ABSTRACT

KACAR, NECIP BARIS. Fitting Clearing Functions to Empirical Data: Simulation Optimization and Heuristic Algorithms. (Under the direction of Reha Uzsoy.)

Clearing function (CF) models, which relate the expected output of a capacitated production resource in a planning period to some measure of its workload, have shown considerable promise for modeling workload-dependent lead times in production planning. This fundamental workload-dependent lead time problem, also known as planning circularity, is due to the fact that cycle time depends on the level of resource utilization in the system, which is determined by the allocation of products to resources made by the production planning procedure. In this thesis we focus on fitting CFs from empirical data which is the most prevalent way to model complex stochastic systems. We use a simulation model of a re-entrant bottleneck system as a surrogate for a real-world semiconductor wafer fabrication environment in order to collect empirical data and compare planning models using different CFs in terms of profit realization. We consider two CF forms: product based CF and load based CF. We apply multiple linear regression (MLR) with three stepwise selection procedures to product based CFs. For load based CFs, we develop simulation optimization and heuristic algorithms to improve the initial regression fits. We implement the load based CF form in the allocated clearing function (ACF) model and compare planning models using product based CF to one using load based CF in extensive computational experiments. We base our comparison on the profit realization in simulation using non-parametric Friedman Tests. Results indicate that the MLR models including the previous period's variables in the regression perform better in the high utilization cases. Stepwise selection procedures applied to same model do not yield significantly different results. Load

based CFs perform better than Product Based CFs in terms of profit realization in simulation. Load based CFs can be further improved by using simulation optimization procedures and heuristics.

Fitting Clearing Functions to Empirical Data: Simulation Optimization and Heuristic Algorithms

by
Necip Baris Kacar

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Industrial Engineering

Raleigh, North Carolina

2012

APPROVED BY:

Dr. Reha Uzsoy Committee Chair	Dr. Brian Denton
Dr. James R. Wilson	Dr. Russell E. King

DEDICATION

To my family

who always supported and encouraged me

all my life

BIOGRAPHY

Necip Baris Kacar, was born in 1983 in Istanbul, Turkey. He graduated from American Robert High School in 1999 and received his Bachelor of Science degree in Mechanical Engineering from Bogazici University, Istanbul, Turkey in 2003. Upon graduation, he attended North Carolina State University for his Master of Science degree in Industrial Engineering and received the degree in May 2009. He continued his Ph.D study at the same university and started to work as a graduate industrial trainee in SAS Institute, Cary, NC. He was elected to the Honor Society of Phi Kappa Phi, and served as treasurer of Industrial and Systems Engineering Graduate Student Association. His research interests include simulation based optimization algorithms for production planning, capacity planning with Work-in-Process (WIP) using clearing functions and supply chain management.

Upon graduation, he will join SAS Institute in Cary, NC, where he has been working as a graduate industrial trainee for three years, as an operation research specialist.

ACKNOWLEDGMENTS

Most importantly, I would like to especially offer my gratitude to my advisor Dr. Reha Uzsoy and appreciate his great support and excellent guidance in my research. We have been working together for five years and all those years he went beyond being just an advisor but offered his help in every way he can. Along with my research, his full support in my professional career gets my deep appreciation. If I am about to get my Ph.D. and start my professional career in the industry that I wanted, the biggest portion of the credit goes to him. From the bottom of my heart, I truly thank you for giving the opportunity working with you and sharing your vast knowledge with me.

I also would like to thank my other committee members Dr. Russell E. King, Dr. Brian Denton and Dr. James R. Wilson for serving in my committee and their valuable comments regarding my thesis. Their feedback greatly improved my thesis and also helped me to learn more.

I would like to recognize and thank to the entire faculty of Department of Industrial Engineering in North Carolina State University who contributed to my education. I already use and will be using what I learnt in this great department. I always felt lucky to learn from the best professors that made me knowledgeable on variety of topics.

I also would like to recognize Hakan Sungur who works in the Department of Industrial Engineering, for always being a great friend and helping me out with all of my bureaucratic work. The staff in our department are very nice people, thank you very much to all of you being friendly and helpful to me. Big thanks to all my friends in the department making this Ph.D study much more enjoyable. I will miss our chats in the department.

I also would like to thank my roommate Aydin Beseli who is also my close high school friend, for making the Ph.D. life more fun and interesting.

I also would like to thank Hui Wang for being nice to me and sharing the amazing trip memories with me across the USA.

I also would like to thank SAS Institute for supporting my Ph.D. education and giving me the opportunity to learn their software used in this thesis, and also providing the very valuable industrial experience.

Lastly, I would like to thank my family. I felt their support in every part of my life. The feeling that you will make them happy and proud by achieving something in your life was very inspiring and also a good motivation for me to work hard to write this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Production planning involves determining the timing and quantity of raw material releases to the plant such that they will emerge as finished products to satisfy customer demand on time. An executable and practical production plan must be feasible with respect to the constraints and business rules of the production facility while optimizing appropriate performance metrics.

Research on production planning has a long history going back to the seminal work of Modigliani and Hohn (1955), Holt et al. (1956) and Holt et al. (1960). A critical issue in these models is how to represent the relationship between the pattern of work releases into the system over time and the resulting output. Understanding this relationship lies at the heart of any production planning model, since we wish to manage the release of raw material into the system in order to coordinate supply and demand in an optimal manner. A critical component of this relationship is the cycle time, the time between work being released into the system and its emergence as finished product. We will use lead time to refer to the estimate of the cycle time used in planning models.

In order to control the timing of releases to the plant so that the finished product will come out on time, an estimate of cycle time is required. However, an inherent problem in estimating the cycle time is that it depends on the level of resource utilization in the system, which, in turn, is determined by the allocation of products to resources made by the production planning procedure itself. This problem of workload–dependent lead times in production planning is known as the planning circularity problem (Asmundsson et al. 2009), and has been studied by many authors (Pahl et al. 2005).

The modeling of the relationship between resource utilization and cycle time has constituted a major challenge for the production planning field since its inception. The problem is particularly difficult when capital-intensive plants must operate in a highly loaded environment where they face the very complex problem of allocating resources to products over time. It is widely recognized that the behavior of most capacitated production resources is governed by queuing, as work arrives randomly over time at resources whose processing capabilities are subject to varying degrees of uncertainty. Queuing models (Hopp and Spearman 2001) have shown a nonlinear relationship between resource utilization and cycle time. This relationship makes the production planning task difficult, since a planning model requires knowledge of cycle times in order to plan the production, while cycle times depend on utilization which is, in turn, determined by the planning procedure itself. The relationship between cycle times and utilization is nonlinear, and its effects are seen well before utilization reaches one.

Most existing production planning models can be grouped under two main headings. The first class of models considers lead time estimates as exogenous parameters independent of resource utilization. Thus, the circularity problem is not considered in these models. This approach is taken in the widely used Material Requirements Planning (MRP) procedure (Vollmann et al. 2005) and most linear and integer programming models (Johnson and Montgomery 1974), (Voss and Woodruff 2003). In MRP, the backward scheduling algorithm offsets the release of orders backwards in time from the time that it is required to emerge as a finished product to satisfy the demand by the value of the lead time. Since these lead times are not updated to reflect changes in resource utilization, this approach may lead to poor results, especially for production systems whose resource loading varies over time and that operate at high utilization levels. In MRP, there is also an infinite capacity assumption that may yield infeasible schedules. A number of authors such as Billington et al. (1983) have formulated the MRP problem as a mixed integer program, again using exogenous lead times.

Another class of exogenous lead time models are planned lead time models (Spitter et al. 2005) where the LP formulation enables the processing of orders in multiple periods during the lead time.

The second class of models uses either detailed scheduling algorithms or a simulation model integrated with mathematical programming in an iterative scheme where the simulation model is to confirm that the release schedule suggested by the plan is realizable in the factory and meets demand. Dauzère-Péres and Lasserre (1994) propose a model to find a locally optimal feasible production plan.

Iterative algorithms combining LP and simulation models have been proposed by several authors, including Kim and Kim (2001); Byrne and Bakir (1999); Byrne and Hossain (2005); and Hung and Leachman (1996). Hung and Hou (2001) replace the simulation model with queuing theory for lead time estimates in order to overcome time consuming simulation runs. Essentially, these algorithms use estimates of cycle times, corresponding to a given resource loading over time and hence a work release schedule, and pass these estimates to an LP model as an input. The LP model in turn, proposes a release schedule based on these cycle time estimates that form the input to the simulation model, which is run again to estimate the new cycle times. Iteration continues until some specified convergence criterion is satisfied. Bang and Kim (2010) develop an iterative approach similar to that of Hung and Leachman (1996), differing in the method of updating the parameters in the LP model after a simulation run. They also use the observations of inventory and throughput levels from the simulation for regrouping products into families to be used in the LP along with lead time estimates.

These iterative simulation-LP approaches can capture the nonlinear relationship between the lead times and resource utilization, but applying this approach to large scale problems may not be feasible due to the time consuming simulation runs to obtain required data. In addition, the convergence behavior of these models is unclear and their computation time requirements may be excessive due to the high computational burden of the detailed simulation models. Moreover, for the models of Hung and Leachman (1996), and Kim and Kim (2001), computational experiments have shown that assessment of their convergence can be difficult (Irdem et al. 2008, 2010) .

Considering previous production planning models, either the nonlinear relationship between work-in-process and output is not well captured as in the first class of the models based on exogenous lead times, or using a simulation model to capture this effect becomes cumbersome and intractable, together with the difficulties in ensuring the convergence of the iterative procedure. Therefore, the effect of capacity loading on lead times is not well incorporated into these models.

Traditional capacity planning models ignore WIP and only consider finished goods inventory. Studies by Ettl et al. (2000) and Liu et al. (2004) for supply chain networks that combine queuing analysis with inventory models to calculate expected WIP and explicitly consider WIP cost in the objective function. An integer programming model by Woodruff and Voss (2004) also includes WIP cost in the objective function and attempts to discretize lead time estimates into brackets depending on resource utilization to model load dependent lead time; however computation burden is intimidating for large scale complex problems. Kekre (1984) also considers WIP explicitly as a function of lead time in his formulation and WIP cost in his objective function. He examines the effects of the lot sizing decision on WIP and lead time.

Given these shortcomings of previous approaches that fail to incorporate the effect of capacity loading on cycle times into models, the idea of clearing functions (CFs) that represent the expected output of a resource in a planning period as a function of the some measure of the work in process inventory (WIP) has been proposed to capture the effects of load dependent lead times. These models have been studied by a number of authors, including Graves (1986); Karmarkar (1989) and Srinivasan et al. (1988). Asmundsson et al. (2006) and Asmundsson et al. (2009) have shown promising results in several studies (Kacar et al. 2012). The advantage of these models is that they can capture the nonlinear relationship between resource utilization and cycle times without requiring time-consuming simulation runs as part of the planning process. It has been shown by several authors (Missbauer 2002; Asmundsson et al. 2006; Selçuk et al. 2007 and Asmundsson et al. 2009) that when the CF is parameterized correctly, these models can yield superior performance over conventional LP models. However, the use of this approach requires effective methods of estimating the clearing functions themselves. Missbauer (2007) shows that the clearing function depends on the work in process at the beginning of the each period, which presents the idea of state-based clearing functions. Selçuk et al. (2007) have developed clearing functions based on the combination of the Pollaczek–Khinchine mean value formula for $M|G|1$ systems and Little's Law assuming arrivals to a station are directly controlled. There are also studies by Agnew (1976) and Spearman (1991) that develop clearing functions from queuing theory. Previous work (Asmundsson et al. 2009) has shown that the estimation of clearing functions from empirical data, especially with the emergence of non-stationary effects on the system, is a nontrivial exercise for which a firm theoretical foundation still remains to be established.

Contributions of this Thesis

In this thesis, we focus on clearing function models and the estimation of CFs from empirical data. We assume that our demand pattern is deterministic. We use a simulation model as a surrogate for a production system to generate data from which clearing functions can be estimated. The randomness in the system is associated with processing times and machine failures. There are several CF forms suggested by Graves (1986), Karmarkar (1989), and Srinivasan et al. (1988); but we focus on Load Based CFs, where output is modeled as a function of the work in process (WIP) available at the beginning of the period and releases in that period, i.e., R_t+W_{t-1}, which we refer to as load of that period (Missbauer 2002). We also examine CFs where WIP and releases are treated as separate independent variables in regression models where output is the dependent variable.

In this thesis, we aim to improve the estimation of clearing functions using a simulation optimization approach. Generally this approach focuses on highly complex, large scale stochastic systems that are difficult to solve analytically or model using stochastic programming. In order to overcome this difficulty, simulation optimization is used where the set of optimal decisions is searched using discrete event simulation models which can include stochastic elements and complexities of a system. (Fu, 1994) provides extensive review of this area. Simulation optimization methods can be considered under two main headings based on whether the parameter state space is discrete or continuous. Different solution procedures are proposed depending on the nature of the parameter state space. For discrete finite state space parameter problems, ranking and selection and multiple comparison approaches are used which basically utilize the statistical analysis of simulation runs. For continuous state space parameter problems, which will be our focus, proposed approaches include response surface and stochastic approximation techniques. Our simulation optimization approach will use the idea of gradient estimation under stochastic approximation techniques, which also will be explained in CHAPTER 6.

Overall, the scope and the goals of this thesis are the following;

- Explore the use of several different linear regression models for estimating clearing functions and compare the performance of the production plans obtained using these models.

- Obtain insight into the structure of the clearing functions and the degree to which multiple regression can be used.

- Develop a simulation optimization algorithm to identify better performing CFs and develop heuristics to obtain good solutions in modest CPU times if possible.

- Compare different functional forms of the CFs in terms of the quality of the resulting production plans.

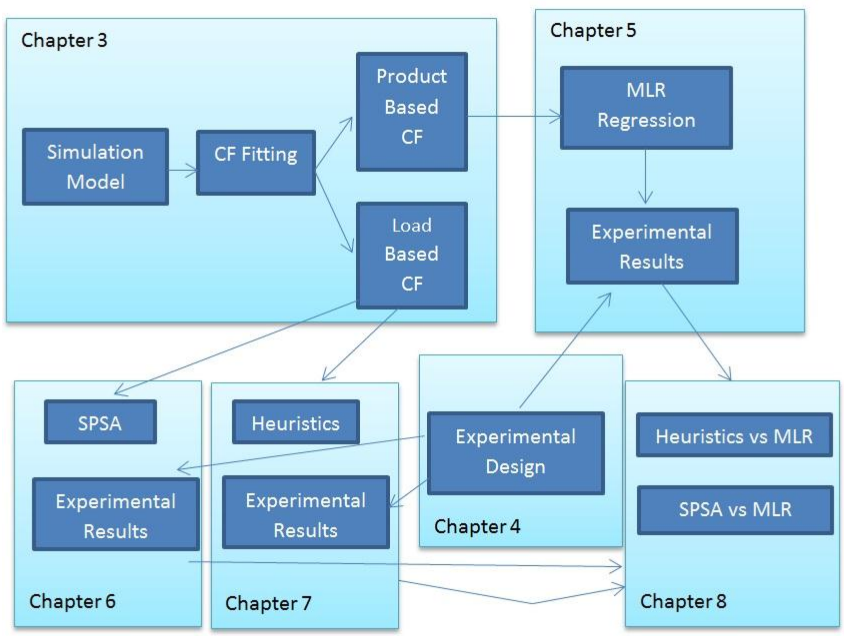


Figure 1-1: Outline of Thesis

The Figure 1-1 illustrates the structure of the thesis. We present our literature review in Chapter 2. In Chapter 3, we will introduce our simulation model and the procedure used to obtain product based and load based CFs. In Chapter 4, we will explain our experimental design used to evaluate the planning models and the forms of

CF fits. In Chapter 5, we will present multiple linear regression models using product based CF. In Chapter 6, we will introduce simulation optimization and in Chapter 7 heuristic algorithms to improve load based CF fits. In Chapter 8, we will compare our algorithms using load based and product based CFs. We will briefly give summary and key findings of this thesis, and end with future directions in Chapter 9.

CHAPTER 2. LITERATURE REVIEW

There are numerous works in the literature presenting mathematical programming models for production planning aimed at allocating capacity to multiple products over time while satisfying demand and optimizing some performance criterion. These algorithms include methods that consider lead time as an exogenous parameter, and iterative methods that combine fixed lead time with simulation. The models Johnson and Montgomery (1974), Voss and Woodruff (2003) focus on stochastic demands are out of scope for this thesis. Recently alternative methods such as clearing function models have been proposed, which form the main focus of this thesis. A review of these different methods is given below, starting with the well-known Material Requirements Planning approach.

The well-known and widely used Material Requirements Planning (MRP) procedure discussed by Orlicky (1975) uses fixed lead time estimates that are independent of resource utilization. A backward scheduling algorithm is used to determine the timing of releases backwards from the due date of the product, using the value of the lead time. The applicability and reliability of this model depend on the accuracy of the lead time estimates, which become more unreliable in highly loaded production environments. In addition to use of fixed lead time in MRP, there is also an infinite capacity assumption that may yield in infeasible schedules, since it assumes that fixed lead times can be maintained regardless of workload, even if the workload exceeds capacity. This infinite capacity assumption is addressed by the Capacitated MRP (MRP-C) procedure developed by Tardif and Spearman (1997) which uses the relation between WIP and cycle time, checks for WIP and capacity infeasibilities and suggests remedies such as delaying due dates or adding capacity. This model assumes that the lead time can be maintained as long as utilization is less than 1, i.e., the resource capacity is not exceeded. Vollmann et al. (2005) evaluates capacity by calculating total hours of man force needed for a given master production scheduling (MPS) using historical data on resource allocation. This Rough Cut Capacity Planning is a straightforward calculation to check for any obvious capacity violations. On the other hand, Capacity Requirements Planning procedure uses more detailed information on individual product routings and operation lead times but is still independent of workload. Billington et al. (1983) also consider capacity explicitly in their mixed integer programming model, including binary variables for setups. They also suggest using minimum lead times in the formulation to represent transfer times, not waiting times. They state that capacity and material flow constraints will implicitly include waiting times in lead time by starting the releases in earlier periods if the capacity constraint is tight thus products are passed through inventory.

Most LP models for production planning Johnson and Montgomery (1974); Hackman and Leachman (1989) also use lead time estimates as an exogenous parameter, and the accuracy of these models, especially at high utilization levels, become questionable. Hackman and Leachman (1989) extends the LP model to incorporate non integer lead times for both lag before and lag after the activity and uneven period lengths, but lead times are still independent of workload. Spitter et al. (2005) proposes the concept of planned lead times for supply chain operations planning where their LP formulation enables the completion of orders before due date while the order can be processed any time between release and due date. This model has the flexibility that it can be processed over multiple periods compared to LP models that assume orders released at the beginning of the period, has to be processed in the same period.

Ettl et al. (2000) and Liu et al. (2004) develop models of supply chain networks that calculates expected WIP from using queuing analysis. They also put WIP cost in their objective function to find the base stock levels that minimize the cost of meeting fill rate requirements. Kekre (1984) analyzes WIP depending on lot sizes and comes up with an equation for WIP as a function of lot size. He states that WIP in front of a resource depends on not individual products separately but combination of all products WIP together. He also formulates WIP explicitly in his constraints and includes WIP cost in the objective function.

Woodruff and Voss (2004) focus on workload dependent lead times and piecewise linearize the nonlinear relationship between resource utilization and lead times with the aid of binary variables that lead to mixed integer programming models that are computationally burdensome and impracticality in highly complex systems.

They also include a constraint to prevent a recently released order passing another that was released earlier. This constraint also allows the model to avoid drastic changes in releases from one period to another, thus reducing the variability in the arrivals to stations.

To address the dependence between workload and lead times, a number of authors have proposed iterative algorithms that combine LP models using fixed lead times with a simulation model. Hung and Leachman (1996) propose an iterative algorithm that estimates lead times corresponding to a given work release schedule from simulation and passes these estimates to an LP model as an input. The LP model in turn, determines a release schedule that forms the input to the simulation model, which is run again to estimate the new lead times. Iteration continues until some specified convergence criterion is satisfied. Kim and Kim (2001) propose a similar approach where loading ratios are used as an estimate of lead times. Loading ratios basically refer to fraction of the releases emerged as finished goods distributed to periods. They extend the idea of Byrne and Bakir (1999) who modify only the right hand side of the capacity constraint by multiplying the nominal capacity by the observed utilization from simulation. Bang and Kim (2010) propose an iterative scheme similar to Hung and Leachman where they also use discrepancies in waiting times, WIP levels and throughput at the bottleneck between the LP solution and the simulation output to update parameters of LP at an iteration. They also stop the simulation run if any discrepancies become excessive. The convergence behavior of these methods is ambiguous and still not well understood Irdem et al. (2008), Irdem et al. (2010). Due to the high computational burden of the simulation runs required to obtain cycle time estimates, Hung and Hou (2001) replace the simulation model with cycle time prediction models based on queuing theory and an empirical method, where they use simulation to collect data in order to develop relationship between machine utilization and cycle time.

Other authors have used detailed scheduling approaches in order to model workload dependent lead times, such as Dauzère-Péres and Lasserre (1994) who propose an iterative alternating lot sizing and scheduling problems solving procedure to find the best possible feasible plan. They show improved performance with their integrated approach based on experiments with several scheduling policies. Zijm and Buitenhek (1996) apply a detailed scheduling algorithm for job shops considering WIP in the shop and using a modified version of the shifting bottleneck method to assign due dates to orders depending on the planned lead times.

Riaño (2003) models the lead times as independent random variables subject to a probability distribution. In this model, expected cumulative output up to period t is equal to cumulative releases up to that period t multiplied by the corresponding probability distribution of the lead time. The requirement of this model is to satisfy the orders with a specified probability. However this approach assumes lead time distributions independent of product types that can be problematic when products share common resources and lead time distributions are not updated depending on the release plan from LP.

The models described above either assume utilization independent lead times, or requires of cumbersome simulation models to validate the feasibility of the plans proposed by the LP models. The necessity of effective and efficient modeling of workload dependent lead time has opened new directions in the literature. In order to overcome these shortcomings of previous production planning models, clearing functions provide a mechanism to relate the expected output X_t of a production resource in a planning period t to the expected work in process (WIP) level W_t over that period. The Clearing Function is similar in nature to the operating curve approach Schoemig (1999) that describes the relation between mean cycle time and throughput, which was developed and used to predict the performance of the manufacturing line. They also show the degrading effect of variability caused by machine down times.

Several examples of clearing functions in the literature are depicted in Figure 2-1. The "Constant level" function places a fixed upper bound on production. It does not have any lead time constraint and assumes instantaneous production no matter what the WIP level is. Graves (1986) proposes a clearing function in the form of $X_t = \alpha W_t$, where output X_t at time t is considered a linear function of WIP. This "constant proportion" function assumes a fixed lead time of $1/\alpha$ can be maintained at all utilization levels. In this model, it is assumed that production facility will be operated in the range that this fixed lead time assumption will hold. This function may yield infeasible levels of output at high WIP levels, and so needs to be limited by a fixed capacity which is shown as the "combined" clearing function. Karmarkar (1989) proposes a non-linear clearing function where output increases as a concave non-decreasing function of W_t, reaching an asymptotic maximum. Srinivasan et al. (1988) propose another clearing function which is a concave, non-decreasing function of WIP. Figure 2-1 shows these types of functions as the "effective" clearing function.

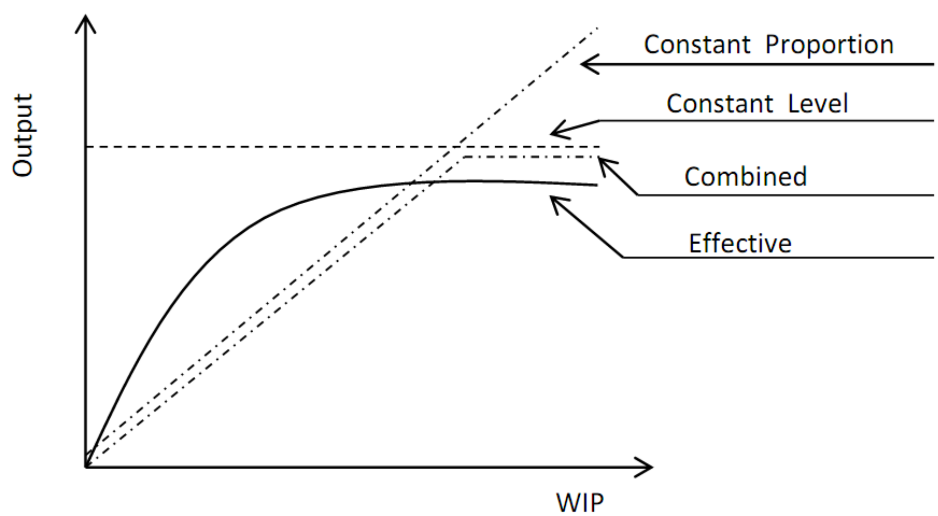


Figure 2-1: Examples of Clearing Functions Karmarkar (1989)

Missbauer (2002) discusses the limitations of clearing function models such as the fact that they limit the output by a function of the expected total load and the distribution of the arrival period of the work expected to contribute to the load in period *t* is not considered. He proposes an aggregate order release planning model that determines the amount of work released in each planning period and can handle different load patterns without requiring additional load balancing parameters. Early clearing function models had difficulty in modeling the behavior of multiple products since when products compete for capacity, then one product may end up waiting indefinitely while other products are processed in very short lead times. In order to solve this problem, Asmundsson et al. (2009) propose an allocated clearing function formulation for multiple products where capacity is allocated to individual products. This model forms the basis for our work in this thesis, and will be described in more detail in CHAPTER 6.

Clearing functions can be derived analytically using steady-state or transient queuing models, or estimated empirically from empirical data. Different authors have implemented somewhat different approaches. Agnew (1976) proposes a throughput function where service rate is a function of the number in queue and suggests using it in optimal control policy context. Spearman (1991) derives a clearing function using closed queuing networks, conjecturing a relation between mean cycle time and WIP, and taking one observation from simulation to specify congestion of the system. Asmundsson et al. (2009) formulate the clearing function as a relationship between the expected throughput of a resource in a planning period as a function of the time-average WIP level at the resource during the period from empirical data. Other authors, such as Karmarkar (1989) and Missbauer (2002) assume the clearing function depends on the expected workload, which they define as the sum of the work in process available at the start of the period and the material released during the period. Zäpfel and Missbauer (1993a) use a simulation model to estimate Clearing Functions based on the expected workload, and observe discrepancies between planned and actual WIP in simulation. Missbauer (2009) shows that the clearing function depends on the work in process at the beginning of the each period due to transition behavior of the system within the planning period and corrects the Clearing Function to minimize the discrepancy between transient and non-transient Clearing Function. Selçuk et al. (2007) have derived transient clearing functions analytically using Pollaczek–Khinchine mean value formula and Little's Law.

In this thesis, we focus on empirical estimation of Clearing Functions from simulation data, which has been the most prevalent approach in the literature especially for complex production systems. In this approach, the production system under study is simulated and data collected from the simulation model in each planning period on the quantities of interest, which depend on the independent variables postulated for the clearing function. Once the data have been obtained, they are fit empirically using some form of regression analysis to a predefined functional form.

In the literature there are two typical functional forms that are suggested to estimate the clearing functions. Karmarkar (1989) proposes the form $f(W) = \frac{K_1 . W}{K_2 + W}$; Srinivasan et al. (1988) suggests using $f(W) = K_1\,(1 - e^{-K_2 . W})$. In both functions,

K_1 represents the maximum possible output in a period and K_2 defines the curvature of the clearing function. These two forms are types of "effective" clearing functions as shown in Figure 2-1. Asmundsson et al. (2009) use the CF form suggested by Srinivasan et al. (1988) to fit CF from simulation data and observe the non-stationary behavior discussed by Missbauer (2009). They modify the CF by fitting a curve that lies below a certain percentage of the data points to minimize the overestimation.

In our research, we consider two different measures of workload. We first consider the WIP $W_{k,t-1}$ and Release R_{kt} of resource k in period t separately, yielding $X_{kt} = f_k\ (W_{k,t-1}, R_{kt})$ and relate them to output X_{kt} using multiple linear regression. Next, we consider W in terms of the sum of the releases within a period and initial WIP levels $X_{kt} = f_k\ (W_{k,t-1} + R_{kt})$, instead of the time averaged WIP that has been studied in previous research.

In our research, we take a different approach from the literature by estimating a clearing function for each individual product based on the release quantities and WIP levels of this product and those of other products in the current and immediately preceding periods. The values of the state variables of interest are regressed against the output of each product. In this way we hope to obtain insight into which variables are important to include in the clearing functions and which can safely be omitted. In the second part, we introduce load based fitting which considers the sum of releases in the period and work in process at the beginning of that period. We regress load against output, and suggest a simulation optimization approach and search heuristic algorithms to improve our fit. This will essentially be an extension of the Asmundsson et al. (2009) approach of using a percentile level of the data to fit the CF, where the parameters are determined by simulation optimization.

