorithm, and the heuristic algorithm results (MIP: the direct optimization of the MIP, DA: the decomposition algorithm, HA: the heuristic algorithm)

		Process Time ~ Discrete Uniform [1,3]						Process Time ~ Discrete Uniform [1,7]					
		Availability ~ 60% or 80%			Availability ~ 0%, 20%, 40%, 60% or 80%			Availability ~ 60% or 80%			Availability ~ 0%, 20%, 40%, 60% or 80%		
Planning Horizon	Number of Tools/Number of Tasks	MIP-DA	MIP-HA	DA-HA	MIP-DA	MIP-HA	DA-HA	MIP-DA	MIP-HA	DA-HA	MIP-DA	MIP-HA	DA-HA
1 Week	10/20	0	0	0	0	0	0	0	0	0	0	0	0
	10/40	0	0	0	0	0	0	16.67	0	-11.11	0	0	0
1 Week	30/60	0	0	0	0	0	0	37.50	12.50	-20.00	N/A	23.33	N/A
	30/120	33.33	0	-25.00	N/A	40.00	N/A	N/A	6.67	N/A	N/A	12.78	N/A

Table 2.6 Average of solution times of weekly planning horizon instances of the direct optimization of the MIP, the decomposition algorithm and the heuristic algorithm in seconds (MIP: the direct optimization of the MIP, DA: the decomposition algorithm, HA: the heuristic algorithm)

		Process Time ~ Discrete Uniform [1,3]						Process Time ~ Discrete Uniform [1,7]					
		Availability ~ 60% or 80%			Availability ~ 0%, 20%, 40%, 60% or 80%			Availability ~ 60% or 80%			Availability ~ 0%, 20%, 40%, 60% or 80%		
Planning Horizon	Number of Tools/Number of Tasks	MIP	DA	HA	MIP	DA	HA	MIP	DA	HA	MIP	DA	HA
1 Week	10/20	1017.55	13.54	0.88	1723.55	10.03	0.76	635.99	56.74	1.17	3142.73	29.75	1.14
	10/40	1040.48	214.19	2.52	3600.00	39.77	2.32	973.04	2437.55	3.03	2853.04	1309.94	2.94
1 Week	30/60	1142.57	126.26	7.50	3600.00	39.69	7.02	2719.13	206.63	9.70	2708.86	3600.00	10.29
	30/120	3600.00	2906.25	23.70	2836.70	3600.00	24.24	3600.00	3600.00	34.44	3600.00	3600.00	36.74

Table 2.5 also indicates that in many cases (65/80) the heuristic algorithm reaches the same result as the direct optimization of the MIP, but in a shorter solution time as Table 2.6 shows. In 37 of these 65 solutions, the direct optimization of the MIP proves optimality. It means that, the heuristic algorithm also gives optimal solution in these 37 cases.

The heuristic algorithm works better than the direct optimization of MIP for all instances with 10 tools: It gives the same objective values as MIP within 3 seconds.

There are cases in which MIP beats the heuristic algorithm, namely instances with 30 tools and high processing time and availability variation, and instances 30 tools and

120 tasks with low processing time variation and high availability variation. Using the heuristic algorithm can be still advantageous in some of these cases due to its significant reduction in computation time.

For all weekly PM scheduling problem instances, the heuristic algorithm clearly beats the decomposition algorithm. In all the cases, the heuristic algorithm finds solutions with same or lower objective values than those of the decomposition algorithm within a fraction of computation time used by the latter.

The heuristic algorithm is the most practical algorithm to solve weekly PM scheduling problems among all three solution methods. While the direct optimization of the MIP works better for some daily PM scheduling problems, the decomposition algorithm is preferred when there is lower processing time variability, availability variability, tool count or task/tool ratio. Overall, none of the algorithms dominates the other two consistently in all experimental points.

2.9 CONCLUSION

Assigning PM tasks over a tool group in a semiconductor fab is a hard problem. Formulating and solving this problem with mixed integer programming is not sufficient for all the cases due to the long solution times. Therefore, we developed a tool based decomposition algorithm to solve the medium to large scale PM scheduling problems in which the direct optimization is not fast enough. We also developed a heuristic algorithm inspired by the analogy between the PM problem and the parallel machine job scheduling problem for even larger problems.

Our experimental design represents fairly realistic fab conditions with tool and PM task characteristics. Our analysis over different situations helped us find out which solution method is preferable for which cases. Since the decomposition algorithm offers

tool based PM schedules, it works well to find a high-quality feasible solution for each tool faster while the direct optimization is a more holistic approach. For weekly PM scheduling, the heuristic algorithm beats the other two solution methods by finding good feasible solutions in a matter of few seconds.

Our next step will be to refine the master problem of the decomposition algorithm to make it work faster and find ways to decrease the solution times for both the direct optimization and the decomposition algorithm

Chapter 3: Lot Size Management

Semiconductor wafers conventionally travel in a wafer fabrication facility (fab) in fixed lot sizes. Wafers move in lots between storage areas and tools (machines in fabs are called "tools") in a fab for various reasons. Therefore, lots in a semiconductor fab behave as what are traditionally called "transfer batches." One reason to move material in lots is to decrease the load on the material handling system in the fab. Also, carrying the same type of wafers together in a lot potentially decreases the number of setups. It is easier to follow the movement of a lot of wafers rather than to follow them separately. Lot sizes do not typically change as the lots move within the fab, except when batches (of lots) are formed in batch processing tools. The current convention in wafer fabrication is a lot size of 25 wafers. Carriers in semiconductor manufacturing are called FOUPs (Front Opening Unified Pods). Every FOUP is identical and designed to carry a maximum of 25 wafers, which facilitates the fixed lot size used in many fabs. Semiconductor manufacturers are currently investigating the advantages and disadvantages of moving to a smaller lot size. Using a different fixed lot size has varying tradeoffs, some of which are investigated here.

In semiconductor manufacturing, many industry practices are based on the common understanding of the actors of the industry. This common understanding directs equipment suppliers to manufacture tools under agreed conditions. As in the lot size case, usually FOUPs are designed to carry 25 wafers at most. Similarly, tools are designed to work efficiently with 25 wafers. However, company needs and purposes are changing and many companies are moving in different directions with respect to lot sizing. They ask suppliers to design different FOUPs or tools for their needs, which generates

additional costs. It is also crucial to incorporate the effect of technology improvements when analyzing lot sizing decisions made by companies.

The main motivation for companies wishing to reduce lot sizes is that the "wait-to-batch" time for lots (wafers waiting each other to keep the lot in tact) is reduced when lot sizes are smaller. (Note that the term "wait-to-batch" typically refers to forming a batch from jobs, which translates into forming a lot from wafers in semiconductor manufacturing). This raises a question that occupies many managers in the industry: Will a lot size decrease directly result into a corresponding over all cycle time reduction? A cycle time decrease yields the benefit of responding more quickly to changes in the consumer market. However, the lower inventories which invariably result from lower cycle times also may limit a company's ability to quickly satisfy short-term demand spikes. Therefore, some companies prefer to hold higher inventories in order to avoid losing their edge in demand satisfaction.

Our ultimate aim is to design a tool that decides on the lot size which minimizes the cost (or cycle time) in different production environments. As a first step in designing such a tool, we analyze the effect of lot size and technology on cycle time using a stylized queueing model. In our models below, the lot size is a positive integer n, while m represents the technology level. This technology parameter encapsulates the effects of first wafer delay (FWD), setup, throughput rate, etc., on the tool. We assume that these effects are manifested in the processing time of lots. FWD is conventionally equal to the overall processing time of one single wafer on the tool (going through all the chambers in the predefined sequence of sub-processes). In a way, FWD occurs because cascading of wafers is not possible for the amount of time it takes to make sure that the first wafer is processed according to specifications and that tool adjustments are done for the

remaining wafers in the lot if needed. When a lot is introduced to the system, no wafer emerges from the tool for a time interval equal to the FWD. After waiting for FWD units of time, wafers are processed with the same rate as the tool's production rate. Decreasing FWD is an important part of decreasing raw processing time and cycle time. Also, in this case setups can be a concern, too, since increasing number of lots may increase the number of possible setups.

In the next section, we give a literature review on modeling and measuring semiconductor fab performance and the effects of lot size. The queueing model is presented and analyzed in Sections 3.2 and 3.3. Section 3.4 focuses on the case where there is a fixed technology in the fab. Section 3.5 includes comparison of the approximation for the fixed technology with simulation. We analyze lot size characteristics in this special case in Section 3.6. We conclude our work after presenting optimal lot size expressions for special cases in Section 3.7.

3.1 LITERATURE REVIEW

Cycle time and WIP reduction is a major focus in semiconductor manufacturing. There is a considerable amount of literature on measuring, evaluating and improving cycle time in semiconductor manufacturing. Simulation is one of the most important tools to model and to measure performance in a semiconductor fab. Detailed simulation execution times can be very long. How much detail to retain in a simulation model is a complex issue. Hung and Leachman (1999) show that accurate estimates of total cycle time and equipment utilization may be obtained using reduced fabrication simulation models that replace operations at low utilization workstations with fixed time lags. The criterion they use for deleting workstations from the model is the standard deviation of lot waiting time. Hunter et al. (2002) build a full scale fab simulation model from SEMATECH data to run special scenarios showing the effects of eliminating

maintenance and changing product types. Comparison of the full scale model to a small scale model built and validated with the same data set shows the inadequacy of the small scale model. Sivakumar and Chong (2001) provide a preliminary analysis of the effect of lot release controls, heuristic dispatching rules, elimination of selected processes, material handling time, setup time, and machine up time on the selected output variables of throughput, cycle time and cycle time spread. The focus is backend manufacturing, i.e., IC packaging and assembly, and IC burn-in and functional test. Lou and Kager (1989) suggest using flow rate control to model the wafer fabrication job shop. They present a robust control policy for shop level scheduling in the semiconductor wafer fabrication facility to reduce WIP. Wood (1996) studies integrated process tools. Integrated tools consist of several process modules connected around a central handler such that the modules can process several wafers from the same lot simultaneously. Wood introduces simple, intuitive models of the cycle throughput and wafer cost of the integrated tools. Tool operations are aggregated under two measurable parameters. The incremental cycle time is the average increase in cycle time resulting from a lot size increment of one wafer and the fixed cycle time is the portion of cycle time that is independent of lot size. These parameters help estimate tool cycle time, throughput, and cost per wafer using Wood's models. In Wood (1997) it is claimed that single wafer processing and integrated tools present an opportunity to decrease cycle time by 50% in semiconductor fabs. Such decreases also imply that optimal cycle times can be achieved with smaller lot sizes. Chen et al. (1998) study the problem of production planning and setup scheduling of multiple products in a facility which runs a single product type at a time. The objective is to determine the setup schedule and production rate that minimize the average total costs, which include the inventory, backlog and setup costs for each

product. The production rate is a decision variable different than it is in the classical economic lot scheduling problem. The combination of fully automated systems and single wafer processing significantly reduces queueing time. Ikeda et al. (2003) introduce a new technology which redesigns fab operations to eliminate excessive processing times. Such a redesign could make single wafer lot sizes more viable.

Queueing models have been a tool to investigate wafer fabs for many years. Chen et al. (1988) design a queueing network representation of a wafer production research plant to estimate the key system performances. In their model, there are approximately 50 tool families and varying lot sizes in the system. Their approximation error is within 10% of actual observations. Wein (1988) uses a Brownian network model to approximate a multiclass queueing network with input control. He shows that scheduling has a great impact on cycle time, with the largest improvements coming from discretionary input control rather than lot sequencing. Kumar and Kumar (2001) describe the scheduling function in semiconductor wafer fabs and identify the key tradeoffs to be evaluated in designing scheduling policies. They survey many commonly used sequencing rules and release policies used in semiconductor manufacturing. Furthermore, they present a queueing network model of a fab and discuss related analytical tools for both performance evaluation as well as dynamic control of such systems.

The effect of lot size on cycle time has been of increasing interest recently in semiconductor manufacturing. Kenyon et al. (2005) assess the impact that lot sizes can have on the operational variables that most influence overall equipment effectiveness, net profits, cycle time, throughput, WIP, and operating expenses. This study shows that progressively smaller lot sizes do not provide continuous improvements on the aforementioned metrics. Furthermore, they show that the impact of lot sizes on cycle

times is not significantly related to setup times. Jaber et al. (2006) investigate the lot sizing problem with respect to reducing setups, reworks, and interruptions to restore process quality. The results indicate that learning in setups and improvement in quality reduces the annual costs significantly, with possible savings of up to 40%. The results also show that accounting for the cost of reworking defective units when calculating the unit holding cost may not be unrealistic given that some researchers suggest using higher holding cost than the cost of money.

Using equation based models, Lopez and Wood (1998) show that serially configured tools significantly reduce the lot size required to achieve a given total throughput capacity. They show that, given the same conditions (such as tool capacity), the serial configuration delivers at least as much throughput as the parallel configuration, with the former being able to attain the throughput capacity at a smaller lot size than the latter. If the lot size and tool configuration can be jointly optimized, then the benefits of the serial configuration become even greater. In another similar study, Lopez and Wood (2003) develop optimal lot sizing and lot release policies for systems of cluster tools in wafer fabs, with the goal of attaining higher reliability and lower cycle times. Lot sizing influences system performance along with lot release policies, and the magnitude of their impact may depend on the system's configuration and reliability.

Wang and Wang (2006) develop a simulation model which can determine, under different bottleneck load conditions, the appropriate lot size to reduce cycle time. They demonstrate that small lot sizes should be used if the bottleneck utilization exceeds expectations. The study also shows that smaller lot sizes decrease cycle time when the lot size is higher than a critical value. Schmidt et al. (2006) investigate the implications of smaller lot sizes via simulation. The quality of the output from a fab's simulation model

depends heavily on the accuracy of the tool modeling. Motivated by this, the authors evaluate current modeling methods using lot sizes smaller than the conventional 25 wafer lot and propose a new method for modeling cluster tools with parallel chambers. Finally, Schmidt and Rose (2007) analyze the interactions and effects on cycle times of fab architecture characteristics, such as lot size, equipment configuration, batch size, equipment front end module, carrier buffering, and AMHS. They group the challenges of lot size reduction into two categories: equipment productivity (batch tools, number of load ports and product dependent setup times) and material handling system (AMHS move rates and AMHS system configuration).

The purpose of this research is to use a relatively simple queueing model to show how lot size, material handling, and technology assumptions interact to affect cycle times. In the semiconductor industry it is often assumed that a decrease in lot size will automatically translate into cycle time improvements. There are two general factors which might mitigate the small lot size benefits. The first is obviously the increased load on the AMHS. The second effect, which is more subtle, is how lot size is correlated with machine operations. In particular, clustering, first wafer delay, and setups all have a complex relationship with lot size characteristics. Although our model is somewhat crude, the results actually align in a general way with detailed fab simulations performed in (Zarifoglu et al. 2008). Furthermore, we hope that the research will spur more detailed quantitative studies, and discussion, on lot sizing decisions.

3.2 A STYLIZED QUEUEING MODEL

We first present a simple queueing model which could represent a tool, or a consolidated portion of the process flow in a wafer fab. Although such a model is obviously somewhat coarse, it still provides good general insights that are borne out in

more detailed models. We believe that such models are a good starting point for a rigorous analysis of the effects of lot size and technology improvement on cycle time, and other important metrics.

The model we use first is an M/G/1 queue, in which the arrivals occur according to a Poisson process, and service times are i.i.d., drawn from a general distribution. Recall that we assume that the lot size is a positive integer n. We assume that the arrival rate to the server is λ/n lots/hr. Since each lot is n wafers, notice that this represents a fixed release rate in *wafers* per hour, even when the lot size is changed. This way, we keep the same "throughput capacity" in the fab for different lot size cases. The service rate is μ/m lots/hr. Thus, the expected processing time for a lot on the server is m/μ hrs/lot. Recall that m is a parameter representing the fab technology level. From the expression above, we see that lower values of m represent better (faster) technology levels.

The expected processing time discussed above is the time between when the transfer carrier enters the server and when all the wafers are processed and collected in the carrier. It is crucial to note that the processing time of a lot is not directly proportional to the number of wafers in the lot. If processing a 25 wafer lot takes 50 minutes, processing a 10 wafer lot may not necessarily take 20 minutes. The 10 wafer lot may take longer since the setup, first wafer delay and other effects are somewhat independent of the lot size. With the parameter m, we consolidate all the setup, first wafer delay, throughput rate effects, etc. on the tool. While our model assumes that the technology parameter m can be adjusted independently of n, we assume it is greater than or equal to n. If $m = n$, then this would be the perfect technology case, where change in lot size directly translates into a commensurate change in processing time: if the processing time

of a 25-wafer lot is 50 minutes, then the processing time of a 10-wafer lot is 20 minutes. However, we often have that $m > n$ which indicates that the benefit, in terms of the total lot processing time, is smaller than the lot size change might imply, due to technological barriers.

The traffic intensity, which is assumed to be less than 1 for stability, is given by $\rho_s = \frac{m\lambda}{n\mu}$. The squared coefficient of variation (c.o.v.) of the processing time on the server is $c_{ss}^2 = \sigma_s^2 \mu^2$, which is independent of the lot size or technology. We obtain this by setting the variance of the processing time to $m^2\sigma_s^2$. The M/G/1 queueing model is depicted in Figure 3.1.

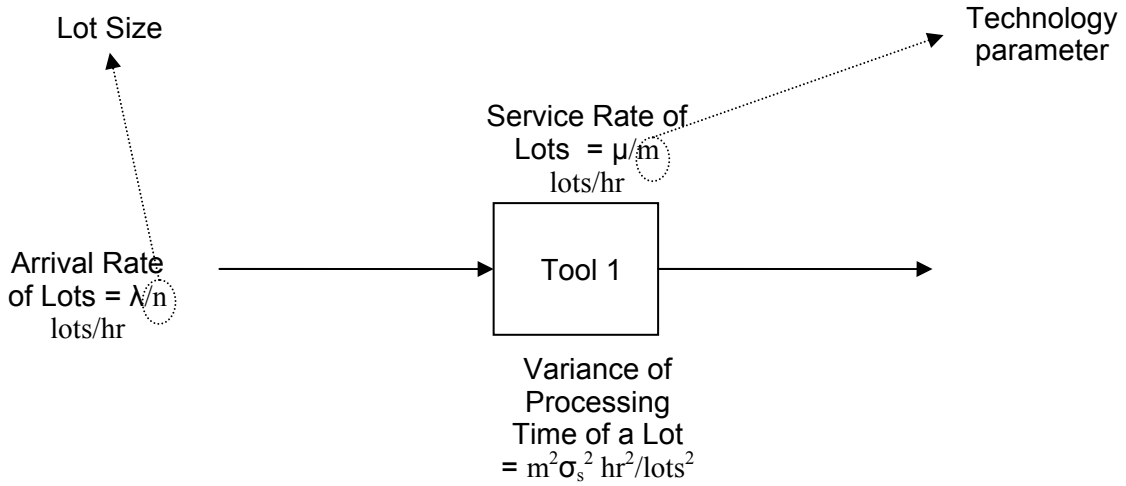


Figure 3.1 Single server queuing model

The performance measure of interest is the long-run average cycle time of lots. In the M/G/1 queue, the average waiting time (in queue), W_{qs} is given by the Pollaczek-Kintchine formula (Gross and Harris 2002):

$$W_{qs} = \frac{\rho_s\left(1+c_{ss}^2\right)}{2\mu_s\left(1-\rho_s\right)}. \tag{3.1}$$

Then, the cycle time expression, $W_{ss}(n,m)$, for this model is the sum of the average service time and average waiting time in queue,

$$W_{ss}(n,m) = \frac{m}{\mu} + \frac{\frac{m^2\lambda}{n\mu^2}\left(1+\sigma_s^2\mu^2\right)}{2\left(1-\frac{m\lambda}{n\mu}\right)}. \tag{3.2}$$

We aim to minimize the cycle time, as defined by (3.2). One can model the problem as a simple mixed-integer programming problem:

$$\begin{aligned} &\min W_{ss}(n,m) \\ &\text{s.t.} \\ &\lambda m - \mu n \le 0 \\ &n - m \le 0 \\ &m \ge 0 \\ &m \in R,\ n \in Z^+ / \{0\} \end{aligned} \quad , \tag{3.3}$$

where the lot size and technology are constrained by the requirement that the maximum traffic intensity is less than 1 and the technology parameter is bounded below by the lot size, as discussed above. The solution to (3.3) is trivial since (3.2) is decreasing in *n* and *m*. The optimal solution then is *n*=*m*=1. Figure 3.2 depicts the cycle time behavior depending on alternative parameter behaviors. These charts are prepared to show cycle time trends by using actual sample data points. When lot size is the only decision variable and the technology is fixed, cycle time has a tendency to decrease as lot size increases. When lot size is fixed, cycle time decreases by technology improvement. We can easily observe that lot size reduction by itself is not enough for cycle time decrease. The third chart shows that reduction of cycle time and improvement of technology reduces cycle time linearly as targeted by (3.3).

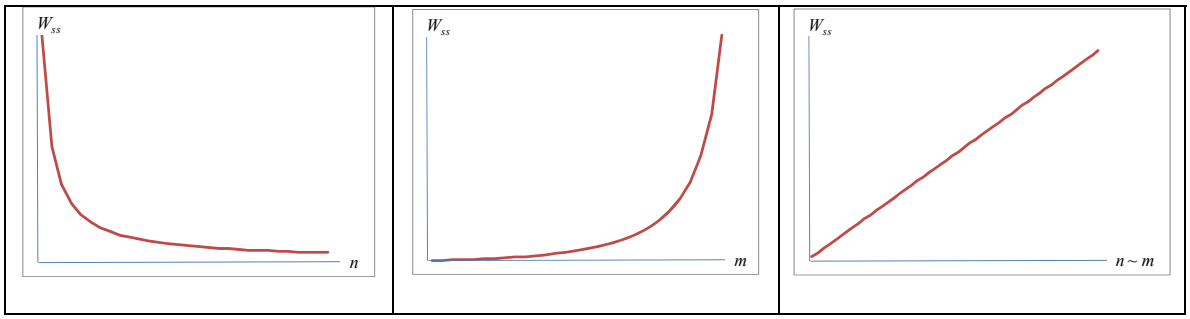


Figure 3.2 Cycle time behavior of single server in relation to lot size and technology

3.3 SINGLE SERVER AND SINGLE MATERIAL HANDLING SYSTEM QUEUEING MODEL

Next, we propose a tandem queueing model, where the second queue approximates the effects of the AMHS. In the previous section, we concluded that single wafer lots with "perfect" technology is the optimal choice when there is no other restriction on technology and the AMHS is not a restrictive factor. Thus, we consider two single server queues in tandem. The first queue represents a tool in the fab, and the second represents the AMHS. Suppose the material handling system has a service rate of τ lots/hr, which is independent of the lot size or the technology of the server. A reasonable assumption in most settings is that AMHS processing time has a very low coefficient of variation. In this model we assume it is 0, i.e., the AMHS processing times are constant, and equal to $\frac{1}{\tau}$ hrs/lot. The traffic intensity at the AMHS node is $\rho_h = \frac{\lambda}{n\tau}$, which is assumed to be less than 1. Since there is no exact formula for the total cycle time in such a system, we derive an approximation using the Queueing Network Analyzer - QNA (Whitt 1983).

The expected cycle time for our network, $W_{ssmh}^{QNA}(n,m)$, is found by summing the cycle time for the tool and the cycle time for the material handling system,

$$W_{ssmh}^{QNA}(n,m) = \frac{m}{\mu} + W_{qs} + \frac{1}{\tau} + W_{qh}. \tag{3.4}$$

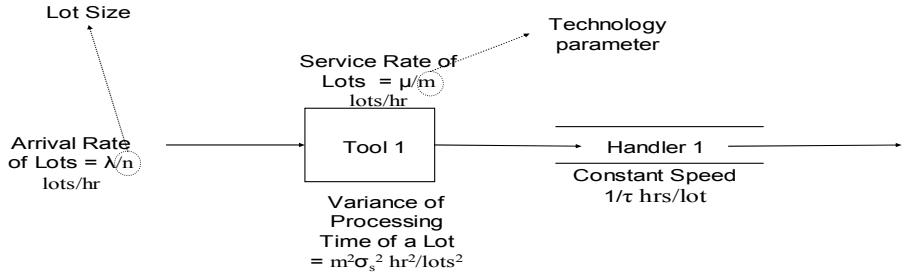


Figure 3.3 Single server single material handling system queuing model

Each of the nodes in our system can be thought of as a G/G/1 queue. The expected waiting time, W_{qj}, for a general node *j* can be approximated by:

$$W_{qj} = \frac{\rho_j \left(c_{aj}^2 + c_{sj}^2\right) g_j}{2\mu_j \left(1-\rho_j\right)}, \tag{3.5}$$

where

$$g_j = \begin{cases} \exp\left[\dfrac{2\left(1-\rho_j\right)}{3\rho_j}\dfrac{\left(1-c_{aj}^2\right)^2}{c_{aj}^2 + c_{sj}^2}\right], & c_{aj}^2 < 1 \\ 1, & c_{aj}^2 \geq 1 \end{cases}, \tag{3.6}$$

and c_{aj}^2 is the squared coefficient of variation for the arrival process. For the first node in our network, this quantity is just 1, since the arrival process is Poisson. However, the second node's arrival process is the departure process from the first node. Unfortunately,

there is no closed form expression for the c.o.v. for this departure process, which is where the QNA is useful.

The key term we need to find out in (5) is the squared coefficient of variation, c_{ah}^2 ,for the arrival process to the AMHS node. Using the QNA procedure gives an estimate of the squared c.o.v. of interarrival times to the material handling system,

$$c_{ah}^2 = \left(\frac{m\lambda}{n\mu}\right)^2 \max\left\{\sigma_s^2\mu^2, 0.2\right\} + 1 - \left(\frac{m\lambda}{n\mu}\right)^2 . \tag{3.7}$$

We can now use the G/G/1 approximation for W_{qh} , and substitute the expression in (3.4) which yields:

$$W_{ssmh}^{QNA}(n,m) = \frac{m}{\mu} + \frac{\frac{m^2\lambda}{n\mu^2}\left(1+\sigma_s^2\mu^2\right)}{2\left(1-\frac{m\lambda}{n\mu}\right)} + \frac{1}{\tau}$$
$$+ \frac{\left\{\begin{array}{l} \frac{1}{\tau}\frac{\lambda}{n\tau}\left[\left(\frac{m\lambda}{n\mu}\right)^2 \max\left\{\sigma_s^2\mu^2, 0.2\right\} + 1 - \left(\frac{m\lambda}{n\mu}\right)^2 + \sigma_h^2\tau^2\right] \\ \left\{\begin{array}{l} I_{\left(\frac{m\lambda}{n\mu}\right)^2 \max\left\{\sigma_s^2\mu^2,0.2\right\}+1-\left(\frac{m\lambda}{n\mu}\right)^2 <1} \exp\left\{\frac{-2\left(1-\frac{\lambda}{n\tau}\right)\left[\left(\frac{m\lambda}{n\mu}\right)^2\left(1-\max\left\{\sigma_s^2\mu^2,0.2\right\}\right)\right]^2}{3\frac{\lambda}{n\tau}\left[\begin{array}{l}\left(\frac{m\lambda}{n\mu}\right)^2 \max\left\{\sigma_s^2\mu^2,0.2\right\} \\ +1-\left(\frac{m\lambda}{n\mu}\right)^2 + \sigma_h^2\tau^2\end{array}\right]}\right\} \\ + I_{\left(\frac{m\lambda}{n\mu}\right)^2 \max\left\{\sigma_s^2\mu^2,0.2\right\}+1-\left(\frac{m\lambda}{n\mu}\right)^2 \geq 1} \end{array}\right) \end{array}\right\}}{2\left(1-\frac{\lambda}{n\tau}\right)} , \tag{3.8}$$

where $I_{(A)}$ is the indicator function for an event A.

