SIMULATION OPTIMIZATION OF DUAL RESOURCE CONSTRAINED CONWIP CONTROLLED LINES USING RESPONSE SURFACE METHODOLOGY

A Thesis Submitted to the

Graduate School of Natural and Applied Sciences of

Dokuz Eylül University

In Partial Fulfillment of the Requirements for

the Degree of Doctor of Philosophy in Industrial Engineering, Industrial Engineering Program

by

Gökalp YILDIZ

September, 2003

İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We certify that we have read the thesis, entitled **"Simulation Optimization of Dual Resource Constrained CONWIP Controlled Lines using Response Surface Methodology"** completed by **Gökalp YILDIZ** under supervision of **Assoc. Prof. Dr. Semra TUNALI** and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Semra TUNALI
Supervisor

Prof. Dr. Hasan Eski
Jury Member

Yrd. Doç. Dr. Adil Alpkoçak
Jury Member

Prof. Dr. Serdar Korukoğlu
Jury Member

Yrd. Doç. Dr. Mehmet Çakmak
Jury Member

Approved by the
Graduate School of Natural and Applied Sciences

Prof.Dr. Cahit HELVACI
Director

ACKNOWLEDGEMENTS

I would first like to thank my advisor Assoc. Prof. Dr. Semra TUNALI for her patience, input and feedback during the conduction of this thesis. She spent many valuable hours reading early drafts of this thesis and made numerous improvements to the manuscript.

I wish to give special thanks to the members of my committee, Prof. Dr. Hasan ESKİ and Assist. Prof. Dr. Adil ALPKOÇAK for their constructive suggestions. I would like to thank Prof. Dr. Hasan ESKİ once more time as our department chair, for his thoughtful and sincere interest in completing this thesis. My sincere thanks also go to Prof. Dr. Cevdet ÖĞÜT, who is the former member of my committee, for his valuable comments on this research.

I would like to acknowledge Dr. Özcan KILINÇCI and Dr. Şeyda TOPALOĞLU for their encouragement and constant support. I would also like to thank my colleagues, Özgür ESKİ, Ceyhun ARAZ, Özlem UZUN, Emrah EDİS, and Rahime SANCAR for their special helps and interests.

I would like to thank the parents of my wife for their endless helps in taking care of my little daughter, Beril. Her sweetie smiles made me stronger in overcoming the problems I faced during the conduction of this thesis. I would like to thank my parents for their neverending supports, encouragements, helps and trust in me. I would also like to thank other members of my family. Their love made good times better and troubles easier to overcome. Finally, if there were to be a dedication of this thesis, it would be to my father, Mehmet, and to my little daughter, Beril.

Gökalp YILDIZ

ABSTRACT

This thesis deals with the use of flexible workers in CONWIP controlled serial manufacturing lines having parallel service facilities. The main objective of this thesis is to propose a worker flexibility policy for this type of system under saturated demand. The primary issue in developing this worker flexibility policy is to decide on the length of periodic controls for evaluating the current system status. To determine the optimum level of this decision variable, we developed a two-stage procedure involving simulation optimization using Response Surface Methodology. In implementing this two-stage procedure, we used a cost-based performance measure. Hence, another objective of this thesis is to develop a unit penalty cost function. To evaluate the performance of the proposed policy, we compared it with other alternative policies widely used in Dual Resource Constrained systems. The simulation models which serve to evaluate the proposed and alternative policies have been developed in ARENA V2.2.

Keywords: Dual Resource Constrained Systems (*DRC*), CONWIP, Simulation Optimization, Response Surface Methodology (*RSM*).

· ÖZET

Bu tezin konusu, paralel makinalara sahip CONWIP kontrollu seri imalat hatlarında esnek işçi kullanımıyla ilgilidir. Bu tezin ana amacı doymuş talep koşulları altında çalışan bu tipte bir sisteme yönelik bir işçi esneklik politikası önermektir. Bu işçi esneklik politikasının geliştirilmesine ilişkin birincil karar değişkeni, mevcut sistem durumunun hangi periyotlarla kontrol edildiğini gösteren periyodik kontrol süresidir. Periyodik kontrol süresinin optimum seviyesini belirlemek için iki aşamalı bir benzetim optimizasyonu yöntemi geliştirilmiş ve Yanıt-Yüzey Metodolojisini de içeren bu iki aşamalı çözüm yönteminin oluşturulmasında maliyet tabanlı bir performans ölçüsü kullanılmıştır. Sonuç olarak, bu tezin diğer bir amacı bu maliyet tabanlı ceza fonksiyonunun geliştirilmesidir. Önerilen işçi esneklik politikasının performansını değerlendirebilmek için önerilen politika, Çift Kaynak Kısıtlı sistemlerde sıklıkla kullanılan alternatif politikalarla karşılaştırılmıştır. Önerilen politika ve alternatif politikaların benzetim modelleri ARENA V2.2'de geliştirilmiştir.

Anahtar sözcükler: Çift Kaynak Kısıtlı Sistemler, CONWIP, Benzetim Optimizasyonu, Yanıt-Yüzey Metodolojisi.

CONTENTS

Chapter Three
A NEW WORKER FLEXIBILITY POLICY FOR CONWIP LINES

Chapter Four
EXPERIMENTAL STUDIES

Chapter Five

CONCLUSION AND FUTURE RESEARCH

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE
INTRODUCTION

1.1 Introduction

Fast and dramatic changes in customer expectations, competition, and technology cause uncertain environments in the global markets of the twenty first century. The success of a manufacturing firm in an uncertain environment depends on how effectively the firm responds to these changes in the markets. In general, the researchers and the manufacturing managers contend that the most important concept to cope with this uncertainty is the manufacturing flexibility. The manufacturing flexibility is the ability of the firm to manage production resources and uncertainty to meet customer requests (Zhang et al., 2003). There are ten dimensions of the manufacturing flexibility (Koste & Malhotra, 1999);

- *Machine Flexibility*: The number and variety of operations a machine can execute without incurring high transition penalties or large changes in performance outcomes.
- *Worker Flexibility*: The number and variety of tasks/operations a worker can execute without incurring high transition penalties or large changes in performance outcomes.
- *Material Handling Flexibility*: The number of existing paths between processing centers and the variety of material which can be transported along those paths without incurring high transition penalties or large changes in performance outcomes.
- *Routing Flexibility*: The number of products which have alternate routes and the extent of variation among the routes used without incurring high transition penalties or large changes in performance outcomes.

- *Operation Flexibility*: The number of products which have alternate sequencing plans and variety of the plans used without incurring high transition penalties or large changes in performance outcomes.
- *Expansion Flexibility*: The number and variety of expansions which can be accommodated without incurring high transition penalties or large changes in performance outcomes.
- *Volume Flexibility*: The extent of change and the degree of fluctuation in aggregate output level which the system accommodates without incurring high transition penalties or large changes in performance outcomes.
- *Mix Flexibility*: The number and variety of products which can be produced without incurring high transition penalties or large changes in performance outcomes.
- *New Product Flexibility*: The number and variety of new products which are introduced into production without incurring high transition penalties or large changes in performance outcomes.
- *Modification Flexibility*: The number and variety of product modifications which are accomplished without incurring high transition penalties or large changes in performance outcomes.

It must be noted that in all of the dimensions, achieving manufacturing flexibility requires additional investments in advanced manufacturing technology and workforce which cause additional costs. So, the trade-offs must be made between the investments and the performance improvement obtained by increasing manufacturing flexibility. In short, more flexibility does not always mean a more economic solution (Beach et al., 2000). This thesis directly deals with the worker flexibility which is the second dimension of the manufacturing flexibility in the list given above. In next section, the worker flexibility will be explained briefly.

1.2 Worker Flexibility

Although, the manufacturing flexibility literature tends to emphasize equipment flexibility and neglect the potential impacts of workers, the workforce plays a vital role in most production processes. Especially, in Dual Resource Constrained (*DRC*) literature, the key element is the worker flexibility (Zhang et al., 2003). A *DRC* production system is one in which all equipment in the shop is not fully staffed and, furthermore, the workers can be transferred from one piece of equipment to another as needed (Treleven, 1989). So, the production in a *DRC* system is limited by both worker and machine. The *DRC* concept was introduced first in the study of Nelson (Nelson, 1967). Although, *DRC* is not a new concept, the advent of Just in Time (*JIT*) to the production environments in 1980's increased the popularity of the *DRC* Systems in the field of Production and Operations Management because *JIT* concept gives the workers broader task responsibility, team participation, and decision-making empowerment. Especially, nowadays' buzzwords such as agility, responsiveness, and leanness increase the need for more skillful workers. Therefore, the value of a worker is more dependent on how many skills he has than how fast he can perform a specific task in today's production environments (Oyen et al., 2001).

The concept of "Flexible Worker" is self-explanatory. The worker who is capable of carrying out more than one task can be called "Flexible Worker". The flexibility of the worker is provided by training the workers in different kinds of tasks. These tasks may be the responsibilities of more than one machine according to his degree of flexibility or the responsibilities of different manual assembly operations in a serial production line, etc. The level of the flexibility of the worker depends on the level of training. At this point, the question "why do we use flexible workers?" may be asked. The first reason may be that "The Flexible Worker Concept" makes the task independent from the worker. For example, in the case of absenteeism, the task may be carried out by another worker. The second reason may be that staffing each machine by one worker may not be economically acceptable. Another reason for adopting worker flexibility is that it allows the manager to balance the workloads of the workstations over time dynamically. Due to the stochastic nature of the

production process, the workloads of the workstations may be changed during the production even if the whole process is balanced. For example, when the congestion of a workstation begins to increase, another worker who is idle at that time can be transferred to that workstation.

In general, the literature including flexible workers can be classified in two groups; The studies employing flexible workers in job shops and in serial manufacturing lines. A typical *DRC* job shop consists of machines, workstations, and departments hierarchically. A group of machines constitutes a workstation, a group of workstations constitutes a department and, a group of departments constitutes a job shop itself. In the case of serial manufacturing lines, the line consists of serially connected workstations each having one machine. Furthermore, in both of these production systems the machines are not fully staffed. Based on the review of current literature, we could state that the *DRC* job shops have received much more attention than serial manufacturing lines. To improve the clarity of the material given in the following Chapters, the terms which are most widely used in these areas are given as follows;

When/Where Rule Pairs: Represent the dynamic worker assignment heuristics. These rules help the worker to decide on when and where to move (Job shops).

Centralized/Decentralized Rules: These rules represent the types of "When Rules". If the "When Rule" is "When the worker becomes idle due to the empty queue at the current workstation", this rule is called "The Decentralized Rule". If the "When Rule" is "After the completion of every job", this rule is called "The Centralized Rule" in *DRC* context (Job Shops).