In this thesis, both our simulation optimization approach and search heuristics for improved Clearing Functions will focus on the load based form where output is a function of the sum of planned input in period *t* and WIP available at the beginning of period *t* for machine *k* given by

$$X_{kt} = f_k \left(W_{k,t-1} + R_{kt}\right) \tag{2.1}$$

We develop CF improvement algorithms that are based on an optimization via simulation approach. (Fu, 1994) has an extensive review paper on this area, discussing several methods. This area has attracted attention due to increased efforts in optimization of stochastic discrete event systems where simulation is used to estimate the performance of system to overcome the computational and inherent complexities of the system. Simulation optimization is an approach in which the space of decision variables is searched to obtain the best performance estimate from simulation. The optimization problem is considered under two categories: discreet and continuous state space. Here, the state space is the collection of the parameters or decision variables in the optimization problem that are subject to change in order to achieve better performance estimates in simulation. The classification and solution methodologies are given in Figure 2-2. The general problem setting of these problems is,

$$min_{\theta \in \Theta} J(\theta),$$

where $J(\theta) = E[L(\theta, \omega)]$ is the performance measure of interest; $L(\theta, \omega)$ the sample performance, with ω denoting the vector of simulation replications and θ the vector of controllable parameters or decision variables. Θ is the set of all possible selections of θ. The optimum value of $min_{\theta \in \Theta} J(\theta)$ can be defined as finding θ^* that will yield $arg\ min_{\theta \in \Theta} J(\theta)$. In these problems, our controllable parameter set θ can be discrete or continuous.

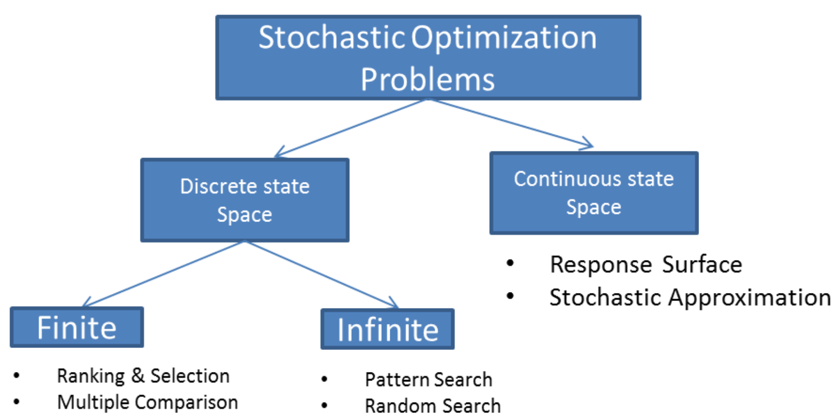


Figure 2-2: Classification of Optimization Problems and Solution Methodologies

We now present solution methodologies for discrete state space problems where the nature of the parameter state space is discrete. Depending on the problem, this state space can be finite or infinite, whereas often the number of choices in the parameter set is finite. Even though the parameter set is infinite, reduction or approximation techniques can be used to reduce the problem state space to a finite set. Under this category, there are two major solution methodologies; Ranking and Selection and Multiple Comparisons. For the Ranking and Selection, in general there are two approaches in literature: Indifference Zone and Subset Selection. The former approach uses a specified probability value p that, with the selected vector of parameters $\lambda_i \in \theta$, $P\{J(\lambda_i) - J(\theta^*) < \delta\} > p$ where δ denotes the so called " indifference zone" i.e., with the selected vector λ_i, $J(\lambda_i)$ is within δ of the optimal value $J(\theta^*)$ with probability p. In the Subset Selection approach, the aim is to choose a subset of at most m $\lambda_i's$ such that one of the λ_i selected will guarantee the probability relation stated above. These two approaches can be used to complement each other in a problem with a large set of possible selections such that Subset Selection can be used to reduce the number of sets and then the Indifference Zone approach can be used to find the λ_i among the reduced set that will be different from the optimum within a specified indifference zone δ. For the other methodology used for discrete finite space problems, Multiple Comparisons, the basic idea is to do all pairwise comparisons of performance of all $\lambda_i \in \theta$ and interpret the difference in terms of confidence intervals to choose the best performing parameter vector λ_i. The comparisons tests may contain paired t tests or multiple comparisons tests such as using Bonferroni inequality to account for multiple comparisons applied simultaneously by reducing the alpha value.

We now discuss continuous state space problems where the parameters are continuous. We start with Response Surface Methodology, where a polynomial function of appropriate degree is fit to the response (performance measure of interest) of the system. There are two phases. In the first phase, a least-squares regression is applied to the initial experimental runs and a new subregion is chosen to be explored through the equation $\theta_{n+1} = \theta_n + a_n \nabla J_n$ where θ_n is *n*th explored subregion , ∇J_n is the estimate of the gradient from the fitted linear response obtained in the first phase and a_n is the step size. This is repeated until gradient gets close to zero that no more regions to be explored are obtained. In the second phase, a quadratic response surface is fitted to the response using more detailed experimental runs and an optimum is obtained analytically from this fit. There is a shift of focus in this methodology in that the main interest becomes finding an improved response over the current conditions than finding the optimum one.

Next, we present the Stochastic Approximation method for continuous state space that our simulation optimization approach will use for estimating gradients. This algorithm begins with best guesses of optimal parameter values which are updated iteratively based on the estimate of the gradient of the performance metric with respect to the parameter. The Response Surface methodology also implements a gradient based algorithm whereas the gradient estimation is obtained from regression model. The basic idea in stochastic approximation is to find the values of θ yielding

$\nabla J(\theta) = 0$ in the original problem setting; however there are problems of local optimality or slow rate of convergence.

The general form of the Stochastic Approximation algorithm is as follows:

Compute $\theta_{n+1} = \Pi_{\Theta}(\,\theta_n + a_n \nabla J_n)$ where θ_n is the parameter set at the *n*'th iteration, a_n is the step size, ∇J_n is the estimate of $\nabla J(\theta)$ and Π_{Θ} is a projection onto Θ which if θ_{n+1} goes out of feasible region, Π_{Θ} returns θ_{n+1} to the feasible region; one such is assigning θ_{n+1} to the previous iteration vector θ_n . In this setting, some of the main questions are, what should the step size a_n be and how ∇J_n should be calculated. For step size, in practice, decreasing the step size when the gradient changes direction has been shown to perform better compared to harmonic series of step sizes $a_n/_n$ that in practice shows slower convergence. For gradient estimation, in general there are four techniques: finite differences, likelihood ratio, perturbation analysis and frequency domain experimentation. The way to estimate the gradient with the finite difference technique is to run multiple simulations to obtain an approximation of the gradient. One version of finite differences is

$\widehat{\nabla}\, J_{n=} [\widehat{\nabla_1}\, J_n \;\ldots.\, \widehat{\nabla_p}\, J_n\,]^T$ where *p* denotes the number of elements in the parameter set.

$\widehat{\nabla_i}\, J_n = \frac{\hat{J}(\,\theta_n + C_i e_i) - \hat{J}(\,\theta_n - C_i e_i)}{2C_i}$ where e_i is the unit vector

c_i's are the difference parameters whose values represent a tradeoff between too much noise (using small values) and too much bias (using large values). In order to have convergence the c_i's should approach zero. This gradient estimation technique is most generally applicable but shows slower convergence due to the computational intensiveness of the technique. In this symmetric finite difference technique, $2p$ simulation runs are required for gradient estimation. In order to overcome the potentially long CPU time caused by many simulation runs, Spall (1998) suggests the Simultaneous Perturbation Stochastic Approximation (SPSA) technique which requires two simulation runs for estimating the gradient. The gradient estimation takes the form

$$\hat{\nabla} J_{in} = \frac{\hat{J}\,(\,\theta_n + c_n\Delta_n) - \hat{J}\,(\,\theta_n - c_n\Delta_n)}{2c_n\Delta_{in}} \quad i = 1, \dots, p$$

$$\Delta_{n=}[\widehat{\Delta_n^1} \;\dots .\, \widehat{\Delta_n^p}]^T$$

Here Δ_n is a p-dimensional random perturbation vector such that each component of Δ_n is independently generated from a zero mean distribution. A simple choice of this type would be ±1 Bernoulli distribution with probability of 0.5 for each outcome ±1. The gradient estimation $\hat{\nabla} J_n$ reflects the simultaneous perturbations of all components of Δ_n in contrast to component by component perturbations in finite difference gradient estimation. Updating θ_n is same as in other stochastic approximation techniques that $\theta_{n+1} = (\,\theta_n + a_n \nabla J_n)$. The gain sequences a_n and c_n

are obtained by choosing nonnegative coefficients for a, c, A. α and γ and then at each iteration a_n and c_n are calculated with the equations a_n=$a/(A+n)^{\alpha}$ and $c_n = c/n^{\gamma}$,respectively. Iteration is terminated if there is little change in several successive iterates or a specified maximum number of iterations is reached.

Perturbation Analysis estimates gradient by applying a what if analysis, computing what would happen if θ, which is the nominal case, had been $\theta + \nabla\theta$ where $\nabla\theta$ is infinitesimally small. One of the main assumptions in this technique is that this small change in parameter will not alter the order in which the events happen in the system. Basically this technique takes the partial derivatives of the performance metric of system with respect to the parameters and estimates the gradient in this way.

The Likelihood Ratio technique uses the idea of differentiating the underlying probability of the system and requires the differentiation property of the probability measure. Expectation of performance measure is expressed as,

$E[L(X)] = \int L(x)dF(\ \theta, X)$ where $F(\theta, X)$ is the cumulative distribution function of random vector X depending on parameter vector θ. Differentiating the previous equation yields, $\frac{\partial E[L(X)]}{\partial\theta} = E[L(x)\frac{\partial \ln(f(\theta,x)}{\partial\theta}]$ so in a single simulation run, estimates of the derivative of performance measure and the performance measure itself can be obtained. The last technique that we present for gradient estimate is Frequency Domain experimentation which uses the idea of oscillating the value of parameters according to sinusoidal function during simulation. The parameter vector is stated as

$$\theta(t) = \theta_0 + \alpha \sin(wt)$$

where α is vector of oscillation amplitudes and *w* the vector of oscillation frequencies. The gradient estimation problem becomes that of estimating the gradient at $\theta(0) = \theta_0$ which is $\nabla J(\theta_0)$. It is achieved by approximating J around θ_0 using a second order Taylor expansion and taking the partial derivatives respect to the parameter vector.

We plan to use a simulation optimization approach, specifically the Simultaneous Perturbations Stochastic Approximation (SPSA) technique, for optimizing our clearing function fitting parameters as explained in Section 3.3. The choice of technique is based on the fact that it requires only 2 simulation runs for gradient estimation at each iteration, compared to *2p* simulation runs for the Symmetric Finite Difference method. The ease of implementation is also an advantage for SPSA. In our setting, our decision variables for the simulation optimization problem are the Clearing Function constraint parameters of each machine: intercept μ and slope β values of each linear segment used to approximate the clearing function. These decision variables are continuous. Formally the general scheme of the SPSA algorithm is as follows; details will be explained in CHAPTER 6.

Definitions:

k : index of Machines;

θ : Decision variables

Θ : Decision variables space, $\theta \in \Theta$

$L(\theta, \omega)$: Profit Realization at Simulation of replication ω with θ

$J(\theta)$: Expected Profit Realization with θ; $J(\theta) = E[L(\theta, \omega)]$

Objective function:

$max_{\theta \in \Theta} J(\theta)$;

Gradient Estimation using SPSA technique at iteration *n*:

$$\Delta_{n=} \left[\widehat{\Delta_n^1} \; \dots . \widehat{\Delta_n^p}\right]^T$$

$$\hat{\nabla} J_{in} = \frac{\hat{J}\,(\,\theta_n + c_n \Delta_n) - \hat{J}\,(\,\theta_n - c_n \Delta_n)}{2 c_n \Delta_{in}} \;\; i = 1, \dots, p$$

Updating Decision Variables:

$\theta_{n+1} = \theta_n + a_n \nabla J_n$ $\quad a_n$ is the gain at iteration n.

Algorithm:

Set *n* =1;

Step 1: Run Simulation with θ_n and obtain $J(\theta)$

Step 2: Calculate $\widehat{\nabla} J_n$

Step 3: Update $\theta_{n+1} = \theta_n + a_n \nabla J_n$

Step 4: if $\left(J(\theta_{n+1}) - J(\theta_n)\right) < \delta$ stop and report $max_{\theta \in \Theta} J(\theta)$; δ is small number

Else *n*=*n*+1, and return to Step 1..

In our research, we will also develop a heuristic algorithm that benefits from the idea of gradient estimation but with a different approach which uses the dual solutions of the LP model for gradient estimates. This approach will be explained in detail in CHAPTER 7. The motivation for the heuristic algorithm is to avoid potential time consuming calculations for gradient estimation due to the simulation runs necessary at each iteration and also the potential large number of iterations for the algorithm to converge.

In this thesis, we will provide the performance comparison of these two different CF fitting approaches and present their results. In the next section we present the simulation model of the testbed factory that will be the source of the empirical data for our fitting techniques.

CHAPTER 3. SIMULATION MODEL

Our data collection of variables of interest for estimation of clearing functions, comparison of performance for our fitted CFs and our algorithms (simulation optimization, search heuristic algorithms) will use a simulation model as a surrogate for the real factory. This model, originally developed by Kayton et al. (1997) will provide the empirical data on output and WIP used to fit the clearing functions, and to assess the performance of the resulting clearing functions by simulating the execution of the production plans obtained using them. Therefore, in the following sections, we present the details of the simulation model, steps to convert LP releases to simulation input, and the data collection procedure used in fitting the CF.

3.1. Simulation Model

Our simulation model of a re-entrant bottleneck system was built with the attributes of a real-world fab environment, studied in previous research (Kayton et al. 1997). The major characteristics of wafer fabrication, including a re-entrant bottleneck process, unreliable machines, batching machines, and multiple products with varying process routings and number of operations are included in the model. Wafers move through the facility in lots of a standard size. There is a distinct re-entrant bottleneck machine representing the photolithography process. The processing times for all other stations are scaled to the bottleneck processing time so that no non-bottleneck station has a utilization approaching that of the bottleneck. The model has batching stations (Stations 1 and 2) early in the process, representing the furnaces which perform the diffusion and oxidation processes, where up to four lots can be processed simultaneously. The remaining stations process one lot at a time.

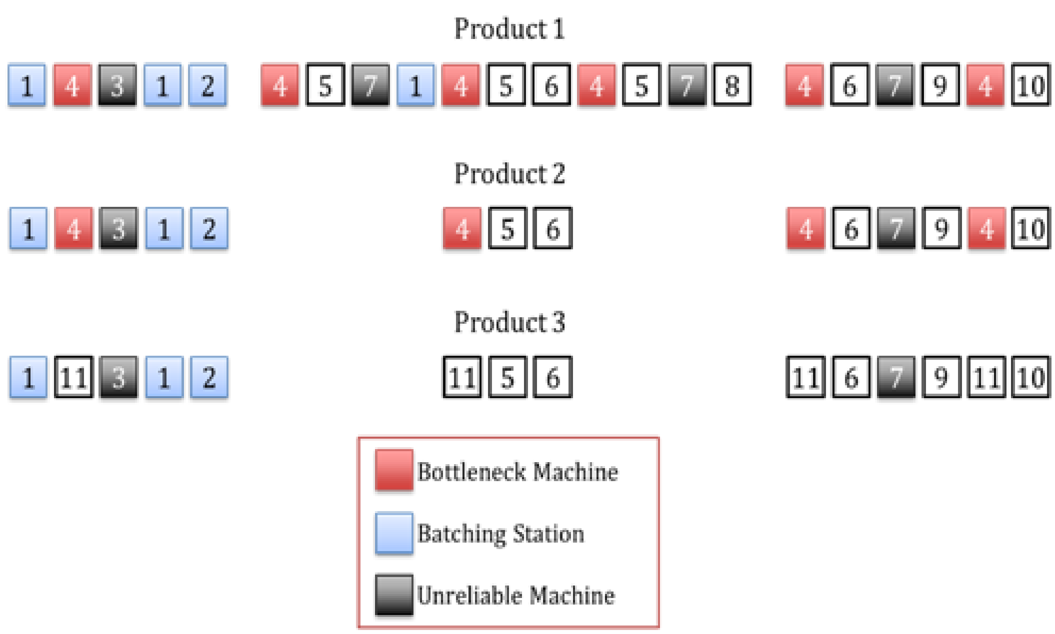


Figure 3-1: Re-entrant Bottleneck Model Process Chart for Products

There are 11 machines (stations) and 3 products in this model. The number of operations for Products 1, 2 and 3 are 22, 14 and 14, respectively. Station 4 is the re-entrant bottleneck machine shown in red in Figure 3-1. There are two servers at Station 4. Products 1 and 2 visit the bottleneck station 6 and 4 times, respectively. Product 3 does not use the bottleneck machine. This product visits Station 11, which is the only station that is allowed to exceed 80% utilization, but is not allowed to exceed the bottleneck utilization. There are two batching machines, machines 1 and 2, shown in blue in Figure 3-1, where up to four lots can be processed simultaneously as a batch. The minimum batch size required is two lots. The batching stations can run lots of different product types in the same batch. There are two unreliable machines, machines 3 and 7, shown in black in Figure 3-1, which are subject to considerably longer failures than the other stations and can cause starvation at the bottleneck station.

3.1.1. Simulation Parameters

The distributions of processing times and failures and their parameters will be presented in this section.

3.1.1.1. Simulation Processing Times

Table 3-1: Simulation Processing Times and Batch Sizes

Machine #	Mean	Std. Dev.	Batch (Min/Max)
1	80	7	(4/2)
2	220	16	(4/2)
3	45	4	1
4	40	4	1
5	25	2	1
6	22	2.4	1
7	20	2	1
8	100	12	1
9	50	4	1
10	50	5	1
11	70	2.5	1

The processing times for all stations are summarized in Table 3-1. All processing times follow a lognormal distribution, and are given in minutes. The standard deviations of process times are within 10 percent of the mean. The processing times for all operations on a given machine, regardless of product type, are assumed to follow the same distribution. This is clearly an approximation of the real situation in practice, but was validated by management during the original study by Kayton et al. (1997) as a reasonable starting point.

3.1.1.2. Failure Distribution Parameters

Table 3-2: Failure Distribution Parameters

	MTTF				MTTR			
Machine #	Alpha	Beta	Mean	Std. Dev.	Alpha	Beta	Mean	Std. Dev.
3	7200	1	7200	84.9	1200	1.5	1800	52.0
7	7200	1	7200	84.9	1200	1.5	1800	52.0

The mean time to failure (MTTF) and mean time to repair (MTTR) distributions follow gamma distributions with parameters given in Table 3-2. Machines 3 and 7 are unreliable machines that can produce a product in a very short time but can starve the bottleneck due to poor availability. The mean and standard deviation values are calculated using the formulas for gamma distribution as follow:

$$\mu = \alpha\beta$$

$\sigma^2 = \alpha\beta^2$

The values are in minutes. Availability can be calculated as follows:

$$A = \frac{MTTF}{MTTF+MTTR} = \frac{7200}{7200+1800} = 0.80, \tag{3.1}$$

This can be interpreted as on average machines are operating 80% of the time. In the simulation model, the failures are implemented as preemptive. We will experiment with different levels of variability in down and repair time in our experiments, while holding the average availability constant.

3.1.2. Simulation Details

The simulation model is created in Simulation Studio 1.6 (SAS) (www.sas.com) that runs on a Intel PC with a Intel(R) Pentium D CPU 3.40 GHz processor and 2GB of RAM, under Microsoft Windows XP Professional.

The period length for the production planning models is 7 days. Our planning horizon is 26 periods, thus simulation is run for 26 periods. An LP solution provides noninteger release quantities that need to be converted into whole quantities. The transfer of release inputs from LP to simulation will be discussed in Section 3.2. The release entities are created at the beginning of each day by reading the data from the converted releases to whole numbers. After presenting our simulation model, we will discuss the conversion of LP releases to simulation input and data collection in the next sections.

3.2. Conversion of LP Releases to Simulation Input

The release schedule that is found by LP models specifies the number of lots of each product to be released in each planning period, and may be fractional. Simulation requires an integer number of lots, so in order to give the release schedule suggested by LP as an input to simulation, the fractional values need to be rounded to integer values.

In addition to being fractional, the release quantities suggested by the LPs are weekly quantities. In our simulations, we assume that the planned releases for a planning period are uniformly distributed over that period. Hence we convert the weekly release schedules to daily release schedules for that period.

In order to address these issues, we use an algorithm that rounds some release values up and some values down. We do not simply round up all values since the cumulative effect could lead to significant excess production. In our algorithm, we first divide the planned release quantity for each period, which is one week in our study, into equal daily release quantities by dividing the weekly release quantity by seven for all periods. The release quantity of the first day of that period is always rounded up to its nearest integer value, and the difference between the planned fractional releases and obtained integer releases is recorded. If the cumulative difference is greater than 0.001, the next value is rounded down to its nearest integer, otherwise it is rounded up. Our objective is to maintain the difference between cumulative release schedule of the planning horizon from LP and cumulative adjusted release schedule for simulation input close to zero. Thus the planned LP release schedules and simulation input schedules are matched closely, enabling us to compare LP outputs to simulation outputs more accurately. The steps of this algorithm can be summarized as follows.

Step 1: Divide the weekly schedule into equal daily schedules by dividing the release schedule from LP by seven.

Step 2: Round up the first release day of that period.

Step 3: Check whether cumulative differences of actual releases from LP and obtained integer releases are greater than 0.001 or not. If the quantity is greater, round down the corresponding fractional release. If it is less than 0.001, than round up the corresponding fractional release. Go to the next daily schedule.

Step 4: When the calculations are done for one weekly schedule, start over the process with the next weekly schedule, following Steps 1, 2 and 3.

The steps are summarized as a pseudocode below. The value of Rounded(R_{it}) corresponds to obtained integer value of R_{it} using the operations explained in the previous paragraph.

Set *t*=1; *t* is the planning period. T is the planning horizon

Step 1: R_{daily}=R_{weekly}/7

Step 2: RoundUP(R_{1t})

Step 3: do d=2 to 7; d is number of the day of the week.

if $(\sum_{i=1}^{d} R_{it} - \sum_{i=1}^{d} Rounded(R_{it})) > 0.001$

Then   RoundDown(R_{it});

Else   RoundUP(R_{it});

End;

Step 4: Set t=t+1 until t=T. Continue with steps 1-4.

Using the steps above, we obtain whole numbers for release lots of each day that gets into the simulation. After the lots are released into the system before their first operation, the products are sequenced in the order of products 1, 2 and 3 repeatedly. The excess amounts of products are put in front of other products following the same sequencing logic. For example, we have 4 lots of product 1, 3 lots of product 2, and 2 lots of product 3, then the sequencing of lots will be 1-1-2-1-2-3-1-2-3. This sequence will processed in the station in the order from left to right, product type 1 being first to be processed. We use this procedure to obtain more uniform output rates among products. Lots are dispatched in First-in-First-Out order on all other operations after their first one.

In the next section we discuss how the data is collected from the simulation model to use in fitting the clearing functions.

3.3. Data Collection for Fitting

In this study, we investigate two forms of clearing function where in the first form output is a function of releases within the period and WIP at the beginning of the period considered separately whereas in the second form output is a function of sum of release and WIP. These functional forms are given below.

$$X_{kgt} = f_k\left(W_{k,\mathrm{g},t-1}, R_{kgt}\right), \tag{3.2}$$

$$X_{kt} = f_k\left(W_{k,t-1} + R_{kt}\right). \tag{3.3}$$

For these CF functional forms, the data required in fitting is different. For the load based functional form, $X_{kt} = f_k\left(W_{k,t-1} + R_{kt}\right)$, CFs are fit based on each machine, not individual products, where X_{kt} represents the total output of machine *k* in period *t*. Note that in our simplified model, all operations at a machine have the same processing time distribution, regardless of product type or operation, thus variables are in units of quantities. The quantity $W_{k,t-1}$ represents the total number of lots available in WIP at the beginning of the period at machine *k* and R_{kt} the total number of lots released to machine *k* in period *t*. The product based functional form $X_{kgt} = f_k\left(W_{k,g,t-1}, R_{kgt}\right)$ distinguishes between product types, *g* being the product index. Thus CFs are based on product type in addition to which machine they are processed on.

In summary, the data needed to fit a function of form $X_{kt} = f_k\left(W_{k,t-1} + R_{kt}\right)$ for each machine k are the total releases within a period R_{kt}, total number of WIP at the beginning of the period $W_{k,t-1}$ and total output in that period X_{kt}. For the functional form $X_{kgt} = f_k\left(W_{k,g,t-1}, R_{kgt}\right)$, we need the individual quantities of each product mentioned above. Denoting g as the product index, note that $\sum_{g\in G} X_{kgt} = X_{kt}$, $\sum_{g\in G} W_{k,g,t-1} = W_{k,t-1}$ and $\sum_{g\in G} R_{kgt} = R_{kt}$.

3.3.1. Outline for Data Collection

Our focus in this thesis is to fit CFs from empirical data. We use our simulation model to obtain our data. In order to obtain data that has basic coverage of several WIP $W_{kg,t-1}$ and Release R_{kgt} scenario combinations and corresponding outputs X_{kgt}, we experiment with several long simulation runs corresponding to different bottleneck utilization levels. The reason for having runs with several bottleneck utilizations is to observe many different combinations of $W_{k,g,t-1}$ and R_{kgt} so as to obtain CFs that better reflect the characteristics of the resources over a wide range of operating conditions. We follow the steps below to obtain data to fit our clearing function. Here, we note that, the release plans used in these long simulation runs are created randomly with the aid of the normal distribution and executed in the simulation in order to obtain the required data for fitting. For data collection purposes only, we choose to run the simulation for 91 periods using the randomly created release plans. In our experiments, we will use 26 periods for our planning horizon in the LP and simulation to compare our production planning models using different CF functional forms, for which the experimental design will be explained in the next chapter. We define D_{gt} as demand of product g at period t.

Step 1: Seven different demand realizations corresponding to average bottleneck utilization levels of 49%, 60%, 70%, 77%, 87%, 94% and 99% are generated assuming normally distributed demand in each period. A release schedule is then obtained by setting the release quantity for each product in each period to the demand for that product in that period i.e., $R_{gt} = D_{gt}$.

Step 2: For each schedule, the simulation model is run for 91 periods with five independent replications, collecting the X_{kgt}, R_{kgt} and $W_{k,g,t-1}$ data required for each machine. Note that the randomness in the simulation is associated with machine failures and processing times; demand is assumed to be deterministic and known.

Step 3: All the data is combined into one file and plotted for each machine. Using the simulation model and parameters explained in Section 3.1, the plots of two functional forms $X_{kgt} = f_k\left(W_{k,\mathrm{g},t-1}, R_{kgt}\right), X_{kt} = f_\mathrm{k}\left(W_{k,t-1} + R_{kt}\right)$ are obtained. Recall that we refer to sum of releases within a period and WIP at the beginning of that period as resource load. The plots and discussion are presented in the next section.

3.3.1.1. Plots of Empirical Data for Selected Machines

We have eleven machines in our system. For the sake of brevity we include only the plots for the most interesting machines: machine 1 which is the first machine for all products, and a batching machine; machines 3 and 7 which are the unreliable machines; and station 4 which is the bottleneck station consisting of two servers. For the functional form $X_{kt} = f_k\left(W_{k,t-1} + R_{kt}\right)$, the plots of other machines are straightforward, showing a linear accumulation of data without much deviation. For the functional form $X_{kgt} = f_k\left(W_{k,g,t-1}, R_{kgt}\right)$, plots are presented in units of total output, total initial WIP and total releases over products for purposes of illustration.