For this tandem system, we can once again formulate a simple mixed-integer program:

$$\begin{aligned} &\min W_{ssmh}^{QNA}(n,m) \\ &\text{s.t.} \\ &-\mu n+\lambda m \le 0 \\ &-\tau n+\lambda \le 0 \\ &n-m \le 0 \\ &m \ge 0 \\ &m \in R, n \in Z^{+}/\{0\} \end{aligned} \quad . \qquad (3.9)$$

Although this problem is more complicated than (3.3), it is still relatively easy to solve. Since (3.8) is decreasing in m for the constraints defined by (3.9), the minimum cycle time is realized when $m=n$. Therefore, the minimum cycle time can be obtained by enumerating n over the range of feasible lot sizes. In the next section, we use the approximation just derived to explore some cases in which there is a particular relationship between the technology level and the lot size. As can be seen in Figure 3.4, cycle time behavior of single server and single material handling system is similar to Figure 3.2. These charts are also plotted using actual sample data points. Since material handling system is a constraint in (3.9), its effect can be seen when lot size reduction and technology improvement go hand in hand in the third chart.

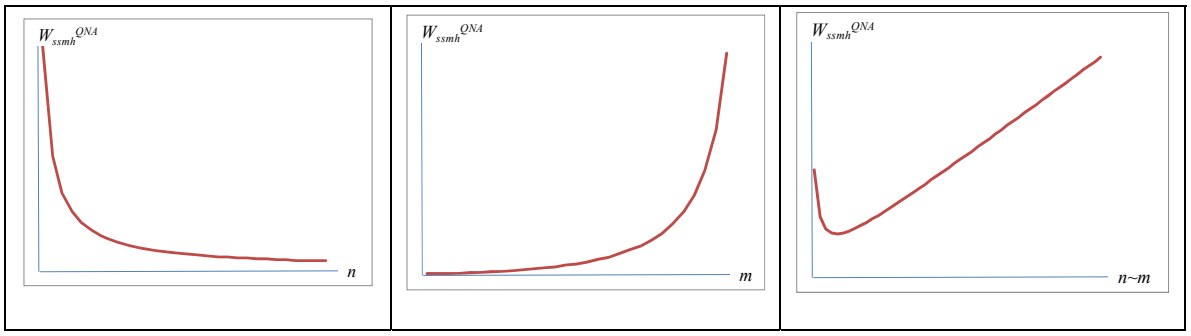


Figure 3.4 Cycle time behavior of single server single material handling system in relation to lot size and technology

3.4 FIXED TECHNOLOGY

Recall that $m = n$ corresponds to the case of "perfect" technology, i.e., the case in which technological innovations allow one to enjoy the maximum benefit of decreasing lot sizes. Since that is an ideal case, in this section we explore other cases in which technological innovation is less than ideal. We assume that there is a corresponding technology for each lot size which is defined by a linear relation, $m = a + \left(1 - \frac{a}{c}\right)n$, in which the current technology, c, matches the lot size, and the technology parameter is bounded below by the parameter a. Since n and m are coupled via the above expression, we can express the optimization problem with only one decision variable. However, due to the complexity of (3.8), a completely analytical solution is out of reach.

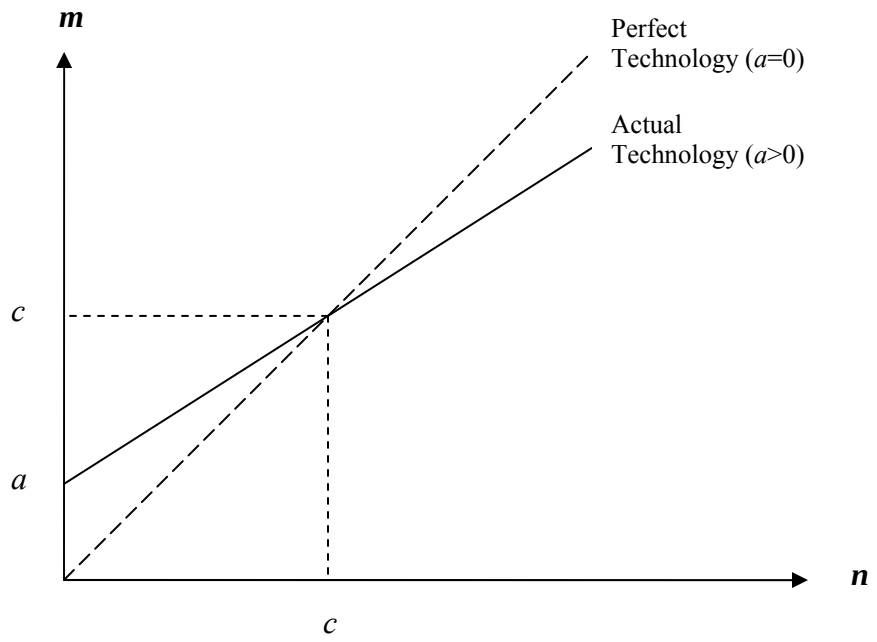


Figure 3.5 Technology - lot size relation

While a limits the technological improvement, c represents the current technology. In the perfect technology case, the technology limit, a, is 0. For lot sizes

smaller than the current size, the technology factor is "worse" than the perfect technology line as a increases. For bigger lot sizes, the technology factor is better than the perfect technology line. Bigger lot size along with smaller lot size is an option for companies especially running with little product differentiation. Although some companies consider moving to larger lot sizes, it comes with some extra investment on carrier size and material handling capabilities in addition to equipment changes, as in conversion to smaller lot sizes. Since current technology is configured according to current lot sizes, when moved to bigger lot sizes, even deviation up to some extent from current technology can be still better than the perfect technology case where technology change goes hand in hand with lot size. The corresponding optimization model is

$$\begin{aligned} &\min W_{ssmh}^{QNA}(n) \\ &\text{s.t.} \\ &-\mu n + \lambda\left[a + \left(1 - \frac{a}{c}\right)n\right] \le 0 \\ &-\tau n + \lambda \le 0 \\ &-n \le 0 \\ &n \in Z^{+}/\{0\} \end{aligned} \quad . \tag{3.10}$$

3.5 COMPARISON OF APPROXIMATION WITH SIMULATION

In the previous three sections, we developed an approximation for cycle time of a queuing model of a compact fab. We check the quality of the approximation by comparing it to a set of scenarios of simulations we run. The method we follow is to form a 95% confidence interval from multiple replications of each scenario and check if the value of the approximation of the corresponding scenario is within the 95% confidence interval derived from simulation results.

Table 3.1 Parameters and factor levels

Parameter	Value	Name of the factor
a	0	perfect technology
	5	average technology
	10	poor technology
λ	7 lots/hr	low base arrival rate
	9 lots/hr	high base arrival rate
σ_s^2	0 $lots^2/hr^2$	low server base service time variance
	0.01 $lots^2/hr^2$	average server base service time variance
	0.1 $lots^2/hr^2$	high server base service time variance
c	25	
μ	10 lots/hr	
τ	Varying values	
σ_h^2	0 $lots^2/hr^2$	
n	Integral values from 1 through 50	

The parameters we base our scenarios are listed in Table 3.1. There are three levels of technology limit, a, perfect, average and poor. Base arrival rate, λ, defines the arrival rate of lots independent of the lot size. Actual arrival rates are calculated by dividing base arrival rate by lot size. The main target by using a base arrival rate as an experimental factor is to keep the size of the wafer load same across experiments. Low base arrival rate corresponds to less frequent arrivals while high base arrival rate

corresponds to more frequent arrivals. Base service time variance, σ_s^2, defines service time independent of the technology. Actual service time variance is calculated my multiplying squared technology parameter by the base service time variance. Levels of base service time classify service variance from no variance to high variance. Current lot size, c, is 25 as conventionally used in the industry. Base service rate, μ, defines service rate of lots independent of the technology. Actual service rate is calculated by dividing base service rate by technology parameter. Material handling service rate, τ, is the service rate of lots by the material handling server. Values of material handling service rate is determined in relation to actual arrival rate of lots such that it maintains its utilization at less than 1 for the smallest lot size within a set of experiments (Values can be seen in Table 3.2). Our assumption of keeping material handling service rate constant across lot sizes is isolating improvement of AMHS from the production technology. We assume that material handling service time variance, σ_h^2, is zero. Lot sizes range from 1 to 50. However, within experiment sets we use feasible lot sizes depending on the utilization on the server.

Table 3.2 gives a detail list of sets of scenarios. For example, the first row corresponds to a set of scenarios, in which arrival rate is 7 lots/hr, handler service rate is 7.01 lots/hr and technology limit is 0. These experiments are repeated for three different service time variance levels. Since each lot size is feasible to run on the current settings of server and handler, we have lot sizes listed from 1 to 50 for these sets of scenarios. The first row corresponds to 150 different scenarios. Varying handler service rate values make high base arrival rate case busier or looser for the smallest lot size possible.

Table 3.2 Simulation scenarios

n	λ (lots/hr)	τ (lots/hr)	a	σ_s^2 (lots^2/hr^2)
1-50	7	7.01	0	0, 0.01, 0.1
2-50	9	7.01	0	0, 0.01, 0.1
1-50	9	9.01	0	0, 0.01, 0.1
8-50	7	0.88	5	0, 0.01, 0.1
17-50	9	0.88	5	0, 0.01, 0.1
17-50	9	0.54	5	0, 0.01, 0.1
13-50	7	0.54	10	0, 0.01, 0.1
20-50	9	0.54	10	0, 0.01, 0.1
20-50	9	0.46	10	0, 0.01, 0.1

Totally, there are 1080 scenarios. We run 100 replications of each scenario in the Arena (Rockwell Automation 2007) simulation software. Each replication is 2500 days long, 250 of which are warm-up period. Results are collected and 95% Confidence Intervals are created by Process Analyzer tool of Arena (Rockwell Automation 2007).

Table 3.3 presents the comparison of approximation values to simulation results. 'Confidence Interval Check" column shows how many of the approximation values of the scenarios are within the Confidence Intervals formed from simulation results out of the total number of scenarios with corresponding parameter values. For example, first row says that for low base arrival rate with corresponding server service rate when technology limit is 0 and server processing time variance is 0, out of 50 different lot size cases, cycle time approximation values of the 27 lot sizes are within the Confidence Interval of their

corresponding simulation results. Also, averages of the relative differences (errors) of cycle time approximation to simulation results of corresponding scenarios (*Relative Difference = |Cycle time approximation of a scenario-Cycle time output of a scenario|/ Cycle time output of a scenario*) is 0.66%. We provide comparison of approximation and simulation results to give an idea of how close these two are in addition to confidence interval checks. Also, the last column is an indicator of how narrow confidence interval is. For the first row, average of proportion of half-length of 50 different lot size scenarios to average of simulation results of same 50 different lot size scenarios is 0.68%, which makes average simulation results of these scenarios a good estimator of averages of means of corresponding 50 scenarios.

Table 3.3 Comparison of Approximation Values with Simulation Results

			n	λ	τ	a	σ_s^2	Confidence Interval		Comparison of Averages of Relative Errors						Half-Length / Simulation Mean						Percentage of Scenarios within Confidence Interval	Average Relative Error	Average Half-Length Proportion
								# of Scenarios within Confidence Interval	Total # of Scenarios	Average	Standard Deviation	Number of Scenarios	95% Half-Length	95% Low-End of Confidence Interval	95% High-End of Confidence Interval	Average	Standard Deviation	Number of Scenarios	95% Half-Length	95% Low-End of Confidence Interval	95% High-End of Confidence Interval			
				(lots/hr)	(lots/hr)		(lots^2/hr^2)																	
Low Server Base Service Time Variance	Perfect Base Technology	Scenario Group 1	1-50	7	7.01	0	0	27	50	**0.66%**	1.00%	50	0.28%	**0.38%**	**0.93%**	**0.68%**	1.85%	50	0.51%	**0.17%**	**1.19%**	54.00%	3.44%	1.23%
		Scenario Group 2	2-50	9	7.01	0	0	37	49	**1.24%**	0.43%	49	0.12%	**1.12%**	**1.36%**	**1.55%**	0.45%	49	0.13%	**1.42%**	**1.67%**	75.51%		
		Scenario Group 3	1-50	9	9.01	0	0	38	50	**1.94%**	5.22%	50	1.45%	**0.50%**	**3.39%**	**1.78%**	1.68%	50	0.46%	**1.32%**	**2.25%**	76.00%		
	Average Base Technology	Scenario Group 4	8-50	7	0.88	5	0	0	43	**5.39%**	19.34%	43	5.78%	**-0.39%**	**11.17%**	**0.72%**	1.71%	43	0.51%	**0.21%**	**1.23%**	0.00%		
		Scenario Group 5	17-50	9	0.88	5	0	9	34	**1.95%**	2.08%	34	0.70%	**1.25%**	**2.65%**	**1.61%**	1.41%	34	0.48%	**1.14%**	**2.09%**	26.47%		
		Scenario Group 6	17-50	9	0.54	5	0	0	34	**4.13%**	5.06%	34	1.70%	**2.43%**	**5.83%**	**1.55%**	1.40%	34	0.47%	**1.08%**	**2.02%**	0.00%		
	Low Base Technology	Scenario Group 7	13-50	7	0.54	10	0	0	38	**7.15%**	4.80%	38	1.53%	**5.62%**	**8.68%**	**0.68%**	1.90%	38	0.60%	**0.07%**	**1.28%**	0.00%		
		Scenario Group 8	20-50	9	0.54	10	0	0	31	**4.08%**	4.19%	31	1.47%	**2.61%**	**5.55%**	**1.30%**	1.90%	31	0.67%	**0.63%**	**1.97%**	0.00%		
		Scenario Group 9	20-50	9	0.46	10	0	0	31	**6.82%**	6.00%	31	2.11%	**4.71%**	**8.93%**	**1.28%**	1.89%	31	0.67%	**0.61%**	**1.95%**	0.00%		
Medium Server Base Service Time Variance	Perfect Base Technology	Scenario Group 10	1-50	7	7.01	0	0.01	44	50	**1.17%**	6.11%	50	1.69%	**-0.53%**	**2.86%**	**1.20%**	1.54%	50	0.43%	**0.77%**	**1.63%**	88.00%	2.18%	2.06%
		Scenario Group 11	2-50	9	7.01	0	0.01	48	49	**0.52%**	0.36%	49	0.10%	**0.42%**	**0.62%**	**2.61%**	0.89%	49	0.25%	**2.36%**	**2.86%**	97.96%		
		Scenario Group 12	1-50	9	9.01	0	0.01	48	50	**1.70%**	8.44%	50	2.34%	**-0.64%**	**4.04%**	**2.79%**	1.49%	50	0.41%	**2.37%**	**3.20%**	96.00%		
	Average Base Technology	Scenario Group 13	8-50	7	0.88	5	0.01	19	43	**4.24%**	19.50%	43	5.83%	**-1.59%**	**10.07%**	**1.23%**	1.30%	43	0.39%	**0.84%**	**1.61%**	44.19%		
		Scenario Group 14	17-50	9	0.88	5	0.01	30	34	**1.53%**	1.08%	34	0.36%	**1.16%**	**1.89%**	**2.83%**	1.81%	34	0.61%	**2.22%**	**3.44%**	88.24%		
		Scenario Group 15	17-50	9	0.54	5	0.01	20	34	**1.88%**	0.88%	34	0.30%	**1.59%**	**2.18%**	**2.62%**	1.46%	34	0.49%	**2.13%**	**3.11%**	58.82%		
	Low Base Technology	Scenario Group 16	13-50	7	0.54	10	0.01	3	38	**4.17%**	14.18%	38	4.51%	**-0.34%**	**8.68%**	**1.13%**	1.38%	38	0.44%	**0.70%**	**1.57%**	7.89%		
		Scenario Group 17	20-50	9	0.54	10	0.01	18	31	**2.53%**	5.51%	31	1.94%	**0.59%**	**4.46%**	**2.20%**	2.14%	31	0.75%	**1.45%**	**2.96%**	58.06%		
		Scenario Group 18	20-50	9	0.46	10	0.01	11	31	**2.67%**	5.13%	31	1.80%	**0.86%**	**4.47%**	**2.05%**	1.80%	31	0.63%	**1.41%**	**2.68%**	35.48%		
High Server Base Service Time Variance	Perfect Base Technology	Scenario Group 19	1-50	7	7.01	0	0.1	33	50	**15.80%**	66.93%	50	18.55%	**-2.75%**	**34.35%**	**7.74%**	2.18%	50	0.60%	**7.14%**	**8.35%**	66.00%	15.55%	8.83%
		Scenario Group 20	2-50	9	7.01	0	0.1	37	49	**7.72%**	3.34%	49	0.94%	**6.79%**	**8.66%**	**9.51%**	3.26%	49	0.91%	**8.59%**	**10.42%**	75.51%		
		Scenario Group 21	1-50	9	9.01	0	0.1	37	50	**20.54%**	91.18%	50	25.27%	**-4.73%**	**45.82%**	**9.48%**	3.23%	50	0.89%	**8.59%**	**10.38%**	74.00%		
	Average Base Technology	Scenario Group 22	8-50	7	0.88	5	0.1	15	43	**19.50%**	75.68%	43	22.62%	**-3.12%**	**42.12%**	**7.40%**	1.34%	43	0.40%	**7.00%**	**7.80%**	34.88%		
		Scenario Group 23	17-50	9	0.88	5	0.1	18	34	**10.79%**	13.00%	34	4.37%	**6.42%**	**15.17%**	**10.29%**	1.86%	34	0.63%	**9.67%**	**10.92%**	52.94%		
		Scenario Group 24	17-50	9	0.54	5	0.1	16	34	**12.50%**	15.59%	34	5.24%	**7.26%**	**17.74%**	**9.75%**	1.71%	34	0.57%	**9.17%**	**10.32%**	47.06%		
	Low Base Technology	Scenario Group 25	13-50	7	0.54	10	0.1	4	38	**22.52%**	76.93%	38	24.46%	**-1.94%**	**46.98%**	**6.63%**	1.03%	38	0.33%	**6.30%**	**6.96%**	10.53%		
		Scenario Group 26	20-50	9	0.54	10	0.1	17	31	**14.33%**	26.31%	31	9.26%	**5.07%**	**23.59%**	**9.96%**	2.72%	31	0.96%	**9.00%**	**10.92%**	54.84%		
		Scenario Group 27	20-50	9	0.46	10	0.1	16	31	**15.21%**	25.13%	31	8.84%	**6.36%**	**24.05%**	**9.39%**	2.30%	31	0.81%	**8.58%**	**10.21%**	51.61%		
Overall								545	1080	7.06%						4.04%						50.46%		

Table 3.4 Comparison of Approximation Values with Simulation Results - Relative Error Bounds

			n	λ (lots/hr)	τ (lots/hr)	a	σ_s^2 (lots^2/hr^2)	Confidence Interval Check		Relative Error Bounds						Average Relative Error Bound
								# of Scenarios within Confidence Interval	Total # of Scenarios	Average	Standard Deviation	Number of Scenarios	95% Half-Length	95% Low-End of Confidence Interval	95% High-End of Confidence Interval	
Low Server Base Service Time Variance	Perfect Base Technology	Scenario Group 1	1-50	7	7.01	0	0	27	50	**1.30%**	2.21%	50	0.61%	**0.69%**	**1.91%**	4.78%
	Average Base Technology	Scenario Group 2	2-50	9	7.01	0	0	37	49	**2.84%**	0.73%	49	0.20%	**2.64%**	**3.05%**	
		Scenario Group 3	1-50	9	9.01	0	0	38	50	**3.65%**	6.04%	50	1.67%	**1.98%**	**5.33%**	
		Scenario Group 4	8-50	7	0.88	5	0	0	43	**6.55%**	23.79%	43	7.11%	**-0.56%**	**13.67%**	
		Scenario Group 5	17-50	9	0.88	5	0	9	34	**3.68%**	3.75%	34	1.26%	**2.42%**	**4.93%**	
		Scenario Group 6	17-50	9	0.54	5	0	0	34	**5.87%**	7.01%	34	2.36%	**3.51%**	**8.22%**	
	Low Base Technology	Scenario Group 7	13-50	7	0.54	10	0	0	38	**7.77%**	5.75%	38	1.83%	**5.95%**	**9.60%**	
		Scenario Group 8	20-50	9	0.54	10	0	0	31	**5.58%**	6.75%	31	2.38%	**3.20%**	**7.95%**	
		Scenario Group 9	20-50	9	0.46	10	0	0	31	**8.36%**	8.79%	31	3.09%	**5.27%**	**11.46%**	
Medium Server Base Service Time Variance	Perfect Base Technology	Scenario Group 10	1-50	7	7.01	0	0.01	44	50	**2.52%**	8.62%	50	2.39%	**0.13%**	**4.91%**	4.41%
	Average Base Technology	Scenario Group 11	2-50	9	7.01	0	0.01	48	49	**3.14%**	0.86%	49	0.24%	**2.90%**	**3.38%**	
		Scenario Group 12	1-50	9	9.01	0	0.01	48	50	**4.67%**	10.93%	50	3.03%	**1.64%**	**7.70%**	
		Scenario Group 13	8-50	7	0.88	5	0.01	19	43	**5.81%**	22.86%	43	6.83%	**-1.02%**	**12.64%**	
		Scenario Group 14	17-50	9	0.88	5	0.01	30	34	**4.34%**	2.07%	34	0.70%	**3.64%**	**5.04%**	
		Scenario Group 15	17-50	9	0.54	5	0.01	20	34	**4.57%**	1.46%	34	0.49%	**4.08%**	**5.07%**	
	Low Base Technology	Scenario Group 16	13-50	7	0.54	10	0.01	3	38	**5.59%**	17.15%	38	5.45%	**0.14%**	**11.04%**	
		Scenario Group 17	20-50	9	0.54	10	0.01	18	31	**4.99%**	8.41%	31	2.96%	**2.03%**	**7.95%**	
		Scenario Group 18	20-50	9	0.46	10	0.01	11	31	**4.93%**	7.42%	31	2.61%	**2.32%**	**7.54%**	
High Server Base Service Time Variance	Perfect Base Technology	Scenario Group 19	1-50	7	7.01	0	0.1	33	50	**26.01%**	76.00%	50	21.07%	**4.95%**	**47.08%**	26.83%
	Average Base Technology	Scenario Group 20	2-50	9	7.01	0	0.1	37	49	**19.28%**	7.32%	49	2.05%	**17.23%**	**21.33%**	
		Scenario Group 21	1-50	9	9.01	0	0.1	37	50	**33.16%**	98.84%	50	27.40%	**5.76%**	**60.56%**	
		Scenario Group 22	8-50	7	0.88	5	0.1	15	43	**29.11%**	82.17%	43	24.56%	**4.55%**	**53.67%**	
		Scenario Group 23	17-50	9	0.88	5	0.1	18	34	**23.25%**	15.69%	34	5.28%	**17.98%**	**28.53%**	
		Scenario Group 24	17-50	9	0.54	5	0.1	16	34	**24.39%**	17.02%	34	5.72%	**18.67%**	**30.11%**	
	Low Base Technology	Scenario Group 25	13-50	7	0.54	10	0.1	4	38	**31.46%**	84.17%	38	26.76%	**4.70%**	**58.23%**	
		Scenario Group 26	20-50	9	0.54	10	0.1	17	31	**27.24%**	30.95%	31	10.89%	**16.35%**	**38.14%**	
		Scenario Group 27	20-50	9	0.46	10	0.1	16	31	**27.24%**	28.04%	31	9.87%	**17.37%**	**37.11%**	

Table 3.4 presents the average relative error bound (the relative difference of the cycle time approximation of cycle time to the farthest point of the corresponding confidence interval). For the low and medium variation cases is 4.59%. This bound is lower around the optimal lot size value while it is higher in the bounds.

Of 1080 scenarios, approximation values of 545 are within the confidence interval. The best fit ones are the scenarios with average server base service time variance case, 241 over 360. The worst fits are the ones with low server base service time variance scenarios, 111 over 360.

Approximation values are overall 7.06% relatively higher than simulation values. The closest results come from the average server base service time variance cases as in confidence interval check, which is 2.19%. Although confidence interval check points out that high server are service time variance case is better than low server base service time variance case, their relative difference from simulation values tell reverse, 15.55% vs. 3.44% correspondingly.

Half-length proportions to means of simulation results, which can be interpreted as narrowness of confidence intervals, change from one set of scenarios to another. Scenarios with low server base service time variance has narrowest confidence intervals, 1.23%. Average server base service time cases have pretty narrow confidence intervals, too, 2.06%.

Having a better working approximation for medium server base service time variance case is good for our purpose since usually a representative fab would be closer to this case. Usually, raw processing times have very low variation (most of the case, no variation). However, the complexity of the system, varying external and internal effects, such as break downs, first wafer delays and setups, bring some variation to the processing times. Therefore, it makes sense to translate this real-life situation to our model as average server base service time variation.

Table 3.5 Average fab scenarios

n	λ (lots/hr)	τ (lots/hr)	a	σ_s^2 (lots^2/hr^2)	Confidence Interval # of Scenarios within Confidence Interval	Confidence Interval Total # of Scenarios	Comparison of Averages of Relative Errors Average	Standard Deviation	Number of Scenarios	95% Half-Length	95% Low-End of Confidence Interval	95% High-End of Confidence Interval
12-50	8	0.685	5	0.01	22	39	**1.29%**	0.88%	39	0.28%	**1.01%**	**1.57%**
n	λ (lots/hr)	τ (lots/hr)	a	σ_s^2 (lots^2/hr^2)	Confidence Interval # of Scenarios within Confidence Interval	Confidence Interval Total # of Scenarios	Half-Length / Simulation Mean Average	Standard Deviation	Number of Scenarios	95% Half-Length	95% Low-End of Confidence Interval	95% High-End of Confidence Interval
12-50	8	0.685	5	0.01	22	39	**1.58%**	0.71%	39	0.22%	**1.35%**	**1.80%**
n	λ (lots/hr)	τ (lots/hr)	a	σ_s^2 (lots^2/hr^2)	Confidence Interval # of Scenarios within Confidence Interval	Confidence Interval Total # of Scenarios	Relative Error Bounds Average	Standard Deviation	Number of Scenarios	95% Half-Length	95% Low-End of Confidence Interval	95% High-End of Confidence Interval
12-50	8	0.685	5	0.01	22	39	**2.90%**	1.23%	39	0.39%	**2.51%**	**3.28%**

Table 3.5 represents an average fab. The approximation values are within the 95% Confidence Interval of the simulation results for 22 cases of the 39 scenarios. The overall approximation values are 1.07% higher than the simulation values relatively. Also, the half-length proportion to the averages of simulation results is 1.58% and relative error bound is 2.90%. We can conclude that our approximation works well for real-life situations.