Full Cross Training/Partial Cross Training: Full cross training is to provide a training programme to the workers to make them fully flexible, that is, the manager can assign a worker to all kinds of tasks in the shop. In the case of partial cross training, the worker can be assigned to more than one task but to not all the tasks (Job shops and Serial Lines).

Homogenous Workers/Heterogeneous Workers: Homogenous workers are identical workers meaning that they can make a specific task at the same pace. However, heterogeneous workers have not similar skills to make a task at the same speed (Job shops and Serial Lines).

Preemptive job/Nonpreemptive job: Preemptive job is a job that it can be transferred to another worker whether it has already started. However, nonpreemptive jobs should be completed once they have started (Serial Lines). Although, it was not stated in the literature about *DRC* job shops, the jobs are nonpreemptive.

Collaborative job/Noncollaborative job: Collaborative job means that more than one worker can work on it simultaneously. Noncollaborative jobs can not be divided into smaller parts (Serial Lines). Although, it was not stated in the literature about *DRC* job shops, the jobs are noncollaborative.

Let us assume that the serial manufacturing line consists of serially connected workstations each having more than one machine. This system is the special case of the job shops and the serial manufacturing lines. If the number of workers in this system is lower than the total number of machines, this system will be *DRC*. Currently, there is only one study related to this issue (Oyen et al., 2001). Where a serial manufacturing line with workstations each having parallel machines and operating under CONstantWork-In-Process (CONWIP) control was evaluated and two worker flexibility policies in the case of collaborative and noncollaborative jobs were proposed. These policies are called "Expedite Policy (*EP*)" and "Pick and Run Policy (*PR*)". The authors proved that the *EP* is optimal under every sample path for collaborative jobs and they also showed that the *PR* is optimal when the CONWIP level is equal to at least the number of workers for noncollaborative jobs. The CONWIP is a control mechanism which allows the *WIP* level to be constant. It was introduced by Spearman (Spearman et al., 1990) in 1990. Under the CONWIP control, the process of a new job at the beginning of the line is possible if and only if a job completes its process at the end of the line. The CONWIP systems are

evaluated by Closed Queuing Networks (*CQN*) in the case of single resource. However, if the system is *DRC*, and dynamic worker assignment heuristics are under consideration, the analytical evaluation of CONWIP systems is impossible in most cases. An alternative to analyze these kinds of systems is simulation which avoids the restricted assumptions of mathematical models by representing the dynamics of a system over time.

In the study of Oyen et al. (Oyen et al., 2001), one of the most important assumptions is that all the workers can be pooled in every workstation providing that ample equipment or machines are available for all of the workers. This is the ample capacity case. In the ample capacity case, it is assumed that there are sufficient equipment or machines at each workstation to ensure that a worker is never blocked for lack of a machine (Hopp & Spearman, 2001). For example, if the line has five workstations and five workers, all the workers can be pooled in the first workstation, the second workstation, and so on. They showed that their worker flexibility policies minimize the cycle time of the line when the CONWIP level is greater or equal to the total number of workers under the assumption given above. However, in real production environments, it may not be possible to supply sufficient equipment at each workstation. In most cases, it is impossible to place all the workers in a same workstation. These impossibilities are caused by some inefficiencies or economical reasons. First, it may not be economical to buy more than one jig, tool, fixture, test equipment or machine, etc. Second, the layout of the line may not allow pooling of the workers. Even if it is possible to place all workers in a same workstation, this may cause inefficiencies. Moreover, continuously changing the place of the worker (i.e., after each completion of job) may cause the motivation problems, and may increase the quality related problems. Lastly, if the moving times of the workers are not negligible due to the layout, moving after each job completion may decrease the performance of the line dramatically.

From this point of view, in most cases, unconstrained workstation capacity (i.e., ample capacity) is not an applicable concept. Therefore, this is not a realistic assumption in most of the production systems. If this assumption is violated, the *EP*

and *PR* policies will not produce optimal results due to the worker blockages at the workstations. Hence, this thesis is based on the violation of the above assumption given in the study of Oyen et al (2001). If all the workers can not be pooled in a workstation, a new flexibility policy should be introduced. This thesis exactly concentrates on this type of problem involving noncollaborative jobs with homogenous workers. In next section, the statement and the importance of the problem will be given.

1.3 Statement and Importance of Problem

The problem stated in this thesis is to develop a new worker flexibility policy to optimize the effective capacity of the CONWIP controlled serial manufacturing line under the workstation capacity constraint. The workstation capacity constraint may refer to the limited availability of equipment or machines at the workstations. As it was mentioned in Section 1.1, increasing the level of manufacturing flexibility causes additional costs. Hence, there is a trade-off between increasing the effective capacity of the CONWIP line and the cost caused by increasing the capacity. So, there should be a cost function to evaluate the alternative flexibility policies. In general, the system considered has the following properties;

- The system consists of M workstations in tandem configuration. The production and inventory control is provided by CONWIP and the demand is saturated (i.e., the capacity of the system is not constrained by demand). The CONWIP level (i.e., the number of CONWIP cards) is K.
- Each workstation consists of identical parallel machines. The number of parallel machines at workstation "i" is denoted as c_i for all i=1,...,M. Each workstation has its input and output buffers. The output buffer of workstation "i" serves as the input buffer of workstation "i+1". Since the demand is saturated, there is no output buffer at the last workstation. Unlimited availability of the raw material is assumed in front of the first workstation

- Machines are not subject to breakdowns.
- Each identical machine in a workstation can be operated if and only if a worker is available. The total number of machines in the line is greater than the total number of workers (i.e., *N*) attended to the line, (i.e., *DRC* system).
- The system is capable of producing just one type of job (i.e., single class job).
- The jobs are transferred between the workstations asynchronously. There is no time delay for the transfer of jobs.
- The jobs are processed at the workstations with respect to the rule "First in System First Serve (*FISFS*)".
- Any information, such as CONWIP card, is transmitted instantly.
- The workers are homogenous.
- The workers are full cross-trained.
- The jobs are noncollaborative.
- The preemption of job is not allowed.

The main contribution of this thesis is to propose a worker flexibility policy for a CONWIP controlled serial manufacturing line producing a single class job and operating under the saturated demand and the conditions given above. The primary issue in developing this worker flexibility policy is to decide on the length of periodic controls for evaluating the current system status. To determine the optimum level of this decision variable, we developed a two-stage procedure involving simulation optimization using Response Surface Methodology (*RSM*) and we used a cost-based performance measure in implementing this two-stage procedure. Hence, another contribution of this thesis is to develop a unit penalty cost function. This thesis provides a novel way to integrate the different research areas, such as *RSM*, *DRC* systems, and CONWIP control, which give valuable insight into the development of this worker flexibility policy. We hope that this flexibility policy will help to more effectively implement the decisions, such as when and where to move the workers while the line is operating.

In summary, the aim of this thesis is to increase the effective capacity of a CONWIP controlled serial manufacturing line at a minimum cost by applying worker flexibility. In next section, the thesis outline is given.

1.4 Thesis Outline

As it was mentioned before, this thesis focuses on the use of flexible workers in CONWIP controlled serial manufacturing lines. The outline of this thesis can be summarized as follows;

In Chapter Two, the background information and the literature related to our problem are given. In general, the problem dealt with in this thesis has four main topics, which are *DRC*, CONWIP, Simulation Optimization and *RSM*. First, the *DRC* systems, their characteristics, dynamic worker assignment heuristics, and other special topics on *DRC* subject are explained in detail and the studies on *DRC* are classified. Second, the CONWIP, which is a new approach to the production and inventory control mechanisms, is investigated and the relevant studies are classified. Lastly, the simulation optimization and *RSM* are explained.

In Chapter Three, the cost function which was developed to compare the alternative flexibility policies is explained. The proposed worker flexibility policy and its parameters are given. Lastly, the two-stage simulation optimization methodology to minimize the cost function developed is explained.

In Chapter Four, the experimental studies which were carried out to compare alternative flexibility policies under different operating conditions, the factors affecting the performance of the proposed policy, and an application of the two-stage simulation optimization methodology are given.

In Chapter Five, the results of the experimental studies given in Chapter Four are summarized. The insights gained from these experimental results are discussed. At last, the future research possibilities about the problem and its extensions are given.

CHAPTER TWO
BACKGROUND INFORMATION AND LITERATURE REVIEW

2.1 Introduction

The subject of this thesis is related to four main issues; *DRC*, CONWIP, Simulation Optimization and *RSM*. First, the literature related to the *DRC* is given. The *DRC* systems, their characteristics, dynamic worker assignment rules, and other special topics on *DRC* subject are explained in detail and the studies on *DRC* are classified. Second, the CONWIP, which is a new approach for production and inventory control, is investigated and the studies employing both the *DRC* systems and the CONWIP control mechanism are discussed. Third, the place of *RSM* in the area of Simulation Optimization and the related literature are given. Lastly, the *RSM* is explained.

2.2 Dual Resource Constrained Systems

A *DRC* production system is one in which all equipment in the system is not fully staffed and, furthermore, the workers can be transferred from one piece of equipment to another as needed. The production in a *DRC* system is limited by both worker and machine. So, the most important characteristic of *DRC* system is the flexible workers with moving capabilities. Although, the term *DRC* is used in the literature just for job shops, any production system which is not fully staffed can be considered as *DRC*. Interestingly, we noted that the two review studies in this area ((Treleven, 1989) - (Hottenstein & Bowman, 1998)) did not categorize the studies on serial manufacturing lines with flexible workers under the name of *DRC*.

The studies on *DRC* can be categorized into two groups; The ones using flexible workers in job shops and the ones using flexible workers in serial manufacturing lines. A typical job shop consists of machines, workstations, and departments, hierarchically. A group of machines constitutes a workstation, a group of workstations constitutes a department and, a group of departments constitutes a job shop itself. In the case of serial manufacturing lines, the line consists of serially connected workstations each having one or more identical machines. In both of these production systems, the machines are not fully staffed. Based on the review of the current literature, we could state that the *DRC* job shops have received much more attention than serial manufacturing lines.

2.2.1 Dual Resource Constrained Job Shops

The distinguishing feature of job shop production is low volume. The manufacturing lot sizes are small. Job shop production is commonly used to meet the specific customer orders, and there is a great variety in the type of work done (i.e., great variety in routings). Therefore, the production equipment must be flexible and general purpose to meet the requirements of a great variety of jobs. Moreover, the skill level of workers in job shops must be relatively high so that a range of different work assignments can be carried out more effectively (Groover, 2001).