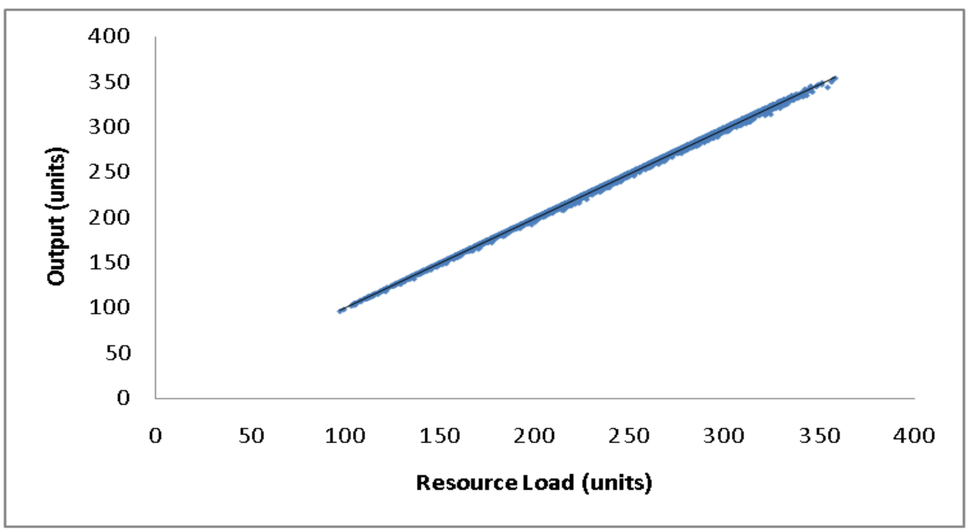


Figure 3-2: Machine 1 Output vs. Resource Load

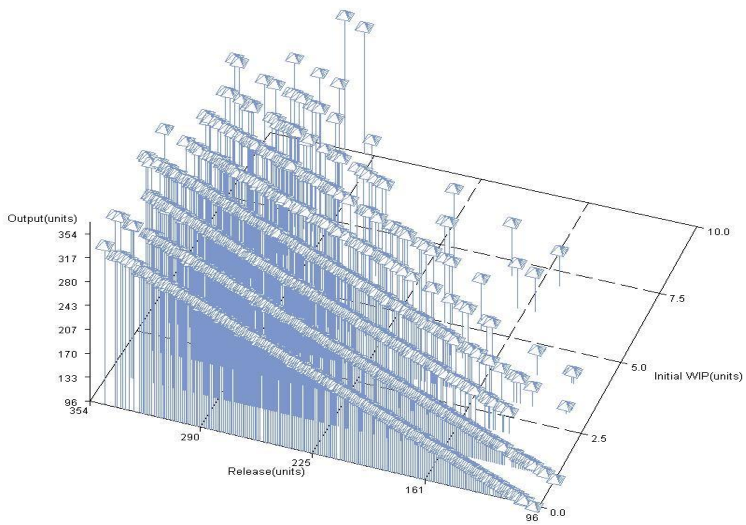


Figure 3-3: Machine 1 Output vs. Release and Initial WIP

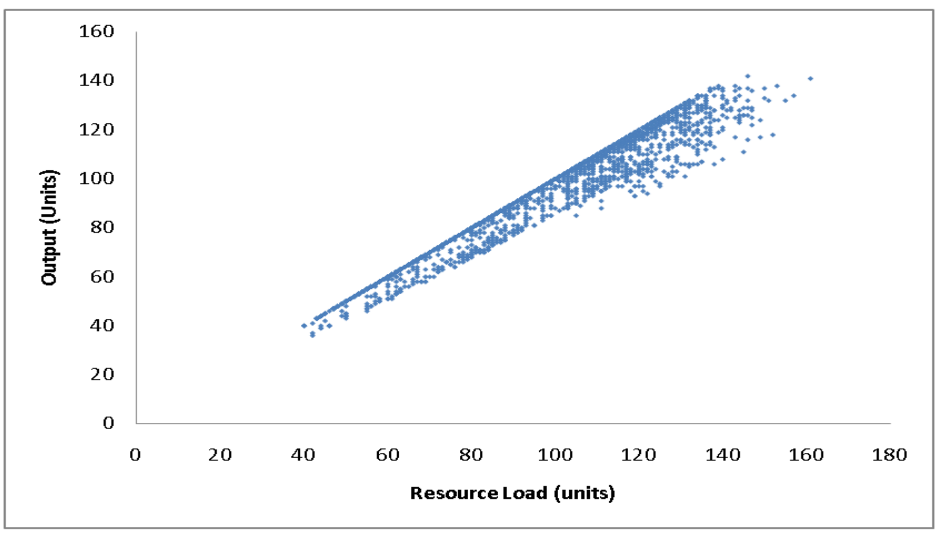


Figure 3-4: Machine 3 Output vs. Resource Load

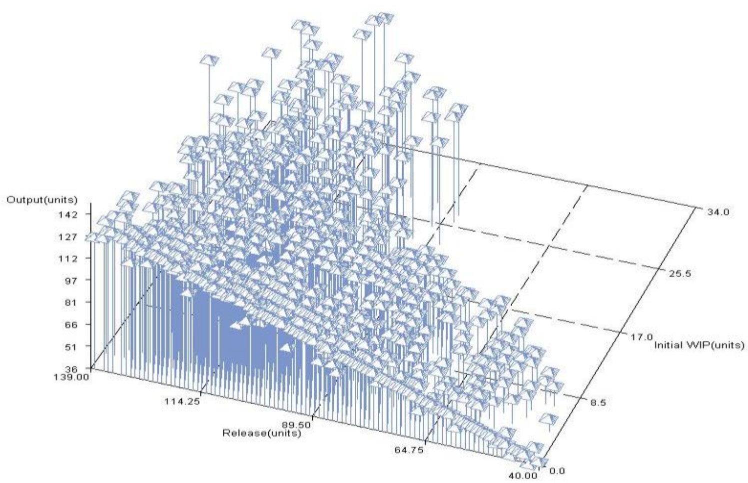


Figure 3-5: Machine 3 Output vs. Release and Initial WIP

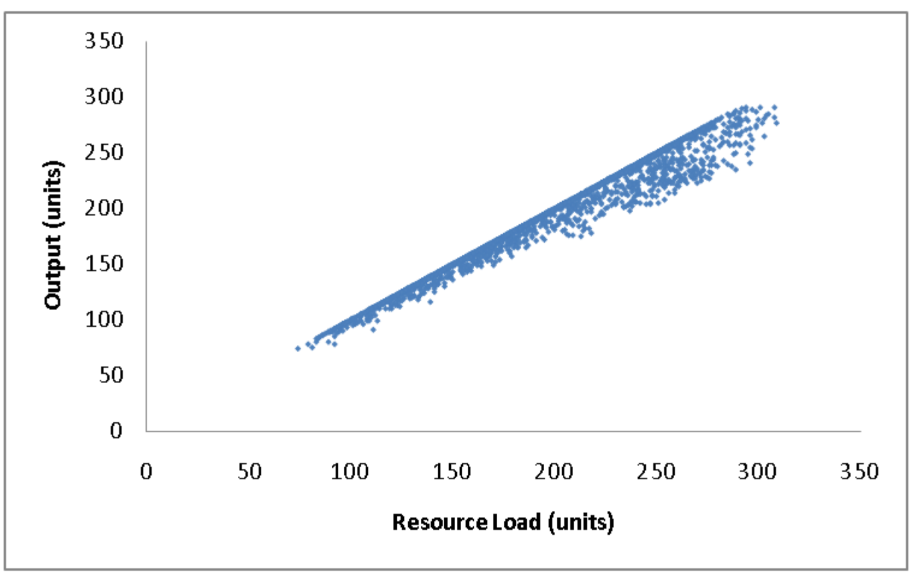


Figure 3-6: Machine 7 Output vs. Resource Load

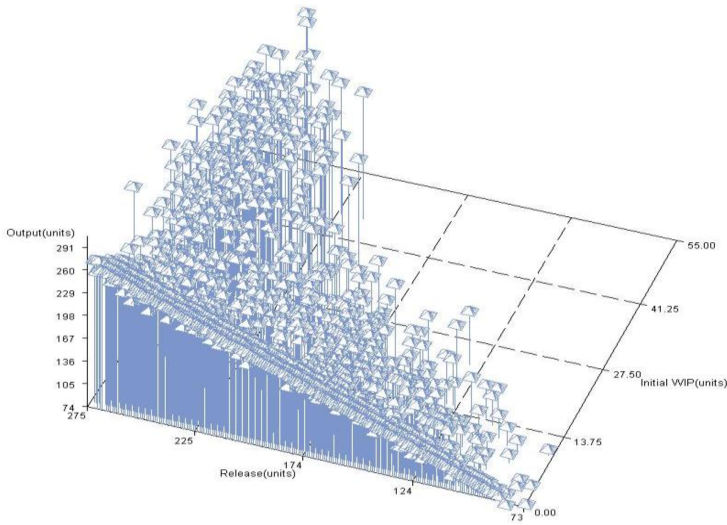


Figure 3-7: Machine 7 Output vs. Release and Initial WIP

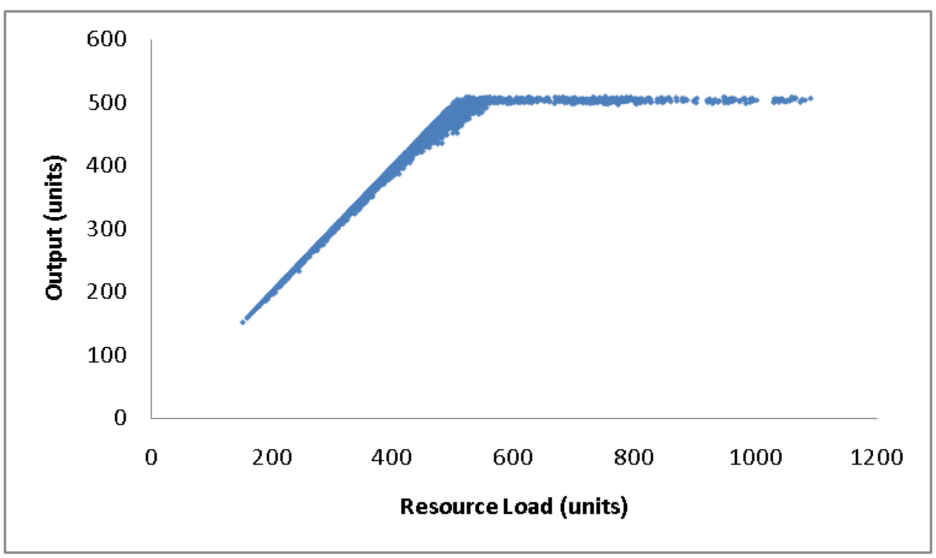


Figure 3-8: Machine 4 Output vs. Resource Load

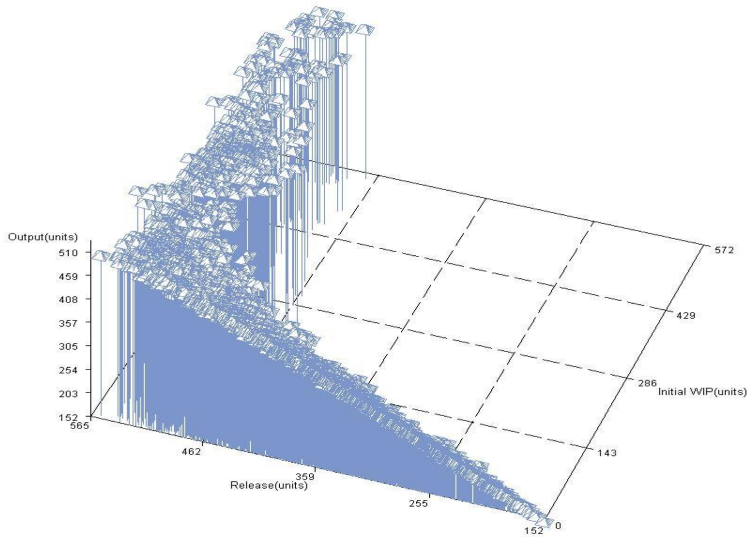


Figure 3-9: Machine 4 Output vs. Release and Initial WIP

Figure 3-2 shows the plot for machine 1 which is the one of the batching machines. We observe a linear relation between output and resource load. Similarly Figure 3-3 shows that for a given of initial WIP, we observe a linear relationship between output and release. There is minimal variability in output on this machine since it is not subject to significant failures or variability in flow, as it is the first machine in the line. Figure 3-4 shows the plot of machine 3, which is an unreliable machine. On this plot, we also observe a linear relation between output and resource load since the data is mostly accumulated on the upper limit with deviations below the upper limit due to failure. Figure 3-5 shows the effects of failure on the unreliable machine 3, the data is spread over the area. Figure 3-6 shows the plot for machine 7, another unreliable machine, which is very similar to that for machine 3.

Figure 3-8 shows the plot of machine 4, which is our bottleneck machine. On this plot we observe a good linear relation until it reaches its capacity value. When this machine reaches its capacity, increasing resource load does not increase the amount of output, causing the curve to level off. Figure 3-9, depicting the situation for the bottleneck machine, shows that when the machine approaches its capacity limit, the data is more scattered in that region. Note that in Figure 3-9, there are no observations at the right upper corner of the plot, since for given high release and WIP levels; we do not observe any low output observations. From the plots for both functional forms, we see that the observations appear to follow a concave functional form when the resource load or release for a given initial WIP increases. This is crucial in terms of implementing those functional forms in optimization models, since as discussed in Asmundsson et al. (2009), the formulation depends on the concavity assumption of the CF in order to obtain a convex set of constraints for the capacity.

The plots for other machines are very similar for both functional forms to the plot of machine 1 shown in Figure 3-2 and Figure 3-3. We observe good linear relations between output and resource load and also same linear relation applies for output and release for a given level of initial WIP for these machines. We will now discuss fitting our linear functions to different segments of the plot in the next section.

3.3.2. Fitting Clearing Functions to Data

In this section we discuss the functional form $X_{kt} = f_k\,(W_{k,t-1} + R_{kt})$, where the output is assumed to be a function of the total load R_{kt} + $W_{k,t-1}$. The discussion of the $X_{kgt} = f_k\,(W_{k,\mathrm{g},t-1}, R_{kgt})$ functional form, where beginning WIP and period releases are treated as separate independent variables, will be given in Section 5.2.

We observe from the plots that there is a linear relation between outputs and resource load. By inspection, we see that for machines that show a good linear relation such as machine 1 shown in Figure 3-2, we can apply simple linear regression to whole data set to estimate the intercept and slope of the clearing function. However, we observe changes in the slope of the data values as the resource load increases by examining the other plots of interest as in Figure 3-4, Figure 3-6 and Figure 3-8. In order to obtain fits that will represent the data more accurately, we decided to divide the data into parts and focus on them separately. We refer to those parts as segments, which have starting and ending resource load values and use the data contained in these boundaries. By examining the plots, we decided to divide the data into three segments. The third segment will represent the capacity limit of the machine. For this segment, there is no regression applied, that it has properties of slope zero and intercept equal to the capacity limit. The first and the second segments will include the estimation of intercept and the slope values of the linear section. Basically, we consider the linear portion of the data as two segments. To do this, we find the range of the linear section by finding the minimum and maximum resource load. We observe that for the low resource load values, the linear relation is pretty good. Then, by inspection we decide that the lower 40% of this range will give the boundaries for the first segment and the next 40% will form the boundaries for the second segment. Denoting the maximum observed value of the resource load by L, the first segment covers the interval $[0,0.4L]$ and the second $[0.4L,0.8L]$. All resource load values greater than $0.8L$ fall into the third segment, and represent the maximum output of the machine. Again, we note that, the reason for dividing the linear part of the data into two segments is to capture any change of slope or intercept values when the machine gets closer to its capacity limit instead of using one line which could give less accurate estimates. The plots of the fitted CFs of selected machines are in Appendix. We also experimented to divide the linear part of the data into more than three segments, e.g. four and five but did not observe any improvements in our results. Thus throughout the thesis, we use total three segments, including the third segment representing the capacity limit to fit our CFs. As mentioned before, the third segment represents the capacity limit of the machines, which will be estimated by dividing the period length by the mean processing times of the machines given in Table 3-1. For machines 3 and 7, due to the presence of machine failure we take into consideration of the average availability in the period, meaning that after dividing the period length by the mean process times, we also multiply it by the availability to have a better estimate of capacity limit as given in the equation below;

$$Capacity\ limit = \frac{Period\ Length}{Mean\ Processing\ Time} * Availability$$

In the following chapter we discuss our experimental design that will be used for performance comparison of developed models that will be explained in CHAPTER 5, CHAPTER 6 and CHAPTER 7.

CHAPTER 4. EXPERIMENTAL DESIGN

In the previous chapters, we gave the details of our simulation model and then presented the two CF functional forms that are used in this study. These two CF functional forms used in production planning model will result in different release plans, and we want to compare their performances based on our experimental factors.

Our experiments are designed to examine the effects of two different factors on the performance of the production planning models using the two different CF estimation procedures discussed in the previous chapter: the bottleneck utilization at different demand patterns, and the severity of the machine failures. The details of incorporating these two CF functional forms into mathematical modeling will be explained in the following Chapters 5 and 6. Our planning horizon is 26 periods, where one period is equal to one week, and one week is considered as seven days. Recall that our demand is deterministic.

4.1. Bottleneck Utilization with Different Demand Patterns:

It is well known from queuing theory that the nonlinear relationship between resource utilization and cycle times becomes more severe at high utilization levels. Hence one would expect any type of clearing function to perform similar to classical LP models that use fixed lead time, at low utilization levels, but to show its capability to represent the non-linear relationship between cycle time and utilization better at higher utilization levels. By the same token we would expect the differences in performance between the LP models using the different CF fits to be more marked at higher utilization levels. Hence we experiment with two average bottleneck utilization values of 0.7 and 0.9 over all periods. The utilization level is achieved by varying the demand of all products. In the next section we present the demand patterns that we will use to test the performance of the LP models.

4.1.1.1. Low Coefficient of Variation (CV) Mixed Demand pattern

We have two demand cases we consider. The first case is a demand pattern with low coefficient of variation (CV) that will give approximately 70% bottleneck machine utilization. The second case is a low CV demand pattern that will give approximately 90% bottleneck machine utilization. We create mixed demand cases where we try to achieve uncorrelated demands across products while maintaining an average bottleneck utilization of 70% and 90%. Demand patterns are created randomly from an uniform distribution with a 10% coefficient of variation (CV) for each product in each period, which we refer as low CV demand case. Figure 4-1 and Figure 4-2 illustrate examples of the two cases of the low CV mixed demands.

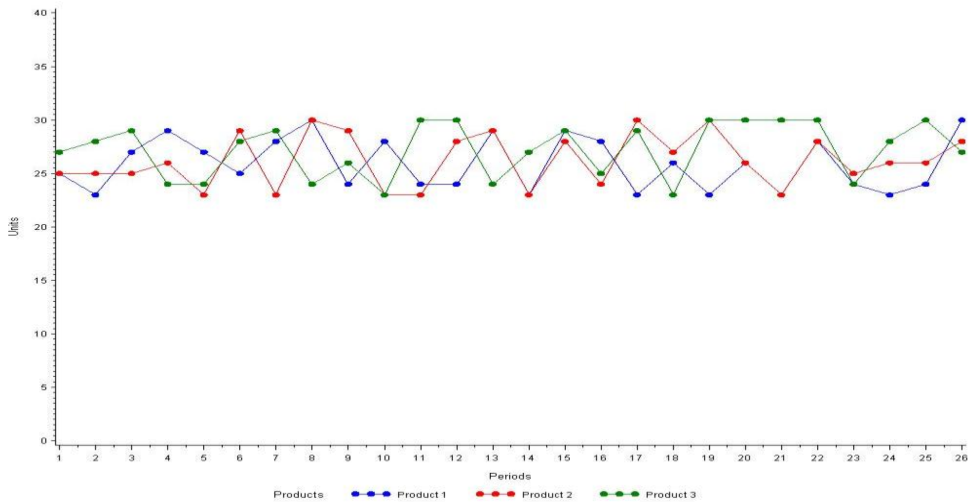


Figure 4-1: Low CV Low Utilization Demand

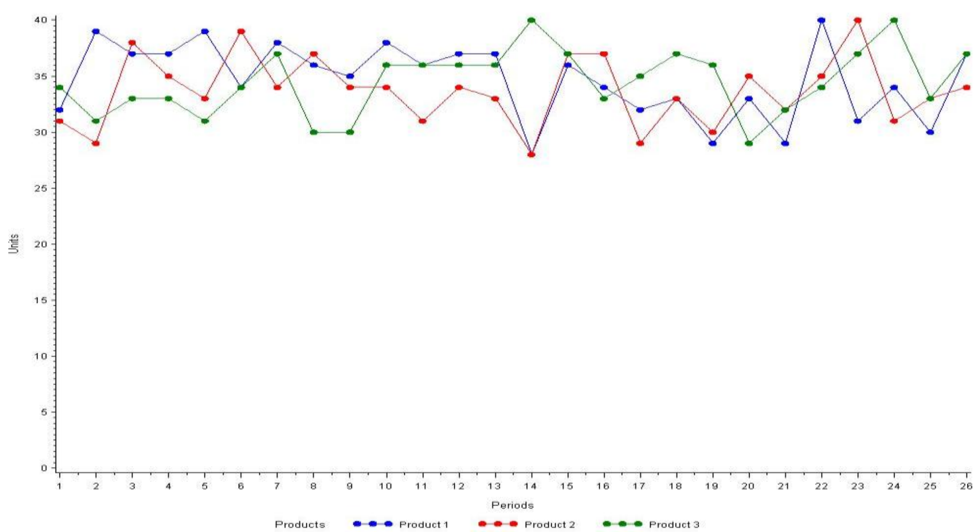


Figure 4-2: Low CV High Utilization Demand

4.1.1.2. High Coefficient of Variation (CV) Mixed Demand pattern

As in the low CV case, we create random demand patterns again using an uniform distribution with a 30% CV which we refer to as the high CV demand case. We again have demand patterns that will correspond to 70% and 90% bottleneck utilization. Figure 4-3 and Figure 4-4 illustrate the two cases of the high CV mixed demands.

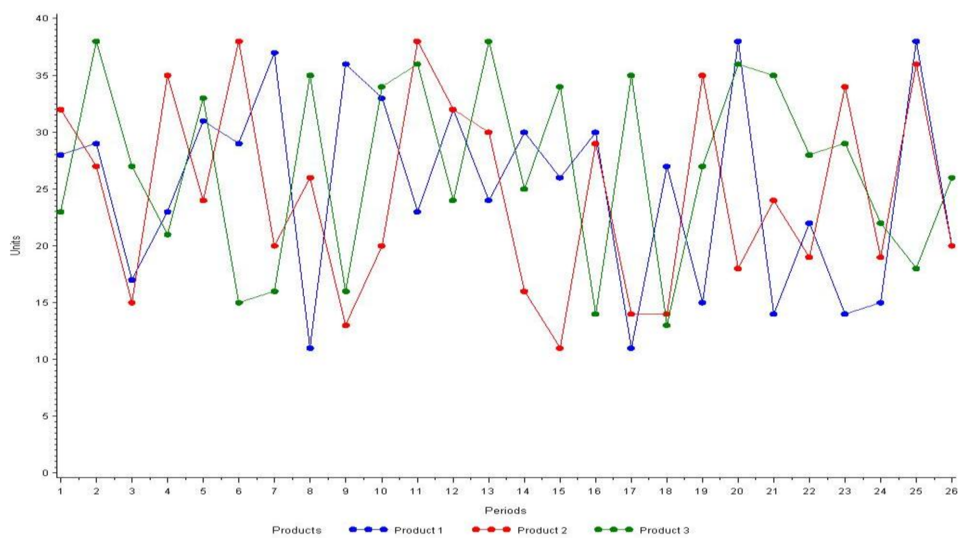


Figure 4-3: High CV Low Utilization Demand

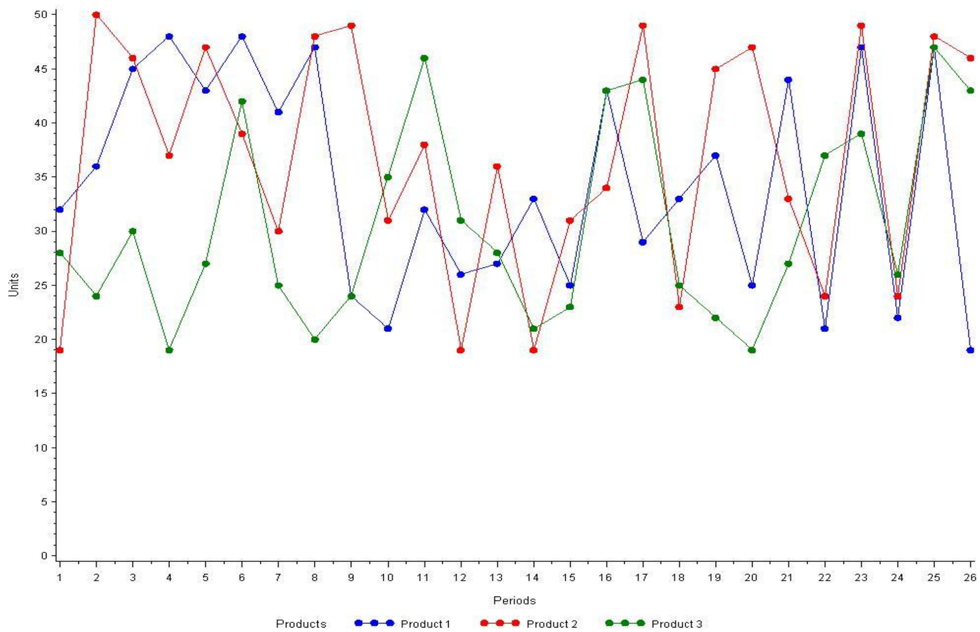


Figure 4-4: High CV High Utilization Demand

4.1.2. Length of MTTF and MTTR

We would expect the performance of planning models to deteriorate under higher machine failures, since these increase the variability of the system. We will consider two cases. In the first case, we use the original parameters of MTTF and MTTR that are given in Table 3-2. We will refer to this case as the Short Failure case. Our second case will be failures with longer MTTF and MTTR times for both failure machines. For this case we double the MTTF and MTTR values by multiplying the alpha parameters of the gamma distribution by 2. We obtain the values shown in Table 4-1 using the mean and variance expressions given in Section 3.1.1.2 .We will refer to this case as the long failure case. This case maintains the same average availability as the short failure case, with significantly higher variability in uptime.

Table 4-1: Failure Distribution Parameters for Long Failure

	MTTF				MTTR			
Machine #	Alpha	Beta	Mean	Std. Dev.	Alpha	Beta	Mean	Std. Dev.
3	14400	1	14400	120.0	2400	1.5	3600	73.5
7	14400	1	14400	120.0	2400	1.5	3600	73.5

We aim to test how the fitting of the clearing functions will be affected by longer MTTF and MTTR values, since as we change this property of the system, we expect changes in clearing functions for both forms and also, how this will change the optimal solution of the LP.

Under the base values of MTTF and MTTR, with MTTF = 7200 minutes and MTTR = 1800 minutes and planning period length being 10080 minutes, on average the machines will fail once in every period. The corresponding empirical data and fitted clearing functions are given in Sections 3.3 and 3.3.2.

Under the Long Failure case, we estimate new clearing functions, as the nature of the underlying production system is substantially altered by the change in machine failures. Note that it is possible, with MTTF = 14400 we do not have any failures in each period but once there is a failure, it will last more than one third of a period, with MTTR =3600.

4.2. Comparison of CF Estimation Algorithms:

Given the details of our experimental factors and two CF estimation procedures in the previous sections, we aim to compare the performance of these estimation procedures. The basis for comparison will be the expected profit of the production system obtained through simulation of its operation when controlled by the planning decisions obtained with the two different CF estimation techniques.

We compare the different methods for estimating clearing functions as follows:

Step 1: Solve the LP using the clearing functions estimated by the different regression models to obtain a planned release schedule.

Step 2: Simulate the operation of the fab under this release plan for fifteen replications with one run length equal to 26 weeks recording the realized output.

Step 3: Compute the expected profit value from the outcome of the simulation which is explained in the following paragraph.

We compare the objective function values realized statistically using fifteen replications of the simulated execution of the plans suggested by the different models using different CF fit estimates. To accomplish this we enter releases from LP models as input to the simulation model following the approach described in Section 3.2 to obtain the realized inventory, backlog and output levels at each period. This allows us to calculate the realized profit by multiplying these values by their corresponding costs based on the each of 15 replications. We use values of 50 for backlog cost, 35 for WIP cost, 3 for material cost, 15 for inventory cost and 60 for revenue. We assume that WIP cost is high due to restrictions of space in the fab and use the same cost values for all products. Thus the primary factor affecting the relative desirability of products is their capacity consumption pattern. We use the same cost values in the LP models.

The experimental design is summarized in Figure 4-5 below.

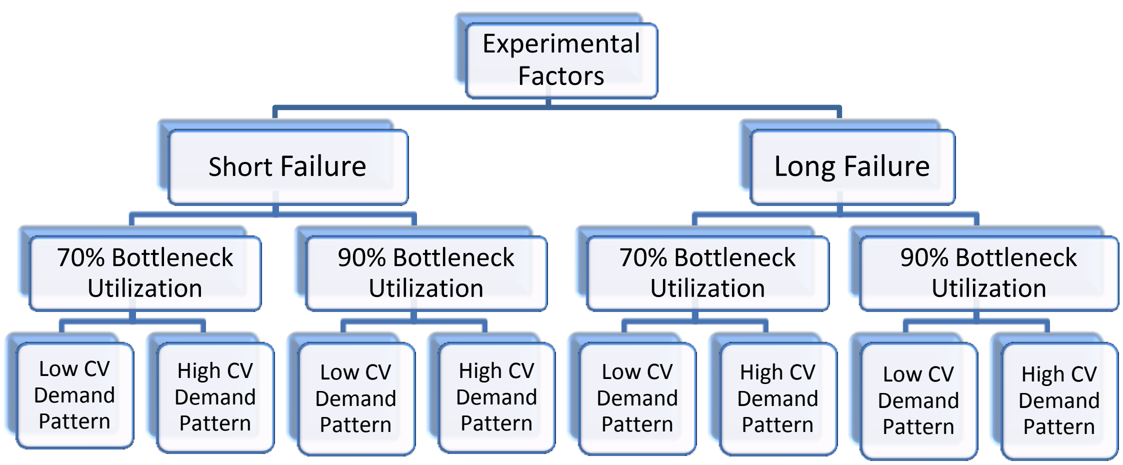


Figure 4-5: Schema of Experimental Factors

The factors that are listed in the figure above will be used to test the multiple linear regression, simulation optimization and search heuristic algorithm models.

Next, we present our CF estimation models starting with multiple linear regression fitting model.

CHAPTER 5. CLEARING FUNCTION FITTING BY MULTIPLE LINEAR REGRESSION

5.1. Linear programming (LP) Model

For all the planning models presented in this thesis, we shall define an operation l to be the processing of a product at a specific machine k. We define $L(k)$ to be the set of all operations l taking place on a specific machine k. Recall that all operations for all products that take place on a specific machine require same processing time. We also define $K(l)$ to be the machine k where operation l takes place.

We define the following notation:

Indices:

g : product index.

k : machine index.

l : operation index.

t : period index.