Table 3.3 indicates that if we form confidence intervals of absolute errors within a scenario group (every row in Table 3.3 represents a scenario group which consists of feasible lot sizes for a specific set of parameters), low server base service time and medium server base service time cases have reasonable confidence intervals such that 95% of different lot size scenarios are within. Approximation does not work as efficiently for high server base service time cases.

Table 3.6 presents the behavioral comparison of approximation and simulation results.

Table 3.6 Comparison of Optimal Lot Sizes and Cycle Times of Approximation and Simulation Results

			n	λ (lots/hr)	τ (lots/hr)	a	σ_s^2 (lots^2/hr^2)	Minimum Cycle Time of Approximation	Minimum Cycle Time of Simulation	Lot Size Appoximation Minimizer	Lot Size Simulation Minimizer	Relative Difference of Minimum Cycle Times	Difference of Cycle Time Minimizer Lot Sizes
Low Server Base Service Time Variance	Perfect Base Technology	Scenario Group 1	1-50	7	7.01	0	0	0.613	0.576	2	2	6.34%	0
		Scenario Group 2	2-50	9	7.01	0	0	1.272	1.240	2	2	2.54%	0
		Scenario Group 3	1-50	9	9.01	0	0	1.220	1.208	2	2	0.97%	0
	Average Base Technology	Scenario Group 4	8-50	7	0.88	5	0	6.232	6.007	17	17	3.75%	0
		Scenario Group 5	17-50	9	0.88	5	0	13.667	13.507	33	32	1.19%	1
		Scenario Group 6	17-50	9	0.54	5	0	14.579	14.222	33	32	2.51%	1
	Low Base Technology	Scenario Group 7	13-50	7	0.54	10	0	7.684	7.222	29	27	6.40%	2
		Scenario Group 8	20-50	9	0.54	10	0	11.000	10.682	42	41	2.98%	1
		Scenario Group 9	20-50	9	0.46	10	0	11.477	11.005	43	41	4.29%	2
Medium Server Base Service Time Variance	Perfect Base Technology	Scenario Group 10	1-50	7	7.01	0	0.01	0.880	0.862	2	2	2.14%	0
		Scenario Group 11	2-50	9	7.01	0	0.01	2.271	2.226	2	2	2.00%	0
		Scenario Group 12	1-50	9	9.01	0	0.01	2.166	2.133	2	2	1.56%	0
	Average Base Technology	Scenario Group 13	8-50	7	0.88	5	0.01	9.436	9.234	20	21	2.19%	-1
		Scenario Group 14	17-50	9	0.88	5	0.01	23.163	22.685	35	35	2.11%	0
		Scenario Group 15	17-50	9	0.54	5	0.01	24.457	23.816	36	35	2.69%	1
	Low Base Technology	Scenario Group 16	13-50	7	0.54	10	0.01	10.548	10.286	34	34	2.55%	0
		Scenario Group 17	20-50	9	0.54	10	0.01	16.583	16.226	48	48	2.20%	0
		Scenario Group 18	20-50	9	0.46	10	0.01	17.155	16.744	49	48	2.45%	1
High Server Base Service Time Variance	Perfect Base Technology	Scenario Group 19	1-50	7	7.01	0	0.1	3.294	3.225	2	2	2.14%	0
		Scenario Group 20	2-50	9	7.01	0	0.1	11.303	11.018	2	2	2.58%	0
		Scenario Group 21	1-50	9	9.01	0	0.1	10.670	10.576	2	2	0.89%	0
	Average Base Technology	Scenario Group 22	8-50	7	0.88	5	0.1	36.992	33.082	23	22	11.82%	1
		Scenario Group 23	17-50	9	0.88	5	0.1	107.238	97.867	38	34	9.57%	4
		Scenario Group 24	17-50	9	0.54	5	0.1	111.481	101.641	40	34	9.68%	6
	Low Base Technology	Scenario Group 25	13-50	7	0.54	10	0.1	34.544	30.957	42	37	11.59%	5
		Scenario Group 26	20-50	9	0.54	10	0.1	65.046	59.783	50	45	8.80%	5
		Scenario Group 27	20-50	9	0.46	10	0.1	66.704	61.409	50	45	8.62%	5

All the perfect base technology cases give the same lot size for approximation and simulation. High server base service time variance cases deviate the most in terms of the best lot sizes. The best set of scenarios that the behavior fits is the one with medium server base service time variance.

In the next section, we use our queueing model to explore the relationship between technology factors and the optimal lot size.

3.6 ANALYSIS OF LOT SIZE CHARACTERISTICS

In real wafer fabs, the concept of an "optimal" lot size is quite situation dependent. Countless varying system parameters and unexpected conditions affect the performance of different lot size options. Our analysis of a stylized queueing model obviously simplifies a complex fab. However, the goal of the analysis is to see if we can spot general trends and tendencies which indicate intrinsic relationships between lot size and cycle time performance. Our model is represented by a few adjustable parameters, and we now summarize what these parameters represent.

Parameter a is the lower limit on the technology parameter, which represents the highest level of achievable technology. While lower a stands for better technology, higher a stands for worse technology. Parameter c is the current technology level. λ is the base arrival rate to the system and μ is the base service rate of the first server. τ is the service rate of the AMHS. The base variance σ_s^2 consolidates the variability effects of the server and σ_h^2 consolidates the variability effects of the AMHS. The c.o.v.'s $c_s^2 = \mu^2\sigma_s^2$ and $c_h^2 = \mu_h^2\sigma_h^2$ better encapsulate variability since they are unitless.

Let us first make some general observations about our model. As the lower limit on the technology parameter decreases, the optimal lot size decreases. In conjunction

with this the optimal cycle time decreases also. If the current technology can be improved upon, it makes more sense to move to smaller lot sizes. However, if significant improvements to technology cannot be made, the optimal lot size may actually be higher than the current lot size.

In our model, as in nearly every stochastic production model, a higher variance always increases the cycle time. Although the marginal change in cycle time may be small, an increased variance generally shifts the optimal lot size to slightly bigger values. Usually, the waiting time in the first queue (the processing queue) is the biggest portion of the cycle time, since we assume no variability in AMHS times. A higher base arrival rate increases both the overall cycle time and the optimal lot size. Finally, we note that decreasing the lower technology limit has the greatest effect on the optimal lot size.

Next, we present an analysis of lot size and cycle time characteristics using the parameters listed in Table 3.7. We later present another analysis, by taking the averages of the extreme values of the factors. We assume this represents average fab behavior.

We first list all the experiments in Table 3.7. Then we group and analyze them based on factors from Figures 3.6 through 3.13. Figure 3.14 represents an average fab followed by its analysis.

Table 3.7 Chart List

Charts	a	c	λ (lots/hr)	(lots/hr)	τ (lots/hr)	σ_s^2 (lots^2/hr^2)	σ_h^2 (lots^2/hr^2)
Chart 1	0	25	7	10	7.01	0	0
Chart 2	0	25	7	10	7.01	0.1	0
Chart 3	0	25	9	10	7.01	0	0
Chart 4	0	25	9	10	7.01	0.1	0
Chart 5	0	25	9	10	9.01	0	0
Chart 6	0	25	9	10	9.01	0.1	0
Chart 7	5	25	7	10	0.88	0	0
Chart 8	5	25	7	10	0.88	0.1	0
Chart 9	5	25	9	10	0.88	0	0
Chart 10	5	25	9	10	0.88	0.1	0
Chart 11	5	25	9	10	0.54	0	0
Chart 12	5	25	9	10	0.54	0.1	0
Chart 13	10	25	7	10	0.54	0	0
Chart 14	10	25	7	10	0.54	0.1	0
Chart 15	10	25	9	10	0.54	0	0
Chart 16	10	25	9	10	0.54	0.1	0
Chart 17	10	25	9	10	0.46	0	0
Chart 18	10	25	9	10	0.46	0.1	0

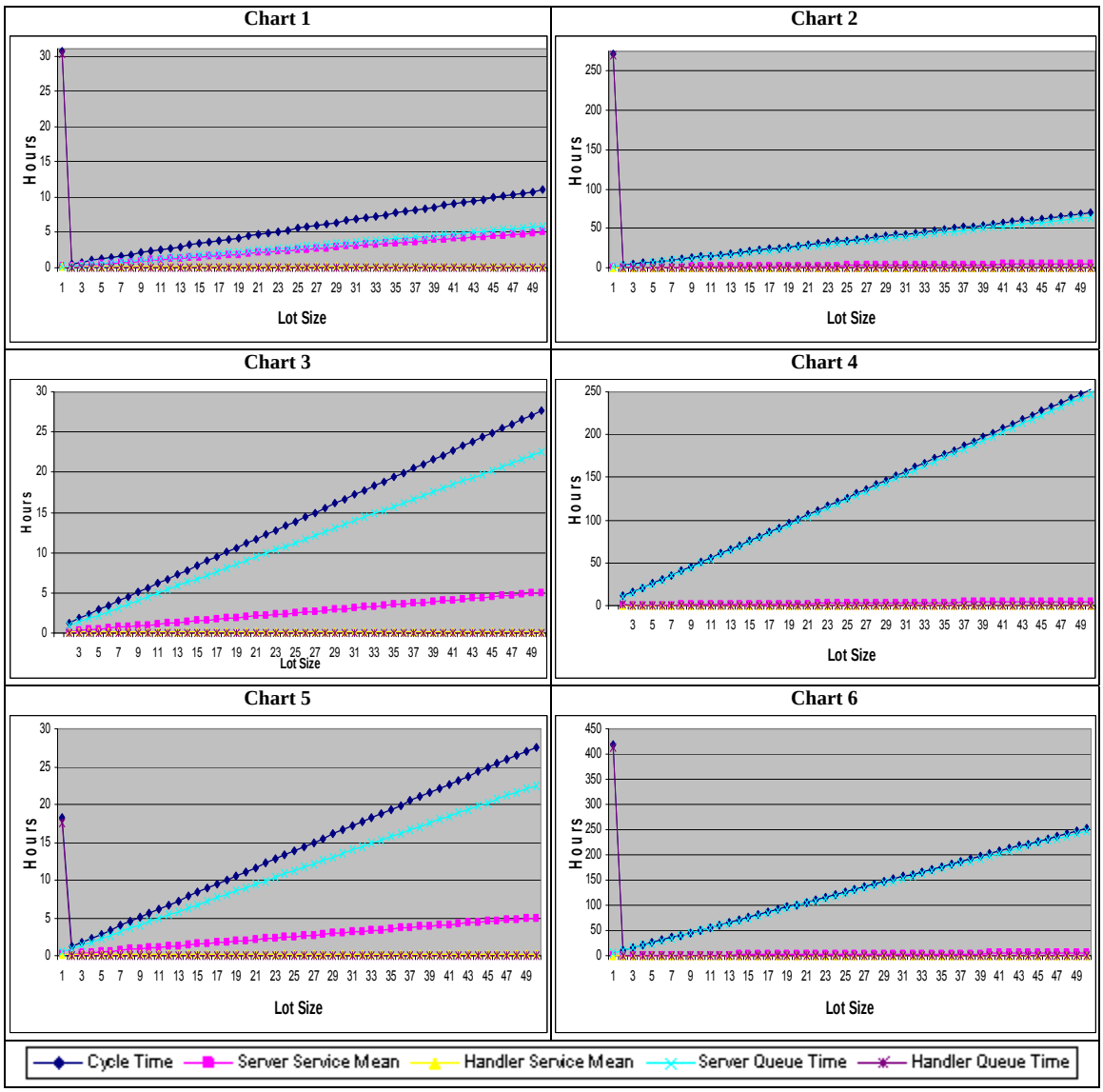


Figure 3.6 *a*=0

The first case we examine is when $a = 0$, that is there is no limit on the potential technological gains, which we call as perfect technology. In this case, the decrease in cycle time when moving to smaller lot sizes is very sharp and the smallest cycle time is achieved very close to the smallest possible lot size. Since the technology is the best that can be achieved, cycle times are also very low. The gain in cycle time is the biggest from the lot size decrease due to the linear tendency of the cycle time graph. This gain

becomes more significant when the base arrival rate is high (Charts 3, 4, 5 and 6). In this case, server queue time is the highest portion of the cycle time.

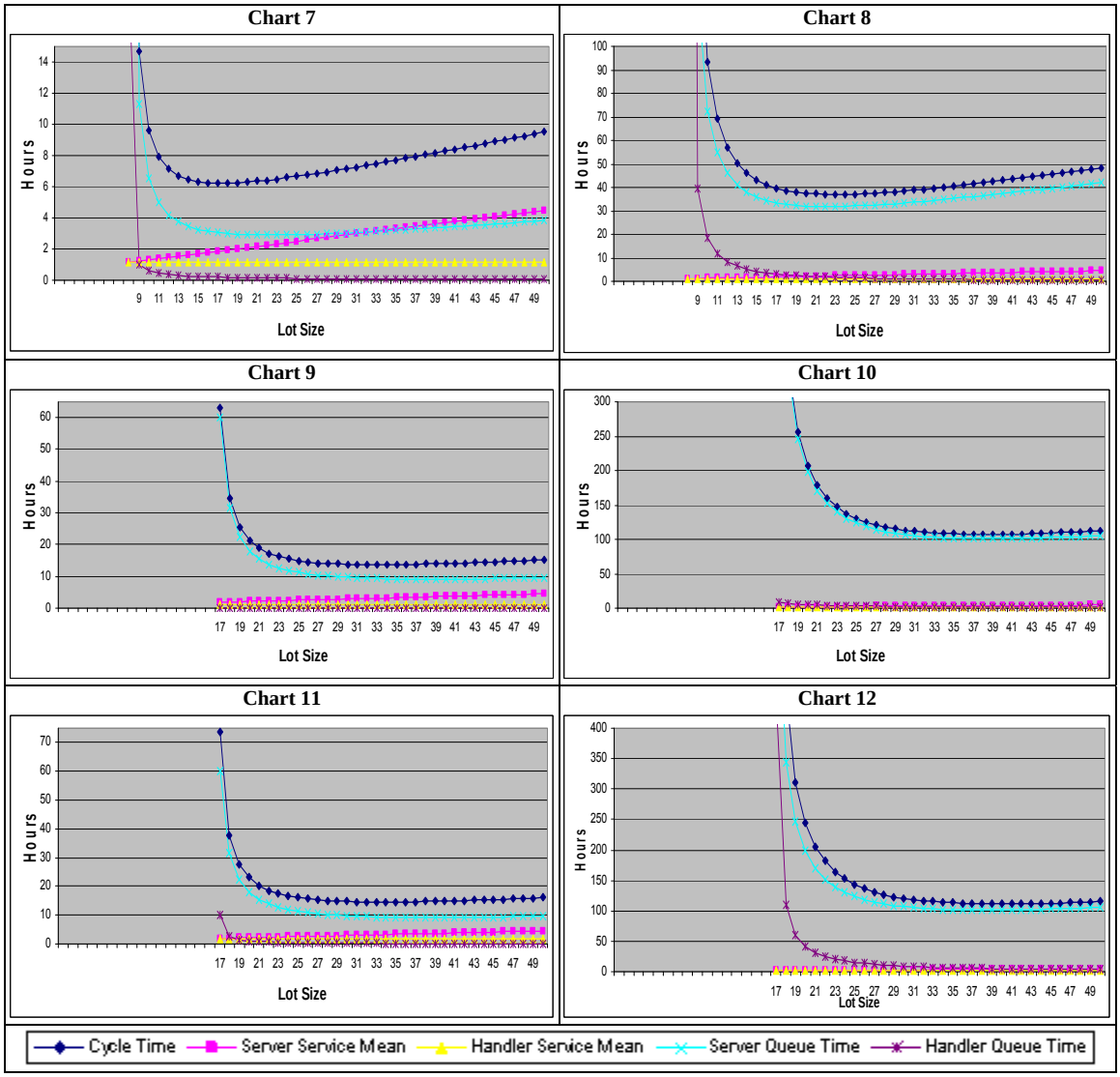


Figure 3.7 *a*=5

As the best technology limit becomes higher, the optimal lot size shifts to relatively higher values. Usually, the gain in cycle time (due to smaller lot sizes as compared to the lot size of 25) is not very significant. The most significant cycle time decrease comes with low arrival rate and low server service time variation (Chart 8). It is optimal to keep current lot size in two alternative cases, one being low arrival rate and

high server service time variation, the other being high arrival rate and low server service time variation with low base handler utilization (Charts 8 and 9). For the rest of the cases, it is better to move to bigger carrier sizes than the current one (Charts 10, 11, and 12).

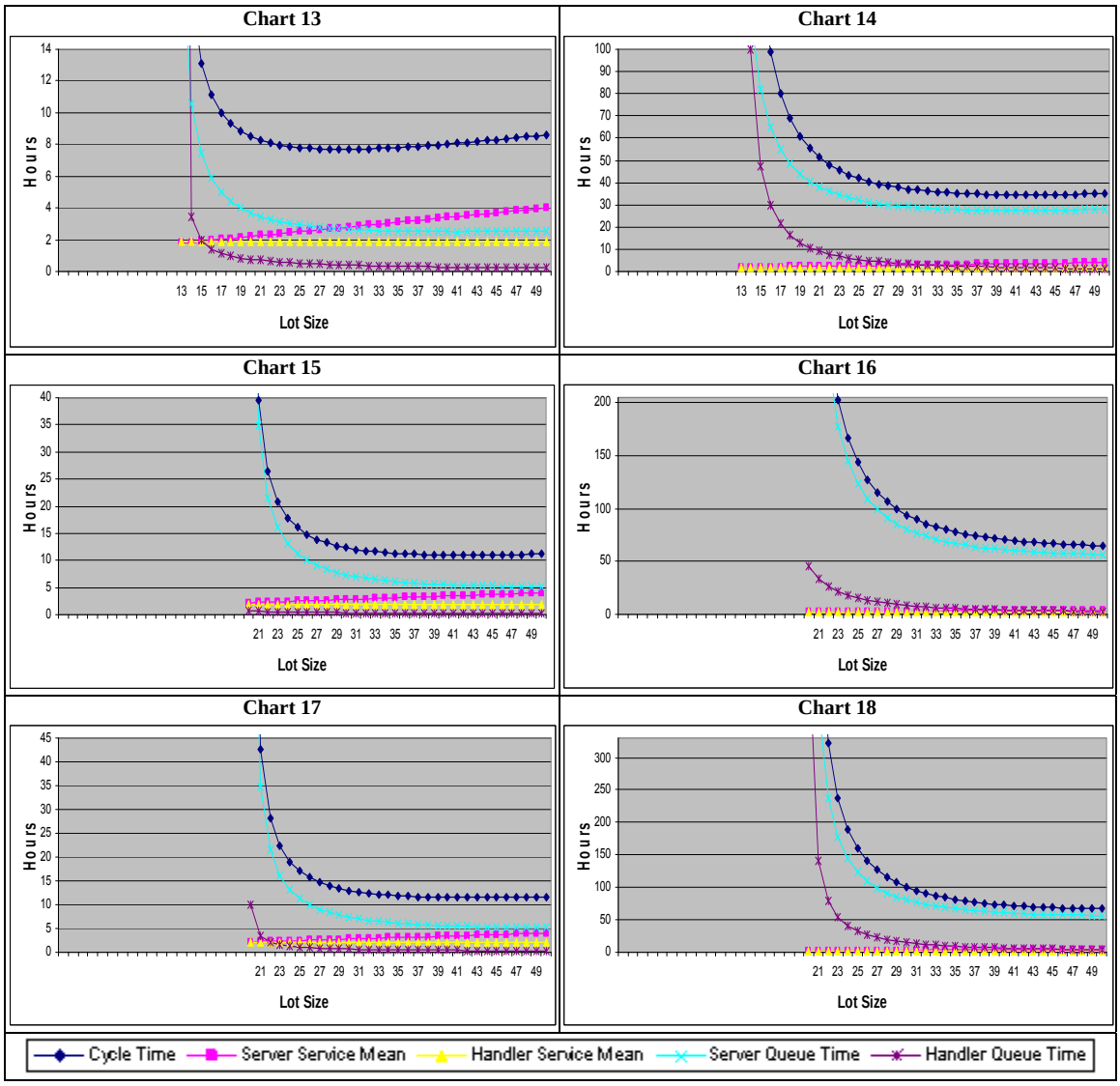


Figure 3.8 a=10

Next we examine the case where the lower bound on the technology parameter is relatively high: $a = 10$. This implies that there are limited feasible technological gains. In this case, transitioning to lower lot sizes drastically increases the cycle time as can be seen in Figure 3.8. As arrival rate and variance increase, optimal lot sizes are even higher.

In environments with a high traffic intensity and low variation, cycle time increases the fastest when moving from the current size to smaller lot sizes (see Charts 15 and 17).

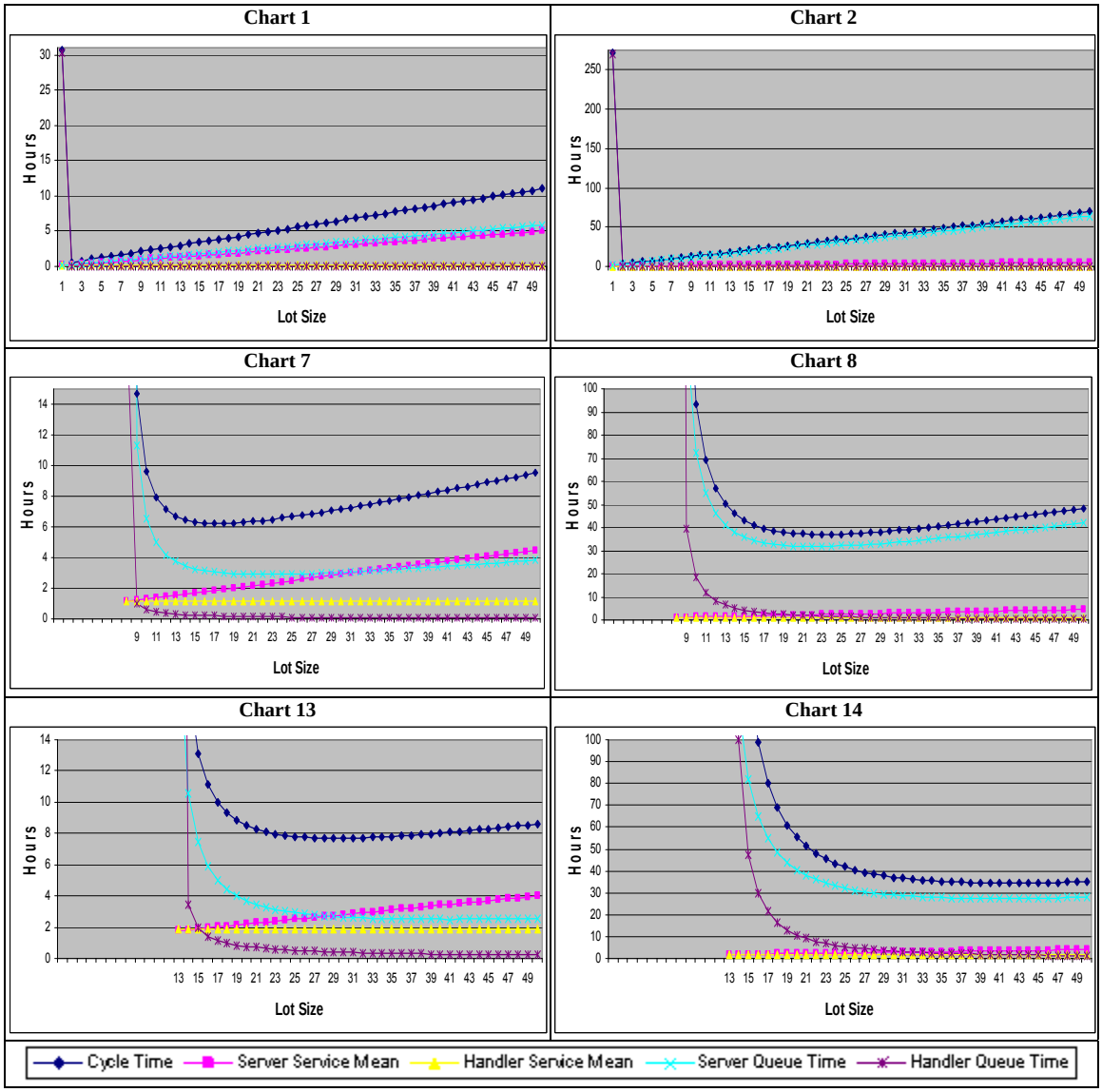


Figure 3.9 λ=7 lots/hr

Systems with lower arrival rates have smaller cycle time values with respect to systems with higher arrival rates. As technology gets better, the gain in cycle time decrease becomes higher while moving to smaller carrier sizes. Low variation is another factor for observing cycle time decrease.