A considerable amount of research has been conducted on decision rules used in operation of job shops. These studies can be categorized into two streams; One stream has focused on the machine limited job shops. A machine limited job shop is one in which all equipment is fully staffed, that is, the capacity of the job shop is limited just by the number of machines. The research on machine limited job shops primarily has focused on dispatching rules (Treleven, 1989) and most of these studies neglected the potential impact of workers. However, the workers play an important role in increasing the effectiveness of job shops. So, to fill the gap in this area, another group of studies have examined the *DRC* job shops where the capacity is limited both by machine and worker. Since the number of machines is greater than the number of workers in a typical *DRC* job shop, the workers move between the

workstations. So, in a *DRC* job shop, besides job dispatching, there is another issue to be dealt with; "When" and "Where" to move the workers. Therefore, the analysis of a *DRC* job shop is more complicated than the analysis of a machine limited job shop.

There are a lot of studies on job shops explaining the behavior of the flexible workers under different operating conditions. One of the most cited studies is the *DRC* job shop review of Treleven (Treleven, 1989). In this study, the author gives the classification of *DRC* studies according to the different characteristics. In another review study, (Hottenstein & Bowman, 1998), the authors review sixteen studies to gain insight into what has been gained about the use of flexible workers in *DRC* systems. In both of these review studies, it was emphasized that the *DRC* job shop studies were focused primarily on hypothetical cases to evaluate different operating conditions and decision rules. The typical characteristics which need to be identified to carry out a research on a hypothetical *DRC* job shop are;

- The Characteristics of Shop
  - *The number of workers*: The total number of workers attended to the shop.
  - *The number of machines*: The total number of machines in the shop.
  - *Staffing Level*: The ratio between the number of workers and the number of machines.
  - *The number of workstations*: The number of identical parallel machine groups.
  - *The number of departments (or divisions)*: The number of workstation groups.
  - *Worker utilization*: The average worker utilization in the shop.
- The Characteristics of Operators
  - *Worker transfer time*: The total time required to move between workstations and to setup the new machine.
  - *Degree of worker flexibility*: Represents the level of cross training (full or partial)
    - *Single level worker flexibility*
    - *Multi level worker flexibility*
  - *Type of worker flexibility*: Represents whether the workers are homogenous or

heterogeneous.

- The Characteristics of Jobs
  - * *Routing pattern*: Represents how the routings of jobs are generated.
  - * *Due date assignment*: Represents how the due dates are generated
- The characteristics of rules
  - * *Dispatching*: Represents how the jobs are selected by workers for operation.
  - * *"When" rule*: The rule for determining the time of worker transfer.
  - * *"Where" rule*: The rule for determining the workstation where the worker will be transferred.

As it was mentioned before, these characteristics are necessary to identify the typical properties of a hypothetical *DRC* shop. The most widely used tool for analyzing *DRC* job shops has been simulation. These characteristics given above can be explained as follows;

The characteristics of shop: A typical shop consists of machines, workstations, and departments (or divisions). Identical parallel machines constitute a workstation and, each workstation (or several) constitutes a department (or a division). The workers are transferred between the workstations (not machines). In some cases, the transfer between departments may be restricted. The staffing level of shop is a ratio between the number of workers and the number of machines in a shop. For example, if the number of machines and workers in a shop are 24 and 12, respectively, the staffing level of this shop is 50%. The worker utilization depends on mean service times and arrival rate (e.g 90% average worker utilization).

The characteristics of Operators: The first characteristic of a worker is whether the transfer of a worker between the workstations requires time or not. This transfer delay may include the transfer time and the setup time for the new machine. The second characteristic is the degree of the worker flexibility. As it was given before, the workers may use different types of machines if they are cross-trained. If the workers are full cross-trained, they can operate all the machines in the shop (i.e., they can be transferred to every workstation and to every department in the shop). If the

workers are partial cross-trained, they can not operate all the machines in the shop, so, they can be transferred just to the workstations for which they are trained. This characteristic can be divided into two parts; the single-level worker flexibility and the multi-level worker flexibility. In single-level worker flexibility, each worker is cross-trained at the same number of workstations or departments. For example, if the shop consists of six workstations each having two identical parallel machines, and if the number of workers is six (i.e., 12 machines 6 workers in the shop, 50% staffing level), the degree of worker flexibility of two represents that each worker can be transferred to just two workstations. In multi-level worker flexibility, each worker is not cross-trained at the same number of workstations. For example, two workers can be transferred to just two workstations, the other four workers can be transferred to three workstations. Then the degree of worker flexibility in this system would be (2*2+4*3)/6=2.67. Multi-level worker flexibility allows non-integer degree of worker flexibility. The third characteristic is the type of the flexibility. If the workers are homogenous, the same level of cross training provides same level of ability. So, the workers can operate the machines at same level of efficiency. If the workers are heterogeneous, they do not have the same efficiency levels.

The characteristics of job: The first characteristic of a job is the job routing pattern. Since the shops are hypothetical, the routings of jobs should be generated. The routings of jobs can be between the routing of pure job shop and the routing of pure flow shop. The routing pattern in pure job shops is as follows; each job selects the first workstation randomly with equal probability. The job, then, can be processed at the workstations with equal probability or exits from the system. Jobs can be processed at same workstation more than once in pure job shops. In pure flow shops, the jobs are processed at each workstation in sequence. This is same as serial manufacturing line. The second characteristic is the due date assignment. Since one of the most important performance measures is related to tardiness or lateness of jobs in job shop type production, due dates should be assigned to the jobs. For example, Total Work Content (*TWC*) is one of the due date assignment method. In this method, the due date of a job is calculated as the sum of arrival time of job and the allowance factor multiplied by the total processing time of the job. The allowance

factor is used for setting tight or loose due dates.

The characteristics of rules: In *DRC* shops, three types of rules are used for coordinating the flow of jobs and workers. The first one is dispatching rule. When the worker finishes his current job at his current machine, he selects a new job from the queue of the current workstation if there is any. Dispatching represents this type of selection criterion (e.g., Earliest Due Date (*EDD*), the worker selects the job which has the earliest due date). The second one is "when" rule. The "when" rule represents the eligibility of worker for transfer. For example, if the worker is eligible for transfer when he finishes his current job, this "when" rule is called as centralized control. If the worker is eligible for transfer when he completes all the jobs waiting in the queue of the current workstation, the rule is called as decentralized control. The third and the last one is "where" rule. It represents how the target workstation is selected by the worker for transfer (e.g., the largest number of jobs in queue (*LNQ*), the worker is sent to the workstation which has the largest number of jobs in its queue).

The literature review on *DRC* job shops was based on the typical characteristics given above. The first hypothetical study on *DRC* shops is Nelson's (Nelson, 1967). The model shop in this study consists of two workstations each having two identical machines. The number of workers is two. It must be noted that the concept of "worker efficiency" in *DRC* literature was first used by Nelson (i.e., fundamentals of heterogeneous workers). He treated the timing of worker transfers probabilistically. A parameter, d, represented the probability of a worker being eligible for a transfer when the job he currently working on is completed. In another study, (Nelson, 1970), Nelson evaluated the effects of centralized worker assignment for heterogeneous workers (i.e. workers are partially efficient in some workstations). The worker assignment rules were first expressed as "when/where rule pairs" by Fryer (Fryer, 1973). He also extended the shop structure by adding departments (or divisions). In another study (Fryer, 1974), Fryer evaluated the effects of a new parametric "when" rule. The "when" rule that he suggested works as follows; The rule considers the workers eligible for transfer on job completion only if there were q or fewer jobs in

queue. Hogg et al. gave the network approach for multiple resource constrained systems (Hogg et al., 1975).

Fryer (Fryer, 1975) evaluated the effects of shop size and worker flexibility in *DRC* job shops. The fixed characteristics of his hypothetical *DRC* shop are the followings; Staffing level is 50%. Average worker utilization is 90%. Worker transfer times are negligible. Single level worker flexibility was assumed. Workers are homogeneous. Job routing pattern is hybrid (i.e., between pure job shop and pure flow shop). In hybrid job routing that he used, the job routings are generated randomly with the probabilities being $1/(M+1)$ that a job will exit from the shop or will transition to a particular one of the M workstations through which it has not previously been routed. Due date assignment rule was not used because there were no performance measure related to tardiness. There are four experimental factors in this study. The experimental factors are the shop size and the organizational structure (1), dispatching (2), when (3), and where rules (4). The shop size and the organizational structure has four levels; 1. One department, two workstations, four machines (i.e., there are four machines, and two workers in the shop totally. This is same as Nelson's model. Workers are full cross-trained and can move between these two workstations). 2. One department, four workstations, eight machines (i.e., there are eight machines, and four workers in the shop. Workers are full cross trained). 3. Three departments each consists of four workstations (i.e., there are twenty four machines, and twelve workers in the shop. Workers are full cross- trained. Workers can move between the departments as well as workstations.). 4. Three departments each consists of four workstations (i.e., there are twenty four machines, and twelve workers in the shop. Workers are partial cross-trained. Workers can move between the workstations, the transfers between the departments are restricted). There are two factor levels for dispatching. These are "First-in-System-First-Served (*FISFS*), and Shortest Processing Time (*SPT*). The factor levels for "when" rules are centralized and decentralized controls. There are also two levels of "where" rule; to the workstation containing the job that has been in the shop for the longest period of time (*FISFS*), to the workstation which has the longest queue (*LNQ*). There is another "where" rule in this study to provide transfers between the departments; If the

workstation assignment rule results in the worker being idle, he is then assigned to the department having the most total jobs in queue at all its workstations. If a workstation is already fully staffed, then the jobs in queue at that workstation are not included in the total. This rule is used in the third level of "the shop size and the organizational structure". The performance measures used in this study are mean flow time, flow time variance, and the number of transfers. Based on the experiment results, he concluded that the direction of change in flow time measure can be generalized across various combinations of shop size and worker flexibility.

The study of Weeks and Fryer (Weeks & Fryer, 1976) is the extended version of the previous study given above (Fryer, 1975). The hypothetical *DRC* shop which they used is the same as the fourth level of "the shop size and the organizational structure". So, the shop consists of three departments and the workers can not move between the departments. They extended their model by including the due date assignment rule. They also included the due date based performance measures and used four experimental factors which are the dispatching rules, when-where rules and the due date assignment rules. For dispatching rules, they used three levels; *FISFS*, *SPT*, and Least Slack per Remaining Number of Operations (i.e., *SPR*, selects the job that has the lowest ratio of job slack "due date minus current date minus total remaining processing time" to the number of remaining operations). The "when" rules are centralized and decentralized. The "where" rules are *FISFS*, *LNQ*, *SPR*. For due date assignment, they used two methods for generating four levels of due date assignment rules. While the first method was "Constant" rule, the second method was total work content (*TWC*). So, their experiment consisted of 72 design points. The performance measures in this study are the mean flow time, variance of flow time, mean lateness, variance of lateness, proportions of jobs late, and total number of transfers. They used regression analysis to estimate the effects of these factors and stated that the most important factor is the due date assignment for the performance measures related to the lateness. The *SPT* decreases the mean flow time and its variance. The "when" rule affects the number of transfers. The effects of "where" rules were not found as significant.