Decision Variables:

$Y_{g,t,l}$: number of units of product g that have completed processing at the l'th operation in period t.

$Y_{gt}(k)$: output quantity of product g from machine k in period t; $Y_{gt}(k) = \sum_{l \in L(k)} Y_{g,t,l}$

$X_{g,t,l}$: release quantity of product g in period t at operation l. This is equal, by definition, to the output of the previous operation in the product's routing in that period. Hence $X_{g,t,l} = Y_{g,t,l-1}$, where operation $l-1$ is the immediate predecessor of operation l in the routing of product g. We assume that each operation will have one predecessor and one successor, although our formulations are easily extended to incorporate multiple predecessors and successors.

$X_{gt}(k)$: release quantity of product g in period t at machine k, given by $X_{gt}(k) = \sum_{l \in L(k)} X_{g,t,l}$

Y_{gt} : output quantity of product g in period t. $Y_{gt} = Y_{g,t,l'}$ where l' is the last operation for product g.

X_{gt} : quantity of product g released into the first station in the line in period t.

$W_{g,t,l}$: WIP of product g at operation l at the end of period t.

$W_{gt}(k)$: WIP of product g during period t at machine k, given by $W_{gt}(k) = \sum_{l \in L(k)} W_{g,t,l}$.

I_{gt} : units of product g in finished goods inventory at the end of period t.

B_{gt} : units of product g backlogged at the end of period t.

Parameters:

h_{gt} : unit inventory holding cost for product g in period t.

b_{gt} : unit backlogging cost for product g in period t.

ω_{gt} : unit WIP cost for product g in period t.

D_{gt} : demand for product g during period t.

$M(k,g)$: index set of the independent variables included in the regression model for machine k of product g.

μ_{kg} : Intercept of the clearing function for product g at machine k.

β_{kg}^{n} : Coefficient of variable $n \in M(k,g)$ in the clearing function for product g at machine k.

C_k : Maximum capacity of machine k in units of products. Since in our simulation model all operations of all products at the same machine require the same amount of time, this is a valid representation.

The formulation can now be given as follows:

Objective function:

$$min \sum_{g\in G} \sum_{t=1}^{T}\sum_{\mathrm{k=1}}^{\mathrm{K}} \omega_{gtk}\, W_{gt}(k) + \sum_{g\in G} \sum_{t=1}^{T} h_{gt}\, I_{gt} + \sum_{g\in G} \sum_{t=1}^{T} b_{gt}\, B_{gt} \qquad (5.1)$$

Subject to:

WIP Balance:

$$W_{g,t,l} = W_{g,t-1,l} + X_{g,t,l} - Y_{g,t,l} \quad g \in G \;, t = 1, \dots, T \;, \mathrm{l} \in L \qquad (5.2)$$

Inventory Balance:

$$Y_{gt} + I_{g,t-1} - B_{g,t-1} - I_{gt} + B_{gt} = D_{gt} \;\; g \in G \;, t = 1, \dots, T \qquad (5.3)$$

Capacity Constraints:

$$Y_{gt}(k) \leq \mu_{kg} + \sum_{n \in M(k.g)} \beta_{kg}^{n} MRV_{g}^{n}, \quad g \in G, \quad t = 1, \dots, T, k \in \mathrm{K} \qquad (5.4)$$

$$\sum_{g \in G} Y_{gt}(k) \leq C_k \quad , t = 1, \dots, T, k \in K \qquad (5.5)$$

Variable Nonnegativity:

$$W_{g,t,l}, Y_{g,t,l}, I_{gt}, B_{gt}, X_{g,t,l} \geq 0 \; g \in G \quad , t = 1, \dots, T, k \in K, l \in L \qquad (5.6)$$

where MRV_{g}^{n} denotes the appropriate variables identified in the regression models discussed in the next section. For example, MRV_{g}^{n} variables consist of variables $W_{gt}(k)$ or $X_{gt}(k)$ which are summations over the relevant operations.

In this formulation, we aim to minimize the sum of finished goods inventory holding, WIP holding and backordering costs across the planning horizon. The first constraint sets (5.2) and (5.3) represent WIP and finished inventory balance for each product at each operation. The capacity constraints (5.4) represent the clearing functions estimated for each product using multiple linear regression as described in the following section. Finally, the total output of all products at each machine is constrained not to exceed the overall capacity of the machine in the planning period given in (5.5). Solutions show that (5.5) is only tight for product 3 at machine 11 since that machine is only used by product 3.

5.2. Multiple Linear Regression (MLR) Models

Based on exploratory experiments, we focused on the three linear regression models below. For brevity we shall state the models for Product 1 only; those for other products are specified in an analogous manner. Recalling that *g* is product index, we present the models below.

Model 1: Under this model the output of each product at machine *k* in period *t* is determined by the releases of all products in that period and the WIP level of each product at the start of period *t*,

$$Y_1(k,t) = \mu_{k,1} + \beta_{k,1}^{1} X_{1,t}(k) + \beta_{k,1}^{2} W_{1,t-1}(k) + \beta_{k,1}^{3} X_{2,t}(k) + \beta_{k,1}^{4} W_{2,t-1}(k) + \beta_{k,1}^{5} X_{3,t}(k) + \beta_{k,1}^{6} W_{3,t-1}(k) .$$

Model 2: This model considers the same set of variables as Model 1, with the addition of the releases of Product 1 in the immediately preceding period,

$$Y_1(k,t) = \mu_{k,1} + \beta_{k,1}^{1} X_{1,t}(k) + \beta_{k,1}^{2} W_{1,t-1}(k) + \beta_{k,1}^{3} X_{2,t}(k) + \beta_{k,1}^{4} W_{2,t-1}(k) + \beta_{k,1}^{5} X_{3,t}(k) + \beta_{k,1}^{6} W_{3,t-1}(k) + \beta_{k,1}^{7} X_{1,t-1}(k).$$

Model 3: This model considers the variables in Model 2 with the addition of the releases of all products in the immediately preceding period.

$$Y_1(k,t) = \mu_{k,1} + \beta_{k,1}^{1} X_{1,t}(k) + \beta_{k,1}^{2} W_{1,t-1}(k) + \beta_{k,1}^{3} X_{2,t}(\mathrm{k}) + \beta_{k,1}^{4} W_{2,t-1}(k) + \beta_{k,1}^{5} X_{3,t}(k) + \beta_{k,1}^{6} W_{3,t-1}(k) + \beta_{k,1}^{7} X_{1,t-1}(k) + \beta_{k,1}^{8} X_{2,t-1}(k) + \beta_{k,1}^{9} X_{3,t-1}(k) .$$

The models are summarized in Table 5-1 below.

Table 5-1: Summary of Multiple Linear Regression Models

	Independent Variables								
	$X_{1,t}(k)$	$W_{1,t-1}(k)$	$X_{2,t}$(k)	$W_{2,t-1}(k)$	$X_{3,t}(k)$	$W_{3,t-1}(k)$	$X_{1,t-1}(k)$	$X_{2,t-1}(k)$	$X_{3,t-1}(k)$
Model 1	✓	✓	✓	✓	✓	✓			
Model 2	✓	✓	✓	✓	✓	✓	✓		
Model 3	✓	✓	✓	✓	✓	✓	✓	✓	✓

The specific forms of these models are likely to be somewhat specific to our experimental environment. For example, we very seldom had cycle times of more than two periods at any station; if longer cycle times were incurred, additional release variables would be likely to enter the regression equations. For each of these models, we applied three different regression selection criteria. These step-wise regression approaches can be found in Draper and Smith (1981):

1) ADJUSTED R SQUARE SELECTION (ARSQ):

The Adjusted *R*-Square method finds the subset of independent variables that best predict a dependent variable by linear regression in the given sample, selecting the subset of dependent variables yielding the highest adjusted R-square value.

2) FORWARD SELECTION (FORWARD):

For each independent variable, this method calculates *F*-statistics that reflect the variable's contribution to the model if it is included. The *p*-values for these *F* statistics are compared to a threshold significance level. If no *F*-statistic has a significance level greater than the specified threshold level, the procedure stops.

Otherwise, the method adds the variable with the largest *F*-statistic to the model. The method then recalculates *F*-statistics for the variables not yet included, and the evaluation process is repeated. Thus variables are added to the model one by one until no as yet unincluded variable produces a significant *F*-statistic.

3) Backward Elimination (BACKWARD)

This selection technique begins by calculating *F*-statistics for a model, including all independent variables. Variables are then deleted from the model one by one until all the variables remaining in the model produce *F*-statistics that are significant at the threshold significance level at each step, the variable showing the smallest contribution to the model is deleted.

We will present the experimental results based on our experimental design in the previous chapter, in the following section.

5.3. MLR Models Experimental Results

For brevity we use a triple to denote each scenario. In the first position, S stands for short failures and L for long failures. In the second position, the numbers 70 and 90 correspond to 70% and 90% bottleneck utilization, respectively. In the final position the words lowCV and highCV stand for 10% and 30% coefficient of variation (CV) for each product demand, respectively. V1A stands for Model 1 with Adjusted *R*-square selection. F and B denote Forward and Backward selection, respectively, while V2 and V3 denote Models 2 and 3 respectively.

Our first comparison is based on the realized profit values shown in Figure 5-1 To further probe the relationship between the performance of the different models, the Friedman Test (Conover 1980) is used to statistically compare the mean of profit values of all algorithms at a significance level of p=0.05. It is a non-parametric test used to compare observations repeated on the same subjects, and uses the ranks of the data rather than their actual values to calculate the statistic. In particular, this test does not make any distribution assumption on the data. We perform fifteen replications in a simulation run where each replication number corresponds to a block across the models. Within each block, we rank the profit values of the models. For each block (simulation replication), we perform the ranking and then apply the test statistic of the Friedman Test provided by Conover (1980).

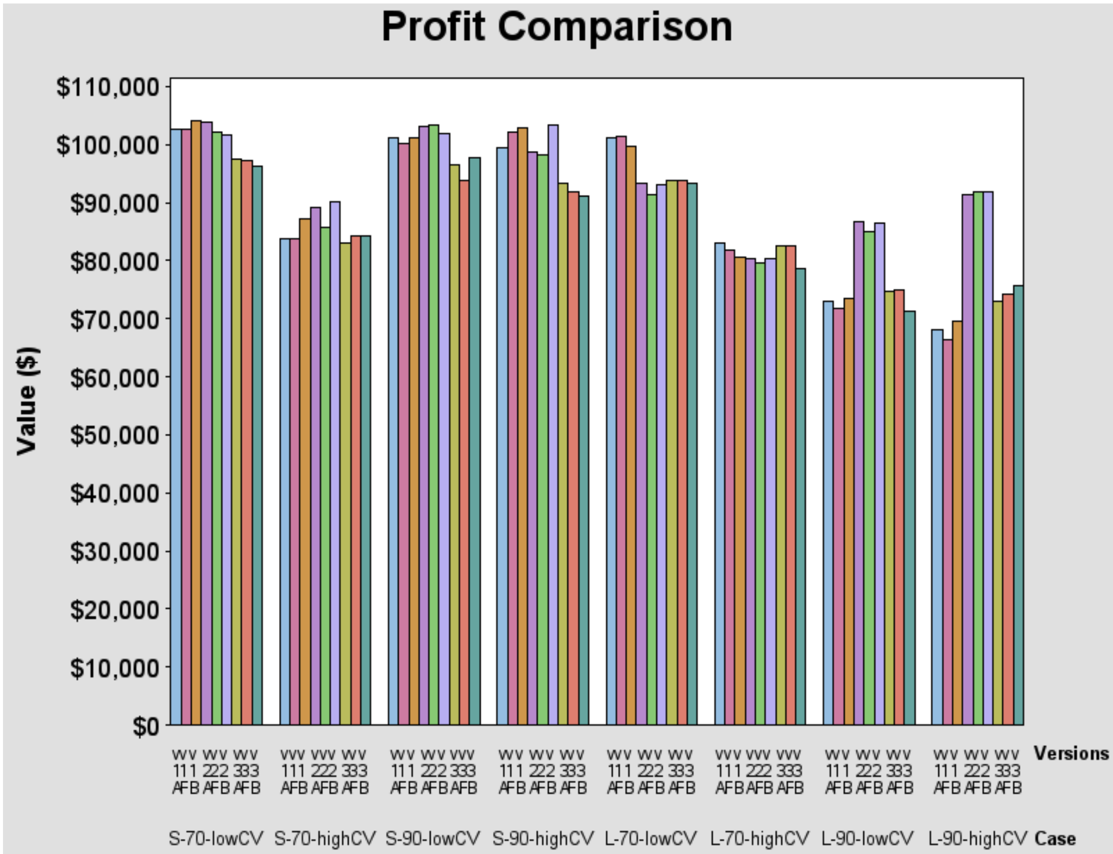


Figure 5-1: Profit Comparison of Models

The results from this test are shown in Table 5-2. Identical colors in the model column represent distributions that are not statistically different from each other at 95% confidence level. These results suggest that there is no single dominant model or selection method that consistently yields higher profit values than the others. However, in general, the same models with different selection procedures tend not to be statistically different at a 95% confidence level. It can be seen from Table 5-2 that in most cases, two of the three selection methods are statistically indistinguishable.

For example, for the S-90-lowCV case, for all individual models, two out of three selection methods result in no statistical difference. For all 70% utilization cases for any demand or length of failure, Version 1 performs best with any selection procedure except in the S-70-highCV case where Model 2 takes the lead. This suggests that, under low utilization, the models including only variables corresponding to that period (V1) perform better than models including variables related to the immediately preceding period (V3). This is intuitive; at low utilization the production system is likely to be able to process all work released to it in the same period, without carrying large amounts of WIP into future periods and maintaining low cycle times.

Table 5-2: The Friedman Test of Models

S-70- lowCV		S-70-highCV		S-90-lowCV		S-90-highCV	
Mean	**Model**	**Mean**	**Model**	**Mean**	**Model**	**Mean**	**Model**
103998	V1B	90067	V2B	103250	V2F	103446	V2B
103849	V2A	89198	V2A	103198	V2A	102819	V1B
102662	V1F	87287	V1B	101969	V2B	102144	V1F
102618	V1A	85683	V2F	101198	V1A	99380	V1A
102021	V2F	84366	V3B	101061	V1B	98776	V2A
101725	V2B	84349	V3F	100283	V1F	98338	V2F
97381	V3A	83793	V1A	97706	V3B	93248	V3A
97317	V3F	83779	V1F	96631	V3A	91896	V3F
96346	V3B	83065	V3A	93824	V3F	91230	V3B
L-70- lowCV		L-70-highCV		L-90-lowCV		L-90-highCV	
Mean	**Model**	**Mean**	**Model**	**Mean**	**Model**	**Mean**	**Model**
101312	V1F	82976	V1A	86597	V2A	91845	V2F
101052	V1A	82643	V3A	86441	V2B	91788	V2B
99572	V1B	82476	V3F	85074	V2F	91271	V2A
93728	V3F	81852	V1F	74872	V3F	75785	V3B
93714	V3A	80622	V1B	74669	V3A	74148	V3F
93228	V2A	80470	V2B	73406	V1B	72888	V3A
93223	V3B	80340	V2A	73014	V1A	69634	V1B
93038	V2B	79619	V2F	71844	V1F	68121	V1A
91459	V2F	78559	V3B	71358	V3B	66478	V1F

At high utilization cases, Model 2 is superior for all demand and failure cases. Since the cycle times are longer, variables that also include the information on the preceding period of that product becomes significant and result in higher profit production plans. This model includes the release of each individual product in the immediate prior period in the regression model. This can be expected since for high utilization, outputs of products are more likely to be affected by the releases of preceding periods; cycle time is long, and individual product released in the prior period may not have exited the resource. Hence output of a given product in a period is significantly affected both by the releases of that product in the current period (Model 1) and by preceding period releases (Model 2).

It is interesting to examine the cost breakdowns shown in Figure 5-2 to see where the differences in profit between models arise. In the vast majority of cases, as would be expected, the total revenue generated by all models is almost equal; the differences arise from the different ways in which the models choose to incur costs to satisfy demand. Material costs are almost constant across all models, since the models all generate approximately the same total output. In the long failure cases, the performance of all models deteriorates due to the higher level of variability created by the less frequent but longer machine failures. The cost structure of each model under different selection procedures is quite similar, consistent with the findings from the Freidman analysis.

In most cases, WIP cost is almost as much as all other costs combined. This suggests that the difference in profit between regression models is determined by which one holds less WIP than the others. Holding a large amount of WIP implies overestimation of capacity since the production plans obtained from LP models using CFs predict that the planned outputs can be obtained from the planned releases and WIPs. However, simulation results show that more products stay in the queue of the resource as WIP than planned WIP obtained from the production planning model. As the relationship between output and variables of release and WIP is established by the product based CF $X_{kgt} = f_k\left(W_{k,\mathrm{g},t-1}, R_{kgt}\right)$, we conclude that the fit may not represent the capabilities of the production system well, compared to the load based CF $X_{kt} = f_k\left(W_{k,t-1} + R_{kt}\right)$ presented in CHAPTER 8. We observe for other cost elements that the backlog and inventory cost trade off against each other; when there is more inventory, it means less backlog cost and vice versa.

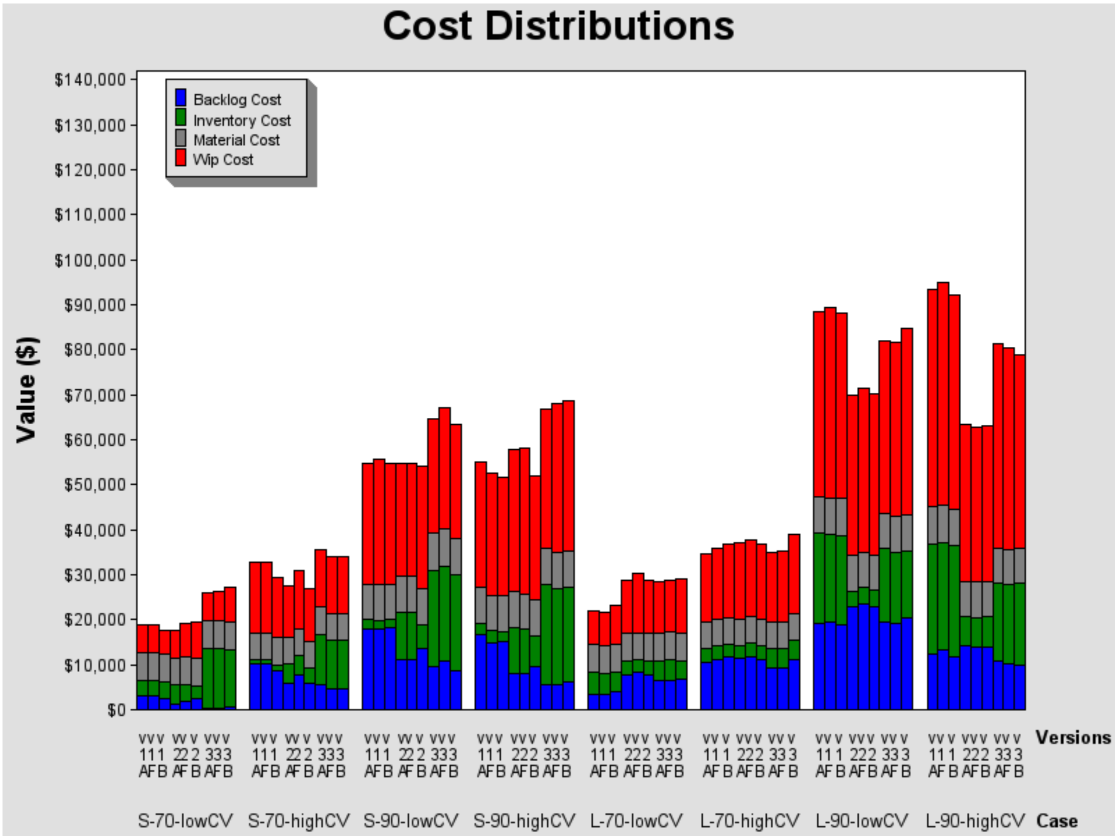


Figure 5-2: Cost Distribution of Models

In order to further explore the differences between the models we calculated the adjusted R^2 values for all models on all machines. While these results are not presented in detail, in almost all cases very high adjusted R^2 values were obtained, despite the quite substantial differences in profit yielded by the different models. Different selection procedures result in different numbers and choices of variables; but we show that in the profit comparison, for most of the cases, different selection methods for the same regression model are not statistically different from each other. This suggests that the adjusted R^2 value may not be a good basis for comparison between different regression models. Albey et al. (2010) show that least-squares regression may not be satisfactory since there might be strong correlation between residuals and WIP levels.

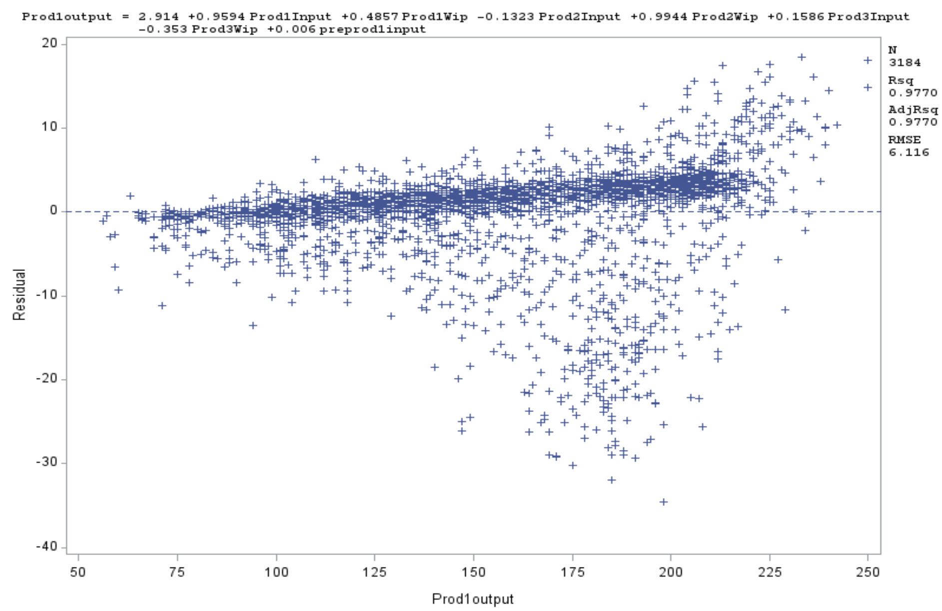


Figure 5-3: Residual Plot of fitting of Machine 7 with Regression Model V2A

The Figure 5-3 shows the residual plot for the independent variable of output of product 1 using the regression model V2A. There is a trend in the spread of residuals that as the value of the output increases, the residuals become mostly accumulated above zero reference level. We can also observe the effect of machine failures from the residual values which can go down to negative 35. We can deduct that this regression do not satisfy the requirement of residuals being normally distributed with mean zero and standard deviation one.

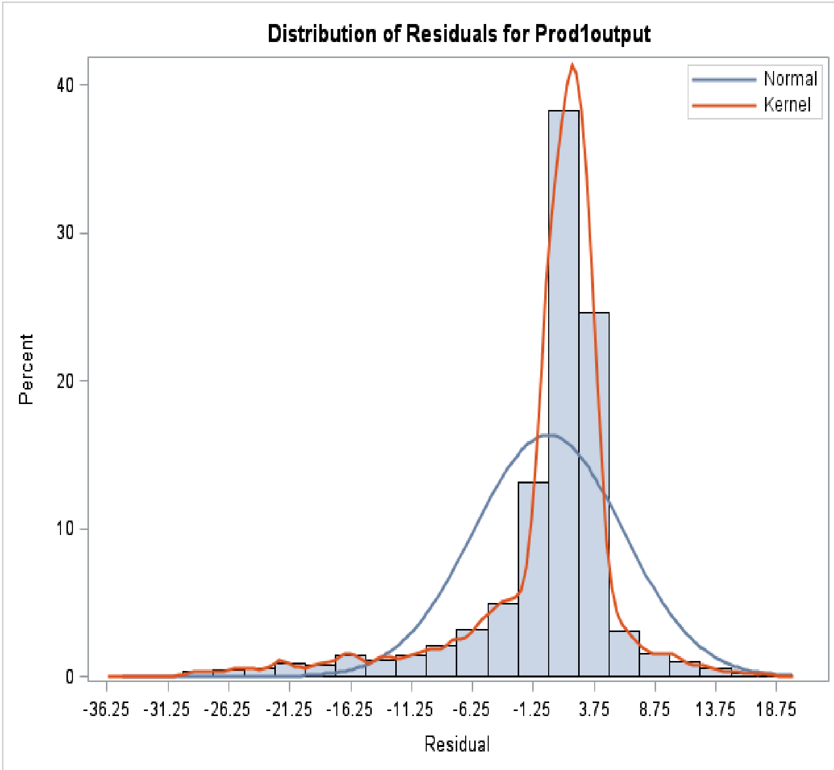


Figure 5-4: Normality plot of residuals

We observe from Figure 5-3 that residuals are not normally distributed and Figure 5-4 confirms this observation that the histogram is off the normal distribution line which is depicted with blue. Those findings also show why R^2 is not a good indication of fit since the regression model has flaws in considering the distribution of the residuals.

Next we present our search for an improved CF fitting algorithm and present its results.

CHAPTER 6. SEARCH FOR IMPROVED CF FITTING ALGORITHMS

In the previous Chapter, we presented an estimation procedure for a product based CF of the form $X_{kgt} = f_k \left(W_{k,g,t-1}, R_{igt}\right)$, where the CFs are developed for individual products. In this chapter we present procedures for estimating the load based functional form $X_{kt} = f_k \left(W_{k,t-1} + R_{kt}\right)$ in which we estimate aggregate CFs for each machine using the total resource load of that machine, and the planning model allocates resource capacity to the individual operations of each product. We use total resource load since the processing time at a machine is the same for all operations that take place on that machine. If the processing time of a machine differed among product types and operations, we could calculate the resource load in units of time. In this chapter, we present a simulation optimization algorithm used to optimize our estimates of load based CFs. In the next chapter, we develop heuristics to search for CF estimates that yield production plans with improved profitability. Throughout this chapter, all algorithms address the issue of estimating load based CFs of the form $X_{kt} = f_k \left(W_{k,t-1} + R_{kt}\right)$.

The basis for the CF-based production planning model in this chapter is the Allocated Clearing Function (ACF) formulation of Asmundsson et al. (2009) and Asmundsson et al. (2006). The LP formulation of this model differs from the MLR LP formulation presented in the previous chapter in its capacity constraints. The objective function and the WIP balance and inventory balance constraints stay the same. In the ACF model, we estimate an aggregate CF for the entire resource and allocate the output it generates to operations, whereas in the previous chapter, we estimate CFs for individual products. The CF used in this model is based on the planned workload, defined as the sum of the initial WIP and the releases in that period, i.e. $X_{kt} = f_k \left(W_{k,t-1} + R_{kt}\right)$. The estimation of a Clearing Function of this form is explained in Section 3.3.2.

We note that all operations for all products that take place on a specific machine require the same processing time. This assumption is not restrictive, as shown in Asmundsson et al. (2009), since both output and WIP at a station can be defined in terms of processing time rather than units of output when processing times differ between operations. We also assume that semi-finished inventory is not held within the production process, but that the output of each operation immediately becomes available to the next operation in the product's routing. External releases are made only into the first operation on each product's routing. Decision variables, constraints and parameters are as follows:

Indices:

g : product index.

k : machine index.

l : operation index.

t : period index.

Decision Variables:

$Y_{g,t,l}$: number of units of product g that have completed processing at the l'th operation in period t.

$X_{g,t,l}$: release quantity of product g in period t at operation l. This is equal, by definition, to the output of the previous operation in the product's routing in that period. Hence $X_{g,t,l} = Y_{g,t,l-1}$, where operation l-1 is the immediate predecessor of operation l in the routing of product g. We assume that each operation will have one predecessor and one successor, although our formulations are easily extended to incorporate multiple predecessors and successors.

Y_{gt} : output quantity of product g in period t. $Y_{gt} = Y_{g,t,l'}$ where l' is the last operation for product g.

X_{gt} : quantity of product g released into the first operation in the line in period t.

$W_{g,t,l}$: WIP of product g at operation l at the end of period t.

I_{gt} : units of product g in finished goods inventory at the end of period t.

B_{gt} : units of product g backlogged at the end of period t.

$Z^k_{g,t,l}$: Fraction of capacity at machine k allocated to operation l of product type g in period t.

Parameters:

h_{gt} : unit inventory holding cost for product g in period t.

b_{gt} : unit backlogging cost for product *g* in period *t*.

ω_{gt} : unit WIP cost for product *g* in period *t*.

D_{gt} : demand for product *g* during period *t*.

C(k) : index set of the line segments used to approximate the CF at machine *k*.

μ_k^s : Intercept of the *s*'th clearing function segment describing machine *k*.

β_k^s : Slope of *s*'th clearing function segment for machine *k*.