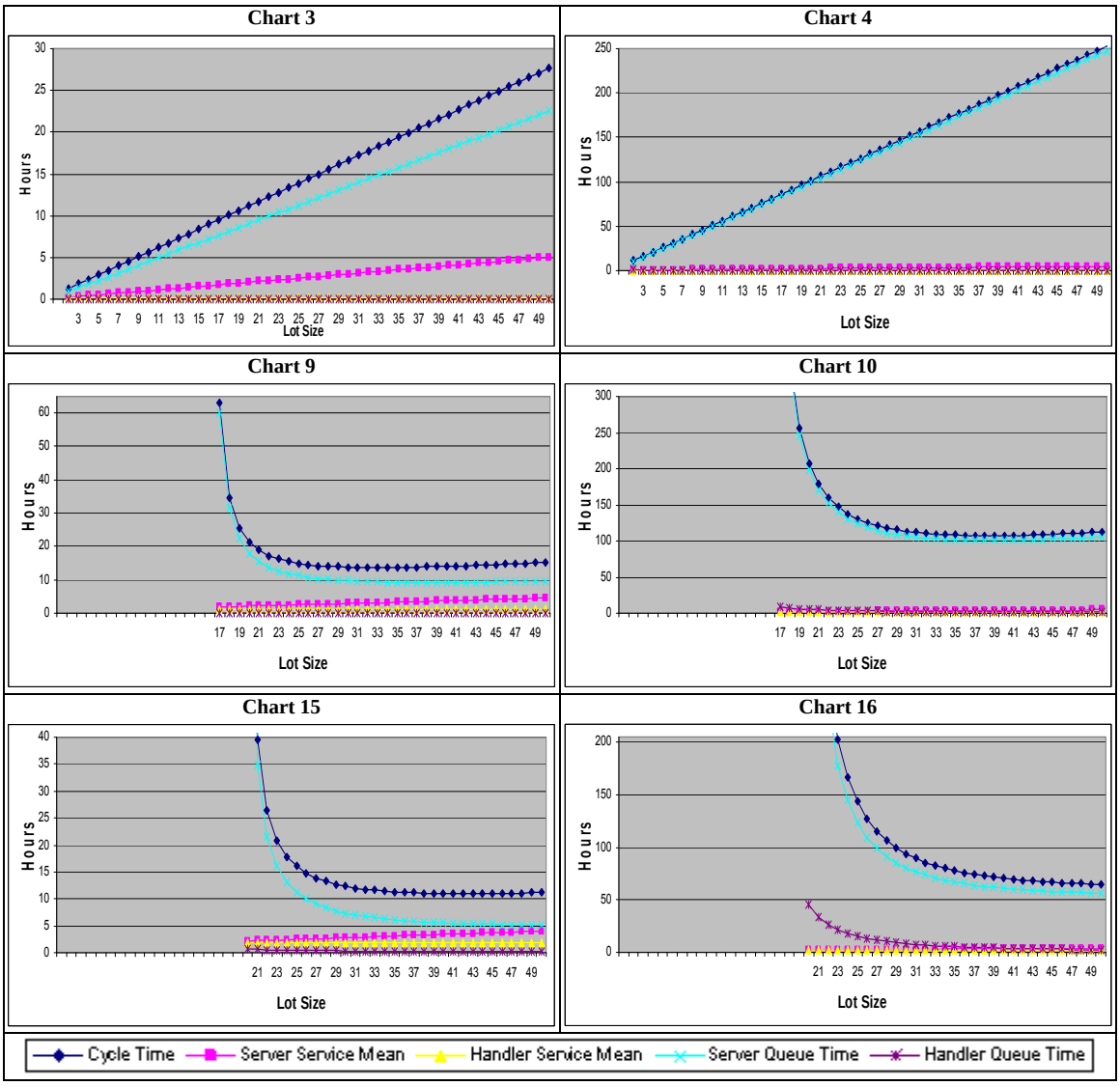


Figure 3.10 λ=9 lots/hr, highest ρ_s<0.95

In the charts in Figure 3.10, we examine the case where traffic intensity is very high, which is typical in a semiconductor environment. A higher traffic intensity naturally implies larger cycle times. More importantly, relative to lower traffic intensity cases, it also increases the lowest feasible lot size achievable, depending on the technology. When technology improvements are more limited, it is better to move to larger lot sizes. According to this model, it only makes sense to go to smaller lot sizes than the current one when technology is close to the best possible (cf. Charts 3 and 4). Handler queue

time is not a significant portion of the overall cycle time since the handler never becomes a bottleneck.

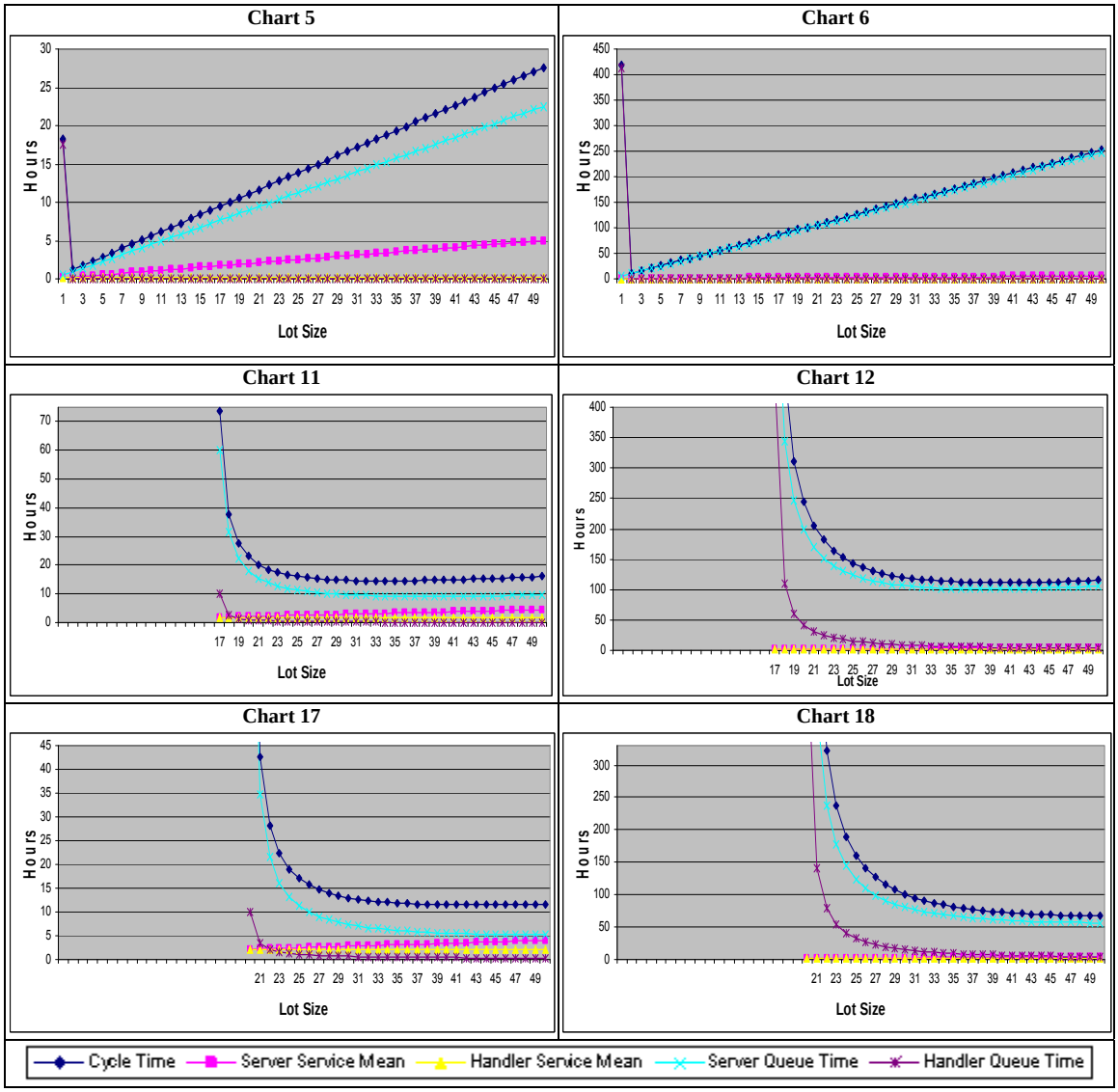


Figure 3.11 λ=9 lots/hr, highest ρ_s>0.95

The highest cycle times are obtained when arrival rate and handler utilization are high. Handler queue waiting becomes significant when variance is high. When the technology is not perfect, it is better to move to larger lot sizes. So in poor technology and high utilization environment, small lot size is not a good option, even it is better to use larger size carriers than the current size.

Chart 1

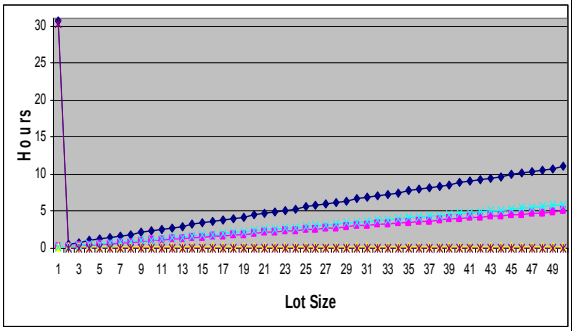


Chart 3

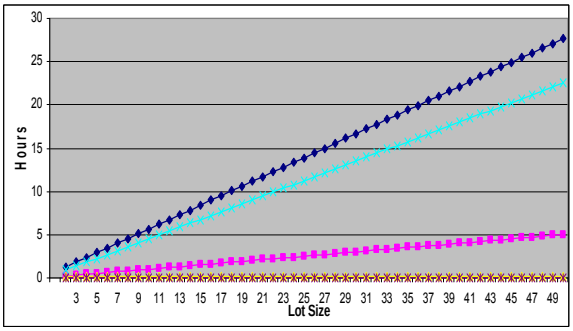


Chart 5

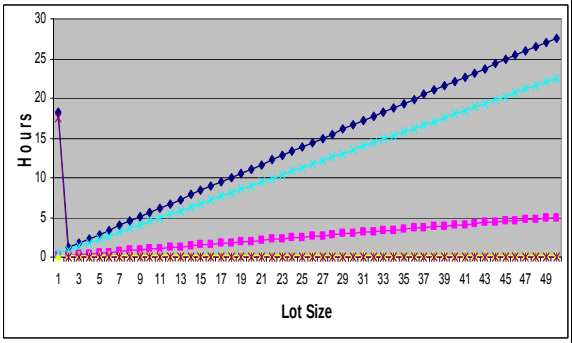


Chart 7

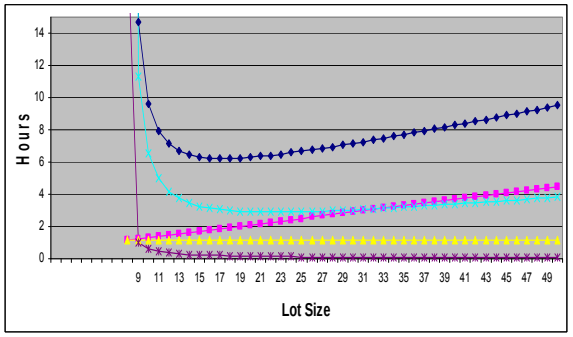


Chart 9

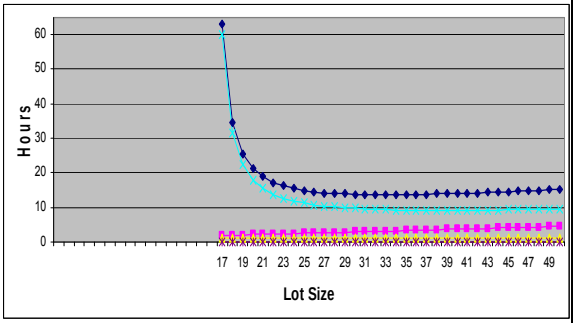


Chart 11

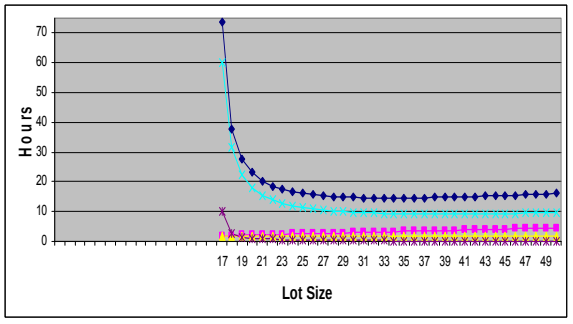


Chart 13

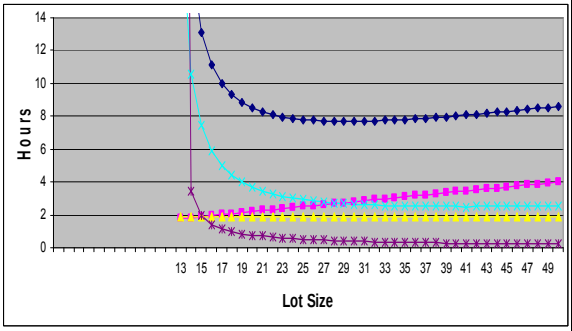
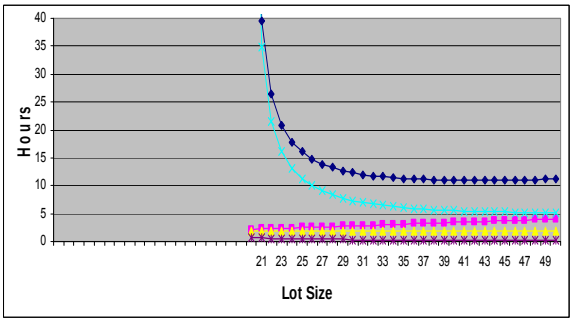


Chart 15



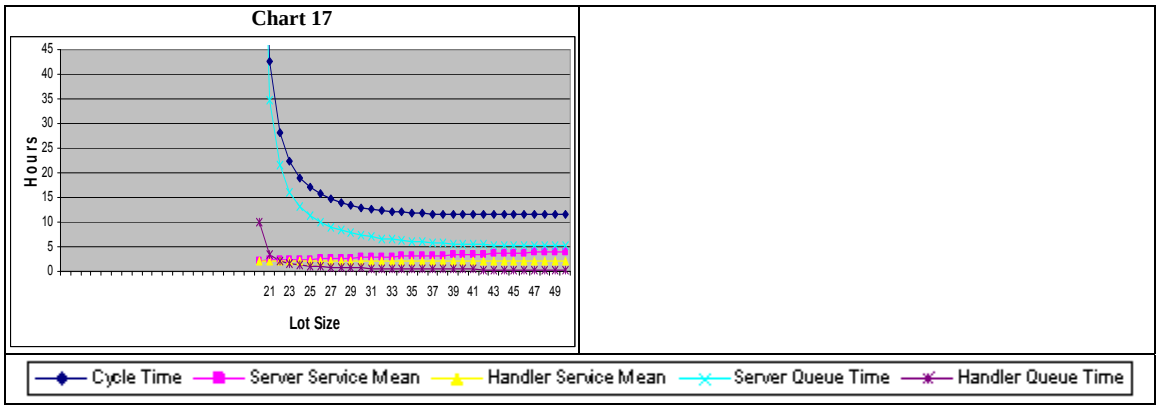


Figure 3.12 σ_s^2=0 lots^2/hr^2

As expected, lower server service time variance causes lower cycle times. As technology becomes poorer, the optimal lot size moves to higher values. For low arrival rates, server service time becomes the biggest portion of the overall cycle time for larger carrier sizes (Charts 1, 7, 13).

Chart 2

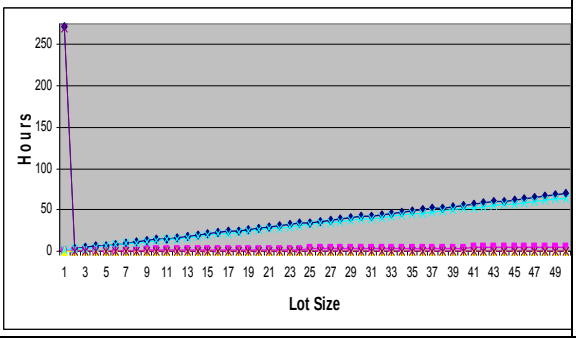


Chart 4

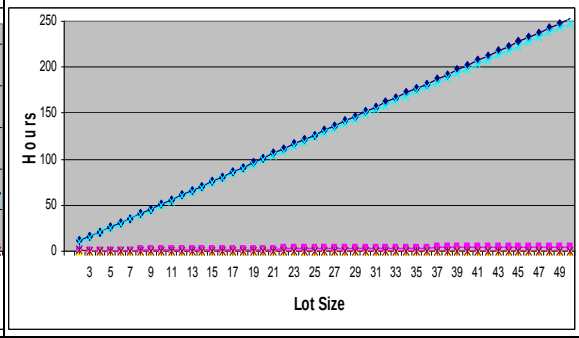


Chart 6

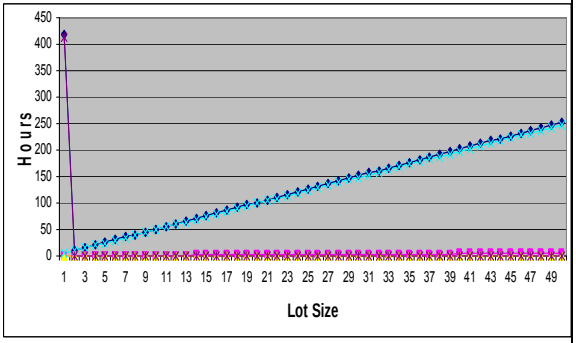


Chart 8

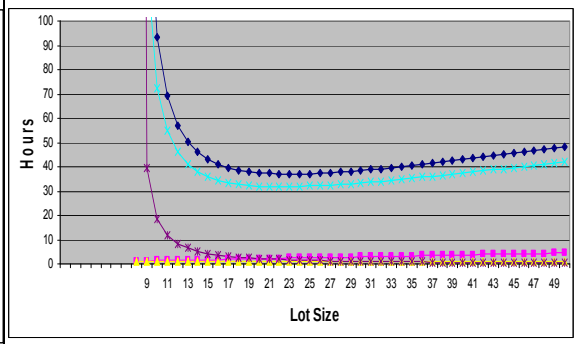


Chart 10

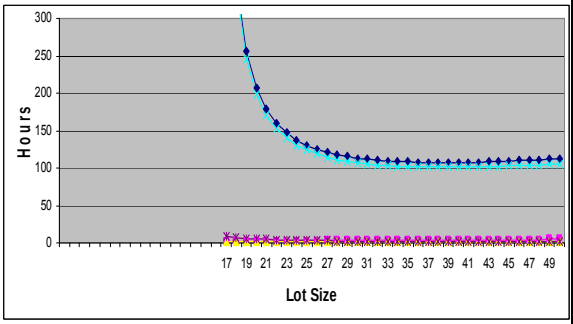


Chart 12

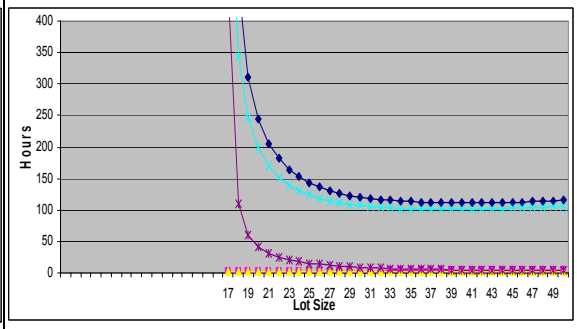


Chart 14

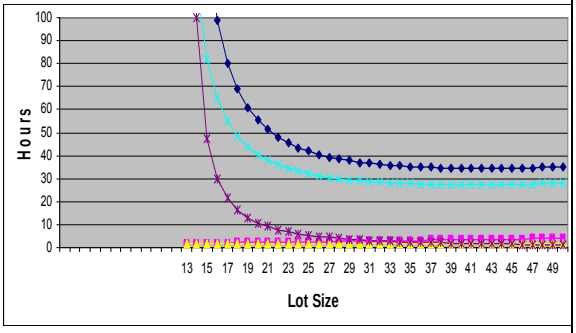
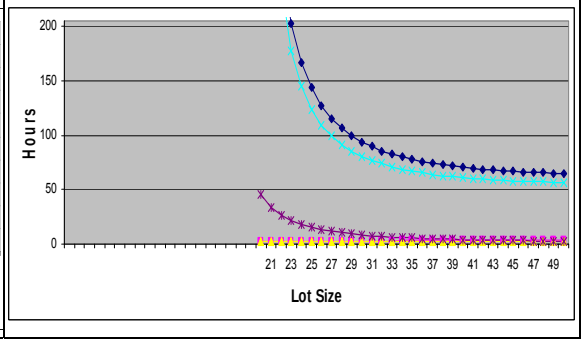


Chart 16



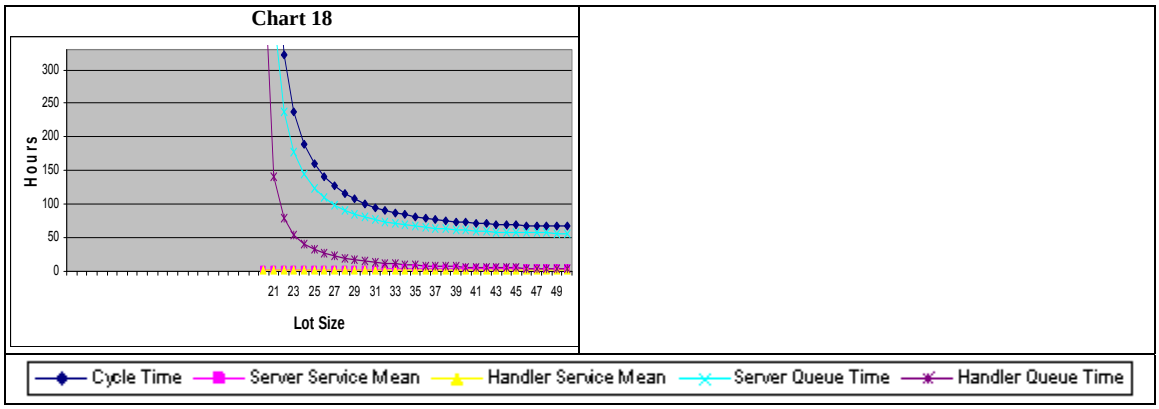


Figure 3.13 σ_s^2=0.1 lots^2/hr^2

Higher server service time variance ends up with higher cycle times. The behavior of the overall cycle time is almost same with server queue waiting time. Except for the perfect technology case, small lot sizes are never an option. It is always better to move to larger lot sizes or to keep the current lot size.

An interesting point to note is that in all of our experimental settings, the single wafer lot size is *never* optimal. It is either infeasible or produces very large cycle times due to high congestion. To further explore the limits of moving to smaller lot sizes consider the parameter set in Figure 3.14.

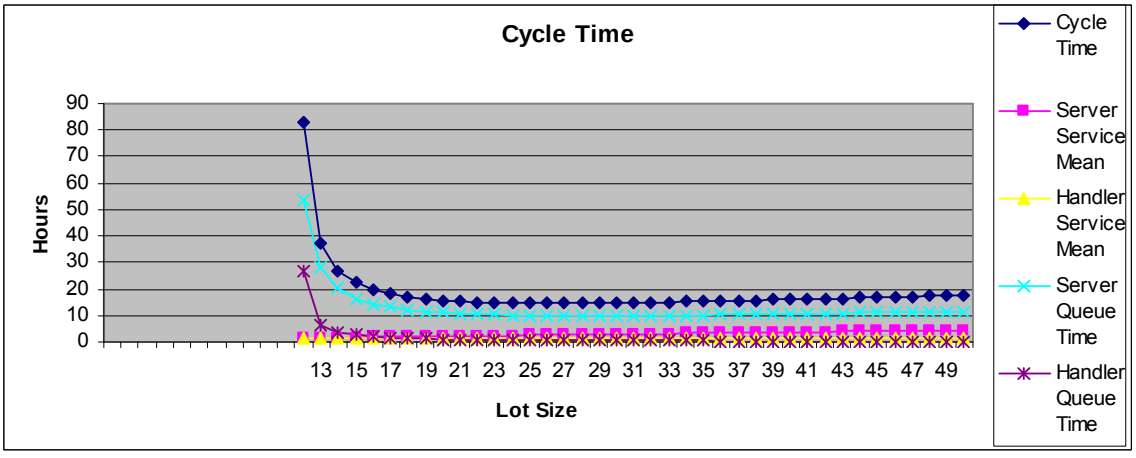


Figure 3.14 a=5, c=25, λ=8 lots/hr, μ=10 lots/hr, τ=0.685 lots/hr, σ_s^2=0.01 lots^2/hr^2, σ_h^2=0 lots^2/hr^2

Relative to our set of experiments, Figure 3.14 represents the cycle time behavior of various lot sizes of a "typical" fab by using the average values of our experimental factors. In such a setting this example indicates that it is not feasible to go to a lot size that is less than 12 without inducing very high cycle times. In fact, it is better to keep the current lot size of 25. While changing to larger lot sizes does not hurt cycle time that much, moving to a smaller lot size has a significant negative effect on the cycle time. In this example, the server queue time is the biggest portion of the cycle time.

3.7 SPECIAL CASES OF LOT SIZE CHARACTERISTICS

In this section we derive an analytical expression for the optimal lot sizes for a special case, using the cycle time approximation obtained in Section 3.4. We use the continuous extension (in the lot size *n*) of the cycle time formula (3.8), and differentiate to obtain the first order condition for a minimum. The cycle time expression is differentiable on three intervals, which are: $c_{ss}^2 \le 0.2$, $0.2 < c_{ss}^2 < 1$, and $c_{ss}^2 \ge 1$. For reasonable parameter values, the cycle time expression is convex for positive values of *n*. Therefore, for practical purposes we can use the corresponding local minimizer as the proposed optimal lot size.

We analyze the case in which $a = 0$, i.e., when there is no bound on the achievable technology level. Also we assume that material handling speed is constant: $c_{sh}^2 = \sigma_h^2 \tau^2 = 0$. There are three different ranges within which the cycle time is differentiable: low, medium, and high base service time variation. We focus on the medium variation case, since this is the most reasonable representation of a semiconductor fab.