The worker transfer delays were first considered in the study of Gunther (Gunther, 1979). In previous studies, the effects of worker assignment rules were measured by total number of transfers. However, since there was not a time delay for worker transfers in these studies, the productive capacities of the workers were not changed. In this study, the workers were not considered as productive while they were moving. The hypothetical system in this study consists of one workstation having two identical parallel machines. The worker utilization was estimated as 75%. Since there was just one worker in the system and the system was single stage, the characteristics such as degree and type of worker flexibility, and routing pattern were not applicable. The experimental factors in this study were dispatching rules, the worker transfer delays, "when" and "where" rules. He employed *FISFS* and *SPT* as dispatching rules. The "when" rule used in this study is as follows; A server is allowed to change his current machine when he is idle. He may also change at any time t if his job has been completed, jobs are in queue at other machine, and $t\text{-}t_0 \geq d$. Here t_0 is the arrival time of the worker at his current machine and d is a parametric weight. The "where" rules suggested for the case in which the transfer times are not negligible are as follows; 1. $\min_{j \in m}\{s_j + a \cdot \tau_{ij}\}$, where m is the number of parallel machines; s_j is the shortest processing time of the jobs in queue at machine j; τ_{ij} is the worker transfer time from machine i to machine j; and a is a parametric weight. 2. $\max_{j \in m}\{q_j /(b \cdot \tau_{ij} + 1)^c\}$, where q_j represents the number of jobs in front of the machine j; b and c are parametric weights. 3. *FISFS*. The performance measures in this study were the mean flow time, the variance of flow time, the percent of time worker spends in transferring and total number of transfers. Based on the results of the experiment for different levels of a, b, c and d, Gunther concluded that increasing the time required for transfer increases the mean flow time. Delaying the transfer (i.e., greater d) due to negligible transfer times results in lower mean flow times. If d is equal to the infinity, this case is same as the decentralized control. In another study (Gunther, 1981), Gunther included the concept of "information access delays". Information access delay is the time required for a supervisor to obtain the updated information necessary to make the proper decision.

Treleven and Elvers (Treleven & Elvers, 1985) evaluated eleven (five of them is new) "where" rules when the worker transfer times are not negligible. Their shop consisted of three departments each having three workstations. Each workstation consisted of two machines (i.e., 18 machines in the shop totally). The average worker utilization in a shop was 90%. Worker transfer times were not negligible, and distributed negative exponentially with the mean of 0.67 time units. Workers were partial cross-trained as in the study of Weeks and Fryer (1976). So, the workers can not be transferred between the departments and they were homogenous. The routing pattern used in this study was same as the pattern used in Fryer (1975). The due date assignment method was *TWC* with the allowance factor of four. The "when" rule was decentralized control. There were three experimental factors in this experiment; The staffing level, dispatching, and where rules. For staffing, two levels were considered; these were 50% and 67% (i.e., eighteen machines operated by nine and twelve workers, respectively). The dispatching rules were *FISFS*, *SPT*, *EDD*, Critical Ratio, (i.e., *CR*, selects the job that has the lowest ratio of due date minus current date to total remaining processing time), and *SPR*. The "where" rules were *FISFS*, *SPT*, *EDD*, *CR*, *SPR*, and *LNQ* (i.e., the classical "where" rules). In addition to the classical rules, the new "where" rules suggested by Treleven and Elvers were *AFISFS* (i.e., Earliest average entry into the system of all jobs in its queue), *ASPT* (i.e., Shortest average processing time of all jobs in its queue), *AEDD* (i.e., Earliest average due date of all jobs in its queue), *ACR* (i.e., Lowest average critical ratio of all jobs in its queue), and *ASPR* (i.e., Lowest average slack per remaining number of operations of all jobs in its queue). The performance measures which were used in this study are similar to the performance measures which were used in the study of Weeks and Fryer (1976). The only difference is that Treleven and Elvers used the mean queue time and the queue time variance instead of the mean flow time and the flow time variance. Based on this experiment, they concluded that none of the "where" rules had an effect on the shop performance that was significantly different from the others. So, the selection of "where" rule was not very important. The selection criteria should be the cost of rule or the ease of use.

Elvers and Treleven (Elvers & Treleven, 1985) evaluated the effects of routing pattern in *DRC* shops. They used three different routing pattern; JJJ, (i.e., job shop routing pattern (hybrid)), JJF (i.e., a routing pattern where two-thirds of the jobs had a job shop routing and one-third had a pure flow shop routing), and JFF (i.e., a routing pattern where one-third of the jobs had a job shop routing and two-thirds had a pure flow shop routing). In JJJ, there were two departments, eight workstations, sixteen machines and twelve workers. In JJF, there were two departments eight workstations, twenty four machines, and eighteen workers. In JFF, there were two departments, eight workstations, forty eight machines and thirty six workers. The staffing level was 75% in these three routing patterns. The average worker utilization was 90%. The worker transfer delay was distributed negative exponentially with the mean of 0.81 hours. The transfer of a worker between the departments was not allowed. So, the workers were partial cross trained. Single level flexibility was assumed and the workers were homogenous. The due date assignment method was *TWC* with the allowance factor of four. The "when" rule was fixed and it was set to decentralized control. The "where" rule was also fixed and the workers select the target workstation by *LNQ*. In addition to the routing patterns, another experimental factor in this study was dispatching rule; The dispatching rules used in the experiment were *FISFS*, *SPT*, *EDD*, *CR*, and *SPR*. The performance measures of this study was same as in the study of Treleven and Elvers (1985). Based on the results of their experiment, they concluded that these results can be generalized to apply to situations involving different routing patterns such as pure flow shops.

Treleven (Treleven, 1987) proposed three new timing mechanisms based on the needs of the workers at workstations. He called these rules as "Pull" rules. The hypothetical shop they employed was similar to the one used in the study of Treleven and Elvers (1985). The parametric "pull" rules that he suggested were as follows; 1. The workstation i pulls a worker when the ratio N_i/L_i satisfies the condition $N_i/L_i \geq r$, where N_i is the number of jobs waiting in the queue of workstation i, L_i is the number of workers currently assigned to the workstation i, r is the predetermined threshold value. The workstation i pulls a worker from the workstation j in which the ratio N_j/L_j satisfies the condition $N_j/L_j \leq s$, where s is a threshold value. 2. The workstation i

pulls a worker when the sum of processing times in the queue of workstation *i* per the number of workers in workstation *i* is greater or equal than a threshold value. The workstation *i* pulls a worker from the workstation *j* which has a value of same ratio lower and equal to a threshold value. 3. The workstation *i* pulls a worker when the waiting time of the job which has been in queue the longest is greater or equal than a threshold value. The workstation *i* pulls a worker from the workstation *j* in which the ratio N_j/L_j satisfies the condition $N_j/L_j \leq s$. So, the dispatching, "when" and "where" rules were incorporated into these rules. Based on this experiment, the centralized control was not suggested as a good selection when the transfer times were significant and the new rules were found to be superior in comparison to the classical "push" rules. However, the proposed rules require special handling which cause additional cost.

Treleven (Treleven, 1988) compared the performances of flow and queue time variances in evaluating the different dispatching or worker assignment heuristics in *DRC* and machine-limited job shops. The system he used was the same as the system used in the study of Treleven and Elvers (1985). The only difference was in the transfer times. The transfer times in this study were negative exponentially with the mean of 0.54 time units. There were two experiments in the study of Treleven. In the first experiment, the flow time and the queue time variances were compared when the dispatching rules were *FISFS* and *SPT*. The "where" rules applied in this experiment were same as the rules given in the study of Treleven and Elvers (1985). The staffing level was accepted as 67%. The "when" rule was fixed and it was assumed as decentralized control. In the second experiment, the *DRC* and machine-limited job shops were compared with respect to the flow and queue time variances. The experimental factors in this experiment were the type of job shop (i.e. *DRC* vs. machine limited) and the dispatching rules (i.e., *FISFS*, *SPT*, *EDD*, and *SPR*). The staffing level in the second experiment was accepted as 50%. The "when-where" rules were fixed, and they were decentralized and *LNQ*, respectively. Based on the experimental results, he concluded that using the mean queue time as a performance measure in evaluating the dispatching rules gave more meaningful results than the mean flow time.

Park and Bobrowski (Park & Bobrowski, 1989) extended the coverage of the *DRC* studies by adding the job release mechanism to the system. In previous studies, the jobs were sent to the shop when they arrived to the system. However, in this study, the arrival time of job was not automatically equal to the release time of the job (the reader may refer to the review of Wisner for order release policy research, (Wisner, 1995)). The *DRC* shop they studied consisted of five workstations each having two machines. The staffing level was accepted as 50%. The average worker utilization was 90%. The worker transfer times were negligible. The degree of worker flexibility was an experimental factor and it included both partial and full cross trained workers. The single level worker flexibility was assumed. The workers were homogenous. The routing pattern was same as in the previous studies (i.e. hybrid). The dispatching and "where" rules were fixed and they were *CR* and *LNQ*, respectively. There were three factors in their experiment; The combination of "when" rule and the degree of worker flexibility, the due date assignment, and the job release mechanism. The combinations of "when" rule and the degree of worker flexibility were the followings; 1. Centralized - workers are full efficient at each workstation (i.e. the degree of worker flexibility is five). 2. Centralized - workers are full efficient at three workstations (i.e., the degree of worker flexibility is three). 3. Decentralized - workers are full efficient at each workstation (i.e., the degree of worker flexibility is one). 4. Decentralized - workers are full efficient at three workstations. 5. No flexibility - each worker is full efficient at just one workstation. The due date assignment rule has two levels; Tight and loose due dates. The due date assignment method was the modified version of *TWC*. In *TWC*, the allowance factor is set to a constant number. However, in this study, the allowance factor was a random variable distributed triangularly. For tight due dates, the parameters -min, mod, max- of the triangular distribution were set to 4, 5, and 6, respectively. For loose due dates, the parameters were 5,8, and 10. Two types of job release mechanisms were used in this study; Backward Infinite Loading (*BIL*), and Forward Infinite Loading (*FIL*). The primary performance measure in this study was the total cost per day. The cost function included the cost terms related to the inventory, the late penalty, and the worker transfer costs. Since the early deliveries of jobs were not allowed, the finished jobs wait in a Finished Good Inventory (*FGI*). So, the inventory

cost included both *WIP* and *FGI*. Based on the experimental results, they concluded that the *BIL* and *FIL* perform equally well with respect to the cost function. Moreover, they found out that the best performance improvement can be obtained by the initial introduction of the worker cross-training. For example, if the degree of worker flexibility is increased from one to two, the performance of the shop increases. However, when it is increased from two to three, they observed that the level of performance was not increasing at the same level as earlier case (i.e, incremental flexibility).