The formulation can now be given as follows:

Objective function:

$$min\ \sum_{g\in G}\ \sum_{t=1}^{T}\sum_{\mathrm{k=1}}^{\mathrm{K}}\omega_{gtk}\,W_{gt}(k)+\sum_{g\in G}\ \sum_{t=1}^{T}h_{gt}\,I_{gt}+\sum_{g\in G}\ \sum_{t=1}^{T}b_{gt}\,B_{gt}$$

Subject to:

WIP Balance:

$$W_{g,t,l}=\ W_{g,t-1,l}+X_{g,t,l}-\ Y_{g,t,l}\qquad g\in G\ \ ,t=1,\ldots,T\ ,l\ \in L$$

Inventory Balance:

$$Y_{gt}+I_{g,t-1}-B_{g,t-1}-I_{gt}+B_{gt}=d_{gt}\ \ g\in G\ \ ,t=1,\ldots,T$$

Clearing Function (Capacity) constraint:

$$Y_{g,t,l}\leq Z_{g,t,l}^{k}\mu_k^s+\beta_k^s\left(X_{g,t,l}+W_{g,t-1,l}\right)\ g\in G,t=1,\ldots,T\ ,l\ \in L,k\ \in K(l),s\in C(k)$$

Output Allocation Constraint:

$$\sum_{g\in G,l\in L(k)} Z_{g,t,l}^{k} = 1 \;\; t = 1, \ldots, T\,, k \in K$$

Variable Nonnegativity:

$W_{g,t,l}, Y_{g,t,l}\,, I_{gt}, B_{gt}, X_{g,t,l}, Z_{g,t,l}^{k} \geq 0$ *for* $g \in G \;\; , t = 1, \ldots, T\,, k \in K, l \in L$ We consider the sum of releases of the product and the work in process (WIP) at the beginning of the period as the load of the system at that period. In this Load Based Model, the clearing function relates the expected output in a planning period to the planned load in that period of the system.

This formulation uses the $Z_{g,t,l}$ variables to allocate capacity among operations. These variables scale up the WIP of each operation of product g at the beginning of period t to estimate the total WIP (or workload) of all products in that period to obtain an upper bound on the output of operation of product g at machine k. The detailed development of this formulation is given in Asmundsson et al. (2009), to which the reader is referred for further details.

In the following sections, we present the simulation optimization algorithm, specifically Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm, and in the next chapter three search heuristics to improve the fitting of these load based clearing functions. Our algorithms aim to determine a clearing function fit that will maximize the expected realized profit when the resulting clearing functions are used to plan production. These algorithms begin from an initial fit that is the set of CF parameters obtained from the least-squares fit of the load based CF form $X_{kt} = f_k \left(W_{k,t-1} + R_{kt}\right)$ given in Section 3.3.

We will consider the CF of the system in two ways. In the first, we assume that the CF for each machine does not change over time as in the SPSA All and CF All procedures shown in Figure 6-1 below. In our second approach we develop CFs for each machine in each period as in the SPSA Each, CF Each and CF KW procedures. We want to observe whether having the same CF for each machine in all periods affects the performance of the CF based planning models as demand varies over time. One common CF across all periods as in SPSA All and CF All can also be considered as an expectation over CFs of all periods as in SPSA Each and CF Each.

To serve as a benchmark, one of our simulation optimization approaches will focus on optimizing the release schedule directly to improve expected profit, without any use of a clearing function. This latter approach allows the release schedule to be optimized for each given demand pattern, but requires lengthy simulation runs as part of the planning process. In contrast, our other procedures use simulation optimization to obtain a clearing function which can then be used in an LP model to generate release schedules without the need for simulation during the planning process. The advantage of our CF based procedures is that the simulation optimization required to fit the CFs is conducted offline, and is not part of the routine planning process.

The algorithms are summarized in Figure 6-1 below. We will introduce the heuristic algorithms in CHAPTER 7.

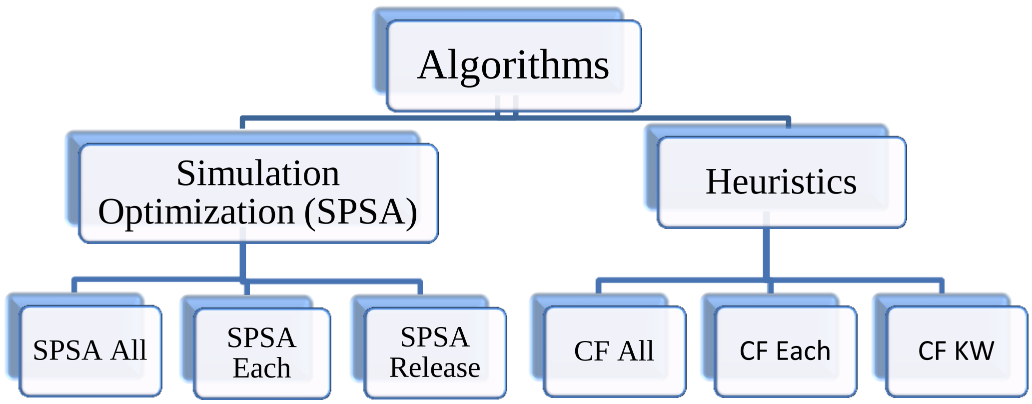


Figure 6-1: Summary of Algorithms

We first discuss the generic Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm that forms the basis of all our different approaches. We then discuss its specific customization to each of our procedures: fitting a stationary CF for all periods (SPSA All), customized CFs for each period (SPSA Each), and the direct optimization of releases without CF (SPSA Releases).

6.1. Simulation Optimization Algorithm

The SPSA algorithm is a Stochastic Approximation method for continuous state space. This procedure implements a gradient based algorithm to find the values of the decision variable vector θ yielding the gradient of the objective function

$\nabla J(\theta) = 0$. At each iteration n, the procedure updates the decision variables $\theta_{n+1} = \theta_n + a_n \nabla J_n$ based on the current gradient estimate ∇J_n with respect to θ.

We first present the general scheme of the SPSA algorithm and later define the specific decision variables used in each SPSA approach.

Decision variables:

θ : Decision variables.

Θ : Decision variables space $\theta \in \Theta$.

Objective function:

$L(\theta, \omega)$: Profit realization in simulation run ω with decision variables θ

$J(\theta)$: Expected profit realization over all simulation runs ω $J(\theta) = E_{\omega}\,[L(\theta, \omega)]$

Our objective is to maximize the expected realized profit *J(θ)* by updating variables using the gradient estimation procedure of Simultaneous Perturbation Stochastic Approximation (SPSA). The procedure is explained in following section.

6.1.1. Gradient Estimation and Updating $\boldsymbol{\theta}$

Let $\Delta_{n=}[\widehat{\Delta_n^1} \; \ldots . \widehat{\Delta_n^p}]^T$ denote a *p*-dimensional random perturbation vector such that each component of Δ_n is independently generated from a zero mean distribution at iteration *n*. The dimension p depends on the dimension of the decision variable θ

which differs among the SPSA approaches presented later in this chapter. The gradient at iteration *n* is estimated as

$$\hat{\nabla} J_{in} = \frac{\hat{J}(\theta_n + c_n\Delta_n) - \hat{J}(\theta_n - c_n\Delta_n)}{2c_n\Delta_{in}} \quad i = 1, \dots, p$$

Instead of applying component by component perturbations to θ_n and running simulations for each component, as in the Finite Difference method (Fu, 1994), simultaneous perturbations for all components are generated independently from a zero mean distribution for the p dimensional Δ_n vector. This Δ_n vector basically defines how far we want to move from the current solution θ_n to calculate the gradient estimation. Spall (1998) gives ±1 Bernoulli distribution with probability of 0.5 for each outcome ±1 as an example of such distribution. He also notes that a uniform distribution with mean zero can also be used by dividing the distribution into two pieces in order to exclude values near zero, since the gradient estimate goes to infinity when Δ_n approaches zero. Recall that p is the number of components in the θ vector. The component separated form of the gradient estimate of the general equation of $\hat{\nabla} J_n$ for iteration *n* is given by

$$\begin{bmatrix} \hat{\nabla} J_{1n} \\ \hat{\nabla} J_{2n} \\ \vdots \\ \hat{\nabla} J_{pn} \end{bmatrix} = \frac{\hat{J}(\theta_n + c_n\Delta_n) - \hat{J}(\theta_n - c_n\Delta_n)}{2c_n} \begin{bmatrix} 1/\Delta_n^1 \\ 1/\Delta_n^2 \\ \vdots \\ 1/\Delta_n^p \end{bmatrix}$$

After we obtain our gradient estimate, we obtain the decision variable vector θ_{n+1} at iteration *n+1* as

$$\theta_{n+1} = \theta_n + a_n \nabla J_n,$$

where a_n is the gain sequence. The component-wise updating at iteration *n* will take the form

$$\theta_{i,n+1} = (\theta_{in} + a_n \nabla J_{in}), \quad i = 1, \dots, p.$$

The SPSA gain sequences a_n and c_n at each iteration *n* are calculated as

$$a_n = a/(A+n)^{\alpha} ,$$

$$c_n = c/n^{\gamma} .$$

These gain sequences a_n and c_n are updated at each iteration and their values decrease as the iterations proceed. The gain sequences, a_n and c_n, constitute non-negative multipliers of the *a*, *c*, and *A* parameters specified at the beginning of the procedure. The values of the α and γ parameters are also specified at the beginning of the procedure. The choices of the *a*, *c*, *A*, α and γ parameters are important in determining the convergence behavior of the SPSA procedure. We will experiment with different values to identify good choices of these coefficients. We now present the iteration of the algorithm.

6.1.2. SPSA Algorithm

In the previous section, we explain how the gradient estimation is obtained using SPSA and the decision variables θ_n are updated using this gradient estimate. We now present the algorithm in pseudocode. Note that at the first iteration, the initial decision variable vector θ_1 is obtained from the multiple linear regression fitting procedure for the load based CFs explained in Section 3.3.2.

Step 1: *n*=1

Step 2: Use θ_n to run simulation to obtain $J(\theta_n)$.

Step 3: Calculate $\widehat{\nabla} J_n$ using the random perturbation vector.

$\Delta_{n=}[\widehat{\Delta_n^1} \; \ldots . \widehat{\Delta_n^p}]^T$ generated at iteration *n*. Compute the gradient estimate

$$\hat{\nabla} J_{in} = \frac{\hat{J}(\theta_n + c_n\Delta_n) - \hat{J}(\theta_n - c_n\Delta_n)}{2c_n\Delta_{in}} \; i = 1, \ldots, p.$$

Step 4: Compute the new solution $\theta_{n+1} = \theta_n + a_n \nabla J_n$ and the corresponding objective function value $J(\theta_{n+1})$.

For a chosen tolerance value $0 \leq \varphi \leq 1$

If $\left(\frac{J(\theta_{n+1})}{J(\theta_n)}\right) \leq \varphi$ then recover θ_n and apply a new, independent random perturbation vector starting from Step 3.

Else use θ_{n+1} for the next iteration.

Step 5: if $\left(J(\theta_{n+1}) - J(\theta_n)\right) < \delta$ stop and report $max_{\theta \in \Theta} J(\theta)$, where δ is a small number.

Else set $n=n+1$, and return to Step 2.

In our SPSA procedures, we perform 100 iterations. Step 5 of the algorithm implies that it is possible to stop the algorithm if there is no significant improvement in the objective function value for a given number of iterations. In the original SPSA algorithm of Spall, updating proceeds regardless of whether the objective value is improved or not. In our version, the user defined tolerance parameter φ allows the objective function value $J(\theta_{n+1})$ to decrease by some fraction of the previous iteration's objective value $J(\theta_n)$. If a larger decrease is observed a new θ_{n+1} vector is obtained by applying a new independent random perturbation to the current θ_n vector. The goal of this condition is to keep the decision vector θ in a region that is more likely to contain improved solutions. Note that we do not require the objective values to improve at each iteration, so that even if there is a small decrease in the objective value at one iteration, there may still be a good chance of obtaining a better objective value in future iterations. The φ parameter specifies the maximum decrease in the objective function value permitted at any iteration.

In this thesis we use SPSA procedure to improve our CF fits to obtain higher expected profit from the simulation. We will treat this simulation optimization algorithm a bit differently since the decision variables θ in the simulation optimization are actually parameters used in the LP. Furthermore, the input to the simulation is the release quantities obtained from the LP. In order to lay out our algorithm, we will consider the solution of the LP and the processing of the release quantities thus obtained for simulation input as intermediate steps between obtaining θ and running the simulation to obtain the expected profit realization $J(\theta)$. Hence after we update our θ, we will solve the LP to obtain release quantities $R(\theta)$, then run the simulation to estimate $J(\theta)$ which is still a function of θ since the parameters and solution of the LP change as we update θ. Our approach is illustrated in Figure 6-2.

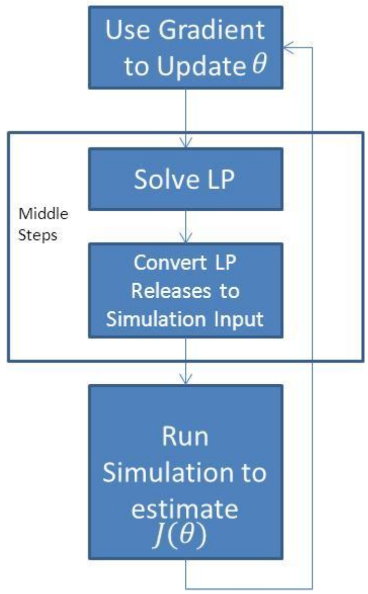


Figure 6-2: Simulation Optimization Iterative Algorithm

In the next section, we explain our two different approaches to fitting CFs using SPSA. We will then describe the SPSA algorithm used to optimize the release schedule directly without use of an intermediate planning model.

6.2. SPSA Approaches for fitting CFs

Recall that our main goal is to optimize CF fit parameters such that the resulting planning models result in the maximum expected profit. Thus for our first approach, SPSA All, we assume a CF that remains constant across time periods. Hence the decision variables optimized by SPSA are the CF parameters μ_k^s and β_k^s for each machine. In the SPSA Each procedure we fit a separate CF for each machine in each period. Thus the decision variables in the SPSA Each procedure are the CF parameters for each machine in each period, μ_{kt}^{s} and β_{kt}^{s}.

6.2.1. SPSA ALL

In the previous sections, we noted that our decision variables are the slope and intercept parameters of the piecewise linearized clearing functions for each machine. Initial values of these variables are obtained by the fitting procedure given in Section 3.3.2. Thus the decision variables are $= \{\mu_{k}^{s}\ ,\beta_{k}^{s}\}$ for each linear segment s; recall from Section 3.3.2 that we have two segments for each machine k for which the load based regression model is applied to estimate slope and intercept values. The third segment defines the capacity limit and is not included in the decision variables. Thus, we have two slopes and two intercept parameters for each machine that constitute our decision set. We have eleven machines in our fab model, so the total number of decision variables is 4 x 11 = 44.

Recall that our planning horizon is 26 periods. In this approach, we will consider one CF for each machine for all periods. Thus, the CF is treated as a property of the machine and remains unchanged over the planning horizon.

In the previous section, we note that the choice of parameter values used in the SPSA algorithm is important for successful convergence to a good, hopefully globally optimal solution. Spall (1998) provides guidelines for choosing the values of the coefficients, but also points out that tuning of the coefficients by experiment is required, which is the approach that we adopt while tuning our coefficients.

6.2.1.1. Coefficient Determination for SPSA All

We begin by describing the generation of the random perturbation vector Δ_n. We use a Bernoulli distribution with probability of 0.5 for each outcome. We choose values of ±0.1 and ±0.001 for the outcomes of the intercept and slope perturbation vectors respectively. These choices are based on the need to avoid excessively large or small perturbations in the decision variables which will be used in gradient estimation. If the perturbation is large, the gradient estimate may lose its reliability when the two points from which the gradient is estimated are far apart. On the other hand, if the two points used for gradient estimation are very close to each other, random noise may affect the gradient estimate. It is hard to judge *a priori* what values are appropriate for the perturbation vector, so we experimented with several choices of outcome values and obtained better results with the following values. For the initial intercept estimates, half-widths of a 95% confidence interval provided by the regression procedure in Section 3.3.2 for all machines range between 0.2 and 1.8. For the initial slope estimates, the 95% CI half-widths range from 0.003 to 0.011. These ranges also guide us in how much we want to deviate from the current decision vector for gradient estimation; we aim to move less than one half-width to explore more points in the region of the confidence intervals.

We now discuss our choices of non-negative values for the initial values of the control parameters a, c, A, α and γ of the SPSA algorithm. Recall that at each iteration n, the gain sequences a_n and c_n are calculated as

$a_n = a/(A+n)^{\alpha}$,

$c_n = c/n^{\gamma}$.

The values of α and γ are selected first, and those of the other parameters are adjusted by experiment. Spall (1998) suggests that effective values for α and γ are 0.602 and 0.101, respectively. They base these proposed values on their experiments and those of other researchers. We adopted the same values for α and γ in our experiments. Spall also provides guidelines for the values of A and c, but in our experiments these values did not work well, and we tuned those parameters by experimenting with several values. First, we set A and c equal to 1. The coefficient c affects the gradient estimation as we calculate the vectors as such $\theta_n - c_n\Delta_n$ and $\theta_n + c_n\Delta_n$. We prefer to control our finite difference points by setting Δ_n as explained above and c = 1. When we define the gain sequence coefficients $a_n = a/(A+n)^{\alpha}$, we set α and A as explained above. The choice of a is important, since it impacts the new decision vector $\theta_{n+1} = \theta_n + a_n \nabla J_n$ by scaling down J_n . We impose this scaling on a such that the difference $\theta_{n+1} - \theta_n$ will be in the range of about 0.001 and 0.1 for absolute changes in slope and intercept variables at the first iterations, respectively. We experimented with several a values, resulting in different absolute differences in decision variables, but found that keeping the absolute difference approximately in the range of our perturbation vector Δ provided superior results. We choose a = 10^{-5} for the intercept variables and a=10^{-9} for slope variables. Recall that the gain sequence $a_n = a/(A+n)^{\alpha}$ is adjusted by the iteration number n, implying that the absolute difference $|\theta_{n+1} - \theta_n|$ will decrease at each iteration. Note that the Bernoulli outcomes for μ_k^s and β_k^s variables are, 0.1 and 0.001 respectively, which result in different magnitudes of gradient estimates $\hat{\nabla} J_i, i \in \{\mu_k^s, \beta_k^s\}$ since the magnitudes of Δ_i values are different for the intercept and slope related variables. Therefore different coefficient values a in the gain sequence a_n for the μ_k^s and β_k^s variables are necessary since we update the current solution as $\theta_{i,n+1} = (\theta_{in} + a_n \nabla J_{in})$.

In Step 4 of the SPSA algorithm, our φ parameter determines how much the objective value is permitted to decrease at any iteration. If the objective function decreases by more than this tolerance, we recover the θ_n vector and calculate a new gradient estimate with a new randomly created perturbation vector Δ_n. In the SPSA All approach, and also in the other approaches that will be introduced in the next sections, we set $\varphi = 0.99$. Even though this seems to be very restrictive, it shows good performance by confining θ to a region where the objective value is pushed to improve. We also experimented with several φ values, such as 0.97, 0.95 but their performances were not as good as that obtained using φ =0.99.

A summary of the SPSA control parameter settings used in the experiments is given in Table 6-1.

Table 6-1: Control Parameter Values for SPSA All

Coefficients	Intercept	Slope
Δ	±0.1	±000.1
c	1	1
γ	0.101	0.101
α	0.602	0.602
A	1	1
a	10^{-5}	10^{-9}
φ	0.99	

The experimental results will be presented in the later sections after we introduce our other SPSA approaches.

6.2.2. SPSA Each

In the previous approach, the CF for each machine is constant over all periods. In the SPSA Each approach, we estimate different CFs for each period for each machine to see if we can achieve better expected profit by allowing the CFs to change over time. Thus our decision variables are $\{\mu_{kt}^{s}, \beta_{kt}^{s}\}$ in the SPSA Each approach with the addition of the period index *t*. This results in 11 machines x 2 Segments x 2 fitting variables x 26 Periods = 1144 decision variables to optimize with SPSA. The steps of the SPSA algorithm are as in the SPSA All approach.

We will use the control parameter values given in Table 6-1 and provide the experimental results and comparison of this approach to the previous one in later sections.

6.3. SPSA Release

In the two previous approaches, SPSA All and SPSA Each, we are optimizing the CF fit variables using the SPSA algorithm. In our final approach, we want to optimize the release variables directly without using an intermediate planning model. In the previous approaches, we optimize the fit variables, but essentially every time we change the fit, we are getting a new set of releases. Then, we execute the plan in the simulation. SPSA Release will provide a more direct approach for our purpose of achieving “best” profit values, since instead of solving LP with new fits at each iteration, we start with an initial set of releases and search over the release space to find better performing release schedules. The basic iterative scheme is given in Figure 6-3.

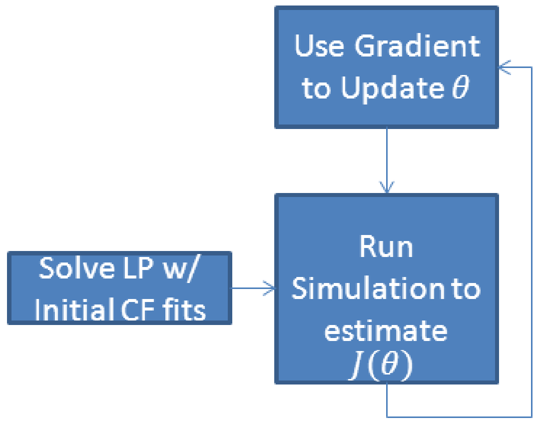


Figure 6-3: SPSA Release Iteration

In this approach, we redefine $\theta = \{X_{gt}\}$, that is the quantity X_{gt} of product g released in period t. The steps of SPSA iteration are as given in Section 6.1.2.

6.3.1. Coefficient Determination for SPSA Release

We have a different random perturbation vector Δ_n in the SPSA Release approach since the magnitudes of the decision variables being optimized are different from those in SPSA All and SPSA Each. We again use a Bernoulli distribution with probability of 0.5 for each outcome. We choose ±0.5 for the outcomes of the random perturbation vector for each product in each period. We aim to move 0.5 units in release decision space in order to avoid large jumps in the objective function $J(\theta_n)$ and likewise large values for the gradient estimate ∇J_n. We use the same c, A, α and γ values as in the previous approaches. The gain sequence parameter a is again adjusted by experimentation in order to obtain changes in releases at an iteration close to the perturbation vector value of 0.5. We again aim to explore the region close to previously visited points. The values of the control parameters used are given in Table 6-2. We use the same perturbation vector value and coefficient values for the release variables of all products.

Table 6-2: Control Parameter Values for SPSA Release

Coefficients	Release
Δ	±0.5
c	1
γ	0.101
α	0.602
A	1
a	10^{-4}
φ	0.99

We have three products and our planning horizon is 26 periods. The total number of decision variables to optimize in this approach is 78 (3 products x 26 periods). Note that, this number is larger than the number of decision variables in the SPSA All approach (44) but significantly lower than that for the SPSA Each approach (1144). In the next section, we will present our experimental results and discuss our findings.

6.4. Results of SPSA Approaches

The testbed that we use is that used in CHAPTER 5 for the Multiple Linear Regression (MLR) experimental design with the same demands and failure parameters in order to make a fair comparison between the two CF forms $X_{kgt} = f_k\left(W_{k,g,t-1}, R_{kgt}\right)$ and $X_{kt} = f_k\left(W_{k,t-1} + R_{kt}\right)$ that we will present in CHAPTER 8.

In all approaches, we use the same SPSA algorithm; therefore we perform 100 iterations for all approaches. It is possible to terminate the algorithm using the condition $\left(J(\theta_{n+1}) - J(\theta_n)\right) < \delta$ for a defined small δ value but we continue iterations to observe any possible improvements in profit realization. For all SPSA approaches, we perform 15 simulation replications for each of the two gradient estimation simulation runs $\hat{J}(\theta_n + c_n\Delta_n)$ and $\hat{J}(\theta_n - c_n\Delta_n)$ and for the simulation run for evaluating the objective value with the updated decision variable $J(\theta_{n+1})$ at all iterations. We use the average of 15 replications in a simulation run for calculating the gradient estimate and reporting the expected profit value.

We start our analysis by comparing profit values in Figure 6-4. We pick the best achieved profit realization value among the 100 iterations for all SPSA approaches. Later in this section, we will present Profit vs. Iteration plots for sample cases and discuss the behavior of the SPSA algorithm.

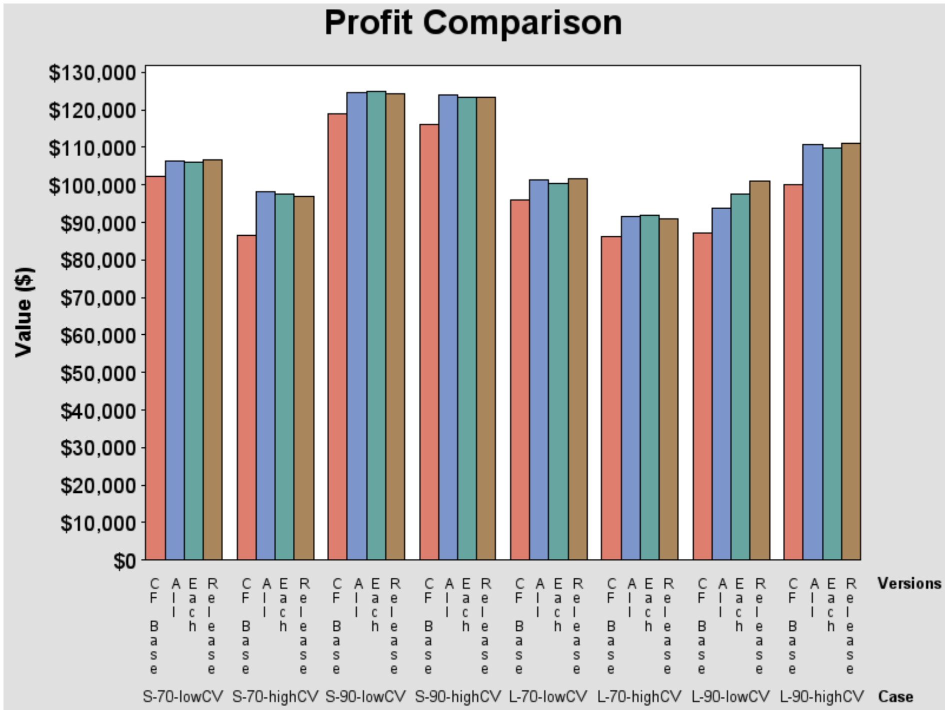


Figure 6-4: SPSA Approaches Profit Comparison

In Figure 6-4, All, Each and Release stand for SPSA All, SPSA Each and SPSA Release approaches, respectively. CF Base stands for the expected profit value obtained by executing the release solution from an LP model using the initial CF fits of the Load Based CF form, which are the starting values of the decision variables for SPSA All and SPSA Each. The CF Base model uses the CF fits obtained from the regression of the Load Based CF form directly without any perturbations on the estimates. Since our input to the simulation is the release schedule obtained from LP, all SPSA approaches result in the same input to the simulation at the first iteration. This also implies the release solution of CF Base defines the starting release variables for the SPSA Release approach. In other words, the starting point of all these approaches is derived from the same LP solution, and they aim to improve the solution by optimizing either the CF fit variables or the release variables.

Figure 6-4 shows that all the SPSA approaches achieve improvement over CF Base for all cases. For some cases, such as S-70-highCV and L-70-highCV, the improvement is more prominent compared to, for example, the S-70-lowCV case. All the SPSA approaches seem to achieve more or less the same expected profit except in the L-90-lowCV case, where SPSA Release is clearly the leading approach.

Note that the revenues generated by the SPSA approaches and CF Base are very close to each other. This is intuitive since all approaches try to generate cumulative output equal to the cumulative demand by the end of the planning horizon, and revenue is calculated as the product of output and revenue per unit. In order to show how much improvement is achieved, we will focus on costs (inventory, WIP and backlog costs) except for material cost which is more or less constant among the algorithms since we release about same amount of material over the planning horizon to satisfy the total demand. Figure 6-5 shows the cost distribution obtained by the different SPSA approaches.

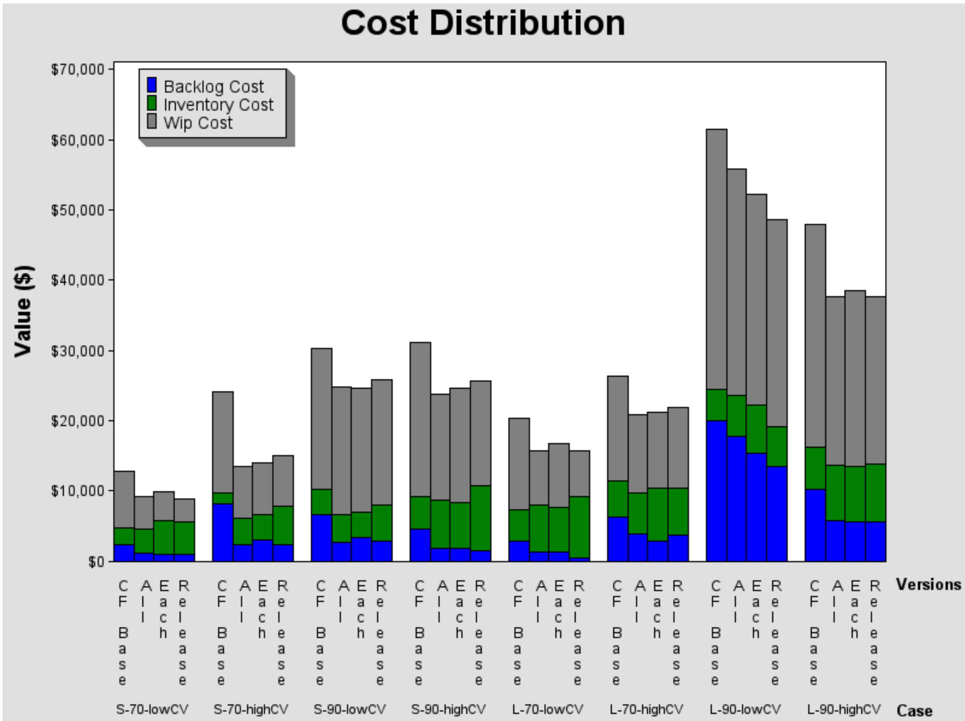


Figure 6-5: Cost Distribution of SPSA Approaches

We observe that in all cases CF Base incurs higher backlog cost than the SPSA approaches. That implies that capacity is overestimated since output does not emerge in time to satisfy the demand. Recall that since our objective function is maximizing profit, the LP will not favor high backlogs. The simulation results also indicate discrepancies between the realized and predicted output under the CF Base model. In order to quantify the cost reductions obtained by the SPSA approaches, we calculate the relative percent cost reduction taking the CF Base as the reference, shown in Table 6-3.