Medium server service time variation ($0.2 < c_{ss}^2 < 1$): The cycle time expression in this case is

$$W_{ssmh}^{QNA}(n) = \frac{n}{\mu} + \frac{\frac{n\lambda}{\mu^2}\left(1+\sigma_s^2\mu^2\right)}{2\left(1-\frac{\lambda}{\mu}\right)} + \frac{1}{\tau}$$
$$+ \frac{\left\{\frac{1}{\tau}\frac{\lambda}{n\tau}\left[1+\left(-1+\sigma_s^2\mu^2\right)\left(\frac{\lambda}{\mu}\right)^2\right]\exp\left\{\frac{-2\left(1-\frac{\lambda}{n\tau}\right)}{3\frac{\lambda}{n\tau}}\frac{\left[\left(\frac{\lambda}{\mu}\right)^2\left(1-\sigma_s^2\mu^2\right)\right]^2}{\left[1+\left(-1+\sigma_s^2\mu^2\right)\left(\frac{\lambda}{\mu}\right)^2\right]}\right\}\right\}}{2\left(1-\frac{\lambda}{n\tau}\right)}. \quad (3.11)$$

The first derivative with respect to *n* is given by:

$$\frac{\partial W_{ssmh}^{QNA}(n)}{\partial n} = \frac{1}{\mu} - \frac{\lambda\left(1+\sigma_s^2\mu^2\right)}{2\mu(\lambda-\mu)}$$
$$+ \exp\left[\frac{2(\lambda-n\tau)\left(-1+\sigma_s^2\mu^2\right)^2}{3\mu^2\left(-\lambda^2+\mu^2+\lambda^2\mu^2\sigma_s^2\right)}\right]\left\{\frac{\lambda}{\mu^2(\lambda-n\tau)}\left[\frac{\lambda^3\left(-1+\sigma_s^2\mu^2\right)^2}{3\mu^2} - \frac{-\lambda^2+\mu^2+\lambda^2\mu^2\sigma_s^2}{2(\lambda-n\tau)}\right]\right\}. \quad (3.12)$$

To find the minimizer we need to find an *n* such that the expression is equal to 0. Unfortunately, the exponential term makes obtaining an analytical solution for the critical point quite cumbersome. However, according to our numerical studies this exponential term is very close to 1 for the parameter range of interest. Therefore, as an approximation, we solve the expression below for *n*:

$$\frac{\lambda}{\mu(\lambda-n\tau)}\left[\frac{\lambda^3\left(-1+\sigma_s^2\mu^2\right)^2}{3\mu^2} + \frac{\lambda^2-\mu^2-\lambda^2\mu^2\sigma_s^2}{2(\lambda-n\tau)}\right] - \frac{\lambda\left(1+\sigma_s^2\mu^2\right)}{2(\lambda-\mu)} + 1 = 0. \quad (3.13)$$

We obtain two solutions for (3.13). One of these solutions is infeasible with respect to the constraints in (3.10). Analyzing the second derivative of (3.11) shows that

the cycle time expression is convex in the feasible region defined by (3.10). Thus, the solution to (3.13) (which is also feasible) is the local minimizer of (3.11). If we cannot find such a solution by solving (3.13), then the optimal lot size is the smallest feasible integer value. This boundary value is determined by the server or AMHS traffic intensity. Otherwise, the optimal lot size value is obtained by solving (3.13) and is given by:

$$n^* = \frac{1}{3\mu^3\tau\left(-\lambda+2\mu+\lambda\mu^2\sigma_s^2\right)} \left(\begin{array}{l} -\lambda^5+\lambda^4\mu-3\lambda^2\mu^3+6\lambda\mu^4 \\ +\lambda^2\mu^2\sigma_s^2\left(2\lambda^3-2\lambda^2\mu+3\mu^3\right) \\ +\lambda^4\mu^4\left(\sigma_s^2\right)^2(\mu-\lambda) \\ + \sqrt{\left\{\lambda(\lambda-\mu)\left[\begin{array}{l} \lambda^8-\lambda^7\mu+3\lambda^3\mu^4\left(\sigma_s^2\right)^2\left(2\lambda^5-2\lambda^4\mu-3\mu^5\right) \\ +4\lambda^7\mu^6\left(\sigma_s^2\right)^3(\mu-\lambda) \\ -\lambda^7\mu^8\left(\sigma_s^2\right)^4(\mu-\lambda)-9\lambda^3\mu^5 \\ +18\lambda^2\mu^6+9\lambda\mu^7-18\mu^8 \\ +\lambda\mu^2\sigma_s^2\left(\begin{array}{l}-4\lambda^7+4\lambda^6\mu \\ +18\lambda^2\mu^5-18\lambda\mu^6-9\mu^7\end{array}\right)\end{array}\right]\right\}} \end{array} \right). \quad (3.14)$$

Since this value is not necessarily integer, one can use the best neighboring integer value. So, a practical approach to finding optimal lot size value is to first check if (3.14) is feasible in (3.10). If it is, use the best neighboring integer value as the optimal lot size value, otherwise use the smallest feasible lot size. Figure 3.15 depicts optimal lot size function behavior depending on λ, τ and σ_s^2. These charts are plotted using actual sample data. Optimal lot size decreases by increasing material handling rate very fast up to a threshold value of τ while it increases by higher variance, σ_s^2. Smaller lot sizes become optimal by lower and higher base arrival rate and utilization while optimal lot size makes a pick around average arrival rate.

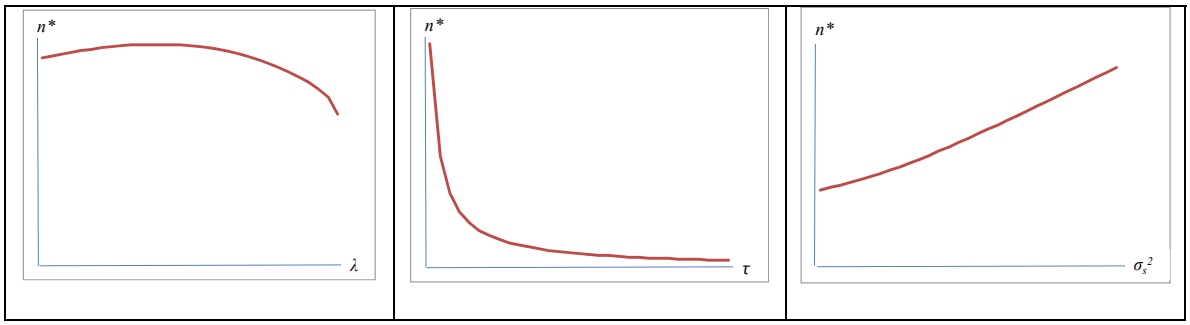


Figure 3.15 Optimal lot size behavior in relation to base arrival rate, material handling rate, and service time variance

3.8 CONCLUSION

One recent trend in semiconductor manufacturing is to promote the use smaller lot sizes in order to achieve lower cycle times. This inspired us to develop a model to gain insight into lot size and cycle time relations. For this purpose, we developed a simple tandem queueing model representing a semiconductor fab with serial processing tools, and material handling system. We then developed cycle time approximations for this model.

Our analysis shows that smaller lot sizes are not always the better option. The optimal lot size in a fab depends on several parameters such as the manufacturing technology, the capabilities of the material handling system and the variability in the system. In many cases, keeping the current lot size or moving to larger lot sizes may actually be better. An improvement in manufacturing technology is the key for smaller lot sizes to improve cycle times. In our model, a single wafer lot is never an optimal.

Future research includes extending our model to allow multiple lot sizes to flow in the system. This is a crucial step since most foundry type fabs, which produce wafers on demand, run multiple lot sizes in the fab at the same time. A further extension would be to attach priorities to the lots. It would also be useful to develop more complex queueing models to represent a more realistic fab.

Chapter 4: AMHS Scheduling

Improvements in semiconductor manufacturing technology have caused a substantial change in every aspect of the manufacturing environment. Moving from 200 mm to 300 mm in the wafer diameter forced an increase in the level of automation in manufacturing due to difficult handling of larger and heavier wafer carriers by human operators. Due to the increased demand for semiconductor products, the scale of production has been increased. Also, demand driven product differentiation results in a large variety of lots running in a fab. Technological improvement is inevitable to satisfy demand and consumer expectation. The number of layers in wafer manufacturing increases as technology progresses. An increased number of layers in a wafer means more operations. Therefore, existing complexity of semiconductor manufacturing is multiplied with time.

Each factor which adds to complexity in wafer production places an additional burden on material handling. The most up to date wafer fabs utilize fully automated material handling systems. Material handling then becomes one of the key parts to provide smooth and prompt means to production.

Over the past decade, most studies have concentrated on better AMHS design in a semiconductor manufacturing environment. Plant layouts which allow smoother vehicle traffic and faster delivery with less cost have been developed. Conventional material handling systems and vehicles have been adapted to the semiconductor manufacturing environment to satisfy the industry's needs in the most appropriate way.

In recent years, AMHS studies have concentrated on developing better policies to provide conflict free fast delivery time policies. Most of these policies are dispatching

policies which work locally or myopically with a limited amount of information available to the decision maker.

The main reason behind having relatively short sighted assignment policies is that there is a limited amount of information. The information infrastructure has not shown improvement at the same rate as the manufacturing environment. The actual production information infrastructure and AMHS information infrastructure are for the most part disintegrated. The information sharing between these two information systems are on a real time basis as needed by a lot or vehicle to be assigned. In short, only the information on lots and vehicles available to be assigned for delivery is available to the decision maker. This prevents the decision maker from developing more long term solutions.

Recently, the industry has been working towards wider integration of production and AMHS information infrastructures. This will give the decision maker a window of information availability while planning. Also, it will help develop better optimization based scheduling solutions for AMHS assignment rather than pure dispatching based methods.

In this study, we first compile an extensive literature review. Then we present new AMHS models. We propose two alternative scheduling algorithms which are classified depending on information availability: myopic and look-ahead algorithms. We evaluate both algorithms by running a comprehensive set of computational experiments. In these experiments, the look-ahead algorithm dominates myopic algorithm in every performance measure. Then we conclude our study with a discussion of the computational experiment results and future research.

4.1 LITERATURE REVIEW

There is a tremendous amount of research specialized in AMHS semiconductor manufacturing. The two main focus areas are AMHS design and AMHS dispatching.

Both design and dispatching studies target having a higher material handling service rate and shorter delivery times with the least amount of conflict.

The most common layout arrangement in semiconductor manufacturing is a bay configuration, in which similar groups of tools are placed together. This layout is widely preferred since it has great advantages in maintenance, utility distribution and handling. Mostly, an overhead monorail system is used to move materials between bays. Two commonly used AMHS configurations along with a bay system are the spine configuration and the perimeter configuration with an overhead monorail system. A carrier mounted on a monorail moves materials between steps. Crossover turntables serve as shortcuts while stockers associated with bays are pick-up and drop-off points. On the tracks of AMHS, intermediate storages for lots transferred in the system are located. These storages are called stockers which serve for a group of tools as a common buffer. While a stocker can serve one or more bays, two or more stockers can serve one bay, too, depending on the system configuration. Each tool has loadports which serve as local tool buffers for lots to be processed on the tool. A variety of vehicle types may run on AMHS tracks. Automated Guided Vehicles (AGVs) are shuttle-like vehicles that run on over tracks. Over-Head Transport (OHT) or Over-Head System (OHS) runs hanging from elevated tracks. An OHT picks a Front Opening Unified Pod (FOUP) filled with wafers from above and carries to its destination. Conveyor belts are recent technology in AMHS which aims to minimize collisions in the system and increase flow of transport. Robot arms pick carriers from conveyor belts and transfer them to storages or tools.

Peters and Yang (1997) propose a spacefilling curve (SFC) heuristic to solve the layout and material handling system design integration problem. Integrated design of AMHS and fab layout is crucial for a fab's efficiency. A bay's area and pick-up/drop-off

point position constraints are considered. The AMHS has a directed flow path and the crossover turntable design must be integrated with the layout and AHMS designs for an effective integrated fab design. Yang et al. (1999) propose a loop-configuration design procedure in which the integrated design problem is formulated as a network flow problem to determine the optimal material handling system design based on a given fab layout. A hybrid tabu search and simulated annealing algorithm that embeds the network flow formulation solves the layout and AMHS design problem in an integrated fashion.

Kuo (2002) models OHT in a fab using three modular-based colored timed Petri net (CTPN) models, which extend time, color sets, module and communication capabilities from the original Petri nets. The OHT transport model uses transport, hoisting, vehicle pushing, nearest vehicle finding, intelligent control, and zone control characteristics. The finding previous station model coordinates with "finding nearest vehicle" algorithm and finds the previous station of the nearest vehicle. The stochastic path generator model generates the path, priority, and commits the completion of the wafers. A simulation technique executed in an object-oriented graphical user interface (GUI) evaluates the performance of the OHT system to determine the number of vehicles in the planning stage and to control the dispatching of the OHT system in the operation stage.

There are two types of AMHS in a wafer fab, one is the interbay which transports cassettes of wafers between bays, the other is the intrabay, which transfers cassettes within a bay from tool to tool. The traditional AMHS isolates the interbay transport from intrabay transport. The wafer delivery from a tool within a bay to another within another bay is conducted through the operation of two stockers. A longer waiting time for wafers by the transport vehicles at the from-stocker and the to-stocker occurs than it would occur

if there was no stocker-to-stocker movement. Therefore, Lin et al. (2003) propose a connecting transport concept, using a different type of vehicle within bays and a single system of interconnected lines. They use a simple mathematical model to determine the minimum number of vehicles for different connecting transports. A connecting transport system is able to move a large number of lots per hour. Wafers can be transferred directly to and from the tool loadports without stocker operation. Waiting time for an empty vehicle is effectively eliminated.

The factors which affect AMHS performance are factory layout, AMHS track layout (spine, perimeter, flexible, with track options such as turntables, turnouts, or high-speed express lanes), transport vehicles (number of vehicles, velocities, vehicle dispatching), production planing and scheduling (throughput rate, WIP, and stocker capacity distribution and loading along the wafer fab), production control, stocker operation management and operator behavior (retrieve trends, delays to output port unloads, and lot requests not from retrieve stockers). Wang and Lin (2004) develop a discrete event simulation model to evaluate the performance of an AMHS for a wafer fab with a zone control scheme avoiding all vehicle collision. The AMHS layout is a custom configuration. Input data analysis shows that the underlying inter-arrival time distribution for most stockers are exponential or Weibull. The number of vehicles significantly affects the average delivery time and the average throughput and can be determined by this simulation. An AGV-based intrabay material handling system is also simulated to determine the number of vehicles to minimize delivery times and maximize throughput. Lin et al. (2005) propose a dynamic connecting transport AMHS in which interbay and intrabay material handling systems are connected and investigate the relation between performance and the number of vehicles. In order to avoid congestion or idle time in the

intrabay system, controlling the upper limit or the lower limit on the number of vehicles in the intrabay system can be a solution. The combination of the existence or non-existence of upper or lower limits on the number of vehicles in the intrabay system forms 4 strategies, upper and lower limits exist, upper and lower limits do not exist, upper limit exists but lower limit does not exist, lower limits exists but upper limit does not exist. Simulation studies show that the flow rate and the existence of upper-lower limit strategy on number of vehicles significantly affect all performance measures. The best strategy for each intrabay is with the upper and lower limits of the vehicle numbers. Controlling the lower limit on the number of vehicles significantly affects the travel time, the waiting time and the empty vehicle utilization.

Li et al. (2005) propose the Intelligent Integrated Delivery (IID) concept that instructs AMHS to deliver remotely stored WIP to the stocker storage location closest to the tool or to a buffer on the tool, before the lots are requested for processing in a just-in-time manner. This reduces the tool idle time. The impact of lot delivery time is further reduced by prioritizing lots for delivery in the AHMS. The delivery time of OHS is defined as the time consumed for the delivery of a lot from the input port or the shelf of departure stocker to the shelf of destination stocker. Excessive waiting time due to the unavailability of a vehicle or poor performance of the stocker causes overtime transport. Kong (2007) considers AMHS layout, transport vehicles, and scheduling logic as determining factor of the AMHS performance and proposes a two-step simulation method. While he simulates the production process on AutoSched AP, he simulates the AMHS process using Automod. The production simulation helps estimate equipment utilization, total around time, WIP, and the throughput of the AHMS. The AMHS

simulation predicts the capability of AMHS system, such as number of vehicles required, throughput and delivery time.

The other major line of research concentrates on scheduling and dispatching in the AMHS. While some rules are based on finding conflict free solutions, others aim to reduce travel times. Lin et al. (2001) study a combination configuration of an interbay system, in which the hallway contains double loops and the vehicle has double capacity whereas most of the literature assumes an interbay system in which the hallway contains only a single loop. The dispatching of cassettes and vehicles can be an issue in the interbay system. This paper studies the relative performance of dispatching rules by simulation. Among the eight dispatching rules compared, a combination of Shortest Distance-Nearest Vehicle (cassette initiated) and First Encounter First Served Rule (vehicle initiated) performs the best. Lin et al. (2004) analyze loading of an automated double-loop interbay material handling system in a wafer fabrication considering the effects of the number of vehicles in the inner and outer loops. They develop simulation models to study the AMHS with a zone control and attempt to avoid any vehicle collision. The hallway contains double loops and vehicles have double capacity in the layout. The dispatching rule is the combination of the shortest distance with nearest vehicle (SD-NV) and the first-encounter-first-served (FEFS) rule. The simulation determines the maximum load. Interbay performance is significantly affected by the number of vehicles. The optimum combination of the number of vehicles in the inner and outer loops can be obtained by response surface methodology.

Liao and Wang (2004) adopt a neural network approach for prediction of expected loop-to-loop delivery times of both priority and regular lots. A neural network is built for each OHT loop, with inputs of transport requirements, automatic material

handling resources of the loop, and the ratio of priority lots in the population. A back propagation method is adopted with the outputs of a discrete event simulation model to train the neural network. Given the estimates of lot delivery times in each loop, the path with shortest delivery time can be determined by solving an integer programming problem. Numerical studies show that the developed approach is effective for the prediction of average delivery times and applicable to the implementation of a transport time estimator for 300-mm fab scheduling and dispatching systems. OHT vehicles usually suffer from severe blocking effects due to simple directed graph configuration of an OHT loop. Liao and Fu (2004) propose an effective, simulation-based, two-phase approach, OHTAD (OHT Allocation and Dispatching) – in a large-scale 300-mm AMHS management which is originally an integer programming problem due to nonlinear constraints. Phase 1 of OHTAD selects the best OHT dispatching policy and Phase 2 of OHTAD determines the number of OHT vehicles by running simulations. Simulation studies show that OHTAD achieves good performance on both moves and carrier delivery times. Liao and Wang (2006) propose an effective OHT dispatching rule, the differentiated preemptive dispatching (DPD) policy, to reduce the possible blocking effects during the transportation of hot lots in a 300 mm OHT system. DPD, inspired by the observation of empirical human operations for carrying hot lots, aims to minimize the delivery time of hot lots while minimizing the impact on the transport of normal lots. Given a system configuration of loading ratio, population of hot lots, and the number of OHTs in the loop, simulation results show that the DPD policy dominates the other rules in delivery performances of hot lots and normal lots. The DPD rule is very useful to streamline shop floor operations for eliminating time delays of hot lots in an automatic environment.

Tyan et al. (2004) present an integrated tool and vehicle (ITV) dispatching strategy based on a state-dependent methodology to consider multiple performance measures in a fully automated fab environment. A manufacturing process and AMHS integrated simulation model is built to examine the performance impact of the ITV dispatching rule measured by cycle time, WIP, on-time delivery, and lot delivery time. The optimization procedure uses a desirability function approach to transform a multiple response problem into a single response problem. The simulation analysis shows that ITV dispatching rule is superior to the use of a static dispatching rule for on-time delivery and other performance measures. Kuo and Huang (2006) propose a multi-mission oriented intelligent vehicle dispatcher to dynamically adjust dispatching rules so that the possible high-risk lots can be resolved to meet all production strategies and manufacturing efficiency. Fuzzy logic is used to develop the multi-mission dispatcher. Simulation case study models are built in Automod. Simulation results suggest that the proposed intelligent multi-mission oriented vehicle dispatcher performs better than others, and most of the production strategies are satisfied.

Work initiated dispatching is started by an available lot searching for an available vehicle. Vehicle initiated dispatching is started by a vehicle searching for an available lot to transport. Buffer initiated dispatching takes place after a busy input buffer becomes available, at which point there are groups of idle vehicles and available lots that need to be matched. Koo et al. (2005) present a vehicle dispatching procedure based on the concept of the theory of constraints, in which the dispatching decisions are made to utilize the bottleneck resource at the maximum level by preventing starvation and blocking. A simulation study shows that this procedure performs better than existing ones, especially in a high load environment with limited buffer space.

Most of the studies on AMHS in the literature use simulation as the base factor to model and evaluate the AHMS performance in semiconductor manufacturing. Nazzal and McGinnis (2007) propose an analytic approach to evaluating AMHS performance in an integrated circuit (IC) fab. They develop a queuing network type model, based on a detailed description of OHT operations, and analyze it as a large scale discrete time Markov chain. The model considers vehicle blocking without the need to include detailed AMHS operations. The overall approach helps them estimate both AMHS throughput and move request delays.

Toba et al. (2005) propose a load balancing method which balances all processing operations of products among multiple semiconductor fabrication lines by using predictive scheduling results. Im et al. (2008) propose an efficient vehicle dispatching rule which minimizes the vehicle blocking and delivery times in AMHS in 300 mm semiconductor manufacturing. The dispatching methodology consists of three phases. In phase 1, the jobs involved in task assignment are selected by waiting time and by window size. In phase 2, a modified Hungarian method is used to identify and select the job that minimizes the average and variance of delivery time. In phase 3, the selected job, according to the category it falls in, i.e. Waiting, Assigned, Loaded, is assigned to the idle vehicle or reassigned to a new idle vehicle. Simulation results show that this methodology outperforms other rules in throughput and delivery times. Kim et al. (2008) propose HABOR, a Hungarian algorithm based OHT reassignment approach. The vehicle dispatching problem is modeled as an assignment problem and is solved by the Hungarian algorithm. HABOR outperforms some of the most popular traditional dispatching rules with regards to number of vehicles and average lead times.

4.2 AMHS MODEL AND DEVELOPMENT OF LOOK-AHEAD AND MYOPIC ALGORITHMS

AMHS is a vital part of a fab to sustain the flow of production. Poor implementation of AMHS scheduling results with delays and failures in lot delivery, overloaded transport systems, congestion, increased inventory and increased transport and process cycle times. Most of the current AMHS dispatching methods neglect the benefit of a look-ahead approach due to limitations of information infrastructure integration between AMHS and production system. Information sharing between these two systems occurs only when an entity of one of these two systems becomes available so these two systems cannot see the status of entities of each other unless they become available. Although there are very well thought myopic dispatching algorithms in the literature, lack of information sharing prevents schedulers to make use of the automation to its full extent. There are recent efforts to increase the level of integration between these two systems. Therefore we analyze and develop solutions for scheduling of AMHS in semiconductor fabs with limited information sharing where there is small ahead of time information availability.

We model the problem using mixed integer programming. We assume a fab layout with spine configuration. Each bay consists of one tool and one stocker. A lot is a batch of product units, wafers, which act as a transfer unit. Each lot has a pre-determined process route. A route consists of steps. A step is an operation that requires a unique tool. In our setting, our lots are defined by their current location, current step and time they are going to be available to be picked up by a vehicle. A vehicle is the carrier part of the AMHS. It can carry one lot a time and travels via the rail tracks of the AMHS.

A fab layout consists of multiple bays which hosts a group of tools. These tools are functionally the same. In the setting we study, we assume one tool for each bay. Each

tool has a number of loadports where lots can be located to be processed. This local storage is called as tool buffer. Each loadport takes one lot. Thus, the number of loadports on a tool determines a tool buffer's capacity. Each bay is served by a stocker which is a common storage area for all the tools in a bay, so it serves as common buffer for tools in a bay. Hence, there are two type of buffers in a bay. If there is not enough capacity on the tools, lots wait to be selected for processing in that tool group's stocker. A legitimate move for a lot defined in our setting can be of three types: from the current operation's tool loadport to next operation's tool loadport, from the current operation's tool loadport to next operation's stocker, and from the current operation's stocker to current operation's tool loadport.

Most current AMHS scheduling is implemented by vehicle dispatching. Roughly speaking, when a lot is available for pickup it picks the closest vehicle available. Due to information infrastructure issues, there is limited communication between job process information system and the AMHS information system. Therefore, a lot signals a move request only when it is available and a vehicle responds to that only when it is available. Recent efforts integrate these two separate information systems up to a degree that each system can see the other for a limited amount of time. This allows us to assume an information availability window of several minutes.

We assume a real-time dynamic system in which we make decisions during the execution. In our problem setting, there is a certain time window in which information is available to the scheduler/decision maker. Within that time window, we assume that the scheduler knows where a lot or a vehicle is located and when it is going to be available, if at all. Our overall aim is to assign vehicles and jobs to each other by maximizing the AMHS performance, which has alternative interpretations.

First, we solve the problem for the lots and vehicles available in the first period in which all relevant information is known. Then we move to the next period, and solve the problem with lot and vehicle information available within that period. Each time we move to the next period, we update lot and vehicle sets and value parameters accordingly. This is a dynamic and online approach. We list the indices, parameters time-dependent variables and decision variables first. Then we explain them in more detail. Definition of assignment model constraints and objectives is followed by look-ahead and myopic algorithms. Since there are numerous parameters and their updates, we group them based on their real-time dynamics and lot-vehicle relations. While we make small explanations in between parameters, main explanations are going to take place after we list all the parameters.

Indices:

$l \in L_t$: lots available in period t (L denotes the set of all lots over the planning horizon)

$v \in V_t$: vehicles available in period t (V denotes the set of all vehicles over the planning horizon)

$b \in B$: bays

s: stocker or tool buffer ($stoc\,\text{ker}$ for stocker and $tool$ for tool buffer)

Time Dependent Indices:

These indices are all location related indices and under the domain or bay or buffer (stocker or tool buffer) indices. They either represent a bay-lot relation or buffer-lot relation.

$b'(l)$: origin bay of a lot l

$s'(l)$: origin buffer of a lot l

$b''(l)$: destination bay of a lot l

$s''(l), s'''(l)$: destination buffer of a lot l

$b'(v)$: origin bay of a vehicle v

$s'(v)$: origin buffer of a vehicle v

$b''(v)$: destination bay of a vehicle v

$s''(v), s'''(v)$: destination buffer of a vehicle v

Fixed Parameters:

These parameters remain same regardless of time period change.

τ : length of a time period

$\delta^{\bullet,\bullet}_{\bullet,\bullet}$: travel time from an origin bay-buffer pair to a destination bay-buffer pair

Time Dependent Parameters:

These parameters are updated every time period depending on decisions given in the previous time period and new availabilities.

An important group of time dependent parameters are lot-vehicle assignment related parameters. Depending on the time when decision making is done, lot-vehicle assignment related parameters are updated. All parameters in this group are relevant to various types of travel time values of specific lot-vehicle assignments. While one set of these parameters calculate actual travel times for a specific lot-vehicle assignment, the other set calculates expected travel times after a specific lot-vehicle assignment. Expected travel times consider move request probabilities from each location to calculate how long the next empty or loaded travel time will take if a certain lot-vehicle assignment is done for a tool-to-tool, tool-to-stocker or a stocker-to-tool movement. Type of the movement is important since it determines the next location of lot or vehicle. Expected travel time parameters contribute to pre-positioning of vehicles for future move

requests by concentrating them around locations where higher move requests arise. This aim is obtained by several constraints and components of the objective function we present in the optimization model later.

d_{lv} : empty vehicle travel time of due to lot l vehicle v assignment

p_{lv} : tool-to-tool delivery time due to lot l vehicle v assignment

q_{lv} : tool-to-stocker delivery time due to lot l vehicle v assignment

r_{lv} : stocker-to-tool delivery time due to lot l vehicle v assignment

ep_{lv} : expected empty travel time of vehicle v if current lot l vehicle v assignment is tool-to-tool

eq_{lv} : expected empty travel time of vehicle v if current lot l vehicle v assignment is tool-to-stocker

er_{lv} : expected empty travel time of vehicle v if current lot l vehicle v assignment is stocker-to-tool

If a vehicle is not assigned in a current time period, expected empty travel time is calculated independent of specific lot assignment.

ed_{v} : expected empty travel time of vehicle v if vehicle v is not assigned to any lot in the current period

Similar to expected travel time of vehicles, expected travel time of lots are updated by a group of parameters. These parameters keep either delivery times or waiting times for next decisions depending on current decisions. Next location of a lot depending on the current decision is the main factor to determine expected delivery time or expected waiting time for the next decision. For example, if a lot is carried to destination stocker both expected delivery time and waiting time are 0 since the lot is already served to its

destination. However, if a lot is moved to a stocker, then the historical waiting time updated each period is associated with that lot as its expected waiting time.

epr_l: expected tool-to-tool or tool-to-stocker-to-tool delivery time of lot l

eqr_l: expected destination stocker-to-tool delivery time of lot l

erp_l: expected origin stocker-to-tool delivery time of lot l

$ewpr_l$: expected waiting time of lot l in the stocker if the next movement is tool-to-stocker-to-tool

$ewqr_l$: expected waiting time of lot l in the stocker if the next movement is destination stocker-to-tool

$ewrp_l$: expected waiting time of lot l in the stocker if the next movement is origin stocker-to-tool

Actual waiting time of a lot is the time between the current period and the time when the lot becomes available for pick up the first time.

w_l: waiting time of lot l until the current period

Location parameters are updated every period depending on current period decisions.

os_{lb}: 1 if lot l's origin bay is b, 0 otherwise

ds_{lb}: 1 if lot l's destination bay is b, 0 otherwise

θ_l: 1 if lot l is located on a tool buffer, 0 otherwise

ω_l: 1 if lot l is located in a stocker, 0 otherwise

β_v: bay in which vehicle v is located

σ_v: buffer in which vehicle v is located (1 for stocker, 2 for tool buffer)

β_l : origin bay of lot l

γ_l : destination bay of lot l

Available tool buffer capacity and stocker capacity in a bay are updated every period. This update is done by taking lots leaving and arriving from and to the location within current period into consideration. Also, similarly, number of move requests originated from a tool buffer or stocker within a bay is calculated similarly. Number of move requests in a location is nothing but same as number of lots within a location. So total number of move requests is same as total number of existing lots. These numbers change every period since each lot which is directed to its destination tool buffer are removed from consideration and new lots (move requests) emerge into the system. Move request probabilities, which are calculated as the ratios of move requests in a certain location to the total number of move requests, are used in expected travel time and delivery time calculations.

c_b : available capacity of bay b tool buffer

k_b : available capacity of bay b stocker

m_b^s : number of move requests originated in bay b tool buffer or stocker in the current period

m : total number of move requests in the system in the current period

π_b^s : probability of a move request originating in bay b tool buffer or stocker in the current period

Finally, availability time and availability time period of lots and vehicles are updated every time period depending on current period decisions. If a lot is assigned to a vehicle, both become available again when vehicle delivers the lot to its destination

location. If a vehicle or lot is not assigned in the current time period, then they become available again for pick up in the next time period. While time is continuous, time period is a discrete term which consists of certain length of time and found by getting the ceiling of division of time by length of time period.