In another study (Bobrowski & Park, 1989), Bobrowski and Park investigated the issue that they dealt with in an earlier study (Park & Bobrowski, 1989). They used the same job shop model with full cross trained workers (i.e., the degree of worker flexibility is five). The "when" and "where" rules were fixed and they were centralized control and *LNQ*, respectively. Moreover, in this study, they used two additional job release mechanisms, and one additional dispatching rule. Another study about the job release mechanisms in *DRC* job shops is the study of Fredendall et al. (Fredendall et al., 1996). They first classified the sources and types of information available for *DRC* job shops, then, they developed alternative procedures to use the information in the job release, dispatching, and worker assignment decision.

Bobrowski and Park (Bobrowski & Park, 1993) suggested new worker assignment rules which used heterogeneous worker efficiency information. The same issue was also investigated before in the studies of Hogg et al. (Hogg et al., 1977) and Nelson (Nelson, 1970) for limited cases. As it was mentioned before, the workers may not have same skills for the same machines. For example, a worker may be full efficient (i.e., 100%) at a workstation but may not be full efficient (e.g., 85%) at another one. So, they can not operate machines equally well. In this type of job shop, the workers may be transferred between the workstations according to the rules using worker efficiency information. Their hypothetical *DRC* shop consisted of nine workstations each having two machines. The staffing level and the average worker utilization were accepted as 50% and 85%, respectively. The worker transfer times were negligible.

The workers were full cross trained but they have not same skills. So, the workers were heterogeneous. Single level worker flexibility was assumed and the routing pattern was hybrid. The due date assignment method was *TWC* with the allowance factor of five. The dispatching rule was *CR*. In experimental studies, they considered two factors; The "when" and "where" rules. The "when" rules used in this study were as follows; 1. Centralized control. 2. Decentralized control. 3. Efficiency rule (i.e., a worker immediately moves the workstation where he is more efficient if it is possible, or stays at the current station until the queue is empty). 4. Follower rule (i.e., a worker moves when he finishes the number of jobs which were there at the arrival). 5. Normalized queue rule (i.e., a worker moves when the normalized queue length is less than the threshold value. The normalized queue length is the queue length divided by the sum of the efficiencies of the workers). The seven "where" rules were evaluated in this study; *LNQ*, Modified *LNQ*, Normalized queue length rule, Efficiency rule, *SPT*, Least total slack loss rule, and Least aggregate slack rule. The performance measure was the mean flow time. They also carried out sensitivity analysis for the best performing rule pairs under different dispatching rules (i.e., *FISFS*, *SPT*, *CR*, and *EDD*). Based on the experimental results, the best rule pair was the rules using efficiency information (i.e., Efficiency rule for "when" and Efficiency rule for "where"). So, if the workers were not homogenous, the rules using worker efficiency information were found to be outperforming the other classical "when-where" rules.

Felan et al. (Felan et al., 1993) made a trade-off analysis between hiring new workers or increasing the degree of the worker flexibility. The shop they used consisted of eight departments each having ten machines (i.e., totally 80 machines in the shop). The staffing level was experimental factor. Four staffing levels were applied (50%, 60%, 70%, and 80%). These staffing levels resulted in 40, 48, 56, and 64 workers in the shop. The average worker utilization was set to 93% for the staffing level of 50%. The worker transfer times were negligible. The degree of worker flexibility was another experimental factor. The levels of this factor were one, two, four and eight (i.e., One; a worker can not be transferred to another department. Two; a worker can be transferred between two departments, and so on).

Single level worker flexibility was assumed and the routing pattern was hybrid but not same as in the previous studies. The routing pattern was as follows; each job flows through the shop from department one to department two in sequence. However, the machines in departments are selected randomly with equal probability. Due date assignment method was *TWC* with the allowance factor of five. The dispatching, "when", and "where" rules were fixed and they were *SPR*, decentralized control, and *LNQ*, respectively. The performance measures of this study were *WIP*, the average tardiness, the percentage of tardy jobs, the utilization, and the number of total transfers. Based on the experimental results, they concluded that the use of either strategy (i.e., hiring new workers or increasing flexibility) can improve the shop performance. However, increasing the number of workers was more effective strategy in terms of shop performances. They also stated that hiring new workers was costly in comparison to increasing worker flexibility. So, lastly, they concluded that increasing the degree of worker flexibility was more effective with respect to cost.

Malhotra and Kher (Malhotra & Kher, 1994) suggested new rules for shops with heterogeneous workers and significant transfer delays. The hypothetical shop consisted of six workstations each having two machines. The staffing level was 50%. The average worker utilization was accepted as 90%. The workers were full cross-trained and heterogeneous. The worker transfer time and the efficiency scheme were experimental factors. The levels of worker transfer times were 0%, 15% and 30% of mean processing time. The levels of efficiency scheme were; Worker efficiency scheme, machine efficiency scheme, and worker&machine efficiency scheme. Single level of worker flexibility was assumed and the job routings were random. The due date assignment rule was *TWC*. Dispatching rule was fixed to the *EDD*. Two "when" rules were applied; Centralized, and decentralized. They evaluated five "where" rules and two of them used both the efficiency and transfer delay information. The performance measures of this study were the mean flow time, the mean tardiness, and the percent of jobs tardy. Based on the experimental studies, they concluded that the most effective strategy was to assign the worker to the workstations in which they were most efficient, and to permit them work there until all the available jobs were finished.

The studies given so far dealt with the similar fundamental characteristics of *DRC* shops. However, there are several studies explaining some special topics on *DRC* shops.

One of these special topics to search for is the effects of learning and worker attrition on performance of a *DRC* job shop. These effects were first evaluated by Malhotra et al. (Malhotra et al., 1993). They extended the classical *DRC* job shop structure by including the learning curves and attrition rates. In previous studies, the typical assumption was that the workers were already full or partial cross trained when they were working in the shop (i.e., learning process is finished). So, in this type of system, there would not be production losses caused by learning process. However, the workers can reach their full efficiencies after a learning process. In learning process, they have not full efficiencies in their new workstations. So, there should be production losses in learning process. Another reason for production losses may be worker attrition. If the worker attrition rate in a shop is higher, the training costs of new workers and the cost of production losses will be higher. As it was mentioned before, these concepts were first introduced by Malhotra et al. They evaluated the effects of these concepts under different learning and attrition rates. They also extended the structure that they used by including the worker transfer delays in one of their next studies (Kher & Malhotra, 1994). In another study (Fry et al., 1995), Fry et al. especially focused on the attrition rates and their effects on the shop performances. Kher et al. extended the structure given in Malhotra et al. by modeling both learning and forgetting effects simultaneously (Kher et al. 1999). The study of Felan and Fry (Felan & Fry, 2001) also used learning and worker attrition and introduced the multi-level worker flexibility.

The second special topic on *DRC* shops is the management of customer priorities. The main issue here is how the *DRC* shop should be scheduled (i.e., both workers and jobs) when the vital customers exist (Kher, 2000), (Kher & Fry 2001).

Another special topic on *DRC* shops is the use of flexible workers in cellular manufacturing. Although, the cellular manufacturing requires the use of flexible

workers, there is a few *DRC* based cellular manufacturing study in the literature. The studies (Morris & Tersine, 1994), (Chen, 1995), (Suresh & Gaalman, 2000), (Jensen, 2000), and (Askin & Huang, 2001) are in this kind. However, some differences in the models exist in comparison to the job shop models in *DRC* literature.

There is only one study (Horng & Cochran, 2001) including the application of classical when-where rule pairs in a pull type environment which is controlled by KANBAN mechanism. This study suggested a decision support methodology called Project Surface Regions (*PSR*). The object of the study was to assist to production managers in determining the appropriate number of multitasking workers and the corresponding dynamic worker assignment rule when the system's behavior changes. They use the *RSM* and *PSR* to figure out which rule should be used.

2.2.2 Dual Resource Constrained Serial Manufacturing Lines

A typical serial manufacturing line consists of workstations and buffers. The role of flexible workers in a serial line may be a sharing a task, or a machine, etc. Balancing the line to achieve efficient utilization of productive capacity is one of the most studied objectives in serial manufacturing lines. The focus of classical industrial engineering studies on line balancing has been on evenly distributing the work in the line over the long term. However, even if a line is balanced with respect to the average loads, it can still become seriously imbalanced in the short term due to the variability (Gel et al, 2000). The negative effects of this variability can be compensated by inventory and by flexible workers. In *DRC* job shop literature, the flexibility of worker is originated from its moving capability. In *DRC* serial manufacturing line literature, there are two types of worker flexibilities; Task sharing (i.e., the workers can do different tasks in their own workstations), Machine sharing (i.e., the workers can use different machines, so they move between workstations). Although, this thesis primarily deals with moving workers, some task sharing schemes are applicable to the moving worker cases. Therefore, the literature related to task sharing will also be given. The studies about moving workers in serial manufacturing lines can be categorized into two parts: The jobs are tied to workers,

and the jobs are not tied to workers. If the jobs are tied to workers, the workers move between the workstations with their jobs in their hands. There are no buffers between the workstations. So, the number of jobs in the system is equal to the number of workers in the system. This type of flexibility is logically similar to the CONWIP systems. If the jobs are not tied to workers, the workers can hand off jobs to the buffers at the workstations.

Ostalaza et al. carried out the first study on task sharing in a serial manufacturing line (Ostalaza et al., 1990). They investigated a specific type of worker flexibility which was referred to as dynamic line balancing (*DLB*). They proposed *SPT/RSPT* work sharing schemes such as; *SPT*: A worker selects the job with shortest processing time from his upstream buffer. *RSPT*: A worker processes the shared task on a job if and only if the number of jobs at downstream buffer is less than *R*. Moreover, they have developed a Markov Chain model of their proposed system for exponential processing times and concluded that the half buffer size of *R* works well if the proportion of shared task to total work is %40. To evaluate different dynamic line balancing strategies, they used the term efficiency which was defined as the actual throughput rate divided by maximum possible throughput rate (Gel et al., 2000).