Table 6-3: Cost Reduction of SPSA Approaches

	Algorithms		
Cases	All	Each	Release
S-70-lowCV	29%	23%	30%
S-70-highCV	44%	42%	38%
S-90-lowCV	18%	19%	15%
S-90-highCV	24%	21%	18%
L-70-lowCV	22%	17%	23%
L-70-highCV	21%	20%	17%
L-90-lowCV	9%	15%	21%
L-90-highCV	21%	19%	21%

From Table 6-3, we observe that all SPSA approaches achieve significant cost savings, ranging from 9% in the L-90-lowCV case up to 44% in the S-70-highCV case. The SPSA approaches either improve the CF fit parameters or the release variables, and each approach yields on average almost 22% cost reduction over all cases. These cost reductions are the main drivers of the higher profit values shown in Figure 6-4.

Figure 6-4 and Table 6-3 indicate that there does not seem to be much difference in performance among the different SPSA approaches. In particular, the SPSA approaches that optimize CF fits provide superior or comparable cost reductions to the SPSA Release procedure in all cases except L-90-lowCV. In order to analyze those differences on a statistical basis, we present the Friedman test analysis with 95% confidence level in Table 6-4 below.

Table 6-4: The Friedman Test of SPSA Approaches

S-70- lowCV		S-70-highCV		S-90-lowCV		S-90-highCV	
Mean	**Model**	**Mean**	**Model**	**Mean**	**Model**	**Mean**	**Model**
106833	Release	98262	All	124902	Each	124079	All
106462	All	97437	Each	124623	All	123361	Each
106207	Each	96998	Release	124194	Release	123248	Release
102380	CF Base	86589	CF Base	118912	CF Base	115985	CF Base
L-70- lowCV		L-70-highCV		L-90-lowCV		L-90-highCV	
Mean	**Model**	**Mean**	**Model**	**Mean**	**Model**	**Mean**	**Model**
101803	Release	91897	Each	101122	Release	111162	Release
101240	All	91742	All	97632	Each	110708	All
100258	Each	90930	Release	93653	All	109848	Each
95965	CF Base	86120	CF Base	87286	CF Base	100019	CF Base

In Table 6-4, we observe that all the SPSA approaches yield statistically significant differences in profit values over those obtained by CF Base as expected. The comparison between the SPSA approaches shows that either the profit values they obtain are not statistically different from each other, as in the S-90-highCV case, or even though they are statistically different, they yield very similar profits (about 1% difference in profit). The case L-90-lowCV is an exception in that the differences in profit between the SPSA approaches are significant compared to the other cases. This analysis shows that performing 100 iterations for each SPSA approach and picking the best profit value over all iterations results in all methods obtaining very similar profit values.

Another observation from Table 6-4 is that even though there are small differences in profit, SPSA Release is the leading approach in four out of the eight cases. For the other cases, SPSA All is either the leading approach, or is statistically indistinguishable from SPSA Each. SPSA Release has an advantage over the other approaches in that it does not need to solve a LP at each iteration and it optimizes the decision variables which enter the simulation directly without any intermediate steps. In SPSA All and SPSA Each, we optimize CF fit variables and need to solve an LP to obtain the releases at each iteration. On the other hand, SPSA All and SPSA Each will provide the best performing CF fits that can be used in LP when we need to optimize the system given a new demand pattern from the same demand distribution without the need to perform SPSA iterations or any simulation. SPSA Release, however, has to re-optimize release variables using SPSA iterations when a new demand pattern occurs.

In the next section, we will examine the iterations of the SPSA approaches.

6.4.1. Iterations of SPSA Approaches

In order to illustrate the behavior of the SPSA algorithm, we present the profit vs. iterations plots for the S-70-highCV and L-90-highCV cases. We choose L-90-highCV case since the performance of SPSA approaches is most separated from each other in this case, as shown in Table 6-4. We include the S-70-highCV case since all SPSA approaches obtain the largest cost reduction (over 38%) seen in Table 7-1 in this case. The plots of the other cases are very similar to that of S-70-highCV.

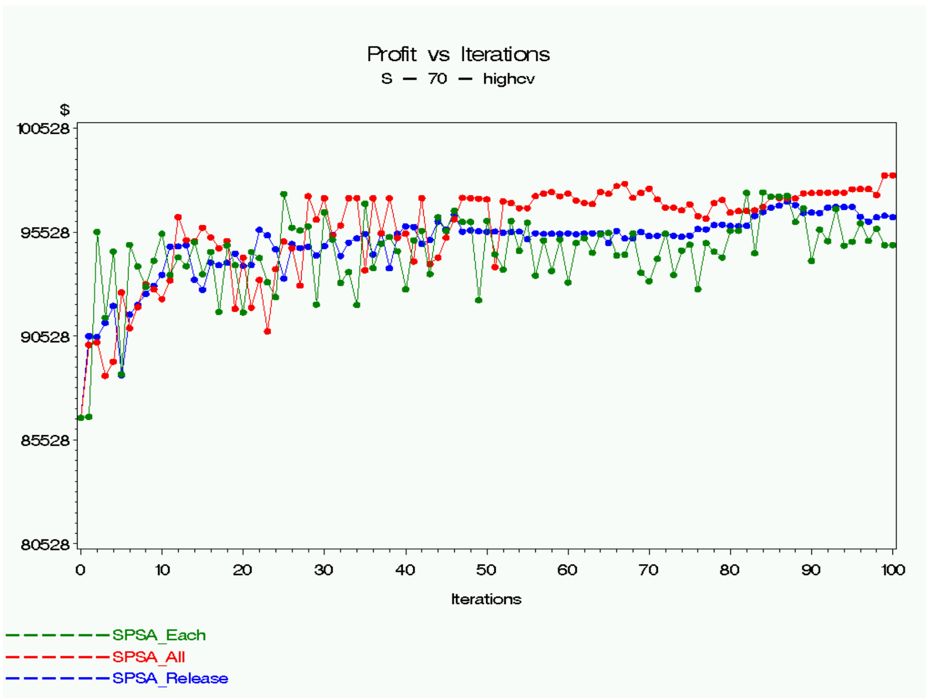


Figure 6-6: Profit vs. Iteration plot for S-70-highCV case

In Figure 6-6, iteration 0 represents the starting point of all SPSA approaches that corresponds to the solution obtained from the CF Base model; the slope and intercept parameters for SPSA All and SPSA Each, and the release schedule for SPSA Release. We observe more fluctuations between iterations for SPSA All and SPSA Each, shown in red and green, compared to SPSA Release shown in blue. We also observe this behavior in the other cases. Both SPSA All and SPSA Each use the intermediate step of solving an LP with new CF fit variables, which may play a role in this fluctuating behavior since, at each iteration the LP has a new feasible region and new extreme points partially defined by the CF constraint. SPSA Release results in smaller changes in releases relative to the SPSA All and SPSA Each since changes in CF parameters may lead to larger changes in releases whereas in SPSA Release changes in releases are directly controlled by the perturbation vector used. We also observe that the best performing iteration is not necessarily the last iteration. In SPSA Each, we obtain the best objective value at the 25th iteration, and at the 87th iteration in SPSA Release. We also see some large decreases in profit value from iteration to iteration. Recall that when the drop in the profit exceeds the tolerance specified by φ, we recover the previous solution and restart the procedure with a new, independently generated random perturbation vector.

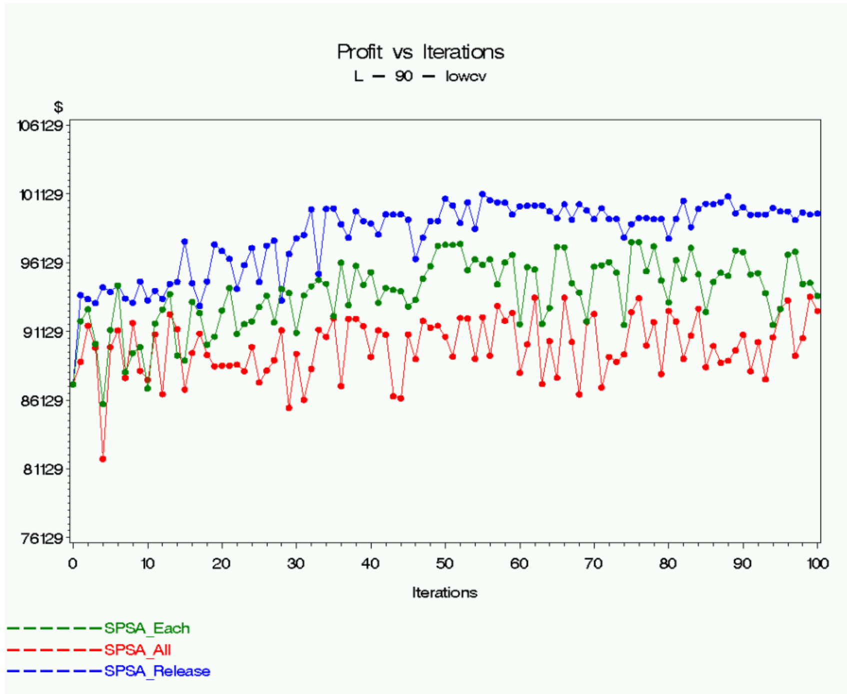


Figure 6-7: SPSA Approach Profit vs. Iteration plot of L-90-lowCV

Figure 6-7 shows the case where the differences in performance between the SPSA approaches are most prominent. In other cases, the lines cross each other and look similar to Figure 6-6. From Figure 6-7, we can see the dominance of SPSA Release over SPSA Each which is also dominant over SPSA All throughout the iterations. We observe the iterations of other cases resemble that of the S-70-highCV.

We now focus on the computation time of the SPSA approaches. Recall that at each iteration we need to perform three simulation runs - two for gradient estimate plus one simulation run for evaluating the new objective function value $J(\theta_{n+1})$ replicated 15 times. For SPSA All and SPSA Each, we need to solve three LPs at each iteration (two for gradient estimation, one to obtain the release schedule corresponding to the new decision variable vector θ_{n+1}) since our decision variables are the CF fit parameters and we need to solve an LP with the new CF fit to obtain the new release schedule. SPSA Release does not require LP solutions in its iterations.

Table 6-5 below shows the average time for one simulation run in the SIMtime column, the average time for one LP solve in the LPtime column and the average time for one iteration in the IterationTime column, all in units of second. Table 6-5 shows the S-70-highCV case time performances. The time average of a simulation run increases by about 10s in high utilization (90%) cases, since there are more entities created in the simulation model.

Table 6-5: Computational Time Performance of SPSA

Model	SIMtime(s)	LPtime(s)	IterationTime(s)	Profit($)
Release	144	N/A	433	96998
All	140	5	436	98262
Each	139	5	433	97364

We observe that the average time for each iteration is more or less the same across the models. SPSA Release has the advantage of not solving LPs that would save about 15s (5*3), but it seems simulation runs take a little bit longer in SPSA Release so that the time performances of all approaches become quite similar. When we convert the seconds into minutes, it means one iteration takes about 7min 12s, so for one case, it takes about 12hrs to finish 100 iterations. We choose to perform 100 iterations for each approach. As shown in Figure 6-6, SPSA All finds its best performing iteration at 25th iteration but this is not predicted beforehand.

In order to avoid time consuming simulation runs to estimate gradient and reduce total computation time of potential many iterations, we develop a number of heuristics presented in the next chapter.

CHAPTER 7. METHODOLOGY OF SEARCH HEURISTIC ALGORITHMS

In the previous chapter, we laid out the steps of the simulation optimization algorithm and how the iterations proceed. We presented the results of SPSA approaches and discuss the computational effort in iterations. In this chapter, we propose a search heuristic to update our decision variables θ without using simulation runs for gradient estimation. Instead we obtain our gradient estimates from the dual solutions associated with the LPs solved at each iteration. The approach is similar to the SPSA procedure except for the gradient estimation portion, which is as follows:

Recall that $max_{\theta \in \Theta} J(\theta)$, where $J(\theta) = E[L(\theta, \omega)]$ is the expected profit realization $L(\theta, \omega)$ is profit realization at simulation run ω with decision variables θ. Our decision variables are the intercept and slope parameters of the clearing functions of each machine, which are fitted by the method described in Section 3.3.2. Note that our decision variable vector θ is continuous, and we seek to maximize expected profit. The usual way to update the variables $\theta_{n+1} = \theta_n + a_n \nabla J_n$ in Stochastic Approximation Techniques is using a gradient estimate ∇J_n in the direction of achieving a better value of the performance metric in the simulation. In this heuristic method, our primary goal is not to find the parameter vector θ^* which gives the global optimum of $J(\theta)$, but rather to find a parameter vector that will give better expected profit realizations than the parameters obtained by CF Base model in modest CPU times.

As in the SPSA approaches, we use the load based CF form for CF fits in our heuristics. Our initial CF estimates are obtained by the least-squares fit of the load based form applied to two segments of the data. The third segment is the horizontal line given by period length divided by the processing time of that machine. The third segment defines the capacity limit as given in Section 3.3. As in the simulation optimization approach, we will not modify the third segment of the fits. Recall that we consider the CF of the system in two ways. The first is a static CF for each machine that does not change over time; the second is a CF for each machine in each period. As in SPSA Each, we allow the CF of the machine to change over periods in CF Each and CF KW.

One of our goals in these heuristic algorithms is to find CFs that will lead to a larger change in expected profit at each iteration. In other words, we want to focus on the “important” variables in our vector θ that will potentially yield larger changes in $J(\theta)$ and hopefully higher expected profit. We refer to changes in $J(\theta)$ as the impact of changes in θ vector on the system. The intuition for this approach using the dual solutions of the LP at each iteration is that if a machine is not heavily utilized compared to other machines, modifying the corresponding CF fit may not lead to significant change in the simulated results. Thus the releases obtained from optimization will not be affected much, since the CF constraint of that machine is not restrictive on the output. We can state the same argument at the segment level. For example, if the CF constraint is mostly binding at segment 1 or 2, modifying a segment whose corresponding constraint is not binding may not affect the release values from optimization. Therefore, our first algorithm will focus on the CFs at the machine and segment levels, and our second algorithm will also include period levels. When we focus our search on the CFs of more highly utilized machines, there is more opportunity to change the θ vector which may lead to better expected profit realizations in the simulation. Our search will seek these opportunities for larger impact by looking at the dual solution of the CF constraint by machine, segments and also, in the second approach, by period. Thus, instead of focusing on the entire variable space and modifying variables arbitrarily, we attempt to find a subset of the variable space that gives higher possibility of favorable change in expected profit. In our search algorithms, larger perturbations $\Delta\theta$ will be applied to those variables appearing in CF constraints with higher values of their associated dual variables. If the dual variable turns out to be zero or close to zero, the variables will not be modified.

In this chapter, we present three iterative heuristics, CF All, CF Each and CF KW. The heuristic CF All uses common CFs across all periods as in SPSA All. We apply weighted least-squares regression by changing the weights of observations in the data collected by the procedure explained in Section 3.3. We perturb the initial fit of CF load with the use of dual solutions and simulated performance at each iteration.

The heuristic CF Each uses the same idea as in CF All but will fit separate CFs for each period as in SPSA Each. Again CF Each perturbs the initial fit of the CF Load. Differently from the previous heuristics, CF KW also considers previous iteration's fit and builds on it with the use of dual solutions and simulated results from the current iteration. The first two heuristics, CF All and CF Each, have a larger search space since we do not roll over the weights assigned to the observations to the next iteration. On the other hand, CF KW stores the weights assigned to the observations and either these weights will be updated in the current iteration or stay same. Therefore the influence of the weights will be carried over for the weighted least-squares regression of the current iteration to obtain our new CF fit.

In the following section, we discuss how we calculate weights using the dual solutions from the optimization models.

7.1. Linear Regression with Weights

The first load based CF fit parameters that we use in optimization are obtained from straightforward linear regression which comes from minimizing the sum of squared errors between predicted and observed values. We referred this model using the initial estimates of load based CF as the CF Base model as in the previous chapter. In this case, the functional form is

$$\widehat{Y_t} = \mu + \beta(R_t + W_{t-1})$$

where μ is the estimated intercept, β the slope estimate and $\widehat{Y_t}$ the estimated output in period t given the sum of release and initial WIP (R_t+W_{t-1}) at period t. These estimates constitute the CFs used in the LP. Recall from Section 3.3 that we collect data in order to estimate our CFs. We have observations of output Y as a function of release R and initial WIP W. We consider these observations without time subscript t since we aggregate the data obtained from executing seven release schedules in simulation as explained in Section 3.3, and we are interested in the values instead of which period we obtain those observations. Essentially, those executions of seven release schedules in the simulation aim to gather many possible combinations of observations of Y and (R+W) to model the production system more accurately. We apply the standard least-squares regression to the data which minimizes the sum of the squared errors (Simulated - Estimated) for estimating the intercept and slope parameters.

$E^2=\sum_i (Y_i - \widehat{Y}_\iota)^2$, where i is the observation index

In our algorithm, we use weighted least-squares, where we assign weights to each observation and minimize the total weighted squared error,

$E_w^2=\sum_i w_i(Y_i - \widehat{Y}_\iota)^2$, where i is the observation index

Observations are not treated equally, but are weighted to influence the parameter estimates. If we do not assign a weight to an observation, this implies the weight w_i of observation i is one, giving no extra influence to that observation in the fitting.

In our search algorithms we will use the weights assigned to observations to modify our parameter estimates. After obtaining our first parameter estimates from the least-squares fit of load based CF form, we start to modify the weights so as to shift the CF segments up or down and evaluate the effects of these changes using simulation. How the weights are assigned to specific observations will be explained in the following section.

7.1.1. Determination of Weights and Assignment to Observations

In this section we describe three heuristics that we refer as CF All, CF Each, and CF Keep Weight, that we use to determine weights and choose the observations to assign those weights to alter the CF fit.

7.1.1.1. CF All

This approach, as in SPSA All, uses a common CF across all periods for each machine. In this approach we take the dual variables from the optimization corresponding to the CF constraint $Y_{gtl} \leq Z_{gtl}^{k}\mu_{k}^{s} + \beta_{k}^{s}\left(X_{gtl} + W_{g,t-1,l}\right)$ and sum them over the products and operations performed by a given machine *k*. Let π_{gtl}^{s} be the dual variable associated with the CF constraint corresponding to product *g*, segment *s*, operation *l* and period *t*. Thus the summation over products and operations of π_{gtl}^{s} give

$$\pi_{tk}^{s} = \sum_{\mathrm{g} \in \mathrm{G}, l \in \mathrm{L}(k)} \pi_{gtl}^{s}.$$

Next we take the average of the π_{tk}^{s} over the T periods in the planning horizon to obtain the average dual price over the periods

$$\bar{\pi}_k^s = \frac{\sum_{t\in T} \pi_{tk}^s}{T}$$

In order to convert the average dual prices to weights we scale our average dual prices by the objective function value of the LP model at that iteration and multiply them by 100 to obtain average dual solutions as a percentage of the objective function value. Thus our weights take the form

$$\omega_k^s = -\frac{\bar{\pi}_k^s}{Objective\ function} 100;$$

We multiply by negative one (-1) in order to have positive values for our weights, since the problem is minimization, we obtain negative values for our dual solutions and when we assign weights to the observations, we aim to work with positive values. The quantity ω_k^s denotes the weight to be assigned to observations *i* of Y_{ik}^s collected for machine *k* and confined in segment *s*. The specific choice of observations will be explained in later paragraphs. In the calculation of ω_k^s, if the value turns out to be less than 1, we set the value of ω_k^s to 1, bringing no extra influence of that observation on fitting.

Kefeli et al. (2011) show that the dual prices associated with the CF constraint are related to the dual price of the capacity. They also show that a more intuitive dual price of capacity is obtained from the dual variable associated with the output allocation constraint $\sum_{g\in G,l\in L(k)} \mathrm{Z}_{gtl}^{k} = 1$. In our heuristics, we use dual solutions and transform them into weights of observations to consider the dual price of capacity when we obtain our new CF fits. The dual prices and their transformed weights enable us to capture the most promising subset of CF fits to perturb and potentially improve. The weights ω_k^s are assigned to observations to influence the fits. The larger the dual variables $\bar{\pi}_k^s$ of a machine k and segment s, the larger the weight ω_k^s; therefore the perturbation on CF fit of machine k and segment s is higher. In a sense, we perturb the fits of those segments that have more potential to yield larger changes in expected profit $J(\theta)$. Next, we discuss how we assign these weights to observations and modify the fit. We first describe the steps for an iteration, and then present the algorithm in pseudocode including the iteration index.

We follow these basic steps assigning the weights:

Step 1: Obtain the initial set of average dual variables $\bar{\pi}_k^s$ by solving the optimization model with the load based clearing function, where standard linear regression is applied to estimate the two segments of the CF.

Step 2: Calculate the differences in outputs by period from optimization and simulation and take the sum over periods. The quantity T is planning horizon.

$$\bar{Y}_g = \sum_{t\in T}(Y_{gt}^{LP} - \hat{E}\left[Y_{gt}^{SIM}\right]) \quad for\ all\ g \in G.$$

Y_{gt}^{LP} denotes the output of product g in period t obtained from the LP solution. Y_{gt}^{SIM} denotes the output of product g in period t obtained from the simulation run using the release schedule of the LP. For a simulation run, we perform 15 replications. $\bar{Y}_g$ denotes the total difference over all periods for each product g.

We sum these totals across the products and record the sign of the total difference.

$\bar{Y} = \sum_{g \in G} \bar{Y}_g.$

This will provide a sense of whether the CFs used in the current LP are overestimating or underestimating the expected output at simulation. If the value $\bar{Y}$ is positive, we are overestimating the CFs since we subtract simulation outputs from LP outputs and a positive sign implies that the LP predicts more output then what is observed in simulation. On the other hand, if the value is negative, then we are underestimating the fits. Simulation outputs are higher than what is estimated in the LP using CFs.

Step 3: Depending on the sign of the total difference value $\bar{Y}$, assign weights ω_k^s to the observations where the residual of the fit $Y_{ik}^s - \widehat{Y_{ik}^s}$ (actual minus predicted output) has the opposite sign of $\bar{Y}$. For example, if $\bar{Y}$ is negative, then it can be interpreted as the CFs are underestimating the output, so we assign ω_k^s to the observations whose residual is positive. The observations with positive residuals imply the observations lie above the fitted line. Recall that $\omega_k^s \geq 1$. When we assign weights to those observations with positive residuals, they will have higher weights in the weighted least-squares regression thus higher influence on the fit. Note that those observations to which we do not assign weights explicitly have a default weight of 1. In this particular example, the fit will be lifted up since observations above the fitted line get the higher influence on the fit while obtaining our new CF fit and therefore we may eliminate the underestimation of CFs.

Step 4: Solve the LP model again and repeat the procedure starting from Step 2. In this algorithm, at beginning of each iteration we use the initial least-squares fit only for applying our weights and obtain our new CF fits to be used in that iteration. Weights will change at each iteration since we obtain different dual solutions solving LP with new CF fits. We record the expected profit obtained using new CF fits at each iteration. Iterations may be stopped when a production plan with a statistically significantly better expected profit value is achieved or may be continued to search for better production plans with higher expected profit value.

We present the illustration of assigning weights in the case of underestimation and overestimation of CFs in Figure 7-1 and Figure 7-2 .

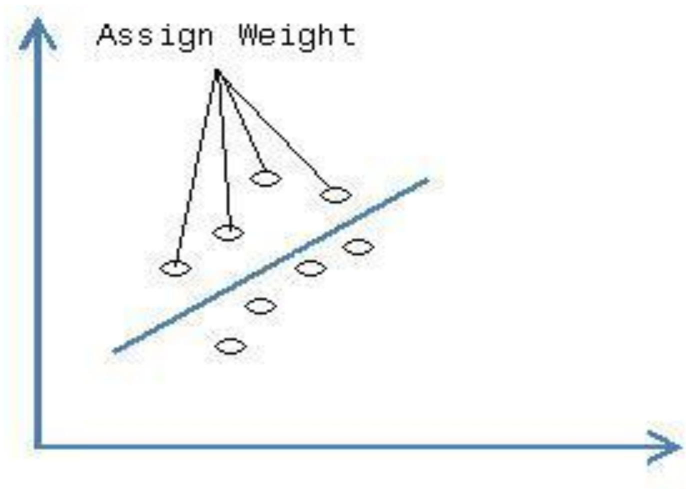


Figure 7-1: Assigning Weights in Underestimation Case

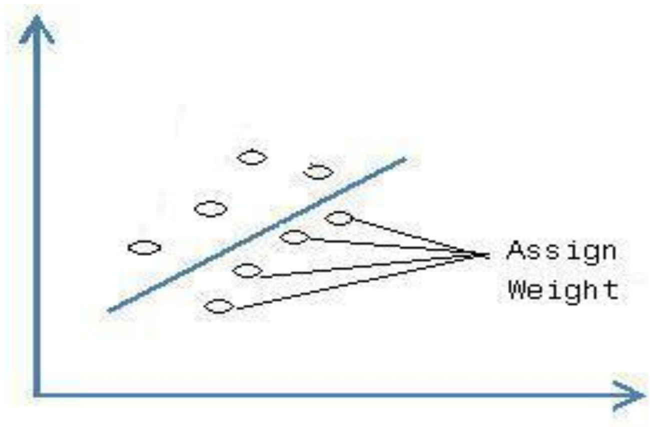


Figure 7-2: Assigning weights in Overestimation Case.

The algorithm assigns weights to the observation over the fitted line in the case of underestimation of CFs shown in Figure 7-1 in order to eliminate the underestimation. The fitted line will shift up after applying the weighted least-squares regression. In the case of overestimation, the assignment of weights is applied to the observations below the fitted line shown in Figure 7-2.

We next present the algorithm in pseudocode:

We define the following notation.

The quantity ρ_{ik}^{s0}=$(Y_{ik}^{s} - \widehat{Y_{ik}^{s}})$ is the residual value of observation i, contained in segment s and observed for machine k. The zero superscript corresponds to the initial least-squares regression that we obtain our residuals from.

We have the following sets of the observations that have either positive or negative residual value.

$R_k^{s+} = \{i: \rho_{ik}^{s} > 0\}$,

$R_k^{s-} = \{i: \rho_{ik}^{s} \leq 0\}$.

Let n be iteration index and θ_n the CF fits at iteration n. The vector θ_o is the initial least-squares regression fit of CF load form. The quantity ρ_{ik}^{s0} are the residuals correspond to θ_o and $J(\theta_n)$ is the expected profit. Recall that we calculate ω_k^s from the dual solutions of $\bar{\pi}_k^s$ and we have iteration subscript n for the dual solutions and weight calculations for the corresponding iteration n where when n equals to 0 corresponds to the initial values coming from the solutions of θ_o.

Step 0: Initialize the first iteration by calculating $\overline{Y_0}$ and ω_{k0}^s from $\bar{\pi}_{k0}^s$ using θ_o.

Set *n*=1;

Step 1: if $\bar{Y}_{n-1} > 0$ then,

Assign $\omega_{k,n-1}^s$ to all $Y_{ik}^s \in R_k^{s-}$

and 1 to all $Y_{ik}^s \in R_k^{s+}$

Else Assign $\omega_{k,n-1}^s$ to all $Y_{ik}^s \in R_k^{s+}$

and 1 to all $Y_{ik}^s \in R_k^{s-}$

Step 3: Apply weighed least-squares regression and obtain θ_n.

Step 4: Solve LP with new CF fits θ_n and run simulation with release schedule obtained by solving LP with θ_n to get expected profit $J(\theta_n)$.

Step 5: Calculate $\overline{Y_n}$ and ω_{kn}^s from $\bar{\pi}_{kn}^s$ using θ_n

n=n+1;

Step 4: if *n* < *MaxIter* then go back to Step 1. Otherwise stop.

In our experiments, significantly better production plans are usually achieved within 10 iterations as will be discussed in Section 7.2.

Next we will discuss the CF for each period.