$a_\bullet$: time when a lot l or vehicle v is available

$at_\bullet$: time period in which a lot l or vehicle v becomes available

Decision Variables:

X_{lv}: 1 if vehicle $v \in V_t$ is assigned to lot $l \in L_t$, 0 otherwise

P_{lv}: 1 if vehicle $v \in V_t$ is assigned to lot $l \in L_t$ for a tool-to-tool movement, 0 otherwise

Q_{lv}: 1 if vehicle $v \in V_t$ is assigned to lot $l \in L_t$ for a tool-to-stocker movement, 0 otherwise

R_{lv}: 1 if vehicle $v \in V_t$ is assigned to lot $l \in L_t$ for a stocker-to-tool movement, 0 otherwise

D_v: Actual empty travel time of a vehicle v

T_l: Actual delivery time of a lot l

ED_v: Expected empty travel time of a vehicle v

ET_l: Expected delivery time of a lot l

F_l: accumulated waiting time of a lot l

EF_l: Extra waiting time of a lot l

Time dependent parameters are updated at the beginning or end of each time period. Parameters d_{lv}, p_{lv}, q_{lv}, and r_{lv} correspond to the pre-calculated empty vehicle travel time, tool-to-tool delivery time, tool-to-stocker delivery time, and stocker-to-tool

delivery time, respectively, for every lot l vehicle v assignment available in that period, where $l \in L_t$ and $v \in V_t$. Vehicle v's expected empty travel time after current period is denoted by parameters ep_{lv}, eq_{lv}, and er_{lv} depending on if the lot l vehicle v assignment in the current period is a tool-to-tool, tool-to-stocker or a stocker-to-tool movement. Parameter ed_v is the expected empty travel time of vehicle v if it is not assigned to any vehicle in the current period. Lot l's expected delivery time after the current period is calculated by parameters epr_l, eqr_l, and erp_l which are the expected tool-to-tool or tool-to-stocker-to-tool, expected destination stocker-to-tool, and expected origin stocker-to-tool delivery times. Parameters $ewpr_l$, $ewqr_l$, and $ewrp_l$ are lot l's expected waiting time in the stocker depending on if the next movement is a tool-to-stocker-to-tool, destination stocker-to-tool, or origin stocker-to-tool movement. How long a lot already has waited until the current period is stored in parameter w_l. The length of a time period is τ. If a lot l's origin bay is bay b, then os_{lb} is 1, otherwise it is 0. If a lot l's destination bay is bay b, then ds_{lb} is 1, otherwise it is 0. If lot l is located on a tool buffer, then θ_l is 1, otherwise it is 0. If lot l is located on a stocker, then ω_l is 1, otherwise it is 0. Available capacity of bay b's tool buffer is c_b. Available capacity of bay b's stocker is k_b.The update of time dependent variables is made possible by a group of pre-determined parameters and indices. Indices $b'(l)$ and $b'(v)$ are used for origin bay of a lot l and a vehicle v, respectively. Indices $s'(l)$ and $s'(v)$ determine if a lot l and a vehicle v originate from a stocker or a tool buffer, respectively (*stocker* for a stocker and *tool* for a tool buffer). $b''(l)$ and $b''(v)$ are the destination bay of a lot and a vehicle, respectively. $s''(l)$ ($s'''(l)$) and $s'''(v)$ ($s'''(v)$) determine if a lot l and a vehicle

v are directed to a stocker or a tool buffer, respectively. Parameter $\delta_{\cdot,\cdot}^{\cdot,\cdot}$ is the travel time from an origin bay-buffer pair to a destination bay-buffer pair. Parameter $a_{\cdot}$ is the time when a lot l or a vehicle v is available. Parameter $at_{\cdot}$ specifies in which time period a lot l or a vehicle v becomes available. Parameters a_v and at_v are updated for all vehicles $v \in V$ since they are followed during the whole planning horizon while there is no update for lots $l \in L$ since they are not followed in the system after their one step movement as long as a move request does not end in a stocker for a particular lot. When the move ends in a stocker for lot $l \in L$, a_l and at_l are updated and lot l becomes available again for pickup from stocker to its destination tool. Similarly, β_v and σ_v, the bay and buffer (tool or stocker) in which vehicle v is located are updated at each period t, depending on the transported lot l's origin bay, β_l, or destination bay, γ_l. β_l is also updated if a lot moves from a tool to a stocker. If the lot makes a tool to stocker movement since the destination tool buffer is full, then a_l and at_l are updated and lot signals a move request when it arrives to its destination stocker. m_{bs} is the number of move requests given at the bay b stocker or tool buffer in the current time period and m is the total number of move requests in the current time period. Since each move request is associated with a certain lot, parameters m_b^s and m are nothing different than lot counts at bay b stocker or tool buffer and total lot count, respectively. Depending on lot-vehicle assignment decisions every time period and new move requests emerged into the system, parameters m_b^s and m are updated periodically.

4.2.1 Lot-Vehicle Assignment Model

Every period, we solve a lot-vehicle assignment problem formulated as a mixed integer problem. The complete set of vehicles, $v \in V_t$, and lots, $l \in L_t$, are available within period t. Binary assignment decision variable X_{lv} determines if a vehicle $v \in V_t$ is assigned to lot $l \in L_t$ or not in period t. P_{lv}, Q_{lv}, R_{lv} are binary decision variables to decide if a move resulting from a lot vehicle assignment is a tool-to-tool, tool-to-stocker, or stocker-to-tool movement, respectively. Type of a move is determined by the origin location of a lot. If the move request of the lot is given while it is on a tool buffer, then it can either to a tool-to-tool or a tool-to-stocker move. If the lot is located in a stocker, then it can only do a stocker-to-tool move. The empty travel time of a vehicle v is denoted by decision variable D_v. Lot l's delivery time is T_l. These values are actual travel times which result from the assignments in period t. There are also expected travel duration variables for the coming periods depending on the decision given in the current period t. The expected empty travel time of vehicle v is ED_v. The expected delivery time of a lot l is ET_l. Expected time variables are important to pre-position vehicles for future move requests and prioritize lots for current decisions. Expected time and waiting time parameters determine expected time variables. Since location based move request probabilities are used to calculate the travel time expectations, vehicles tend to concentrate around locations with high move request probabilities by employing a proper objective function. Also, if a lot's expected travel time is long for its next move depending on a current decision, it can be given priority again by employing a proper objective function. These priorities or pre-positioning are only implicit effects of decisions based on expected travel time values. These effects are explained according to how the expected time variables are used when we fit an objective function to our

assignment problem. If a lot l is not picked in period t, than its idle waiting time accumulates in F_l, otherwise it waits for an extra period which is indicated by EF_l, expected idle time until the new decision. EF_l ensures the length of the current decision period is also taken into consideration while calculating idle time. Waiting time variables also help prioritize pick up of lots in coordination with expected time variables by employing a proper objective function. Simply, if a lot has already waited much longer than other lots in the system, it implicitly has a higher priority of pick up than other lots depending on other components of the objective which we will discuss next. X_{lv}^*, P_{lv}^*, Q_{lv}^*, and R_{lv}^* are optimal values of the decision variables X_{lv}, P_{lv}, Q_{lv}, and R_{lv}, respectively.

An assignment model which is solved in every period t consists of following objective function and constraints, (4.1)-(4.13).

$$\min \sum_{l \in L_t} T_l + ET_l - F_l + EF_l + \sum_{v \in V_t} D_v + ED_v \tag{4.1}$$

s.t.

$$\sum_{l \in L_t} X_{lv} \le 1, v \in V_t \tag{4.2}$$

$$\sum_{v \in V_t} X_{lv} \le 1, l \in L_t \tag{4.3}$$

$$X_{lv} = P_{lv} + Q_{lv} + R_{lv}, l \in L_t, v \in V_t \tag{4.4}$$

$$D_v \ge d_{lv} X_{lv}, l \in L_t, v \in V_t \tag{4.5}$$

$$T_l = p_{lv} P_{lv} + q_{lv} Q_{lv} + r_{lv} R_{lv}, l \in L_t, v \in V_t \tag{4.6}$$

$$ED_v = \sum_{l \in L} ep_{lv} P_{lv} + eq_{lv} Q_{lv} + er_{lv} R_{lv} + ed_v \left(1 - \sum_{l \in L_t} X_{lv} \right), v \in V_t \tag{4.7}$$

$$ET_l = \left\{ \begin{array}{l} \theta_l \left[\begin{array}{l} \left(epr_l + ewpr_l \right) \left(1 - \sum_{v \in V_t} X_{lv} \right) \\ + \left(eqr_l + ewqr_l \right) \sum_{v \in V_t} Q_{lv} \end{array} \right] \\ + \omega_l \left(erp_l + ewrp_l \right) \left(1 - \sum_{v \in V_t} X_{lv} \right) \end{array} \right\}, l \in L_t \tag{4.8}$$

$$F_l \le \sum_{v \in V_t} w_l X_{lv}, l \in L_t \tag{4.9}$$

$$EF_l \ge \sum_{v \in V_t} \tau \left(1 - X_{lv} \right), l \in L_t \tag{4.10}$$

$$\sum_{l \in L_t} \sum_{v \in V_t} ds_{lb} \left(P_{lv} + R_{lv} \right) \le c_b + \sum_{l \in L_t} \sum_{v \in V_t} os_{lb} \left(P_{lv} + Q_{lv} \right), b \in B \tag{4.11}$$

$$\sum_{l \in L_t} \sum_{v \in V_t} ds_{lb} \left(Q_{lv} \right) \le k_b + \sum_{l \in L_t} \sum_{v \in V_t} os_{lb} \left(R_{lv} \right), b \in B \tag{4.12}$$

$$X_{lv}, P_{lv}, Q_{lv}, R_{lv} = 0,1, l \in L_t, v \in V_t \tag{4.13}$$

$$T_l, ET_l, D_v, ED_v, F_l, EF_l \ge 0, l \in L_t, v \in V_t$$

The objective function in (4.1) minimizes the total actual and expected empty travel time and delivery time in addition to the total actual waiting time subtracted from the total expected waiting time. Lower empty travel time and delivery time are good performance indicators in an AMHS. Therefore, objective function terms, $\sum_{l \in L_t} T_l + \sum_{v \in V_t} D_v$,

are the first obvious choices to minimize. However, it is not sufficient since the optimization problem simply ends up with not assigning any vehicles to lots with only these components in the objective function. We need to have some rewards/penalties for assigned/unassigned vehicles and lots for better judgment. Expected empty travel time and delivery time, $\sum_{l \in L_t} ET_l + \sum_{v \in V_t} ED_v$, are the main components of the objective function that serve to reward/penalty purpose. Minimization of expected empty travel time and delivery time enables us to pre-position vehicles better to respond quickly for future move requests. Also, it lets us avoid unnecessary idling of lots considering future moves. We subtract actual waiting time, $\sum_{l \in L_t} F_l$, from the rest of the terms in the objective function to give a sense of priority to the lots that wait in the system longer than others. Minimization of expected waiting time, $\sum_{l \in L_t} EF_l$, ensures a lot being assigned to a vehicle if there is no need for it to wait while there is enough transportation capacity.

Constraints (4.2) and (4.3) ensure that each lot or vehicle can be assigned at most once. Construction of optimization function with added reward and penalties as explained above make sure lot-vehicle assignments take place when required. A vehicle tool assignment can be made for one of three move types, a tool-to-tool, a tool-to-stocker, or a stocker-to-tool move in (4.4). A lot makes a tool-to-tool or a tool-to-stocker movement only if it is located in a tool buffer. A lot which is located in a stocker can only stocker-to-tool buffer. We fix corresponding assignment variables such that a lot can only make move types it is allowed to make before solving the model.

The empty travel time and expected empty travel time of vehicle v is calculated in (4.5) and (4.7), respectively. A vehicle's expected empty travel time depends on location of vehicle after the current decision. Whether a vehicle is assigned or not,

depending on the move request probabilities coming from different locations, expected empty travel time is calculated between vehicle's location after current decision and possible next new location of the vehicle.

The delivery time and expected delivery time of lot l is calculated in (4.6) and (4.8) respectively. If the lot is located in a tool, depending on the destination tool buffer availability, the lot's new location can be either destination tool or destination stocker. If the lot is not assigned to any vehicle, then it stays in its current tool buffer. Then, the next move can either be a tool-to-tool or a tool-to-stocker movement. Each decision has its own expected travel time and expected waiting time in stocker values. If the lot moves tool-to-stocker, then it will take a certain time to move it to the next tool buffer when the new decision is made in addition to the waiting time in the stocker. If the lot makes a tool-to-tool movement, then the expected delivery time is 0 since the lot is already served and it does not appear in the system again. If the lot is in the stocker, then waiting time in that stocker is added to expected delivery time of next stocker-to-tool movement.

While the actual waiting time of lot l is calculated in (4.9), the expected waiting time of lot l is calculated in (4.10). The capacity constraint of the tool buffer of bay b is (4.11). The stocker capacity of bay b is controlled by (4.12). Binary and non-negativity constraints are contained in (4.13). The vehicle-lot assignment model is (4.1)-(4.13).

4.2.2 Parameter Updates in Period *t*

The parameter equations and updates which construct the algorithm and the assignment model are listed in (4.14) to (4.41). As we will see in the look-ahead algorithm next, parameter updates from (4.14) to (4.31) are updated at the beginning of every time period before the assignment problem is solved and parameter updates from (4.32) to (4.41) take place after the assignment problem is solved.

$$d_{lv} = \delta^{s'(v),s'(l)}_{b'(v),b'(l)}, l \in L_t, v \in V_t \tag{4.14}$$

$$\begin{aligned} p_{lv} &= \max\left[a_v - (t-1)\tau + d_{lv}, a_l - (t-1)\tau\right] \\ &+ \delta^{s'(l),s''(l)}_{b'(l),b''(l)} + ld + ud, l \in L_t, v \in V_t, s''(l) = tool \end{aligned} \tag{4.15}$$

$$\begin{aligned} q_{lv} &= \max\left[a_v - (t-1)\tau + d_{lv}, a_l - (t-1)\tau\right] \\ &+ \delta^{s'(l),s'''(l)}_{b'(l),b''(l)} + ld + ud, l \in L_t, v \in V_t, s'''(l) = stoc\,\text{ker} \end{aligned} \tag{4.16}$$

$$\begin{aligned} r_{lv} &= \max\left[a_v - (t-1)\tau + d_{lv}, a_l - (t-1)\tau\right] \\ &+ \delta^{s'(l),s''(l)}_{b'(l),b''(l)} + ld + ud, l \in L_t, v \in V_t, s''(l) = tool \end{aligned} \tag{4.17}$$

Actual empty travel time and delivery time are updated based on the current location of a lot and a vehicle. Empty travel time is the total transportation time between vehicle's current location and current location of the lot which the vehicle is going to pick. Tool-to-tool delivery time is updated in (4.14) if the lot is located in its original tool buffer and it's directed to the destination tool buffer ($s''(l) = tool$). Tool-to-stocker delivery time is updated in (4.15) if the lot is located in its original tool buffer and it's directed to the destination stocker ($s'''(l) = stoc\,\text{ker}$). Finally, stocker-to-tool delivery time is updated in (4.16) if the lot is located in its original tool buffer and it's directed to the destination stocker ($s'''(l) = stoc\,\text{ker}$).

$$ed_v = \sum_{b \in B} \sum_{s = stoc\,\text{ker}, tool} \pi^s_b \delta^{s''(l),s}_{b''(l),b}, v \in V_t \tag{4.18}$$

$$ep_{lv} = \sum_{b \in B} \sum_{s = stoc\,\text{ker}, tool} \pi^s_b \delta^{s''(l),s}_{b''(l),b}, l \in L_t, v \in V_t, s''(l) = tool \tag{4.19}$$

$$eq_{lv} = \sum_{b \in B} \sum_{s = stoc\,\text{ker}, tool} \pi^s_b \delta^{s'''(l),s}_{b''(l),b}, l \in L_t, v \in V_t, s'''(l) = stoc\,\text{ker} \tag{4.20}$$

$$er_{lv} = \sum_{b \in B} \sum_{s = stoc\,\text{ker}, tool} \pi^s_b \delta^{s''(l),s}_{b''(l),b}, l \in L_t, v \in V_t, s''(l) = tool \tag{4.21}$$

If a vehicle is not assigned to a lot, expected empty travel time is updated depending on vehicle's current location and move request probabilities emerging from

every location in (4.18). If the vehicle is assigned to a lot, then depending on the lot's next location since the assignment type is going to determine lot's new place, empty travel time is calculated for each current move type (4.19)-(4.21).

$$epr_l = \pi_{b''(l)}^{s''(l)} \begin{pmatrix} \delta_{b'(l),b''(l)}^{s'(l),s''(l)} \\ +ld \\ +ud \end{pmatrix} + \left(1-\pi_{b''(l)}^{s''(l)}\right) \begin{pmatrix} \delta_{b'(l),b''(l)}^{s'(l),s'''(l)} \\ +\delta_{b''(l),b''(l)}^{s'''(l),s''(l)} \\ +2ld \\ +2ud \end{pmatrix}, l \in L_t, s''(l) = tool, s'''(l) = stoc \text{ ker} \quad (4.22)$$

$$eqr_l = \delta_{b''(l),b''(l)}^{s'''(l),s''(l)} + ld + ud, l \in L_t, s''(l) = tool, s'''(l) = stoc \text{ ker} \quad (4.23)$$

$$erp_l = \delta_{b'(l),b''(l)}^{s'(l),s''(l)} + ld + ud, l \in L_t, s''(l) = tool \quad (4.24)$$

The expected delivery time takes move request probabilities and current and later lot locations into consideration for each of the different current and future move type combinations. If a lot currently does not make a move while it is on a tool buffer, then the expected empty travel time includes both tool-to-tool and tool-to-stocker move options for the next decision depending on the probability of the destination tool buffer being available by a move request is being signaled from that location by that certain location's move request probability in (4.22). If the lot is on a tool buffer but it is currently assigned for a tool-to-stocker move or the lot is on a stocker and it is not assigned to any vehicle, then the same expected delivery time is calculated for both cases as the time between destination stocker and destination tool buffer in addition to load and unload times in (4.23) and (4.24).

$$ewpr_l = \left(1-\pi_{b''(l)}^{s''(l)}\right) w_{b''(l)}, l \in L_t, s''(l) = tool \quad (4.25)$$

$$ewqr_l = w_{b''(l)}, l \in L_t \quad (4.26)$$

$$ewqr_l = w_{b'(l)}, l \in L_t \quad (4.27)$$

$$w_l = t\tau - a_l, l \in L_t \quad (4.28)$$

Waiting time in the stocker if the lot is in a tool buffer but it is not assigned is relevant to if the next tool buffer is going to be available depending on the move request probability from that tool buffer in (4.25). But if the lot makes a tool-stocker movement or if it is already in a stocker, then waiting time in that stocker is assigned to expected waiting time for that lot in (4.26) and (4.27). If the lot is still in the system, the waiting time is accumulated in (4.28).

$$m_b^s = \left|\left\{l : \beta_l = b, \sigma_l = s, l \in L_t\right\}\right|, \forall b \in B, s = stoc\,\text{ker}, tool \tag{4.29}$$

$$m = \left|L_t\right| \tag{4.30}$$

$$\pi_b^s = m_b^s / m, b \in B, s = stoc\,\text{ker}, tool \tag{4.31}$$

Number of lots in a stocker and tool buffer are calculated in (4.29). Total number of move requests in the system is added up in 4.30. Move request probability of a location is the ratio of number of lots in a tool buffer or stocker of a bay to total number of lots in the current time period, as in (4.31).

$$a_v \leftarrow \max\left[\left(a_v + \sum_{l \in L_t} p_{lv} P_{lv}^* + q_{lv} Q_{lv}^* + r_{lv} R_{lv}^*\right)\sum_{l \in L_t} X_{lv}^* + t\tau\left(1 - \sum_{l \in L_t} X_{lv}^*\right), t\tau\right], v \in V_t \tag{4.32}$$

$$at_v = \lfloor a_v / \tau \rfloor + 1, v \in V_t \tag{4.33}$$

$$\beta_v \leftarrow \beta_v\left(1 - \sum_{l \in L_t} X_{lv}^*\right) + \sum_{l \in L_t} \gamma_l P_{lv}^* + \gamma_l Q_{lv}^* + \beta_l R_{lv}^*, v \in V_t \tag{4.34}$$

$$\sigma_v \leftarrow \sigma_v\left(1 - \sum_{l \in L_t} X_{lv}^*\right) + \sum_{l \in L_t} 2P_{lv}^* + Q_{lv}^* + 2R_{lv}^*, v \in V_t \tag{4.35}$$

If a vehicle is assigned to a lot, depending on vehicle's current availability and delivery time of the assigned lot, new vehicle availability is calculated in (4.32). If the vehicle is not assigned in the current period, then it becomes available again at he the beginning of the next period. Also, since our model does not allow a lot or a vehicle a

second move within the same time period, if the vehicle becomes available again before the next time period begins due to short delivery time of the assigned lot, the model assumes that the vehicle still becomes available at the beginning of the next time period. The time period the vehicle is going to be available is the ceiling of the time it is going to be available, calculated in (4.33). The bay location and the buffer location of a vehicle are updated in (4.34) and (4.35), respectively, depending on the vehicle's current location and the move type if it moved.

$$a_l \leftarrow \max\left[\left(a_l + \sum_{v \in V_t} q_{lv}\right)\sum_{v \in V_t} Q^*_{lv}, t\tau\right] + t\tau\left(1 - \sum_{v \in V_t} X^*_{lv}\right), l \in L_t \tag{4.36}$$

$$at_l = \left(\lfloor a_l / \tau \rfloor + 1\right)\left(1 - \sum_{v \in V_t} X^*_{lv} + \sum_{v \in V} Q^*_{lv}\right), l \in L_t \tag{4.37}$$

$$\beta_l \leftarrow \beta_l\left(1 - \sum_{v \in V_t} X^*_{lv}\right) + \gamma_l \sum_{v \in V} Q^*_{lv}, l \in L_t \tag{4.38}$$

$$\sigma_l \leftarrow \sigma_l\left(1 - \sum_{v \in V_t} X^*_{lv}\right) + \sum_{v \in V} Q^*_{lv}, l \in L_t \tag{4.39}$$

A lot's available time is calculated depending on its current available time and delivery time in (4.36) and (4.37) only if it makes a tool-to-stocker movement or it is not picked up at all. If the lot is not picked up, then it becomes available again at the beginning of the next time period. Also, since a second move within the same period is not allowed, if the lot becomes available again before the next time period begins, the model takes that lot into consideration for pick up at the beginning of the next time period. The location of the lot is updated to its destination bay and buffer in (4.38) and (4.39) if it makes a tool-to-stocker movement. Otherwise, it remains in the same location. If the lot makes a tool-to-tool or stocker-to-tool movement (if the lot reaches to its

destination tool buffer), there is no need to update it availability and location since it is considered to be served by the model.

$$c_b \leftarrow c_b + \sum_{l \in L_t} \sum_{v \in V_t} os_{lb} \left(P_{lv}^* + Q_{lv}^* \right) - \sum_{l \in L_t} \sum_{v \in V_t} ds_{lb} \left(P_{lv}^* + R_{lv}^* \right), b \in B \tag{4.40}$$

$$k_b \leftarrow k_b + \sum_{l \in L_t} \sum_{v \in V_t} os_{lb} R_{lv}^* - \sum_{l \in L_t} \sum_{v \in V_t} ds_{lb} Q_{lv}^*, b \in B \tag{4.41}$$

Available tool buffer capacity of a bay and available stocker capacity of a bay are updated in (4.40) and (4.41), respectively. If a lot moves from a bay's tool buffer by making a tool-to-tool or tool-to-stocker movement, then an extra capacity becomes available within that tool buffer. If the lot makes a stocker-to-tool movement to the same bay, then it reduces the availability of corresponding bay's tool buffer capacity by 1. Similarly, a bay's stocker capacity is increased by 1 if a stocker-to-tool movement is made from that bay. The bay's stocker capacity reduces by 1 if there is a lot makes a tool-to-stocker move to that bay.

Although parameter updates are very detailed, they all same the same purpose which is to provide sufficient travel time and delivery time information to the assignment problem solved within look ahead algorithm and used in myopic algorithm. The updating mechanism mainly lies on updating lot and vehicle locations, their available time information and buffer capacities. Next, we sketch the look-ahead and myopic algorithms along with how the parameters are updated during the algorithms.

4.2.3 Look-Ahead Algorithm

The look-ahead algorithm consists of one initial and three basic steps which are repeated for every time period. A lot and vehicle assignment problem is solved every time period with the lots and vehicles available within that time period. Sets of available

lots and vehicles and system parameters are updated between periods. The algorithm ends at the end of the last time period in the planning horizon.

Step 0: Initiate $a_l, a_v, at_l, at_v, \beta_l, \sigma_l, \gamma_l, \beta_v, \sigma_v, c_b, k_b, m_b^s, m, \pi_b^s, l \in L, v \in V, b \in \mathrm{B}, s = tool, stoc\,\mathrm{ker}$.