The study by McClain et al. (McClain et al., 1992) was extended version of the study by Ostalaza et al (1990). McClain et al. investigated the same problem but they modelled each task as a set of sub tasks and proposed new rules that measure the number of subtasks. Their analysis included simulation studies of two-stage lines under different scenarios, and some of three and five-stage lines. The authors concluded that the *DLB* can be used effectively for low inventory systems in which the material handling system requires the jobs to be processed in the order of *FISFS*. Their analysis also confirmed that the simple rule of sending a shared task to the downstream if the buffer was less than half-full (*HFB*), was effective (Gel et al., 2000)

It must be noted that these two studies differ from each other only in task structure and queue discipline for the downstream workstation (i.e., complete tasks, subtasks-*SPT*, *FIFO*). The only decision that the upstream worker should make in both of the studies is whether to process the shared task or send it to the downstream buffer if it is allowed by the buffer management rule (i.e., *RSPT/HFB*). The preemption of tasks was not allowed and the workers were homogenous.

By allowing the preemption of shared tasks and heterogeneous workers concepts, the study by Gel et al. (Gel et al., 2000) brought different dimensions to the issue of work sharing in *DLB* literature. They investigated the same problem as in the studies of Ostalaza et al., and McClain et al. However, they allowed the preemption of the shared task, and different paces of the workers. In addition to these, they assumed that the hypothetical system works under the CONWIP production control mechanism. In this study, they used the Markov Decision Process to model the system they proposed. Their buffer management rule was *HFB* but it was slightly different from the original rule proposed by McClain et al. (1992) due to the reason that the workers were not homogenous in Gel's et al. study (2000). Therefore, the determination of *R*, which was the cutting level of the buffer, showed different behavior. They also proposed a new buffer management rule which was called 50-50 work content (*50-50WC*). This policy was as follows; the upstream worker transfers the shared task to the downstream buffer if the ratio of the work content at upstream station to the total work content in the system is greater than %50 and starts processing it otherwise. They also compared the suboptimality of the *HFB* and *50-50WC* buffer management rules for different constant levels of *WIP* and the proportion of shared task to the total task. To be able to make suboptimality analysis, they used stochastic dynamic programming with value iteration algorithm to obtain the optimal decisions of workers in all possible states of CONWIP levels in the range of 3 to 12. Based on the experimental results, they concluded that the percent suboptimality of *50-50WC* was better at all *WIP* levels than *HFB*'s. However, they showed that these buffer management rules worked best in the environment in which the preemption of shared task was allowed. Lastly, they investigated the factors affecting the performance opportunity of work sharing schemes by comparing the

proposed system with the best static allocation of jobs to the workers.

The flexibility of the workers in the studies explained so far originated from the shared tasks. The number of workstations and the number of workers were equal and the various types of tasks were available providing that one or two tasks may be handled by both workers. The movements of the workers were not needed in these cases. There is another group of studies which explore the worker flexibility in the sense of worker transfers. The following studies which require the full or partial cross training of workers whose number is less than the number of machines explore this issue.

One of the most cited study considering the worker transfers was carried out by Bartholdi and Einstein (Bartholdi & Einstein, 1996). Their study depends on the system known as Toyota Sewn Products System (*TSS*). The operation of *TSS* is as follows; Each worker carries a product from station to station towards the completion; when the last worker finishes his product, he sends it off from the line and walks back to the upstream stations to take over the work of the first worker that he come across. This chain movement of the workers causes the entrance of a new product to the system. If the worker comes across with a busy station in his forward movement, he remains blocked until the downstream busy station becomes idle. In their study, Bartholdi et al. modelled the system using deterministic processing times, but used a very general model of the processing rates. Specifically, processing rates may differ by worker and by position in the line, that is, for every part of every task, every worker may have a unique speed of operation. They showed several important properties about the worker positions in the line. They proved the existing of a fixed point (i.e., a vector describing the starting position on the line for each worker) having the following properties; If all workers begin at their specified positions, they will return to those same positions after the completion of each item. They then proved that if workers are sequenced from slowest to fastest, the fixed point is unique and will be reached eventually, regardless of the starting positions of the workers (Zavadlav et al., 1996). They referred this system where the workers sequenced from the slowest to the fastest as Bucket Brigades. It must be noted that

the preemption of jobs is allowed, and the workers are heterogeneous. Each workstation has just one machine (i.e., identical parallel machines are not allowed) and the jobs can be thought as if they are tied to workers (i.e., there are no buffers between the workstations). Hand off and worker transfer delays were negligible. Bartholdi and Einstein (Bartholdi & Einstein, 1998) described the bucket brigades and its applications. Bartholdi et al. (Bartholdi et al., 1999) gave the closed form formulations of the production rates for two and three worker bucket Brigades. They described all possible asymptotic behavior of bucket brigade production lines with two or three workers, each characterized by his work velocity. In another study (Bartholdi et al., 2000), Bartholdi et al. addressed the case of stochastic processing times, and proved a convergence of behavior between the deterministic and stochastic systems as the number of station goes to infinity.

Zavadlav et al. (Zavadlav et al., 1996) classified the worker flexibility in different types for U-shaped lines; The first one is the Fixed Assignment System. For example, in a two workers three machines serial system, the first worker may be assigned to the machine one and the machine three, and the second worker may be assigned to the machine two where the capacity of one worker is shared between the machine one and the machine three. They modelled this system by using Markov Decision Process, and calculated the system efficiency in balanced case while changing the coefficient of variation of the processing times. They also explained the effects of nonconsecutive assignments on the throughput rate. The second system they studied focused on Overlapping Worker Assignments where the line consisted of fixed assigned machines and shared machines. The upstream and the downstream machines shared in this assignment. Allowing the preemption of the job makes these systems similar to *TSS* but the preemption rules should be regulated according to the limited zone situations. They studied the effects of a buffer management rule (Ostalaza's *HFB*) on the efficiency of a hypothetical line. They also proposed Forward Bumping (i.e. the preemption mechanism in *TSS* is called Backward Bumping) and measured the effectiveness of the proposed preemption scheme. The last system they studied involved the Free-Floating Worker Assignments. This system required full cross-trained workers and considered as the most flexible way to

allocate work. No bumping is necessary in this system due to the reason that the busy machine can be easily passed. Furthermore, they introduced the hierarchy of the worker flexibility levels and their efficiencies by changing the various important factors such as, balance of the line, the process time's variability, and the shared portion of total work content. In this study, the jobs were not tied to the workers, and each workstation had just one machine. The preemption was allowed only in the case of overlapping worker assignments.

Bischak (Bischak, 1996) concentrated on the system in which the workers move like the workers in *TSS* without buffer. They simply evaluated the effect of the following parameters on the performance of the line allowing worker transfers; the number of machines and workers, the presence or absence of a bottleneck machine, the coefficient of variation of the processing time distribution at each machine, and the number of buffers in front of each machines. To evaluate any losses in capacity, they also compared the moving worker system with the static serial line under the condition that the number of machines and workers are equal.

A recent study on employing the flexible workers belongs to Oyen et al. (Oyen et al., 2001). They proposed two methods for worker agility in the case of collaborative and noncollaborative tasks. The proposed methods are called "Expedite Policy (*EP*)" and "Pick and Run Policy (*PR*)". They proved that the *EP* is optimal under every sample path for collaborative tasks. They also showed that the *PR* policy is optimal when the CONWIP level is equal to at least the number of workers for noncollaborative tasks. They investigated the factors affecting the performance opportunity for workforce agility in different types of production environments such as make-to-stock and make-to-order. They also gave some theoretical issues about their general models.

This thesis primarily focuses on a CONWIP controlled system which consists of serially connected workstations each having identical parallel machines, proposes a new worker flexibility policy for this system and compares the performance of the most widely used the *DRC* job shop "when-where" rules with respect to the cost

function suggested.

2.3 Constant Work-in-Process Control (CONWIP)

CONWIP was first introduced by Spearman et al. (Spearman et al., 1990). It is one of the pull type production/inventory control mechanism which incorporates the benefits of both push and pull systems (i.e., Pull type production/inventory control uses the actual occurrences of demand rather than future demand forecasts). How the CONWIP controls the flow of material depends on the type of the demand structure. The demand structure may be either; Saturated or non-saturated.

In saturated demand case, the finished parts leave the system immediately (i.e., It is consumed by the customer immediately), so there is no need for Finished Good Inventory (*FGI*). This type of production environment is also called as capacity-constrained. In the capacity-constrained environment, a production system operates under its maximum capacity which is not bounded by the demand. The CONWIP mechanism works under the saturated demand case as follows; a new job is released to the system each time a job departs the system. This type of job release mechanism provides nearly constant *WIP*. The production capacity of the system controlled by CONWIP is limited by the level of CONWIP, "*K*". The *K* is the maximum number of jobs allowed in the system.

In case of non-saturated demand, the capacity of the production system is limited by the demand. There is a *FGI* at the end of the system. This production environment is called as demand-constrained. The CONWIP mechanism works under the non-saturated demand case as follows; Each job in the system has a CONWIP card attached to it. The total number of CONWIP cards is equal to *K*. When a demand is satisfied from the *FGI*, the CONWIP card which is attached to the job is simply detached and then, it is sent to the first workstation as a production authorization signal. So, if the *FGI* is full (i.e., all the CONWIP cards are in *FGI*) and there is no demand, the manufacturing line will not operate due to the reason that there is no CONWIP card authorizing the production at the first workstation. If the *FGI* is

empty at the moment that demand arrives, there are two possible actions; backlogging, or accepting lost sales. In backlogging, the orders which can not be satisfied from the *FGI* are kept on a backlog list and when a finished job becomes available the backlog demand is filled. If backlogging is not allowed, the customer demands that can not be met from the items in *FGI* are simply lost. The departure rate from the last workstation is called as fill rate. The fill rate is a measure of the system capacity. The fill rate may be less and equal than the demand arrival rate. If the fill rate is equal to the demand arrival rate, the service level (*SL*) becomes 100%, which represents the perfect service quality. If it is less than the arrival rate, *SL* will be less than 100%. There is a trade-off between inventory holding cost and the cost of lost sales (or the cost of backlogging). If the value of *K* is increased, the fill rate increases, and the probability that the arriving demand finds the *FGI* empty decreases, that is, the cost of lost sales or backlogging decreases. However, by increasing the value of *K*, the inventory holding cost increases.