7.1.1.2. CF Each:

This approach, as in SPSA Each, uses different CFs for each period. It is similar to the previous one except that we do not consider the average dual price over all periods, but instead use the dual prices associated with each period π_{tk}^{s}. Recall that $\pi_{tk}^{s} = \sum_{\mathrm{g} \in \mathrm{G}, l \in \mathrm{L}(k)} \pi_{gtl}^{s}$ and the weights now have period subscript t and are calculated as

$$\omega_{tk}^{s} = -\frac{\pi_{tk}^{s}}{Objective\ function} 100.$$

In this approach, we capture more details of the use of capacity over time. Let us say that in one period the demand is considerably lower than the other periods. Then the dual prices of the CF constraints corresponding to that period should be smaller than those in the other periods. We may not need to modify the parameter estimates in that period since it may not impact the production plan that much. On other hand, for periods that have higher demand and thus higher utilization, we may have a better opportunity to modify the parameter estimates to search for a better production plan. Also, in this case we do not lose the information by averaging the dual solutions.

The algorithm steps are the same as CF All except the calculation of ω_{tk}^{s}.

7.1.1.3. CF Keep Weight:

In the previous approaches, at each iteration, when we get the dual solutions and start the fitting process, we take as baseline the initial least-squares fit without any previous weights assigned to observations. Recall that in the previous approaches at each iteration, the residuals ρ_{ik}^{s0} are the same but the duals and calculated weights change. In CF Keep Weight we also take into consideration of the weights from the previous iteration. That means either the weight of an observation is updated in the current iteration or kept the same as its previous value, depending on the sign of the total differences $\overline{Y_n}$ and observation's residual value in the weight assignment step of the algorithm. This also implies that residuals ρ_{ik}^{sn} change as iterations *n* proceed since we build on the previous iteration's fit and no longer start afresh at the initial CF fit before applying weights. In other words, the weights are assigned to the fit that is obtained in the previous iteration. This bookkeeping potentially yields more stable solutions as the iteration progress. The steps of this algorithm are as follows:

Define n as the iteration index and recall θ_o as the initial least-squares regression fit of CF load form. The quantities ρ_{ik}^{sn} are the residuals corresponding to θ_n, and $J(\theta_n)$ is the expected profit. We initialize the first iteration by calculating $\overline{Y_0}$ and ω_{k0}^{s} from $\overline{\pi}_{k0}^{s}$ using θ_o. Recall that CF KW, as in CF Each, uses different CFs for each period. We present the algorithm below for any chosen time period in the planning horizon. It is analogous for all CFs of each period. Recall the sets of observations for machine k and segment s that have positive residuals R_k^{s+} and negative residualsR_k^{s-}.

Step 0: Initialize the first iteration by calculating $\overline{Y_0}$ and ω_{k0}^s from $\bar{\pi}_{k0}^s$ using θ_o.

Set n=1;

Step 1: if $\bar{Y}_{\text{n-1}} > 0$ then,

Update weight ω_{ik}^s of Y_{ik}^s with new weights $\omega_{k,n-1}^s$ where $Y_{ik}^s \in R_k^{s-}$ and keep ω_{ik}^s of Y_{ik}^s where $Y_{ik}^s \in R_k^{s+}$

Else update ω_{ik}^s of Y_{ik}^s with $\omega_{k,n-1}^s$ where $Y_{ik}^s \in R_k^{s+}$ and keep ω_{ik}^s of Y_{ik}^s where $Y_{ik}^s \in R_k^{s-}$.

Step 3: Apply weighed least-squares regression and obtain θ_n.

Step 4: Solve the LP with new CF fits θ_n and run simulation with release schedule obtained by solving the LP with θ_n to get expected profit $J(\theta_n)$.

Step 5: Calculate $\overline{Y_n}$ and ω_{kn}^s from $\bar{\pi}_{kn}^s$using θ_n

n=n+1;

Step 4: if n<*MaxIter* then go back to Step 1. Otherwise stop.

This approach is similar to CF Each in terms of weight calculations for individual periods but we don't start from the initial least-squares fit anymore, we build on the modified fits obtained at each iteration. Essentially, Step 1 in this algorithm is different from the previous ones.

We now present our experimental results for these heuristic algorithms.

7.2. Search Heuristic Algorithm Results

The testbed that we use is same as the one used in Multiple Linear Regression (MLR) and SPSA experimental design with same set of demands and failure parameters. We use the same parameters in order to make a fair comparison between these two fitting approaches and also the heuristics and the SPSA procedures.

First, we will compare the three heuristic algorithms among themselves and then compare them to the SPSA approaches in terms of profit and time performance. In CHAPTER 8, we will compare heuristic algorithms to the MLR solutions from CHAPTER 5. We follow these steps for the comparison of the heuristics.

Step 1: Solve the LP of the ACF model to get the LP outputs and release schedule that are suggested by the LP to produce the desired LP output at the objective function.

Step 2: Give the LP release schedules to the simulation as input. Perform fifteen replications recording the realized output and calculating the realized objective function value.

Step 3: Obtain the dual solutions from LP and calculate the weights to refit the CFs.

Step 4: Repeat Steps 1 through 3 until a better profit realization in simulation is achieved or the maximum number of iterations is reached. In our experiment for this model we use MaxIter=30 where in most of the cases, we obtained more profitable production plans. In our experiments, we do not stop the iterations once we observe a better plan to continue searching for a better one than the best plan achieved until that iteration. When we perform 30 iterations, we observe that either the solution converges such that profit values stay constant as the iterations proceed, or the solution cycles through two or three points.

For all search heuristics, the starting point is the solution achieved with straightforward least-squares fit of the Load-Based CF form which we refer to as the CF Base model. In our reports, we show the best profit realization achieved among all iterations. If the algorithm cannot find a release schedule that yields higher profit than the initial least-squares fit solution, we assign the initial solution as the best one. In a sense, that means we assign the weight of 1 to all observations in the search algorithm. First we present the profit comparison graph with best profit values achieved by the individual search algorithm including the initial profit value achieved by the regular least-squares fit.

Figure 7-3 shows the profit comparison of the search algorithms including the initial solution from least-squares regression that we refer to as CF Base.

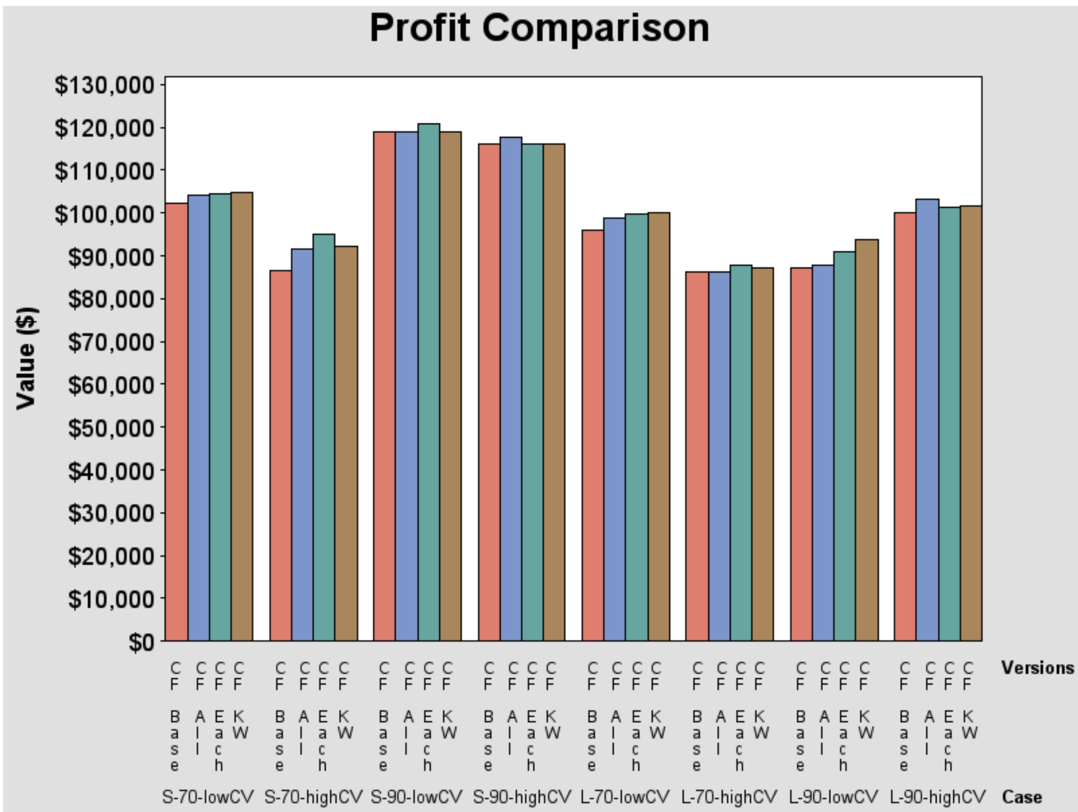


Figure 7-3: Profit Comparison of Search Algorithms

In Figure 7-3, CF All denotes the search algorithm which works with one CF for all periods and considers the average dual solutions assigning the weights to the fit. CF Each refers to individual fits for each period and considers the dual solutions from each period. CF KW (Keep Weight) refers to refitting the function considering the fit from the previous iteration and its current dual solutions at that iteration.

From Figure 7-3, we see that, in all cases, at least one of the search algorithms finds a better solution than CF Base, indicating that the base solution is improved. There is no single dominating search algorithm that is better than the others. In three out of the eight cases, S-70-highCV, S-90-lowCV and L-70-highCV cases, CF Each is the leading algorithm. Again in three out of all cases, S-70-lowCV, L-70-lowCV and L-90-lowCV cases, CF KW is the leading algorithm. In the rest, S-90-highCV and L-90-highCV cases, CF All becomes the leading algorithm. There seems to be a pattern as to which of the algorithms function better. For example, at high utilization (90%), high CV cases, CF All is the leading algorithm. Since all periods are highly utilized, one CF fit is satisfactory and leads to a better solution. For low utilization low CV cases, having individual CF fits leads to higher profit values. We consider each period individually and have a more detailed fit for each period.

In order to show how much improvement is achieved we will again focus on costs (inventory, WIP and backlog costs) except for material cost. We also see that the revenues generated by the algorithms are also close to each other for similar reasons. Figure 7-4 shows the cost distribution of the solutions obtained through the different search algorithms.

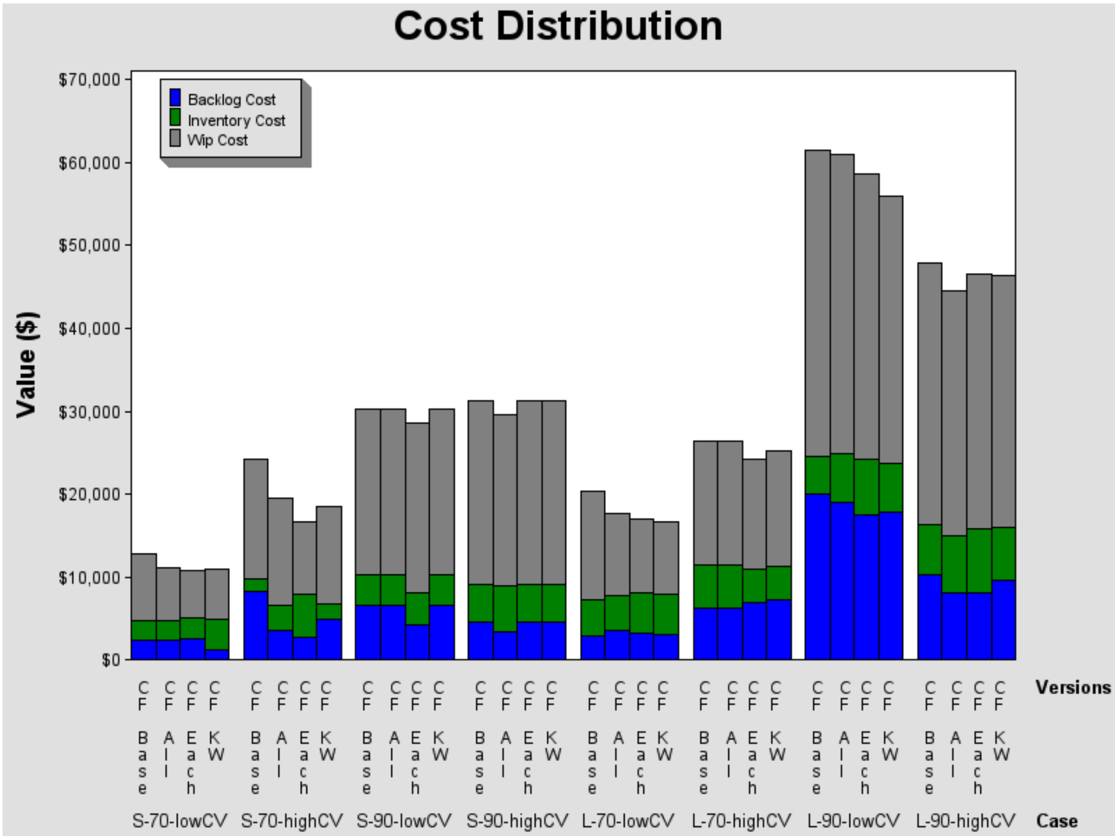


Figure 7-4: Cost Distribution of Search Algorithms

In order to quantify the cost reductions in search algorithms, we calculate the relative percent cost reduction taking the CF Base as the reference as shown in Table 7-1.

Table 7-1: Cost Reduction of Search Algorithms

	Algorithms		
Cases	CF All	CF Each	CF KW
S-70-lowCV	13%	16%	15%
S-70-highCV	19%	31%	23%
S-90-lowCV	0%	5%	0%
S-90-highCV	5%	0%	0%
L-70-lowCV	13%	17%	18%
L-70-highCV	0%	8%	5%
L-90-lowCV	1%	5%	9%
L-90-highCV	7%	3%	3%

We observe that the cost reduction goes up to 31% as seen in S-70-highCV for CF Each. In low utilization cases (70), the improvements are above 10% except for L-70-highCV which is limited to 8%. This shows that there is more room for improvement in low utilization cases than in high utilization cases, where cost reduction ranges from 5% to 9%.

Next, we present the Friedman analysis of the profit comparison of search algorithms in Table 7-2 below.

Table 7-2: Friedman Analysis of Search Heuristic Algorithms

S-70- lowCV		S-70-highCV		S-90-lowCV		S-90-highCV	
Mean	**Model**	**Mean**	**Model**	**Mean**	**Model**	**Mean**	**Model**
104720	CF KW	94990	CF Each	120802	CF Each	117737	CF All
104386	CF Each	92285	CF KW	118912	CF All	115985	CF Base
104111	CF All	91597	CF All	118912	CF Base	115985	CF Each
102380	CF Base	86589	CF Base	118912	CF KW	115985	CF KW

L-70- lowCV		L-70-highCV		L-90-lowCV		L-90-highCV	
Mean	**Model**	**Mean**	**Model**	**Mean**	**Model**	**Mean**	**Model**
100008	CF KW	87914	CF Each	93746	CF KW	103280	CF All
99746	CF Each	87109	CF KW	91077	CF Each	101661	CF KW
98783	CF All	86120	CF All	87971	CF All	101458	CF Each
95965	CF Base	86120	CF Base	87286	CF Base	100019	CF Base

In Table 7-2, the models that are highlighted in grey are not statistically different from each other. Table 7-2 shows that at least one of the search algorithms achieves significantly better profit realizations in simulation than CF Base. In three cases (S-70-highCV, L-70-highCV and S-90-lowCV), CF Each is statistically better than the others and in three cases, (S-70-lowCV, L-70-lowCV and L-90-lowCV) CF KW algorithm has the higher profit values and statistically different than others. We note that in the S-70-lowCV and L-70-lowCV cases, there is not much difference (less than 0.3%) in terms of profit values achieved even though the Friedman Test considers them statistically different. Therefore, the CF Each algorithm performs best in three cases and very close to best in two cases out of a total of eight cases. Recall that both CF Each and CF KW implement CFs for each period but differ in their updating process in iterations. In two cases, CF All performs best in high utilization and high CV cases.

In summary, if there is more room for improvement as shown in cost reduction analysis, CF Each performs better or very close to the other algorithms.

We present the iterations of heuristic algorithms in order to show the progress of algorithms along the iterations. As a sample, we choose the most cost reduction achieved case, S-70-highCV from Table 7-1. Note that, SPSA approaches also perform best in this case in terms of cost reduction.

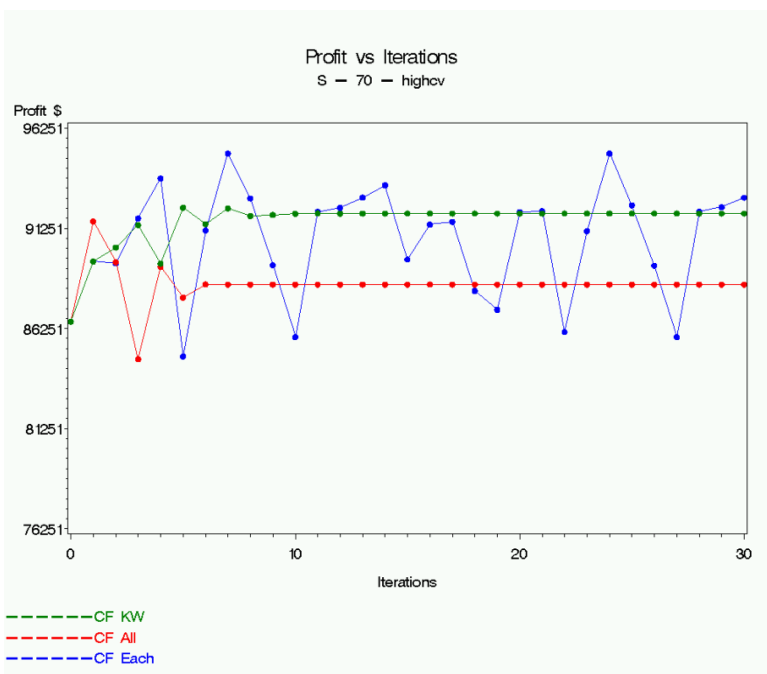


Figure 7-5: Heuristic Algorithms Profit vs. Iteration plot of S-70-highCV

Recall that we perform 15 replications per iteration, and we plot the mean of the replications. From Figure 7-5, we observe that CF All shown in red, converges at the 6th iteration which is not the best solution but better than CF Base profit value. The heuristic CF KW, shown in green, also converges after at the 10th iteration, again converged solution is not the best one but very close to the best profit value achieved. On the other hand, CF Each, shown in blue, seems to cycle through the very similar solutions, with the same best profit values achieved at the 7th and 24th iteration.

We choose another sample case, L-70-highCV whose cost reduction performance is also good compared to other cases.

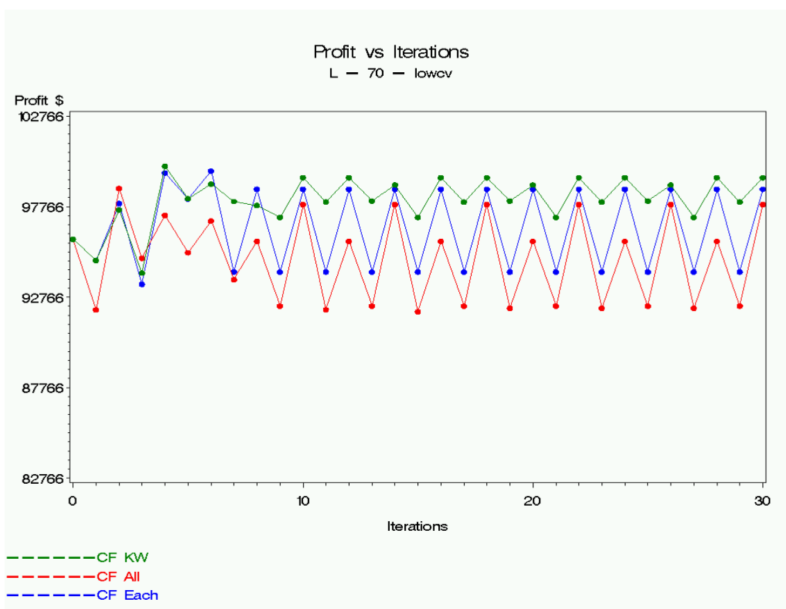


Figure 7-6: Heuristic Algorithms Profit vs. Iteration plot of S-70-highCV

Figure 7-6 shows that all heuristic algorithms cycle through the same solutions after some iterations. That implies that we obtain the same dual solutions from LP and therefore end up assigning the same weights for the same observations at certain iterations. The information that we collect from LP and simulation, and the way we assign weights, makes the algorithms visit the same or very similar points periodically. We also observe these cyclic iterations in other cases. This implies that we achieve the best possible profit values within 30 iterations for these heuristic algorithms.

Next we examine the computational time performance of the heuristics. We again choose the S-70-highCV case for illustration.

Table 7-3: Computational Time Performance of Heuristics

Model	**SIMtime (s)**	**LPtime (s)**	**Regtime (s)**	**IterationTime (s)**	**Profit($)**
CF ALL	146	5	5	156	91597
CF Each	141	5	133	280	94990
CF KW	137	5	164	306	92285

The values are the average of 30 iterations. We note that we perform one simulation run, one LP solve and one regression operation at each iteration. The heuristics use very similar amounts of time in simulation and solving LPs, but differ in the regression operation. Recall that, we perform regression for two segments for each of 11 machines in CF All. However for CF Each and CF KW, we perform regression for each period of the planning horizon (26 periods) in addition to operations of CF

All. That makes the difference in the time for regression operation shown in the Regtime column and note that 133s is almost 5s times 26 periods. For CF KW, Regtime is higher than for CF Each, since we have additional operations for the regression of the new CF fits in addition to the previous CF fits. In summary, CF All has a significant advantage in the time that the regression operation takes compared to others. Recall that, CF All performs better in two cases (S-90-highCV and L-90-highCV) and the rest of the cases, either CF Each or CF KW is the leading algorithm.

In the next section, we present the comparison of SPSA approaches developed in CHAPTER 6 and heuristic algorithms developed in this chapter.

7.3. Comparison of SPSA approaches and Heuristic Algorithms

First, we present Friedman Tests to show statistically that the SPSA approaches achieve better profit values than the heuristic algorithms. We omit SPSA Release in this comparison since all the other SPSA approaches and heuristic algorithms try to improve CF fit parameters but SPSA Release works on release variables.

Table 7-4: The Friedman Test Comparing SPSA and Heuristic Algorithms

S-70- lowCV		S-70-highCV		S-90-lowCV		S-90-highCV	
Mean	**Model**	**Mean**	**Model**	**Mean**	**Model**	**Mean**	**Model**
106462	SPSA All	98262	SPSA All	124902	SPSA Each	124079	SPSA All
106207	SPSA Each	97437	SPSA Each	124623	SPSA All	123361	SPSA Each
104720	CF KW	94990	CF Each	120802	CF Each	117737	CF All
104386	CF Each	92285	CF KW	118912	CF All	115985	CF Base
104111	CF All	91597	CF All	118912	CF Base	115985	CF Each
102380	CF Base	86589	CF Base	118912	CF KW	115985	CF KW
L-70- lowCV		**L-70-highCV**		**L-90-lowCV**		**L-90-highCV**	
Mean	**Model**	**Mean**	**Model**	**Mean**	**Model**	**Mean**	**Model**
101240	SPSA All	91897	SPSA Each	97632	SPSA Each	110708	SPSA All
100258	SPSA Each	91742	SPSA All	93746	CF KW	109848	SPSA Each
100008	CF KW	87914	CF Each	93653	SPSA All	103280	CF All
99746	CF Each	87109	CF KW	91077	CF Each	101661	CF KW
98783	CF All	86120	CF Base	87971	CF All	101458	CF Each
95965	CF Base	86120	CF All	87286	CF Base	100019	CF Base

Table 7-4 shows that the SPSA approaches are statistically better than the heuristic algorithms except in the L-70-lowCV and L-90-lowCV cases, where CF KW provides a comparable result to SPSA All. We can also infer from the cost reduction tables Table 6-3 and Table 7-1 that the SPSA approaches provide higher cost reductions than the heuristic algorithms. In order to show how much better the SPSA approaches are, we pick the best performing SPSA approach and heuristic and compare them in terms of cost reduction achieved in Table 7-5 below. We again consider CF Base as our reference.

Table 7-5: Comparison of Cost Reduction

	Algorithms	
Cases	SPSA	Heuristic
S-70-lowCV	29%	16%
S-70-highCV	44%	31%
S-90-lowCV	19%	5%
S-90-highCV	24%	5%
L-70-lowCV	22%	18%
L-70-highCV	21%	8%
L-90-lowCV	15%	9%
L-90-highCV	21%	7%

Table 7-5 clearly shows that SPSA approaches obtain much better cost reductions, such as 19% to 5% in S-90-lowCV case or 21% to 7% in L-90-highCV case.

In previous paragraphs, we show that SPSA approaches perform better than heuristic algorithms. Also recall that we perform 30 iterations for heuristics where we achieve best possible profit value compared to 100 iterations for SPSA approaches where the profit values pretty much level off but there is still the possibility for improvement even it may be small.

Next, we discuss the computational efforts of both SPSA and heuristics. Since we perform different numbers of iterations for both methods, we first compare the average computational time per iteration and best profit achieved summarized from Table 6-5 and Table 7-3 and obtain Figure 7-7 below for the case S-70-highCV.

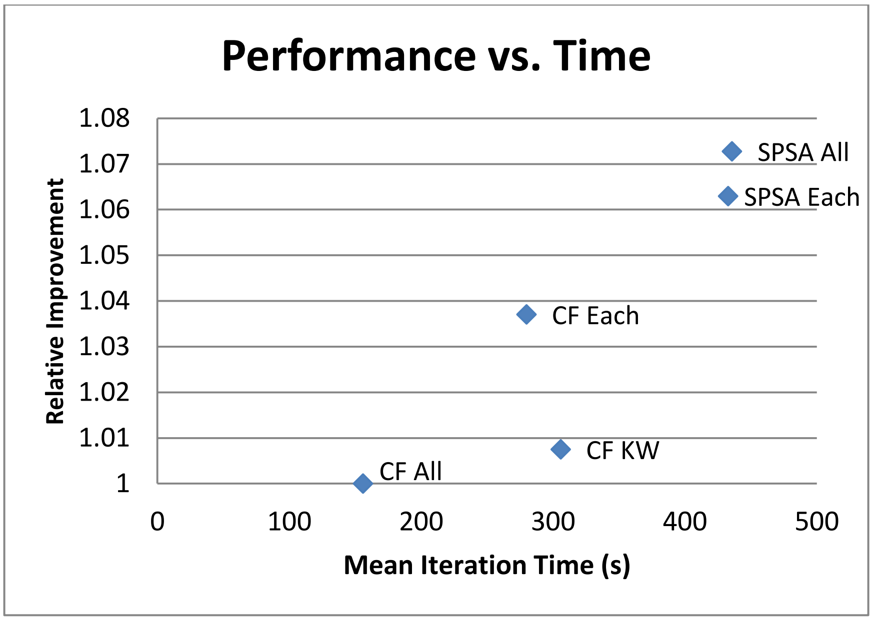


Figure 7-7: Cost-Tradeoff plot of all Algorithms in mean iteration time

In Figure 7-7, we calculate relative improvement by normalizing against the lowest expected profit value, which in this case corresponds to CF All. We observe from Figure 7-7 that, even though the SPSA approaches can take almost up to three times of the time that CF All takes per iteration, they achieve 8% improvement in profit over CF All. Similarly, the average iteration time of SPSA is about 1.4 times that of the longest iteration time algorithm of heuristics, CF KW and achieves about 6% additional improvement. This shows that the SPSA approaches are more computationally intensive but at the same time achieve significantly better results.

Figure 7-7 illustrates the trade-off between computational time and expected profit value achieved. If we spend more time in computation, we can achieve better performance results. Recall that we perform 100 iterations for the SPSA approaches and 30 iterations for the heuristics. We now show another plot that we consider the total time spent until reaching the best profit value for the S-70-highCV case.

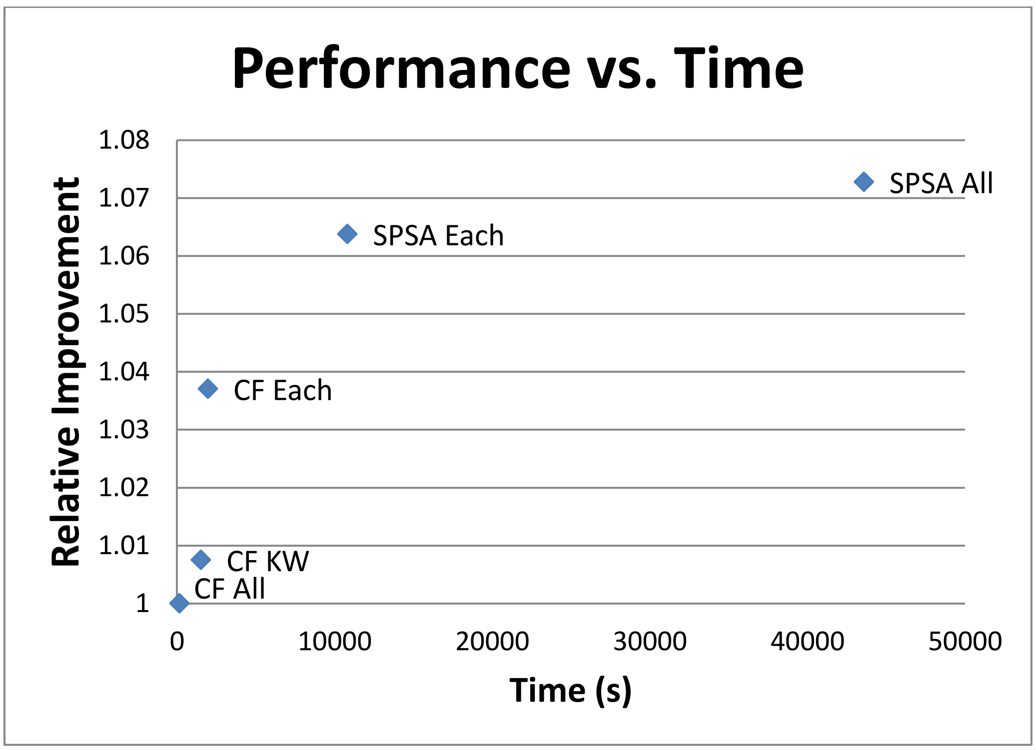


Figure 7-8: Cost-Tradeoff plot of all Algorithms in total computation time

In Figure 7-8 we again show normalized profit but instead of mean iteration time we use the total time spent until reaching the best profit value. We observe the diminishing return of the profit values of the algorithms respect to total time spent. As expected, the algorithms can achieve better profit when they spend more time, but the relationship is not linear. We observe this concave shape in Figure 7-8. We again observe that SPSA algorithms spend significantly more time achieving the best solution compared to the heuristic algorithms.