$L_1 = \{\ \}$, $V_1 = \{\ \}$, $t = 1$

Step 1: $L_t \leftarrow L_t \cup \{l : at_l = t, l \in L_{t-1}\}$, $V_t \leftarrow V_t \cup \{v : at_v = t, v \in V_{t-1}\}$.

Update (4.14)-(4.31).

If $\sigma_l = 1$, $P_{lv}^* = Q_{lv}^* = 0, l \in L_t, v \in V_t$

Else, $R_{lv}^* = 0, l \in L_t, v \in V_t$

Step 2: Solve (4.1)-(4.13).

Step 3: Update (4.32)-(4.41).

$t \leftarrow t + 1$.

If $t > T$,

STOP.

Otherwise,

$$L_t = L_{t-1} \setminus \left\{ l : \sum_{v \in V_{t-1}} X_{lv}^* = 1, l \in L_{t-1} \right\},\ V_t = V_{t-1} \setminus \left\{ v : \sum_{l \in L_{t-1}} X_{lv}^* = 1, v \in V_{t-1} \right\}.$$

Go to Step 1.

4.2.4 Myopic Algorithm

The myopic algorithm is myopic in information availability and greedy in its objective. A lot can only see available vehicles when it becomes available for pickup and calls the closest vehicle for pickup if there is any available vehicle. Otherwise, a lot waits until a vehicle becomes available. If the destination tool buffer is available, then the lot is

transferred tool-to-tool. If the destination buffer is not available, then the lot is moved tool-to-stocker and waits in the stocker to be transferred to its destination tool. After serving each move request, the heuristic checks if lots which couldn't be served in earlier steps of the algorithm could be served at the current step if its destination tool has become available. The algorithm stops at the end of the planning horizon. Since there is no decision period concern in the myopic algorithm, we consolidate available vehicles in set V_a instead of V_t. Since decisions are made per lot l at a time, available lot set L_a consists of only lot l being considered at the time. While updating parameters, we use lot and vehicle sets L_a and V_a instead of L_t and V_t in parameter update equations (4.14)-(4.17) and (4.32)-(4.41).

Step 0: Set clock to 0. Initiate $a_l, a_v, at_l, at_v, \beta_l, \gamma_l, \beta_v, \sigma_v, c_b, k_b, l \in L, v \in V, b \in \mathrm{B}$.

$l = 1$, $V_a = \{\ \}$

Step 1: Set clock to a_l, $V_a \leftarrow V_a \cup \{v : a_{l-1} < a_v \le a_l, v \in V\}$, $L_a = \{l\}$.

If

Update (4.14)-(4.17).

Step 2: If $c_{\gamma_l} > 0$,

Assign $v^* = \arg\min_{v \in V_a} \{d_{lv}, v \in V_a\}$ to lot l, $X^*_{lv^*} = 1$

If $\sigma_l = 2$, $P^*_{lv^*} = 1, Q^*_{lv^*} = R^*_{lv^*} = 0$.

Else, $R^*_{lv^*} = 1, P^*_{lv^*} = Q^*_{lv^*} = 0$.

Else

If $\sigma_l = 2$,

If $s_{\gamma_l} > 0$, assign $v^* = \arg\min_{v \in V_a} \{d_{lv}, v \in V_a\}$ to lot l, $X^*_{lv^*} = 1$,

$$Q^*_{lv^*} = 1, P^*_{lv^*} = R^*_{lv^*} = 0$$

Else, $X^*_{lv} = P^*_{lv} = Q^*_{lv} = R^*_{lv} = 0$

Step 3: Update (4.32)-(4.41), $V_a \leftarrow V_a \setminus \{v^*\}$, $l \leftarrow l + 1$.

If $l \notin L$, STOP.

Else, Go to Step

4.3 COMPUTATIONAL EXPERIMENTS

We test our algorithms with a wide range of experiments. We work on a spine layout which is commonly used in the industry. The layout consists of 8 bays, each of which consists of one stocker and one tool. In a real fab number of bays may be 20 or even more. In our experiments, one tool represents a combination of tools in a bay to keep the model simple. Putting one stocker per bay is a common practice, although in general the number of stockers may vary. Instead of distances, we locate the fab components using vehicle travel times between them. Intrabay distances are 2 minutes each. Between every intersection, there is a 1 minute distance for interbay movement. There is a crossover in the middle of the inner circle of the spine layout. The tracks are bi-directional. We put enough crossovers so that we can work in a collision-free environment and we can ignore the waiting time of vehicles due to traffic. Figure 4.1 depicts the fab layout.

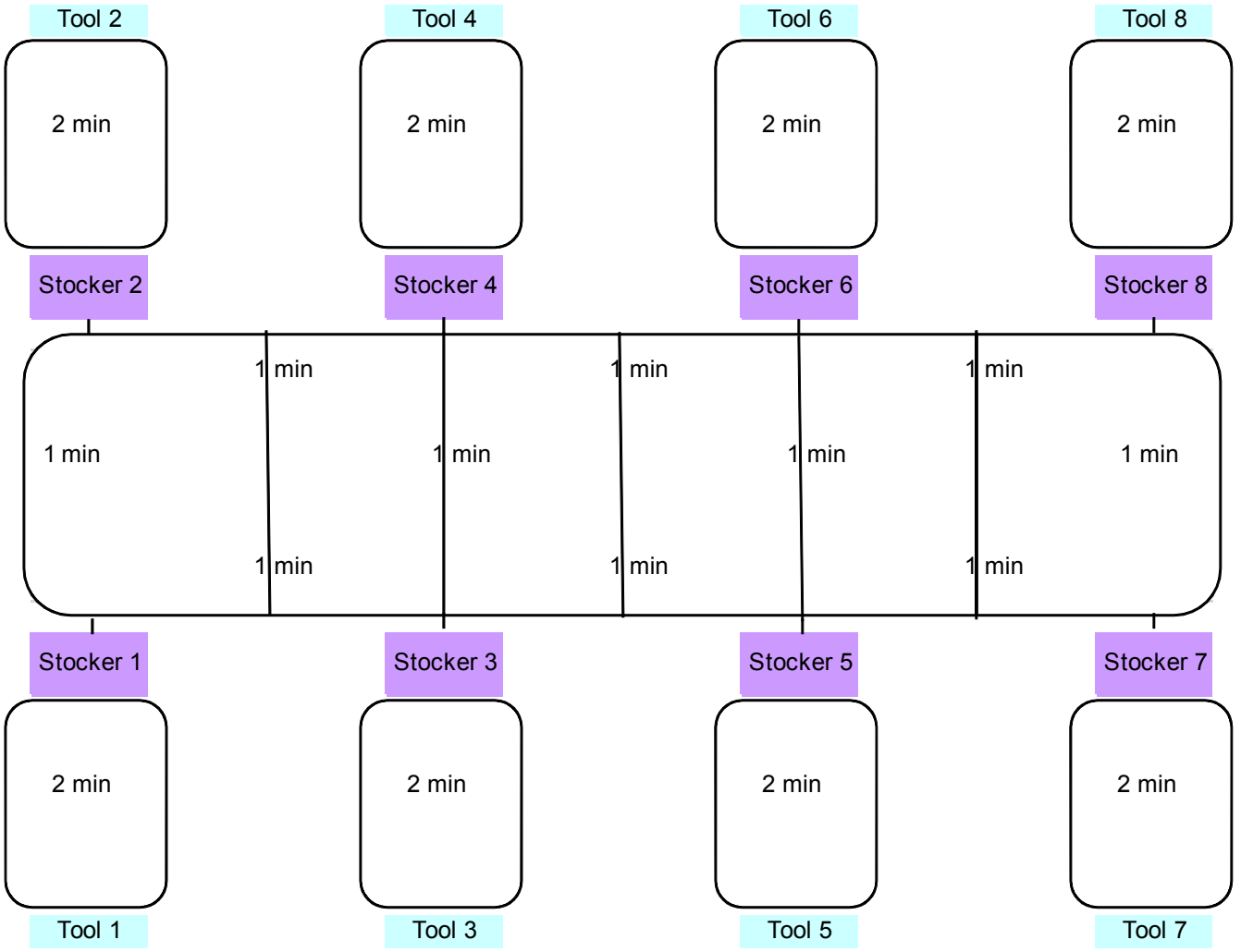


Figure 4.1 Fab Layout

Loading and unloading of the vehicles take 1 minute each. Therefore it takes an extra 2 minutes to deliver a lot from a location to another in addition to the travel time.

We divide the planning horizon into 50 or 100 intervals of 5 or 10 minutes. On average, 1 new move request is signaled every minute or every two minutes. The size of the vehicle fleet is 9 or 12. When deciding on move request rates and vehicle fleet size we considered vehicle utilizations in the experiments. Although the average vehicle utilization ranges from 30% to 99% in the experiments, for most cases the utilization is between 50% and 80%, which is similar to a typical fab. All the vehicles are released to the system in the first time period. Lots appear in the system as move requests. As part of the primitive data we input probabilities of where a lot is located. We call these probabilities "move request probabilities," which can be extracted from the historical data in a fab. They are also updated throughout the look-ahead algorithm as explained in Section 4.2.2. In our experiments, we assume all tools are equally likely to signal a move request as presented in Table 4.1. Also, we assume that stockers are empty at the beginning so the probability of initial move request from stockers is 0.

Table 4.1 Initial move request probabilities

Bay Number	Stocker	Tool
1	0	0.125
2	0	0.125
3	0	0.125
4	0	0.125
5	0	0.125
6	0	0.125
7	0	0.125
8	0	0.125

Since stockers are used for intermediate storage, in reality they have limited capacity. In our experiments, we assume the capacity is infinite, 1 or 0. When it is 0, only tool-to-tool moves are available. In other words, there are no stockers in this case. At time zero, we assume that all the tool buffer capacities are full except for one available capacity per tool buffer on the tools. For one case only, we assumed that the beginning available tool buffer capacity is 0 with infinite stocker capacity. In the experimental charts, we do not explicitly note exact tool buffer capacity since all the tool buffers are full except for one for each tool and a new lot can be placed only if an existing lot is removed from a tool buffer. However, the tool buffer capacity is implicitly assumed to be 3 or 6 which is typical in real fabs. It is important to note that having no stocker is not a common practice in fabs. Almost all existing fabs use stockers in their layout. However, there are some efforts to build fabs with no intermediate storages. A real fab is much more complex than the model we propose in our study. For example, we ignored breakdowns in our model. Also all lots are assumed to have the same priority. Fabs usually keep very high capacity stockers which are in many cases have practically an infinite capacity. Therefore the uncapacitated stocker case is close to a real fab environment in current practice, and the limited stocker case (0 or 1 lot capacity) can be viewed as a way of predicting future practice.

A lot is assumed to be completed when it reaches the destination tool. If it is stays in the stocker or is not picked up at all, then it is not completed. A move may be any of three types: tool-to-tool, tool-to-stocker, or stocker-to-tool depending on where tool is located and where it is directed. If a lot goes from one tool to its destination tool directly, then this is considered one (tool-to-tool) move. If it goes from one tool to the other tool

visiting a stocker in between, then that completion is accomplished with 2 moves. The empty travel time of a vehicle is the time between a vehicle starts moving and picking up a lot, including the empty travel time of the vehicle. It does not include pickup time. The average empty travel time is calculated by dividing the total empty travel time by the total number of moves, which is sum of all tool-tool, tool-to-stocker and stocker-to-tool moves. The delivery time is the time between a vehicle starts picking up a lot and delivers it to its destination. The first and main component is the travel time between lot's origin and destination. The second and third components, load and unload times of lot to and from vehicle are included in the delivery time, too. Existence of the fourth component, which is full or partial empty travel time of the vehicle depends on lot and vehicle availability. If the lot is available before the vehicle becomes available, then whole empty travel time of the vehicle is included in the delivery time. However, if the lot becomes available after vehicle, then vehicle moves to the location of the lot before the lot becomes available. In this case, empty travel time is not included in the delivery time. Depending on vehicle's arrival to lot location, empty travel can be added to delivery time partially. Functions for delivery time can be seen in equations (4.15)-(4.17).

4.3.1 Computational Results

The look-ahead algorithm is coded in GAMS (Rosenthal 2007) and solved using CPLEX 9.0 (ILOG 2003) since it is an optimization based heuristic. The myopic algorithm is coded in Java (Sun Microsystems 2009). Since the sizes of the assignment problems are very small, all the assignment models during these executions have been solved to optimality within either very small or occasionally bigger fraction of a minute.

We run 25 different experiments in total. The performance measure of each experiment is obtained by taking the average of 5 independent runs. Hence, we run a total of 125 distinct instances for each of our two algorithms. Within each experimental point, randomness comes from when and where the move requests are created and initial positions of vehicles from one instance to the other. We do not consider a warm-up period since system starts loaded and the horizon is long enough. Therefore, performance measures are taken over all the run. Table 4.2 lists the experiments and the associated parameters.

Table 4.2 List of experiments and corresponding experimental parameters

Experiment Number	Length of Planning Horizon	Number of Time Periods	Interval Length	Number of Move Requests	Number of Vehicles	Tool Buffer Capacity	Stocker Capacity
1	500	50	10	250	9	0	∞
2	500	50	10	250	9	1	∞
3	500	50	10	250	9	1	1
4	500	50	10	250	9	1	0
5	1000	100	10	500	9	1	∞
6	1000	100	10	500	9	1	1
7	1000	100	10	500	9	1	0
8	1000	100	10	1000	9	1	∞
9	1000	100	10	1000	9	1	1
10	1000	100	10	1000	9	1	0
11	500	100	5	500	9	1	∞
12	500	100	5	500	9	1	1
13	500	100	5	500	9	1	0
14	1000	100	10	1000	12	1	∞
15	1000	100	10	1000	12	1	1
16	1000	100	10	1000	12	1	0
17	500	100	5	500	12	1	∞
18	500	100	5	500	12	1	1
19	500	100	5	500	12	1	0
20	500	50	10	250	12	1	∞
21	500	50	10	250	12	1	1
22	500	50	10	250	12	1	0
23	1000	100	10	500	12	1	∞
24	1000	100	10	500	12	1	1
25	1000	100	10	500	12	1	0

The performance measures we use to evaluate our experiments are chosen from the widely used performance metrics for the AMHS in semiconductor manufacturing. The number of moves is the total number of AMHS moves made. The process of a

vehicle traveling to a lot location, loading the lot, delivering it to the target location (either a tool or a stocker) and unloading the lot is considered a single move. The service ratio is the ratio of the number of lots which are delivered to their respective destination tools to the number of all lots. For example, while a tool-to-tool movement or a stocker-to-tool movement contributes to the numerator of the service ratio, a tool-to-stocker movement does not. The tool-to-tool ratio is the ratio of tool-to-tool moves to total number of moves. The tool-to-stocker ratio is the ratio of tool-to-stocker moves to the total number of moves. The stocker-to-tool ratio is the ratio of tool-to-tool moves to the total number of moves. The empty travel time per move and delivery time per move are average times spent by a vehicle for empty and loaded travel, respectively. The average utilization of a vehicle gives an indication of the load on the system. The average waiting time of a lot is an average of waiting times of lots until they are picked up for delivery to their destination tool. Therefore the time spent in stockers contributes to this performance measure. The maximum waiting time of lots gives an indication of the range of waiting times.

Table 4.3 Comparison of average performances of all experiments

	Overall Average	
	Look Ahead	Myopic
Number of moves	576	538
Service ratio	91.3%	78.4%
Ratio of tool to tool moves	74.8%	59.0%
Ratio of tool to stocker moves	13.3%	21.5%
Ratio of stocker to tool moves	11.8%	19.5%
Empty travel time per move	1.66	5.82
Delivery time per move	5.30	11.76
Average utilization of vehicles	57.1%	81.9%
Average waiting time of lots	38.1	78.7
Maximum waiting time of lots	244	345

Table 4.3 lists the overall average of performance measures for all runs, for Experiments 1 to 25. The look-ahead algorithm achieves a greater number of moves and a higher service rate with a lower vehicle utilization. The tool-to-tool ratio is higher using the look-ahead approach. The main benefit of the look-ahead approach and the information availability window is manifested in the lower empty travel time per move and lower delivery time per move. There is almost a 70% reduction in empty travel time per move as a result of better pre-positioning via the look-ahead approach. Since look-ahead approach considers expected travel times for the future decisions, vehicles are positioned closer to future move request locations. Furthermore, the delivery time reduction is more than 50%. Both of these effects lead to lower vehicle utilization. With the look-ahead approach, excess vehicle availability is utilized to deliver more lots which leads to higher service ratio. Also, the average and maximum waiting times of lots are

reduced by the look-ahead approach. This is again a result of better pre-positioning and improved scheduling ability due to the information availability window.

Table 4.4 Comparison of alternative storage capacity results

	uncapacitated stocker		stocker capacity = 1		stocker capacity = 0	
	Look Ahead	Myopic	Look Ahead	Myopic	Look Ahead	Myopic
Number of moves	683	618	555	586	518	433
Service ratio	89.4%	78.7%	91.9%	76.0%	92.3%	78.8%
Ratio of tool to tool moves	44.9%	40.2%	84.4%	40.1%	100.0%	100.0%
Ratio of tool to stocker moves	29.5%	32.1%	7.9%	30.5%	0.0%	0.0%
Ratio of stocker to tool moves	25.6%	27.7%	7.6%	29.5%	0.0%	0.0%
Empty travel time per move	1.98	5.49	1.55	5.45	1.41	6.57
Delivery time per move	5.80	10.78	5.08	10.68	4.88	14.01
Average utilization of vehicles	68.6%	86.0%	54.2%	82.0%	49.8%	80.0%
Average waiting time of lots	45.3	81.1	35.9	87.2	34.2	74.5
Maximum waiting time of lots	218	285	257	376	265	396

Table 4.4 summarizes experiments based on varying the stocker capacity. The uncapacitated stocker results are the average of Experiments 2, 5, 8, 11, 14, 17, 20, and 23. The single lot capacity stocker results are the average of Experiments 3, 6, 9, 12, 15, 18, 21, and 24. Zero capacity stocker results are the average of Experiments 4, 7, 10, 13, 16, 19, 22, and 25. The look-ahead algorithm shows better performance than the myopic algorithm in all the key performance measures for each level of stocker capacity. The myopic algorithm has a similar service ratio and average lot waiting times for the uncapacitated stocker and zero capacity stocker cases. When the stocker has a one lot capacity, the myopic algorithm shows the worst performance in service ratio and average lot waiting times. The look-ahead algorithm gets better as stocker capacity changes from infinity to 0. The most important result of this analysis is to observe that both algorithms

perform their best in overall performance measures when there is no stocker. It is obvious that the look-ahead algorithm exploits the future information availability to achieve more tool-to-tool moves which result in no stocker use ultimately. Also, the myopic algorithm acquires an implicit look-ahead approach due to the lack of stockers. A lot waiting on a tool unwillingly due to the lack of a stocker can find its destination available within a couple of minutes and be directly transferred to the destination tool buffer instead of being directed to the stocker first and the tool buffer later. Although we sacrifice the safety of having a stocker as an intermediate storage area, we eliminate an extra move to complete a lot by keeping it on the tool for a couple of minutes more. Also, the extra transportation, load and unload times due to the stocker visits are eliminated in addition to the waiting time in the stocker.

Table 4.5 Comparison of alternative vehicle fleet sizes

	number of vehicles = 9		number of vehicles = 12	
	Look Ahead	Myopic	Look Ahead	Myopic
Number of moves	565	501	606	590
Service ratio	89.7%	73.2%	92.7%	82.5%
Ratio of tool to tool moves	76.9%	60.2%	75.9%	60.0%
Ratio of tool to stocker moves	12.2%	20.8%	12.7%	20.9%
Ratio of stocker to tool moves	10.8%	19.0%	11.3%	19.1%
eEmpty travel time per move	1.87	5.86	1.43	5.82
Delivery time per move	5.30	11.85	5.20	11.79
Average utilization of vehicles	65.6%	87.0%	49.5%	78.4%
Average waiting time of lots	44.70	100.28	32.24	61.57
Maximum waiting time of lots	249	381	244	324

Next, we summarize the experiments based on varying the vehicle fleet size. The smaller vehicle fleet size results are compiled from Experiments 2 to 13 and the larger

vehicle fleet size results are compiled from Experiments 14 to 25. Table 5 shows that increasing the number of vehicles increases the total number of moves substantially. The service ratio increases while the tool-to-tool, tool-to-stocker and stocker-to-tool ratios remain stationary. The average lot waiting times are reduced by more than 25% for the look-ahead algorithm. This reduction is almost 40% for myopic algorithm.

Table 4.6 Comparison of alternative interval lengths

	move requests/min = 1			
	Interval Length = 10		Interval Length = 5	
	Look Ahead	Myopic	Look Ahead	Myopic
Number of moves	968	844	548	432
Service ratio	86.0%	71.4%	93.0%	67.0%
Ratio of tool to tool moves	79.7%	59.4%	75.4%	60.3%
Ratio of tool to stocker moves	10.4%	21.0%	13.1%	20.8%
Ratio of stocker to tool moves	9.8%	19.6%	11.5%	18.8%
Empty travel time per move	1.65	5.83	1.63	5.87
Delivery time per move	4.54	11.76	4.83	11.88
Average utilization of vehicles	87.6%	92.3%	54.3%	96.5%
Average waiting time of lots	68	160	21	86
Maximum waiting time of lots	276	543	169	273

Table 4.6 summarizes results for alternative information availability interval lengths while keeping the move request rate the same for fair comparison. The results for the longer interval length are acquired from Experiments 8, 9, 10, 14, 15, and 16. The results for the shorter interval length are acquired from Experiments 11, 12, 13, 17, 18, and 19. Actually, since the interval length is not a concern for the myopic algorithm, the performance results do not vary from the long interval to short. The only difference comes from the length of the planning horizon which is not indicated in this table. However, the length of the decision interval is an important concern for the look-ahead

algorithm. It appears to perform better with shorter interval lengths. Shorter decision intervals reduce idling. The main reason for this is the way the look-ahead algorithm works. Since a vehicle can only make one move within a decision interval, if the decision interval is too long, then the vehicle stays unnecessarily idle. However, shortening decision interval more than enough carries the risk of making the algorithm behave myopically. Finally, the service ratio is higher and waiting times of lots are lower for the look-ahead algorithm.

Table 4.7 Comparison of alternative workloads within equal length of planning horizon

	Length of Planning Horizon = 1000			
	Number of Move Requests = 500		Number of Move Requests = 1000	
	Look Ahead	Myopic	Look Ahead	Myopic
Number of moves	554	583	968	881
Service ratio	93.1%	88.5%	86.0%	68.1%
Ratio of tool to tool moves	74.3%	59.5%	79.7%	59.4%
Ratio of tool to stocker moves	13.5%	21.1%	10.4%	21.0%
Ratio of stocker to tool moves	12.0%	19.4%	9.8%	19.6%
Empty travel time per move	1.70	5.80	1.65	5.83
Delivery time per move	5.77	11.76	4.54	11.76
Average utilization of vehicles	44.7%	67.9%	87.6%	97.6%
Average waiting time of lots	40	48	68	160
Maximum waiting time of lots	349	391	276	543

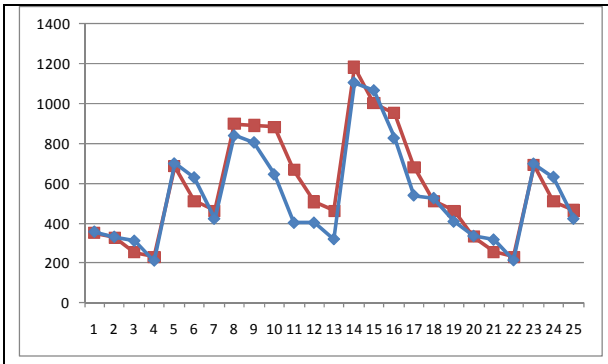
Table 4.7 summarizes results for alternative work loads within the same length of planning horizon. The lighter workload cases are Experiments 5, 6, 7, 23, 24, and 25. The heavier workload cases are Experiments 8, 9, 10, 14, 15, and 16. While the myopic algorithm is significantly affected by reduced quality of performance due to work load, the look ahead algorithm preserves a reasonable amount of performance satisfaction. The look ahead algorithm faces only a 7% decrease in the service ratio although the work load

is doubled. It seems that the AMHS becomes a serious bottleneck for the execution of the myopic algorithm by forcing almost 100% utilization of vehicles.

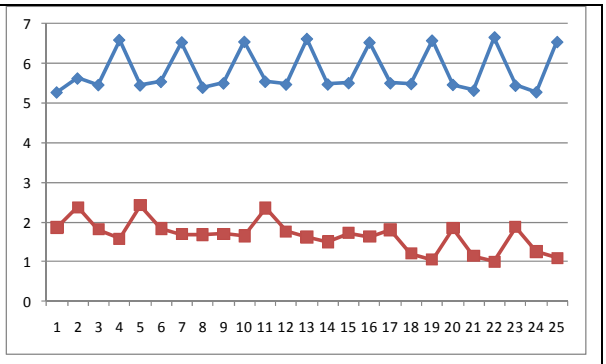
Table 4.8 Comparison of alternative workloads within varying length of planning horizon

	Move Request/min = 0.5		Move Request/min = 1	
	Look Ahead	Myopic	Look Ahead	Myopic
Number of moves	412	435	758	657
Service ratio	92.9%	88.2%	89.5%	67.5%
Ratio of tool to tool moves	75.3%	60.3%	77.5%	59.9%
Ratio of tool to stocker moves	13.2%	20.8%	11.8%	20.9%
Ratio of stocker to tool moves	11.5%	18.9%	10.7%	19.2%
Empty travel time per move	1.66	5.83	1.64	5.85
Delivery time per move	5.81	11.82	4.69	11.82
Average utilization of vehicles	44.1%	68.3%	71.0%	97.1%
Average waiting time of lots	33	39	44	123
Maximum waiting time of lots	271	297	223	408

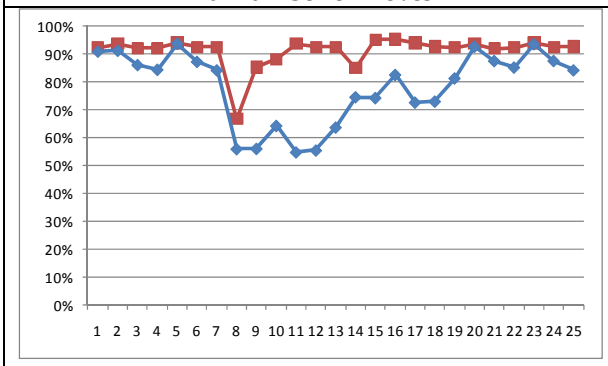
Table 4.8 presents similar cases to those presented in Table 4.7. These results are compiled by varying the planning horizon this time. The workload is represented by the rate of move requests. The lighter workload cases are Experiments 2, 3, 4, 5, 6, 7, 23, 24, and 25. The heavier workload cases are Experiments 8 to 19. The results are similar to the results in Table 4.7.