CONWIP control is more suitable for the production line consisting of a single and fixed routing with single class jobs. However, there are studies explaining the use of CONWIP in settings other than serial lines. Typical objective functions in CONWIP lines are to minimize the CONWIP level (i.e., *WIP* or the cost of *WIP*) to reach a specified throughput rate (i.e., saturated demand) or to reach a specified service level (i.e., non-saturated demand).

The types and properties of CONWIP lines were explained in detail by Hopp and Spearman (Hopp & Spearman, 2001). In general, most of the studies were compared the performances of alternative pull mechanisms (Spearman et al., 1990, Muckstadt & Tayur, 1995a, Muckstadt & Tayur, 1995b, Frein et al., 1995, Gstettner & Kuhn, 1996, Bonwik et al., 1997, Gaury et al., 2000, Dallery & Liberopoulos, 2000, Karaesmen & Dallery, 2000, Marek, et al., 2001, and Liberopoulos & Dallery, 2002). As it was mentioned earlier, there are studies explaining the use of CONWIP in settings other than serial lines or using CONWIP under some special conditions (Golany et al., 1999, Ryan et al., 2000, Ip et al.,2002, Huang et al. 1998, Ayhan & Wortman, 1999, Hazra et al., 1999, Dar-el et al., 1999, Duri et al., 2000, Ovalle &

Marquez, 2003, and Isakow & Golany, 2003). There are also studies about the theoretical aspects of CONWIP systems (Duenyas & Hopp, 1992, and Duenyas et al., 1993).

The majority of the studies on CONWIP lines primarily focused on machine-limited systems. There are also several studies on worker and machine limited systems (i.e., *DRC*) which work logically like CONWIP as it was given in Section 2.2.2 (Bartholdi & Einstein, 1996, Bartholdi & Einstein, 1999, Bartholdi & Einstein, 2000, Zavadlav, 1996, and Bischak, 1996). It must be noted that in all of these studies mentioned, only one machine was considered at each workstation. In other words, parallel machines were not allowed. Based on the literature review conducted in this area, we can state that the study by Oyen et al. (Oyen et al., 2001) is the only one which allows parallel machines.

In the study of Oyen et al. (Oyen et al., 2001), one of the most important assumptions is that all the workers can be pooled in every workstation providing that ample equipment or machines are available for all of the workers. This is the ample capacity case. In the ample capacity case, it is assumed that there are sufficient equipment or machines at each workstation to ensure that a worker is never blocked for lack of a machine (Hopp & Spearman, 2001). They showed that their worker flexibility policies (i.e., *EP* and *PR*) minimize the cycle time of the line when the CONWIP level is greater or equal to the total number of workers under the assumption given above. However, in real production environments, it may not be possible to supply sufficient equipment at each workstation. In most cases, it is impossible to place all the workers in a same workstation. These impossibilities are caused by some inefficiencies or economical reasons. First, it may not be economical to buy more than one jig, tool, fixture, test equipment or machine, etc. Second, the layout of the line may not allow pooling of the workers. Even if it is possible to place all workers in a same workstation, this may cause inefficiencies. Moreover, continuously changing the place of the worker (i.e., after each completion of job) may cause the motivation problems, and may increase the quality related problems. Lastly, if the moving times of the workers are not negligible due to the layout,

moving after each job completion may decrease the performance of the line dramatically.

From this point of view, in most cases, unconstrained workstation capacity (i.e., ample capacity) is not an applicable concept. Therefore, this is not a realistic assumption in most of the production systems. If this assumption is violated, the *EP* and *PR* policies can not produce the optimal results due to the worker blockages at the workstations. Hence, this thesis is based on the violation of the above assumption given in (Oyen et al., 2001). In other words, we propose a worker flexibility policy which does not require pulling all the workers into the same workstation. This policy does not promise to reach the optimum performance but we believe that it is better suited to the real production environments.

2.4 Simulation Optimization

In general, the decision makers use the models which represent the logical and the mathematical relationships between the components of the systems to analyze the existing or proposed systems. Models of systems can be put into two main categories: Optimization and performance models. Optimization models generate a single or a set of decisions for a given set of criteria and constraints. These models use the principles of optimization. Performance models, on the other hand, estimate the measures of system performance for a given set of decisions and system parameters. Performance models use the techniques of stochastic processes, probability, and computer simulation (Altıok, 1997). In this point of view, it may seem that the optimization and the simulation are not same. However, the recent developments in optimization and computer simulation have been causing an increase in the use of simulation as an optimization tool.

The term "Simulation Optimization" can be defined as the process of finding the best values of input parameters from among all alternatives without evaluating each alternative. The objective of simulation optimization is to minimize the resources spent while maximizing the information obtained in a simulation experiment (Carson

& Maria, 1997). In mathematical programming the decision variables are all quantitative and the performance measure is stated as an analytical function of decision variables. This function is known as objective function. Additional analytical expressions may represent relationships between the variables or constraints on the decision variables. So long as all variables are quantitative a wide range of solution methods exists, depending on the continuity and the form of the objective function and constraints. In simulation optimization, the analytical objective function and constraints are replaced by one or more discrete event simulation models. The decision variables are the conditions under which the simulation is run, and the performance measure becomes one of the responses (or a function of several responses) generated by a simulation model.

In general, simulation optimization techniques can be divided into four main categories. The reader may refer to following studies for different classification schemes; (Safizadeh, 1990), (Fu, 1994), (Carson & Maria, 1997), (Azadivar, 1999), (Swisher et al., 2000). The first one is gradient-based approaches. These types of methodologies require the gradient estimation of the performance measure. It is clear that if the input parameters are not continuous, gradient-based approaches can not be used for the purpose of optimization. For example, in many case the simulation model includes discrete input variables such as queue ranking rules, different operating policies, or different design alternatives. The widely used techniques are Finite Difference Estimation (*FDE*), Infinitesimal Perturbation Analysis (*IPA*), Likelihood Ratio Estimation (*LRE*), and Frequency Domain Analysis (*FDA*). The second main category is stochastic optimization approaches. Stochastic optimization methods are the stochastic versions of the gradient based approaches. There are two methods used in stochastic optimization of simulation responses. These are Robbins-Monro and Kiefer-Wolfowitz. The third main category is Response Surface Methodology (*RSM*). The *RSM* is a collection of mathematical and statistical techniques that are useful for the modeling and analysis of problems in which a response of interest is influenced by several variables and the objective is to optimize this response. The last main category is direct search techniques. The term "Direct Search" implies that no information concerning the analytical form of the objective

function is available. In general, direct search methods do not use derivative information, they require only the value of objective function for a given set of values assigned to input variables. These methods can be used for optimization of both discrete and continuous input variables. The most widely used direct search techniques are Genetic Algorithms (*GA*), Tabu Search (*TS*), Simulated Annealing (*SA*), and Nelder Mead's Simplex Search. The simulation optimization method which will be used in thesis is *RSM*. The ease of applicability of *RSM* outperforms the use of other simulation optimization techniques. In next section, the literature and the background of *RSM* are given.

2.5 Response Surface Methodology

The Response Surface Methodology, or *RSM*, is a collection of mathematical and statistical techniques that are useful for the modeling and analysis of problems in which a response of interest is influenced by several variables and the objective is to optimize this response. For example, if the objective is to find the levels of $x_1, x_2, \ldots, x_k$ that maximize the response (i.e., y) of a specific process, the process should be identified as a function of independent variables $x_1, x_2, \ldots, x_k$, that is,

$$y = f(x_1, x_2, \ldots x_k) + \varepsilon \qquad (2.1)$$

where ε represents the noise or error observed in the response "y". Since the error term should be normally distributed with mean and standard deviation of 0 and σ^2, respectively, the expected response is

$$E(y) = f(x_1, x_2, \ldots x_k) = \eta \qquad (2.2)$$

then the surface represented by $\eta = f(x_1, x_2, \ldots x_k)$ is called a response surface (Montgomery, 1991).

In most *RSM* problems, the form of the relationship between the response and the independent variables is unknown. Thus, the first step in *RSM* is to find an appropriate approximation for the true functional relationship between y and the set of independent variables. Usually, a low-order polynomial in some region of the independent variables is employed. If the response is well modeled by a linear function of the independent variables, then the approximating function is the first-order model,

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \varepsilon \tag{2.3}$$

if there is curvature in the system, then a polynomial of higher degree must be used, such as the second-order model (Montgomery, 1991),

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^{\,2} + \sum_{i} \sum_{j} \beta_{ij} x_i x_j + \varepsilon \tag{2.4}$$

Almost all *RSM* problems utilize one or both of the approximating polynomials mentioned before. However, it is not the case that a polynomial model will be a reasonable approximation of the true functional relationship over the entire space of the independent variables. These models usually work quite well only for a relatively small region. The method of least squares is used to estimate the parameters in the approximating polynomials. The response surface analysis is then done in terms of the fitted surface. If the fitted surface is an adequate approximation of the true response function, then the analysis of fitted surface will be approximately equivalent to analysis of the actual system. This type of model is called as Metamodel. The model parameters can be estimated most effectively if proper experimental designs are used to collect the data. Designs for fitting response surfaces are called response surface designs. The response surface design used for fitting response surface directly affects the quality of the estimate. The measure of the quality of the estimate may be, variance-covariance matrix, or the variance of the fitted values. Therefore, it is necessary to use proper design for specific purpose. It is clear that the importance of choosing the proper design is to identify the factor levels

at which the data are collected, and how many replications should be made at each design point to minimize or maximize the design optimality criterion which is selected (Montgomery, 1991).

A typical *RSM* study consists of two main steps. These are the estimation of the response surface and the optimization. Once the surface is estimated, the fitted polynomial can be used to optimize the process by using any kind of unconstrained nonlinear optimization technique. However, the most widely used technique in *RSM* is steepest Decent/Ascent method. The current literature on *RSM* can be summarized based on these two steps;

Estimation Process;

- Factor Selection: This is called factor screening. At this phase, statistically significant factors that effect the response are identified, and insignificant ones are eliminated. The studies (Kleijnen, 1994), (Kleijnen, 1995), and (Kleijnen, 1998), (Morrice & Schruben, 1993), (Murugabaskar, 1993) represents the factor and group screening methods in simulation.
- Optimum Design Selection: At this phase, special response surface design such as 2^k, 2^{k-p}, or Central Composite Design is selected. The books (Box & Draper, 1987), (Diamond, 1989), (Draper, 1981), (Montgomery, 1991), (Myers & Montgomery, 1995) give fundamentals of the response surface designs. Further material on which design to use, the studies of (Donohue et al., 1992), (Donohue et al., 1995), (Hoods, 1993), (Hardin, 1992), (Tew, 1992), (Youssef, 1994) include comparative analysis of response surface designs.
- Variance Reduction Strategies: Once the design is selected, the simulation model is executed at the design points which are identified according to the selected design. The objective of these strategies is to reduce the variance of the fitted model obtained by using selected experimental design. The studies of (Song et al., 1996), (Song et al.,1998), (Nozari et al., 1987), (Hussey et al., 1987), (Tew et al., 1994), (Kleijnen, 1992), and (Kwon et al., 1994) analyze the variance reduction techniques to reduce the variance of the fitted model and its parameters.