For another comparison we pick the first iteration of SPSA approaches to achieve a better profit value than the heuristic algorithms. We include the corresponding profit values and computational times and compare them to the values for the heuristic algorithms.

Table 7-6: Computational Effort of Best Iteration of Heuristics

Model	**Iteration**	**Profit ($)**	**Time (s)**
CF ALL	1	91597	156
CF Each	7	94990	1958
CF KW	5	92285	1530

Table 7-7: SPSA vs Heuristics Iteration Comparison

	CF ALL			CF Each			CF KW		
SPSA	**Iteration**	**Profit ($)**	**Time (s)**	**Iteration**	**Profit ($)**	**Time (s)**	**Iteration**	**Profit ($)**	**Time (s)**
SPSA All	5	92624	2180	12	96251	5232	5	92624	2175
SPSA Each	2	95540	866	2	95540	866	2	95540	866

Table 7-7 shows the first iteration number and the corresponding profit value for the SPSA approaches that exceeds the best profit value achieved by heuristic algorithms. We observe that for this particular case S-70-highCV, SPSA All takes more time to achieve a better profit value than all heuristic algorithms shown in Table 7-6. For example, SPSA All requires 5 iterations taking 2180 seconds to exceed the profit value achieved by CF All in one iteration and 156 seconds. On the other hand, SPSA Each exceeds the profit values of heuristics in the second iteration and requires less time than CF Each and CF KW. This analysis again shows the computational intensity of the SPSA approaches when we need to perform more iterations to improve the solution. At the same time, the SPSA approaches show superior performance in terms of achieving a better profit values than the heuristics.

In the next chapter, we compare both CF functional forms $X_{kgt} = f_k \left(W_{k,\mathrm{g},t-1}, R_{kgt}\right)$ used by MLR and $X_{kgt} = f_k \left(W_{k,\mathrm{g},t-1} + R_{kgt}\right)$ used by SPSA approaches and heuristic algorithms and present their results.

CHAPTER 8. COMPARISON OF MULTIPLE LINEAR REGRESSION FITS AND CF LOAD FITS

In this thesis we consider two types of fits for the clearing function. The first one is Multiple Linear Regression (MLR) fits where we consider the releases and work in process (WIP) at the beginning of the period separately for individual products. It corresponds to the product based functional form $X_{kgt} = f_k \left(W_{k,\mathrm{g},t-1}, R_{kgt}\right)$. As described in CHAPTER 5, we use three models and three selection methods to analyze this fitting procedure. In fitting the load based CFs, we sum the releases of the period and WIP at the beginning of that period of all products and refer it as the load of the period and fit a linear function; the output is a function of the load using least-squares fit. This corresponds to the functional form $X_{kt} = f_k \left(W_{k,t-1} + R_{kt}\right)$. We then attempt to improve our fits by using Simulation-Optimization (SPSA) approaches or heuristics which consist of refitting the clearing function by giving weights to some observations. In order to see which method, MLR or Load Based, fits perform better in terms of expected profit, we first compare MLR to heuristic algorithms and then to the SPSA approaches.

8.1. Comparison of MLR vs. Heuristic Algorithms

We perform the Friedman test for all cases given in Table 7-1. In order to show the results concisely, we pick the best MLR model for each case.

Table 7-1: The Friedman Test of Heuristics vs. MLR

S-70- lowCV		S-70-highCV		S-90-lowCV		S-90-highCV	
Mean	**Model**	**Mean**	**Model**	**Mean**	**Model**	**Mean**	**Model**
104720	CF KW	94990	CF Each	120802	CF Each	117737	CF All
104386	CF Each	92285	CF KW	118912	CF All	115985	CF Base
104111	CF All	91597	CF All	118912	CF Base	115985	CF Each
103998	V1B	90067	V2B	118912	CF KW	115985	CF KW
102380	CF Base	86589	CF Base	103250	V2F	103446	V2B
L-70- lowCV		**L-70-highCV**		**L-90-lowCV**		**L-90-highCV**	
Mean	**Model**	**Mean**	**Model**	**Mean**	**Model**	**Mean**	**Model**
101312	V1F	87914	CF Each	93746	CF KW	103280	CF All
100008	CF KW	87109	CF KW	91077	CF Each	101661	CF KW
99746	CF Each	86120	CF All	87971	CF All	101458	CF Each
98783	CF All	86120	CF Base	87286	CF Base	100019	CF Base
95965	CF Base	82976	V1A	86597	V2A	91845	V2F

In Table 7-1, the same color with different tones (ex: light blue, blue and dark blue) indicate that the models highlighted in different tones are statistically different from each other, but each of these models is not statistically different from the model highlighted in the middle tone of the color, such as blue. This analysis shows that load based CFs with improvements outperform the MLR fits in all cases except the L-70-

HighCV case where MLR model 1 takes the lead. When we only consider the CF Base fit and compare it to the MLR fits, we see that it outperforms the MLR fits in five out of the eight cases. This suggests that even a straightforward least-squares fit to load will perform better in most cases than considering the releases and WIP separately, as in the MLR fits. This shows that the CF Load fits and their improvements yield better profit realization at simulation than any of the MLR fits.

These results also show the difference in solution quality between MLR and Load Based fitting procedures. Our performance metric in our experiments is the expected profit realization at simulation. Thus, we associate the quality of the fits with the profit values realized from the executed plans obtained by using those fits in the LP models. Therefore a fit yielding higher realized profit is preferable, regardless of how it fares on more conventional fitting criteria such as SSE or correlation coefficient. Under this performance metric, we can conclude that CF Load yields more profitable production plans than the MLR fits. This suggests that considering WIP in terms of load has more potential to better represent the production system instead of disaggregating WIP into multiple independent variables for each product and performing regression on them individually. However, this may also be a function of the specific system we study; since all operations on the same machine require the same processing time, and there are no setup times on any machine, these conclusions must be tested under more general conditions in future research.

8.2. Comparison of MLR vs. SPSA Approaches

Recall that the CF Base approach performs better or very comparably to the best product based MLR fit. Our main purpose in the SPSA approaches is to improve the CF Base fit to obtain better profit realizations.

Table 7-2: The Friedman Test of SPSA vs. MLR

S-70- lowCV		S-70-highCV		S-90-lowCV		S-90-highCV	
Mean	**Model**	**Mean**	**Model**	**Mean**	**Model**	**Mean**	**Model**
106462	SPSA All	98262	SPSA All	124902	SPSA Each	124079	SPSA All
106207	SPSA Each	97437	SPSA Each	124623	SPSA All	123361	SPSA Each
103998	V1B	90067	V2B	118912	CF Base	115985	CF Base
102380	CF Base	86589	CF Base	103250	V2F	103446	V2B
L-70- lowCV		**L-70-highCV**		**L-90-lowCV**		**L-90-highCV**	
Mean	**Model**	**Mean**	**Model**	**Mean**	**Model**	**Mean**	**Model**
101312	V1F	91897	SPSA Each	97632	SPSA Each	110708	SPSA All
101240	SPSA All	91742	SPSA All	93653	SPSA All	109848	SPSA Each
100258	SPSA Each	86120	CF Base	87286	CF Base	100019	CF Base
95965	CF Base	82976	V1A	86597	V2A	91845	V2F

Table **7**-2 shows that all SPSA approaches perform better than MLR except in the L-70-lowCV case, where SPSA All and V1F are statistically indistinguishable. Recall that in the heuristics and MLR, there is regression applied to the data. In contrast, the SPSA approaches use simulation optimization without any regression process involved. This also shows that even though the CF Load form yields better expected profit values than MLR fits, CF Load fits can be significantly improved by both heuristics and the SPSA approaches. This is another indication that adjusted R-square measure is not a good measure to define a good quality fit to be used in production planning. Both MLR fits and CF Load fits result in very high adjusted R-square values. In the context of statistics, those fits explain the variance in the data very well, but as shown in the results, this does not correspond to CFs yielding the best performance. It is not necessarily the case that a regression model with high adjusted R-square will result in good CF fits for planning that yield high realized profit.

CHAPTER 9. CONCLUSION

9.1. Summary

In this thesis, we address the problem of estimating CFs from empirical data. We consider two CF forms $X_{kgt} = f_k\ (W_{k,g,t-1}, R_{kgt})$ and $X_{kt} = f_k\ (W_{k,t-1} + R_{kt})$ and incorporate these CF fits into the LP models as presented in CHAPTER 5 and CHAPTER 6 respectively. We introduce our simulation model and its details in CHAPTER 3, which is used for data collection and obtaining the performance metric of the executed production plans. We compare the performance of our planning models on the basis of expected profit of the production system obtained through simulation of its operation when controlled by the planning decisions obtained with the two different CF estimation techniques. We present our experimental design in CHAPTER 4 and introduce eight different scenarios that we use to test our models performance.

In CHAPTER 5, we examine the product based CF form $X_{kgt} = f_k\ (W_{k,g,t-1}, R_{kgt})$ in terms of Multiple Linear Regression and variable selection procedures. Our results show that the same regression models with different selection procedures generally do not yield statistically different release plan executions. On the other hand, models that include the effect of the immediately preceding period's releases perform better, especially in the high utilization cases.

In CHAPTER 6, we introduce the simulation optimization approach using Simultaneous Perturbation Stochastic Approximation (SPSA) to improve our load based fits of the form $X_{kt} = f_k\left(W_{k,t-1} + R_{kt}\right)$. We also develop heuristics to improve our CF fits for the same form, which use dual prices for the gradient estimation to update our fit variables (intercept and slope). The objective of the heuristics is to obtain good solutions in modest CPU times. Our results indicate that the linear regression fits can be improved significantly by using our SPSA approaches and heuristics. The SPSA approaches achieve significant cost savings compared to our initial CF fit introduced in CHAPTER 3. Our comparison also shows that even though SPSA is more computationally intensive, it can achieve better profit values compared to heuristics.

We present a comparison of the two CF forms in CHAPTER 8. Our findings show that using the functional form $X_{kt} = f_k\left(W_{k,t-1} + R_{kt}\right)$ for estimating CFs yields higher expected profit values compared to use of the other functional form $X_{kgt} = f_k\left(W_{k,g,t-1}, R_{kgt}\right)$. In addition, our SPSA approaches and heuristics significantly improve the CF fits of the form $X_{kt} = f_k\left(W_{k,t-1} + R_{kt}\right)$, resulting in higher expected profit values. This suggests that the quality of CF fits of the form $X_{kt} = f_k\left(W_{k,t-1} + R_{kt}\right)$ is better than that for the form $X_{kgt} = f_k\left(W_{k,g,t-1}, R_{kgt}\right)$, in terms of our performance criterion, expected realized profit. The estimates of CF fit can also be further improved by SPSA approaches and heuristic algorithms.

9.2. Future Research Directions

In this thesis, we use a scaled down model of a semiconductor fab. We focused on smaller size model in order to obtain insights of the algorithms that we develop and see how effective they are in our scaled down model. One extension can be to scale up this model to more complex systems and validate the obtained results in a larger system. Our assumption of identical processing times for all operations in a machine can be relaxed and its impact can be tested.

Another extension can be using genetic algorithm or tabu search for searching improved CF fits and compare the results in terms of quality and computation time to SPSA algorithm. SPSA is a tool for us that we use in this thesis but clearly other algorithms can be used for the same purpose.

In this thesis, we develop heuristics that use the dual solutions from the LP planning models to derive weights to influence weighted least-squares regression. We obtain significantly better expected profit by doing so but it is worth investigating extensions to our heuristic algorithms and obtain better performance using less computational time. Then the heuristics can become good candidates to use over computational intensive simulation optimization algorithms such as SPSA.

Another future direction can be to model a robust (Bertsimas and Thiele 2006) CF fit which can adjust itself to different demand realizations the production planning faces. Given the computational burden of the SPSA methods and heuristics, thus a robust approach without need of iterations can be very useful.

Another way of modeling CFs can be to use a nonlinear function instead of dividing the data into segments and using linear functions. Even though the resulting production planning model is nonlinear, it may take away the ambiguity due to segmentation and outer linearization of CFs.

In this thesis we develop multiple regression models. In those models, we omit the interaction terms in order to avoid nonlinear model. It would be an interesting extension to include significant interaction terms in the regression model and solve a nonlinear production planning model.

Another extension can be categorizing which machines need a CF fit and which can be modeled as a fixed lead time (Missbauer 2002). In our small scaled down model, we fit CF to each machines and computational time in LP is not an issue. If we consider a larger system, it might be effective to assign fixed lead time to machines that run on low utilization. Thus, CF fits can be only modeled for machines that are highly utilized. The study can be on how to differentiate between those machines that need CF and those that can be easily modeled with a fixed time.

LIST OF REFERENCES

AGNEW, C. 1976. Dynamic Modeling and Control of Some Congestion Prone Systems. *Operations Research,* 24**,** 400-419.

ALBEY, E., BILGE, Ü. & UZSOY, R. 2010. An Exploratory Study of Disaggregated Clearing Functions for Multiple Product Single Machine Production Environments. *Department of Industrial and Systems Engineering, Bogazici University*

ASMUNDSSON, J. M., RARDIN, R. L., TURKSEVEN, C. H. & UZSOY, R. 2009. Production Planning Models with Resources Subject to Congestion. *Naval Research Logistics,* 56**,** 142-157.

ASMUNDSSON, J. M., RARDIN, R. L. & UZSOY, R. 2006. Tractable Nonlinear Production Planning Models for Semiconductor Wafer Fabrication Facilities. *IEEE Transactions on Semiconductor Manufacturing,* 19**,** 95-111.

BANG, J. Y. & KIM, Y. D. 2010. Hierarchical Production Planning For Semiconductor Wafer Fabrication Based on Linear Programming and Discrete Event Simulation *IEEE Transactions on automation Science and Engineering,* 7.

BERTSIMAS, A. & THIELE, A. 2006. A Robust Optimization Approach to Inventory Theory. *Operation Research,* 54,150-168.

BILLINGTON, P. J., MCCLAIN, J. O. & THOMAS, L. J. 1983. Mathematical Approaches to Capacity-Constrained MRP Systems: Review, Formulation and Problem Reduction. *Management Science,* 29**,** 1126-1141.

BYRNE, M. D. & BAKIR, M. A. 1999. Production Planning Using a Hybrid Simulation-Analytical Approach. *International Journal of Production Economics,* 59**,** 305-311.

BYRNE, M. D. & HOSSAIN, M. M. 2005. Production Planning: An Improved Hybrid Approach. *International Journal of Production Economics,* 93-94**,** 225-229.

CONOVER, W. J. 1980. *Practical Nonparametric Statistics,* New York, John Wiley.

DAUZÈRE-PÉRES, S. & LASSERRE, J.-B. 1994. *An integrated approach in production planning and scheduling,* Berlin ; New York, Springer-Verlag.

DRAPER, N. R. & SMITH, H. 1981. *Applied regression analysis,* Wiley.

ETTL, M., FEIGIN, G., LIN, G. Y. & YAO, D. D. 2000. A Supply Chain Network Model with Base-Stock Control and Service Requirements. *Operations Research,* 48**,** 216-232.

FU, M. 1994. Optimization via simulation: A review. *Annals of Operations Research,* 53**,** 199-247.

GRAVES, S. C. 1986. A Tactical Planning Model for a Job Shop. *Operations Research,* 34**,** 552-533.

HACKMAN, S. T. & LEACHMAN, R. C. 1989. A General Framework for Modeling Production. *Management Science,* 35**,** 478-495.

HOLT, C. C., MODIGLIANI, F. & MUTH, J. F. 1956. Derivation of a Linear Rule for Production and Employment. *Management Science,* 2**,** 159-177.

HOLT, C. C., MODIGLIANI, F., MUTH, J. F. & SIMON, H. A. 1960. *Planning Production, Inventories and Work Force,* Englewood Cliffs, NJ, Prentice Hall.

HOPP, W. J. & SPEARMAN, M. L. 2001. *Factory Physics : Foundations of Manufacturing Management,* Boston, Irwin/McGraw-Hill.

HUNG, Y. F. & HOU, M. C. 2001. A Production Planning Approach based on Iterations of Linear Programming Optimization and Flow Time Prediction. *Journal of the Chinese Institute of Industrial Engineers,* 18**,** 55-67.

HUNG, Y. F. & LEACHMAN, R. C. 1996. A Production Planning Methodology for Semiconductor Manufacturing Based on Iterative Simulation and Linear Programming Calculations. *IEEE Transactions on Semiconductor Manufacturing,* 9**,** 257-269.

IRDEM, D. F., KACAR, N. B. & UZSOY, R. 2008. An Experimental Study of an Iterative Simulation-Optimization Algorithm for Production Planning. *In:* MASON, S. J., HILL, R., MOENCH, L. & ROSE, O. (eds.) *2008 Winter Simulation Conference.* Miami, FL.

IRDEM, D. F., KACAR, N. B. & UZSOY, R. 2010. An Exploratory Analysis of Two Iterative Linear Programming-Simulation Approaches for Production Planning. *IEEE Transactions on Semiconductor Manufacturing,* 23**,** 442-455.

JOHNSON, L. A. & MONTGOMERY, D. C. 1974. *Operations Research in Production Planning, Scheduling and Inventory Control,* New York, John Wiley.

KACAR, N. B., IRDEM, D. F. & UZSOY, R. 2012. An Experimental Comparison of Production Planning using Clearing Functions and Iterative Linear Programming-Simulation Algorithms. *IEEE Transactions on Semiconductor Manufacturing,* 25**,** 104-117.

KARMARKAR, U. S. 1989. Capacity Loading and Release Planning with Work-in-Progress (WIP) and Lead-times. *Journal of Manufacturing and Operations Management,* 2.

KAYTON, D., TEYNER, T., SCHWARTZ, C. & UZSOY, R. 1997. Focusing Maintenance Improvement Efforts in a Wafer Fabrication Facility Operating Under Theory of Constraints. *Production and Inventory Management***,** 51-57.

KEFELI, A., UZSOY, R., FATHI, Y. & KAY, M. 2011. Using a Mathematical Programming Model to Examine the Marginal Price of Capacitated Resources. *International Journal of Production Economics,* 131**,** 383-391.

KEKRE, S. 1984. The Effect of Number of Items Processed at a Facility on Manufacturing Lead Time. *Working Paper Series.* Rochester, NY: University of Rochester.

KIM, B. & KIM, S. 2001. Extended Model for a Hybrid Production Planning Approach. *International Journal of Production Economics,* 73**,** 165-173.

LIU, L., LIU, X. & YAO, D. D. 2004. Analysis and Optimization of Multi-stage Inventory Queues. *Management Science,* 50**,** 365-380.

MISSBAUER, H. 2002. Aggregate Order Release Planning for Time-Varying Demand. *International Journal of Production Research,* 40**,** 688-718.

MISSBAUER, H. 2007. Meta-Models of the Transient Behaviour of a Work Centre to Improve Clearing Function Models. *Department of Production and Logistics Management, University of Innsbruck.*

MISSBAUER, H. 2009. Models of the Transient Behaviour of Production Units to Optimize the Aggregate Material Flow. *International Journal of Production Economics,* 118**,** 387-397.

MODIGLIANI, F. & HOHN, F. E. 1955. Production Planning Over Time and the Nature of the Expectation and Planning Horizon. *Econometrica,* 23**,** 46-66.

ORLICKY, J. 1975. *Material Requirements Planning: the New Way of Life in Production and Inventory Management,* New York, McGraw-Hill.

PAHL, J., VOSS, S. & WOODRUFF, D. L. 2005. Production Planning with Load Dependent Lead Times. *4OR: A Quarterly Journal of Operations Research,* 3**,** 257-302.

RIAÑO, G. 2003. *Transient Behavior of Stochastic Networks: Application to Production Planning with Load-Dependent Lead Times.* Georgia Institute of Technoogy.

SCHOEMIG, A. K. The Corrupting Influence of Variability in Semiconductor Manufacturing. *In:* FARRINGTON, P. A., NEMBHARD, H. B., STURROCK, D. T. & EVANS, G. W., eds. Winter Simulation Conference, 1999. 837-842.

SELÇUK , B., FRANSOO, J. C. & DE KOK, A. G. 2007. Work in Process Clearing in Supply Chain Operations Planning. *IIE Transactions,* 40**,** 206-220.

SPALL, J. C. 1998. Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization. *IEEE Transactions on Aerospace and Electronic Systems*, 34.

SPEARMAN, M. L. 1991. An Analytic Congestion Model for Closed Production Systems with IFR Processing Times. *Management Science,* 37**,** 1015-1029.

SPITTER, J. M., HURKENS, C. A. J., DE KOK, A. G., LENSTRA, J. K. & NEGENMAN, E. G. 2005. Linear Programming Models with Planned Lead Times for Supply Chain Operations Planning. *European Journal of Operational Research,* 163**,** 706-720.

SRINIVASAN, A., CAREY, M. & MORTON, T. E. 1988. Resource Pricing and Aggregate Scheduling in Manufacturing Systems. *Graduate School of Industrial Administration, Carnegie-Mellon University.* Pittsburgh, PA.

TARDIF, V. & SPEARMAN, M. L. 1997. Diagnostic Scheduling in Finite-Capacity Production Environments. *Computers and Industrial Engineering,* 32**,** 867-878.

VOLLMANN, T. E., BERRY, W. L., WHYBARK, D. C. & JACOBS, F. R. 2005. *Manufacturing Planning and Control for Supply Chain Management. ,* New York, McGraw-Hill.

VOSS, S. & WOODRUFF, D. L. 2003. *Introduction to Computational Optimization Models for Production Planning in a Supply Chain,* Berlin ; New York, Springer.

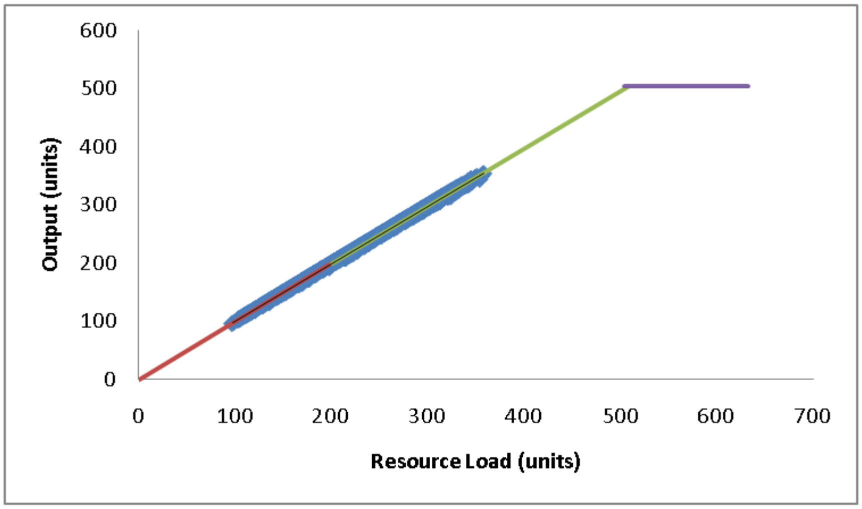
WOODRUFF, D. L. & VOSS, S. A model for multi-stage production planning with load dependent lead times. Proceedings of the International Conference on System Sciences, 2004 Hawaii, United States. 1425-1434.

ZÄPFEL, G. & MISSBAUER, H. 1993a. Production Planning and Control (PPC) Systems Including Load-Oriented Order Release - Problems and Research Perspectives. *International Journal of Production Economics,* 30**,** 107-122.

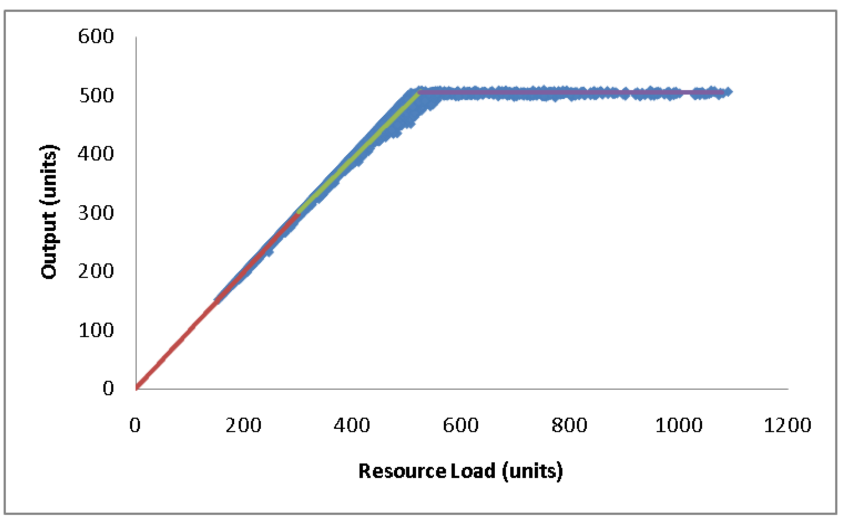
ZIJM, W. H. M. & BUITENHEK, R. 1996. Capacity Planning and Lead Time Management. *International Journal of Production Economics,* 46-47**,** 165-179.

APPENDIX

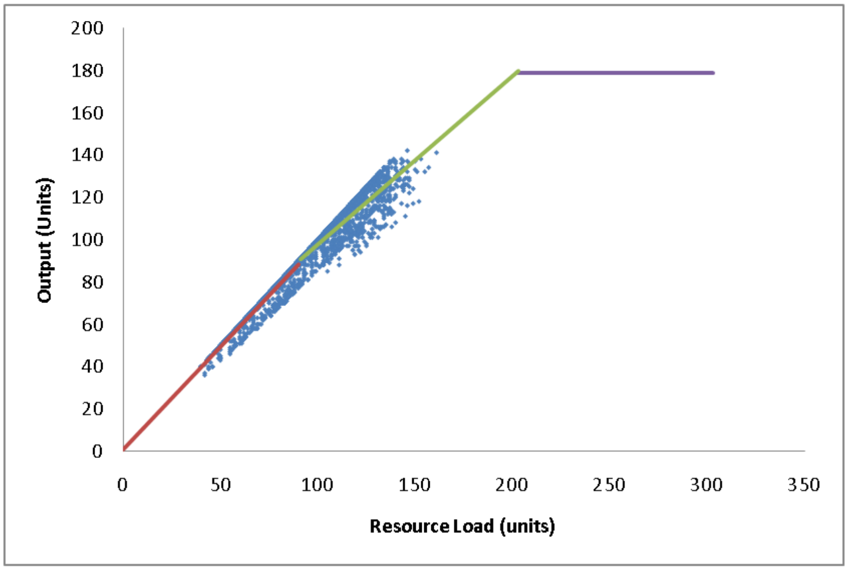
Appendix: The Fitted CFs of Selected Machines



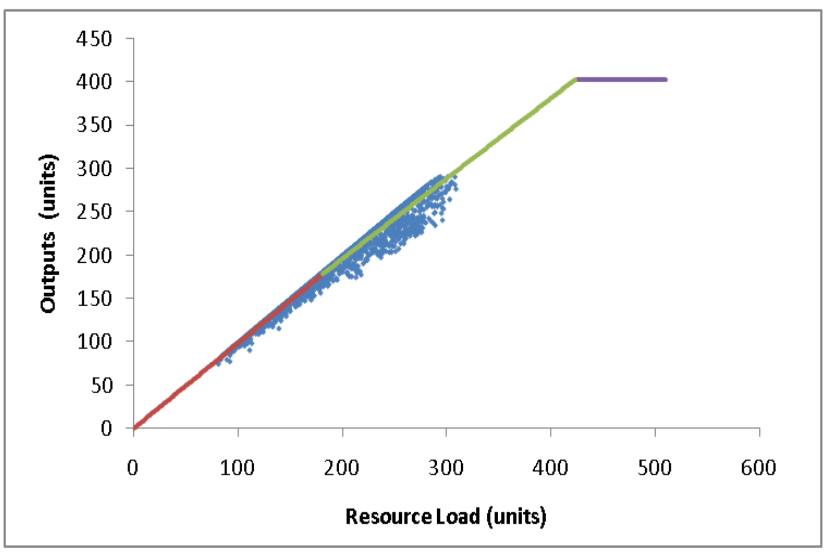
Fitted CF of Machine 1



Fitted CF of Machine 4



Fitted CF of Machine 3



Fitted CF of Machine 7