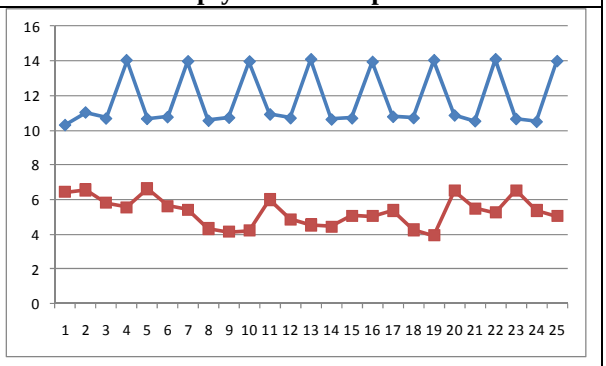
2 a number of moves



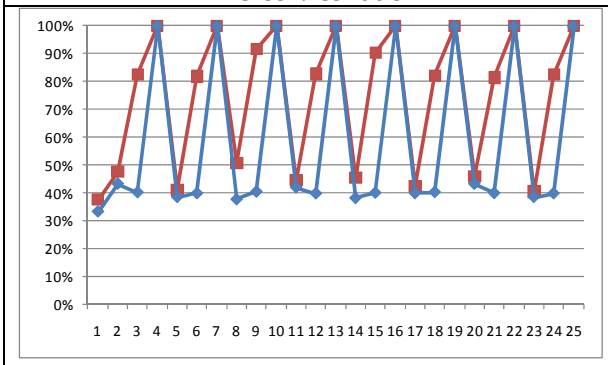
2 b empty travel time per move



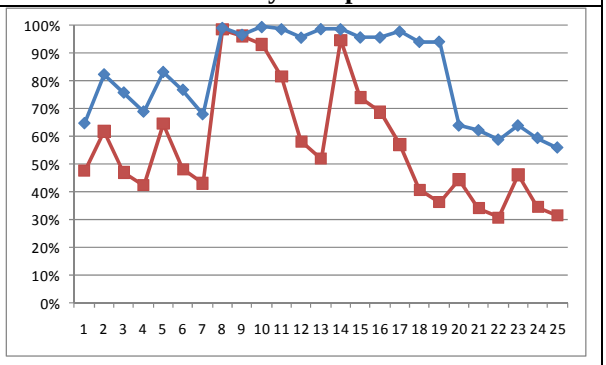
2 c service ratio



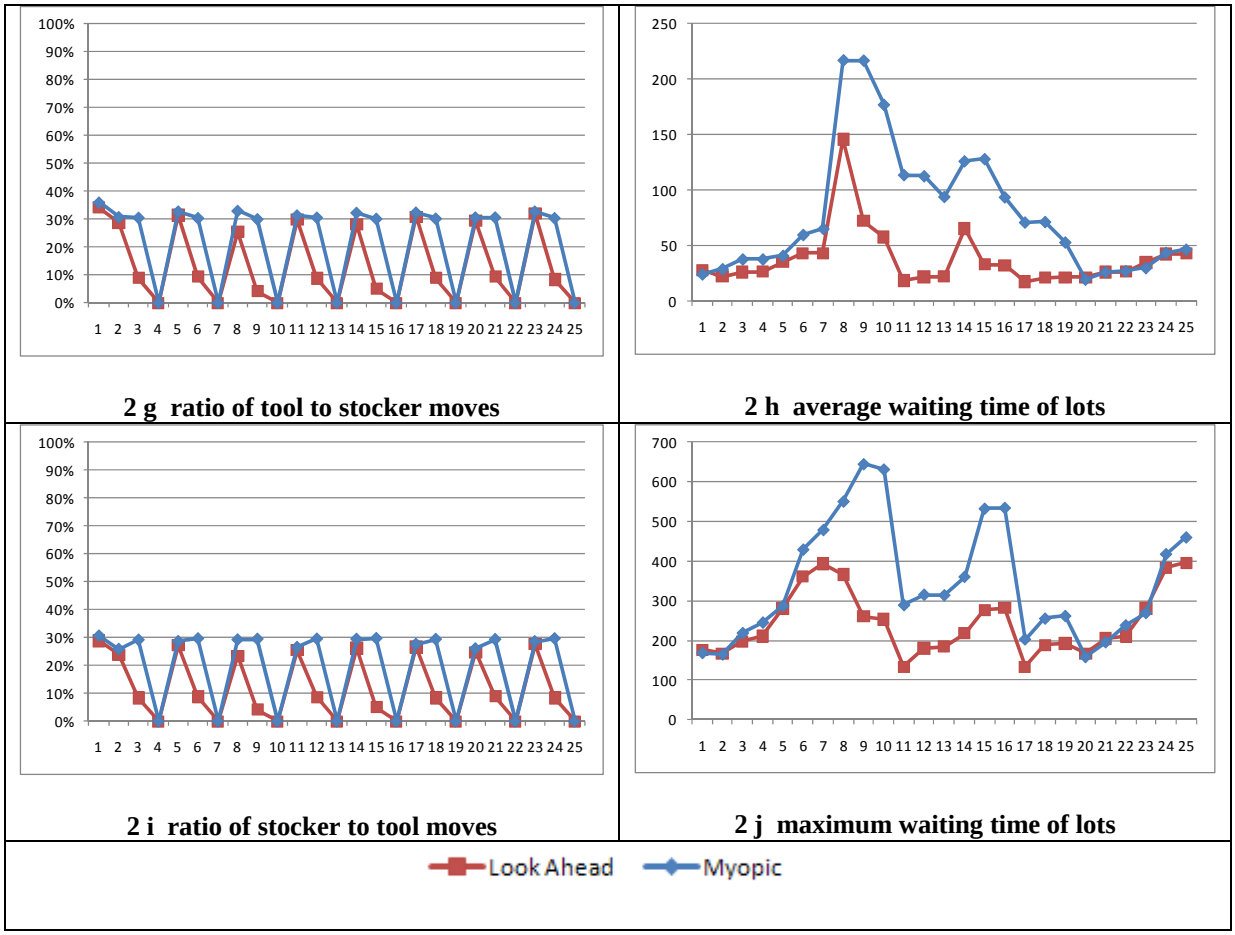
2 d delivery time per move



2 e ratio of tool to tool moves



2 f average utilization of vehicles



2 g ratio of tool to stocker moves

2 h average waiting time of lots

2 i ratio of stocker to tool moves

2 j maximum waiting time of lots

Figure 4.2 Comparison of look ahead and myopic algorithms for all experiments and performance measures

Figure 4.2 depicts a comparison of the look ahead and the myopic algorithm for all the performance measures and for all the cases we have discussed, showing the details of average performance of 5 runs at each experiment. The horizontal axes contain the experiment numbers. The vertical axes contain the corresponding performance measures. The middle range of the charts represents the heavy work load experiments. The larger vehicle fleet size experiments are located in the right side of the charts. For all cases and performance measures, the look-ahead algorithm dominantly performs better.

4.4 CONCLUSION

Continuous improvement in the semiconductor industry forces the actors of the industry and their suppliers to come up with up to date solutions for problems on different scales. Full automation in semiconductor production is common. However, nonintegrated production and material handling automation prevents decision makers in a fab from exploiting the overall potential of jointly optimizing the two systems. Due to limited information sharing between material production and material handling systems, most solutions for AMHS scheduling and dispatching end up being myopic to some degree. However, integrating the information infrastructures of production and material handling systems is an increasing trend in the industry.

In our study, we analyze the AMHS in a limited information sharing environment and a broader information sharing environment. We propose a myopic algorithm heuristic that performs in a limited information sharing environment and an optimization based look-ahead algorithm heuristic that performs in a broader information sharing environment.

The look-ahead algorithm dominates the myopic algorithm in all performance measures for every different parameter setting. This indicates the benefit of information sharing between production and material handling in a fab environment. Another important finding is that whether there is limited or abundant information sharing, using intermediate storage is not a vital necessity as long as one has advanced performance scheduling and dispatching solutions such as the ones we propose.

The next step is to test the look-ahead algorithm in a large scale fab simulation. Although recent fab layouts minimize congestion and conflict due to traffic, observing

performance of the look-ahead algorithm in a simulated high traffic environment will help us develop conflict-free sub-routines for the algorithm to make it more realistic.

Chapter 5: Conclusion and Future Research

Semiconductor manufacturing presents a rich potential source of research topics to scholars in addition to its contributions to technology and our everyday human lives. Industrial engineering and operations research scholars especially have performed invaluable studies in semiconductor manufacturing. This dissertation tries to follow in the footsteps of these scholars.

The second chapter studies a preventive maintenance scheduling problem. The third chapter investigates the lot size management in semiconductor manufacturing. The fourth chapter revisits the material handling scheduling problem from the perspective of optimization. The overall dissertation is a wide coverage of optimization practices in semiconductor manufacturing.

The traditional preventive maintenance research concentrates on the equipment and puts the production and factory dynamics into second plan. However, preventive maintenance is actually nothing less than a production unit without any output from the equipment's perspective. As regular production tasks, it occupies some capacity of the tool for a certain amount of time. Therefore, combining preventive maintenance scheduling with capacity concerns has been one of the main contributions of this dissertation. This approach helps us maintain certain capacity levels throughout the planning horizon by controlling capacity loss over time. Maintaining certain capacity levels on the tool group helps reduce overall system variance and helps decision makers plan robustly. The direct optimization of the proposed mixed integer programming model had serious computational issues. Therefore, we proposed a tool based decomposition

algorithm and a heuristic algorithm. Although the tool based decomposition algorithm computationally has advantages and disadvantages against the direct optimization, the heuristic algorithm provides very quick and quality results for larger size problems. The granularity of the solution profiles is the main cause of large relative gaps for certain cases between decomposition algorithm, heuristic algorithm and direct optimization.

Lot sizing has been one of the most discussed topics in the industry, recently. The lot sizing issue in semiconductor manufacturing is substantially different than traditional lot sizing problems. In the semiconductor industry, main issue is to determine a fixed lot size which is going to run through all the fab in a carrier. We proposed a stylized queueing model that represents production and material handling to investigate the behavior of cycle time for varying lot sizes under various environmental factors, such as technology, arrival rate, service rate and variance, analytically and numerically. The simplicity and the quality of the approximation of the queueing model helps us save enormous amount of simulation time and analyses while elaborating on lot size issues. Although lot size reduction has been one of the most popular practices to achieve cycle time gains, one of the major findings of our analysis is that smaller lot sizes do not aways mean lower cycle times. Technology is the main decision factor of what the lot size should be in a fab. In our stylized model, technology is a function of lot size which is a common perception in the industry. Our analysis shows that smaller lot sizes become a better option if there is no limit on technological advances that can be achieved in the fab. However, if there are too many restrictions on the technology improvement in a fab, then optimal lot sizes tend to be larger. Therefore, assessment of ability of technological improvements takes on a very important role in the lot sizing decision. Also, higher

variance in the system makes larger lot sizes a better option while eliminating variance helps move smaller lot sizes.

Recent efforts of automating material handling and production in the semiconductor manufacturing have overlooked the information infrastructure of the automation. However, utilizing the benefits of the automation better, integration of production and material handling information infrastructures has started to gain some pace. This technological improvement in information sharing abilities has given us the motivation to implement optimization based solutions to AMHS scheduling and dispatching problems. Therefore, we developed a look-ahead algorithm that solves lot-vehicle assignment problems periodically on a real-time basis in an information sharing environment between production and AMHS information infrastructures. The look-ahead algorithm performs dominantly better than current generic myopic approach in service ratio, delivery time, empty travel time and vehicle utilization. We also investigate the effect of usage of intermediate storages (stockers) in the fab. Regardless of the level of information sharing, eliminating stockers in a fab helps improve or preserve service ratio and vehicle utilization along with travel time and delivery time performances. Although the effect of stockers need deeper analysis, we conjecture that building new fabs with less intermediate storages has prospective benefits to the AMHS system and overall performance of the fab by the means of service ratio and cycle time.

Opportunities for future research topics are numerous. Preventive maintenance studies are expanding and recently fall more closely under the rubric of predictive preventive maintenance. Small lot size manufacturing is also attracting a lot of attention among those in the industry. Industry actors are looking for ways to shift paradigms and

move to single wafer lots along with producing larger diameter wafers. Especially, plans about moving from 300-mm diameter wafers to 450-mm diameter wafers is not only going to change equipment and product technology, it will affect fab layout, automation efforts, production and material handling systems, production methodologies along with rising new problems and their solution approaches. Advances in information sharing bring more researchers to the area of AMHS scheduling.

One of the important parts we left untouched in preventive maintenance scheduling problem is the case where capacity loss of consolidated PM tasks is calculated as a function of combination of PM tasks rather than linearly. In this approach, instead of adding up capacity losses when PM tasks are scheduled to the same time period in a tool group, they are consolidated such that they occupy les capacity than they would require if the capacity loss was calculated linearly. This is an example of consolidating PM tasks within the same chamber. While this approach can increase the overall processing time of PM tasks, it definitely provides some capacity gain compared to the case studied in this dissertation. Another important part this dissertation is missing is deciding the group of PM tasks which are going to be implemented during a planning horizon, which is assumed to be a given input in this dissertation. Making this input a decision leads to a higher level, fab-wide PM scheduling problem. Also, we need to give more special attention to tool based decomposition by investigating the cases it works efficiently in more detail to increase its performance. Advantage of very quickly solved tool problems is lost during the solution of master problem. However, there are signs that certain cases have more improvement opportunity for better tool based decomposition algorithm. Also

improving the efficiency of the heuristic algorithm with some optimization perspective to it promises an efficient modified heuristic.

Lot sizing efforts are going to continue in the industry for longer as a part of cycle time improvement efforts. While our analysis is a good fit for self-manufacturing chip producers where lot sizes are fixed and there is little among production lines, we need to improve our model for to include foundry type manufacturers for a wide range analysis. Foundries are manufacturing oriented chip suppliers that make production to order from various customers. Their production lines are diverse, orders are numerous and specifically designed for customer. Order sizes vary significantly. Therefore, we need to extend our model to include varying lot sizes for alternative product types and orders. Also, technology parameter can be broken down to see specific effects of first wafer delay, setup and throughput on lot sizing decision. These extensions have potential to bring our model from its high level discussion to a lower level analysis where scheduling of lots become important.

AMHS scheduling is getting more attention in the industry day by day. Actors of the industry are aware of the importance of information sharing between production and AMHS. Although, our look-ahead approach is a good improvement onto existing models, it requires improvement in the way it utilizes the existing information. The major issue is about the length of decision making period in which the information is used. Determining the length of decision periods is an important part of optimization efforts. Since current advances in the information sharing technology match our decision length assumptions, one time assignment of lots and vehicles within a decision period has been a good start to build a fast working optimization procedure. However, as more ahead of time

information becomes available, it will be important to determine the value of the information. The important questions are how long the decision period should be, how much difference it makes to use all the available information, after what point in time uncertainty becomes a major factor affecting our decision quality. We need to develop a modified assignment problem which allows more than one time assignment of lots and vehicles within a decision period, as the length of decision period becomes longer. Also, an algorithm which is able to track lot movements further than one step has a potential to make a positive difference for better AMHS scheduling.

Semiconductor manufacturing is likely going to continue to be a major research and application area for industrial engineering and operations research scholars. The industry is having a big transformation in manufacturing technology and business models. The need for operations research in the industry has been increasing and new study areas within the industry are opening for interested operations research scientists and experts.

References

Charles, A.-S. , 2003. Floru, I.-R., Azzaro-Pantel, C., Pibouleau L., and Domenech, S., "Optimization of Preventive Maintenance Strategies in a Multipurpose Batch Plant: Application to Semiconductor Manufacturing", Computers and Chemical Engineering, Vol. 27, pp. 449-467.

Chen, H., Harrison, J. M., Mandelbaum, A., Van Ackere, A., Wein, L. M., 1988. "Empirical Evaluation of a Queuing Network Model for Semiconductor Wafer Fabrication", Operations Research 36(2), 202-215.

Chen, F., Yan, H., Yang, J., 1998. "Production Scheduling of Continuous Flow Lines: Multiple Products with Setup Times and Costs", Production and Operations Management 7(4), 387-401.

Chih-Hong, L. and Shih-Chao, C., 2002. "A Preventive Maintenance Forecast Method for Interval Triggers", IEEE/CPMT International Electronics Manufacturing Technology Symposium, pp. 275-277.

Graham, R. L., 1969. "Bounds of Multiprocessing Timing Anomalies", SIAM Journal of Applied Mathematics, Vol. 17(2), pp. 416-429.

Gross, D., Harris, C. M., 1998. "Fundamentals of Queuing Theory: Third Edition".

Hung, Y.-F., Leachman, R. C., 1999. "Reduced Simulation Models of Wafer Fabrication Facilities", International Journal of Production Research 37(12), 2685-2701.

Hunter, J., Delp, D., Collins, D., Si, J., 2002. "Understanding a Semiconductor Process Using a Full-Scale Model", IEEE Transactions on Semiconductor Manufacturing 15(2), 285-289.

Ikeda, S., Nemoto, K., Funabashi, M., Uchino, T., Yamamoto, H., Yabuoshi, N., Sasaki, Y., Komori, K., Suzuki, N., Nishihara, S., Sasabe, S., Koike, A., 2003. "Process Integration of Single-Wafer Technology in a 300-mm Fab, Realizing Drastic Cycle Time Reduction with High Yield, and Excellent Reliability", IEEE Transactions on Semiconductor Manufacturing 16(2), 102-110.

ILOG, 2003. "ILOG CPLEX 9.0 User's Manual", ILOG.

Im, K., Kim, K., Park, T., Lee, S., 2008. "Effective Vehicle Dispatching Method Minimizing the Blocking and Delivery Times in Automatic Material Handling Systems of 300 mm Semiconductor Fabrication", International Journal of Production Research, iFirst, 1-15.

Jaber, M., Y., 2006. “Lot Sizing for an Imperfect Production Process with Quality Corrective Interruptions and Improvements, and Reduction in Setups”, Computers and Industrial Engineering 51, 781-790.

Kenyon, G., Canel, C., Neureuther, B. D., 2005. “The Impact of Lot-sizing on Net Profits and Cycle Times in the n-job m-machine Job Shop with Both Discrete and Batch Processing”, International Journal of Production Economics 97, 263-278.

Kim, B., I., Shin, J., Jeong, S., Koo, J., 2008. “Effective Overhead Hoist Transport Dispatching Based on the Hungarian Algorithm for a Large Semiconductor FAB”, International Journal of Production Research, iFirst, 1-12.

Kong, S., H., 2007. “Two-Step Simulation Method for Automatic Material Handling System of Semiconductor Fab”, Robotics and Computer Integrated Manufacturing, 23, 409-420.

Koo, P., H., Jang, J., Suh, J., 2005. “Vehicle Dispatching for Highly Loaded Semiconductor Production Considering Bottleneck Machines First”, International Journal of Flexible Manufacturing Systems, 17, 23-38.

Kumar, S., Kumar, P. R., 2001. “Queuing Network Models in the Design and Analysis of Semiconductor Wafer Fabs”, IEEE Transactions on Robotics and Automation 17(5), 548-561.

Kuo C.-H., 2002. “Modeling and Performance Evaluation of an Overhead Hoist Transport System in a 300 mm Fabrication Plant”, International Journal of Advanced Manufacturing Technology, 20, 153-161.

Kuo C.-H., Huang, C.-S., 2006. “Dispatching of Overhead Hoist Vehicles in a Fab Intrabay Using a Multimission-Oriented Controller”, International Journal of Advanced Manufacturing Technology, 27, 824-832.

Li, B., Wu, J., Carriker, W., Giddings R., 2005. “Factory Throughput Improvements through Intelligent Integrated Delivery in Semiconductor Fabrication Facilities”, IEEE Transactions on Semiconductor Manufacturing, 18(1), 222-231.

Liao, D.-Y., Wang C.-N., 2004. “Neural-Network-Based Delivery Time Estimates for Prioritized 300-mm Automatic Material Handling Operations”, IEEE Transactions on Semiconductor Manufacturing, 17(3), 324-332.

Liao D.-Y., Fu, H.-S., 2004. “Speedy Delivery: Dynamic OHT Allocation and Dispatching in Large Scale, 300-mm AMHS MAnagement”, IEEE Robotics and Automation Magazine: Speical Issue on Semiconductor Factory Automation, Part 2, 22-32.

Liao, D.-Y., Wang C.-N., 2006. "Differentiated Preemptive Dispatching for Automatic Material Handling Services in 300 mm Semiconductor Foundry", International Journal of Advanced Manufacturing Technology, 29, 890-896.

Lin, J.,T., Wang F.-K., Yen, P.-Y., 2001. "Simulation Analysis of Dispatching Rules for an Automated Interbay Material Handling System in Wafer Fab", International Journal of Production Research, 39(6), 1221-1238.

Lin, J.,T., Wang, F.-K., Wu, C.-K., 2003. "Connecting Transport AMHS in a Wafer Fab", International Journal of Production Research, 41(3), 529-544.

Lin, J.,T., Wang F.-K., Young, J.,R., 2004a. "Virtual Vehicle in the Connecting Transport Automated Material-Handling System (AMHS)", International Journal of Production Research, 42(13), 2599-2610.

Lin, J.,T., Wang F.-K., Yen, P.-Y., 2004b. "The maximum Loading and the Optimum Number of Vehicles in a Double Loop of an Interbay Material Handling System", Production Planning and Control, 15(3), 247-255.

Lopez, M. J., Wood, S. C., 1998. "Systems of Multiple Cluster Tools: Configuration and Performance under Perfect Reliability", IEEE Transactions on Semiconductor Manufacturing, 11(3), 465-474.

Lopez, M. J., Wood, S. C., 2003. "Systems of Multiple Cluster Tools: Configuration, Reliability and Performance", IEEE Transactions on Semiconductor Manufacturing, 16(2), 170-178.

Lou, S. X. C., Kager, P. W., 1989. "A Production Control Policy for VLSI Wafer Fabrication", IEEE Transactions on Semiconductor Manufacturing 2(4), 159-164.

Nazzal, D., McGinnis L., F., 2007. "Analytic Approach to Estimating AMHS Performance in 300 mm Fabs", International Journal of Production Research, 45(3), 571-590.

Peters, B., A., Yang, T., 1997. "Integrated Facility Layout and Material Handling System Design in Semiconductor Fabrication Facilities", IEEE Transactions on Semiconductor Manufacturing, 10(3), 465-474.

Pinedo, M., L., 2008. "Scheduling: Theory, Algorithm and Systems: Third Edition", Springer Science and Business Media, NY.

Ramirez-Hernandez, J. A., and Fernandez-Gaucherand, E., 2003. "An Algorithm to Convert Wafer to Calendar-Based Preventive Maintenance Schedules for Semiconductor Manufacturing Systems", Proceedings of the 42nd IEEE Conference on Decision and Control, pp. 5926-5931.

Rockwell Automation, 2007 . “Arena Contact Center: User’s Guide”, Rockwell Automation.

Rosenthal, R. E., 2007. “GAMS: A User’s Guide”, GAMS Development Corporation.

Schmidt, K., Rose O., 2007. “Development and Simulation Assessment of Semiconductor Fab Architectures for Fast Cycle Times”, SimVis PhD Colloqium.

Sivakumar, A. I., Chong, C. S., 2001. “A Simulation Based Analysis of Cycle Time Distribution, and Throughput in Semiconductor Backend Manufacturing”, International Journal of Production Research 45, 59-78.

Sloan, J. G., and Shanthikumar, T. W., 2000. “Combined Production and Maintenance Scheduling for a Multi-Product, Single-Machine Production System”, Production and Operations Management, Vol. 9(4), pp. 379-399.

Sun Microsystems, 2009. “http://java.sun.com/docs/books/tutorial/”, Sun Microsystems.

Toba, H., Izumi, H., Hatada, H., Chikushima, T., 2005. “Dynamic Load Balancing among Multiple Fabrication Lines through Estimation of Minimum Inter-Operation Time”, IEEE Transactions on Semiconductor Manufacturing, 18(1), 202-213.

Tyan, J., C., Du, T., C., Chen, J., C., Chang, I.-H., 2004. “Multiple response Optimization in a Fully Automated FAB: An Integrated Tool ad Vehicle Dispatching Strategy”, Computers and Industrial Engineering, 46, 121-139.

Wang, F.-K., Lin, J.,T., 2004. “Performance Evaluation of an Automated Material Handling System for a Wafer Fab”, Robotics and Computer Integrated Manufacturing, 20, 91-100.

Wang, C.-N., Wang, C. H., 2007. “A Simulated Model for Cycle Time Reduction by Acquiring Optimal Lot Size in Semiconductor Manufacturing”, International Journal of Advanced Manufacturing 34, 1008-1015.

Wein, L. M., 1988. “Scheduling Semiconductor Wafer Fabrication”, IEEE Transactions on Semiconductor Manufacturing 1(3), 115-130.

Whitt, W., 1983. “The Queuing Network Analyzer”, Bell System Technical Journal, Vol. 62(9), pp. 2779-2815.

Wood, S. C., 1996. “Simple Performance Models for Integrated Processing Tools”, IEEE Transactions on Semiconductor Manufacturing 9(3), 320-328.

Wood, S. C., 1997. “Cost and Cycle Time Performance of Fabs Based on Integrated Single-Wafer Processing”, IEEE Transactions on Semiconductor Manufacturing 10(1), 320-328.

Yang, T., Rajasekharan, M., Peters, B., A., 1999. “Semiconductor Fabrication Facility Design Using a Hybrid Search Methodology”, Computers and Industrial Engineering, 36, 565-583.

Yao, X., Fernandez-Gaucherand, E., Fu and S. I. Marcus, M. C., 2004. “Optimal Preventive Maintenance Scheduling in Semiconductor Manufacturing”, IEEE Transactions on Semiconductor Manufacturing, Vol. 17(3), pp. 345-356.

Yao, X., Fu, M. C., Marcus, S. I., and Fernandez-Gaucherand, E., 2001. “Optimization of Preventive Maintenance Scheduling for Semiconductor Manufacturing Systems: Models and Implementation”, Proceedings of the 2001 IEEE Conference on Control Applications, pp. 407-411.

Yao, X., Xie, X., Fu, M. C., and Marcus, S. I., 2005. “Optimal Joint Preventive Maintenance and Production Policies”, Naval Research Logistics, Vol. 52, pp. 668-681.

Vita

Emrah Zarifoglu was born in Istanbul, Turkey. After completing his high school education in Kahramanmaras Suleyman Demirel Science High School in 1997, he got his B.S. and M.S. in Industrial Engineering at Bilkent University, Ankara, Turkey in 2002 and 2005. He began his Ph.D. working on Operations Research and Industrial Engineering at the University of Texas at Austin under supervision of Dr. Erhan Kutanoglu and Dr. John Hasenbein in 2004. He is currently a member of Institute of Industrial Engineers and INFORMS.

Permanent email: zarifemrah@gmail.com

This dissertation was typed by the author.