Optimization Process;

- Optimization: At this phase, the objective of the experimenter is to decide on whether the process is operating under near-optimum conditions (i.e., optimum system parameters), or operating under remote-optimum conditions. The mathematical optimization technique steepest ascent (or descent) is used when the operating conditions are remote-optimum. If the operating conditions are near-optimum, the objective of the experimenter is to find true response function within a relatively small region around the optimum. The studies of (Kleijnen, 1998), (Joshi, 1998) introduces the steepest ascent/descent methodologies. For further reading, the reader may refer to the books of (Arora, 1989), (Rardin, 1998), and (Law & Kelton, 1992).

A typical *RSM* study consists of the following steps;

Estimation Process

The Studies Before Estimating the Metamodel

- The objectives of developing a metamodel for the simulation model are determined.
- All of the decision variables and their characteristics such as being discrete or continuous are determined. The performance measure(s) is identified.
- The operability regions of all the decision variables are identified. This issue is handled by making sensitivity analysis on each decision variable. The developed and the validated simulation model which represents the real system's behavior is used for this purpose.
- The accuracy of the metamodel with respect to the selected performance measures is identified.
- The validity measures of the metamodel are determined.

- The Factor Screening is applied. In this phase, the important decision variables (i.e., the decision variables which have statistically more effects on the performance measures) is determined.

The Studies for Estimating the Metamodel

- According to the factor screening results, the designs of 2^k full factorial or 2^{k-p} fractional factorial with center points is applied in estimating the metamodel.
- The low and the high levels for each decision variable is determined providing that the operability region of the related decision variable should cover the low and high level of the related decision variable. The low and the high levels for the decision variables are coded as –1, +1.
- Tactical decisions on executing the simulation model are made. The number of replications at each design point, the length of the replication, and the variance reduction technique which will be used is specified.
- After executing the model at each design point, the least square estimation procedure is applied to fit the response to a first-order model.
- At this point, significance of the model parameters, main and interaction effects, Lack-of-Fit, and Curvature effects are analyzed. If the first-order model is proper according to the results, the optimization process starts. If it is not proper, the first-order model with interaction or second-order model is fitted.

Optimization Process

- If the first-order model is proper, the gradient vector of the performance measure is determined. This gradient vector is used in one of the process improvement techniques such as Steepest Ascent/Descent. In these methods, the center of the design is changed to increase the performance of the system. If there is no improvement, the method stops for estimating a new first-order model in new ranges, then, same procedure which was explained will be followed. This process continues until the new design has the

interaction or curvature effects. If it is so, the interaction or second-order model is fitted.

- After fitting second-order model, the canonical analysis is applied to calculate the optimal levels of the decision variables.

CHAPTER THREE

A NEW WORKER FLEXIBILITY POLICY FOR CONWIP LINES

3.1 Introduction

In this chapter, the proposed unit penalty cost function, the worker flexibility policy, and the two-stage simulation optimization methodology are explained.

The main purpose in developing the unit penalty cost function is to have a cost-based criterion to compare alternative flexibility policies. As it was mentioned earlier, the manufacturing flexibility causes additional costs, and there is a trade-off between the cost of increasing the flexibility and the performance improvement obtained by applying flexibility. So, a cost-based criterion can provide a sound comparison of alternative policies. Moreover, since this cost function is embedded into the simulation model of CONWIP line, we used it as a validation tool to compare the simulation results with the analytical results obtained under some special conditions. Figure 3.1 presents the hypothetical manufacturing system studied in this thesis.

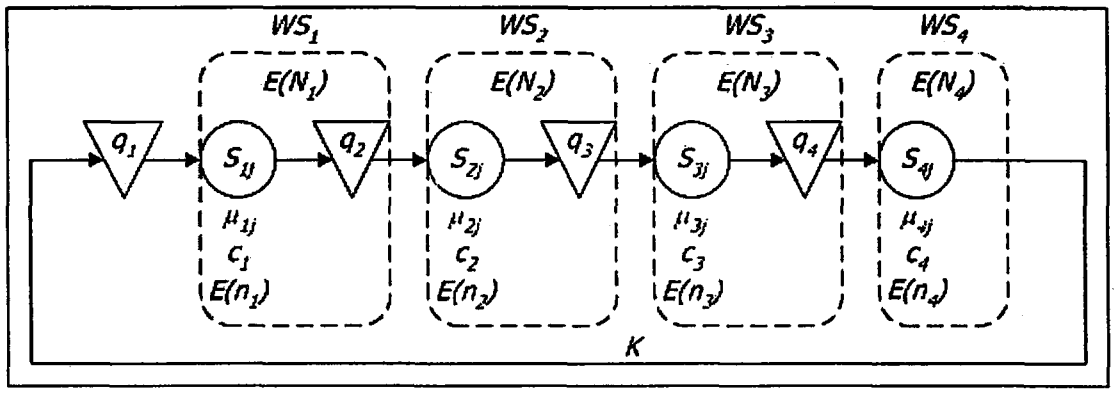

Figure 3.1 The Hypothetical Manufacturing System

The system consists of four workstations which are connected serially. Each workstation (WS_i) has identical parallel machines (S_{ij}) and buffers (q_i). The number of jobs circulating in this system is K. This hypothetical manufacturing system is a typical *CQN*. So, the performances (i.e., cycle time) of CONWIP controlled serial lines under saturated demand can be evaluated analytically. Therefore, the analytical approaches for *CQN*'s are used to validate the simulation model of CONWIP line based on the proposed unit penalty cost function. The abbreviations given in Figure 3.1 are consistent with the notation and definitions given in Section 3.2.

3.2 Notation and Definitions

In following two sections, the notation used both in representing the proposed flexibility policy as a *CQN* problem and also in developing the unit cost function are given. While first section gives the definitions related to the CONWIP line with flexible workers, the second section presents the definitions related to the proposed unit penalty cost function.

3.2.1 Notation and Definitions for CONWIP Line with Flexible Workers

i: The index for workstations, $i = 1,\ldots,M$.

M: The total number of workstations that are connected serially.

j: The index for identical parallel machines in workstation i, $j = 1,\ldots,c_i$

S_{ij}: The identical parallel machines in workstation i. This case can be thought as single machine with parallel service facilities.

c_i: The number of identical parallel machines in workstation i (or, the capacity of a single machine).

q_i: The input and the output buffer of workstation i are q_{i-1} and q_{i+1}, respectively. The q_1 is not included as *WIP* since it is just a waiting space for job orders, or raw material.

$nq_i(t)$: The continuous time stochastic process representing the number of jobs in q_i at time t. We use nq_i to denote this stochastic process.

$E(nq_i)$: The expected number of jobs waiting in q_i.

$nr_i(t)$: The continuous time stochastic process representing the number of jobs being processed in WS_i at time t. We use nr_i to denote this stochastic process.

$E(nr_i)$: The expected number of jobs being processed in WS_i.

WS_i: The workstation i, WS_i, consists of S_{ij}'s and the output buffer q_{i+1}. Since the demand is saturated, there is no output buffer at the end of the line. So, the last workstation consists of just identical parallel machines (i.e., $WS_M \rightarrow S_{Mj}$).

λ: The mean demand arrival rate (i.e., expected number of demand arrivals per unit time).

$1/\mu_{ij}$: The mean service time at S_{ij}. Since the machines in a workstation are identical, mean service times are equal. So, the mean service time at a workstation can be represented by just one index (i.e., $1/\mu_i$).

ρ_i: The machine utilization at WS_i. The utilization ρ_i is defined as $\frac{\lambda}{c_i \cdot \mu_i}$ providing that $\rho_i < 1$.

K: The CONWIP level (i.e., the number of CONWIP cards, or the maximum number of jobs allowed in the system). It is the unique control parameter of a CONWIP line.

T_{pc}: The length of the periodic controls. This is the parameter of our proposed policy.

WL_i: Total workload capacity of first i workstations. We defined WL_i as $\sum_{j=1}^{i} \frac{c_j}{\mu_j}$.

$E(TR)$: The expected throughput rate of the line (i.e., The expected number of units produced per unit time). We denoted it as TR. The reciprocal of TR is the cycle time of the line. It must be noted that the throughput rate of this system will be the arrival rate of the system. So, λ will be equal to TR.

$N_i(t)$: The continuous time stochastic process representing the number of jobs in WS_i at time t, (i.e., the number of jobs in both S_{ij}'s and q_{i+1}, $N_i(t) = nr_i + nq_{i+1}$. We use N_i to denote this stochastic process.

$E(N_i)$: The expected number of jobs in WS_i. It can be thought as average *WIP* level at WS_i. It must be noted that $E(N_i)$ is equal to $c_i \cdot \rho_i + E(nq_{i+1})$.

N(t): The continuous time stochastic process representing the total number of jobs in all WS_i's at time t. We use N to denote this stochastic process.

E(N): The expected total number of jobs in all WS_i's. The relation between $E(N)$ and $E(N_i)$*'s* can be written as $E(N) = \sum_{i=1}^{M} E(N_i)$. Since q_1 is not included in the workstations, $E(N)$ is not equal to K in saturated demand case for our modeling purposes. However, it can be easily written that $K = E(nq_1) + E(N)$.

n: The total number of workers in the line.

$n_i(t)$: The continuous time stochastic process representing the number of workers attended to WS_i at time t providing that $0 \le n_i(t) \le c_i$. As it was mentioned in chapter one, the hypothetical manufacturing system is bounded by two resources (i.e., *DRC*); Machines, and workers. Each machine needs just one worker for processing a job, so the condition $0 \le n_i(t) \le c_i$ is clear. The case $n_i(t) = 0$ represents the state that no machine is utilized by the workers, and there may or may not be a job waiting in q_i. Another extreme case may be $n_i(t) = c_i$. In this state, each machine has its own worker, however, it does not mean that all of the workers are working, they may be in waiting mode for a job. The $n_i(t)$ is one of the most important terms related to the flexibility of the workers. Since it changes over time with respect to the flexibility policy that is applied, this means that the workers are moving between the WS_i's. The workers can move between the WS_i's only when the condition $n < \sum_{i=1}^{M} c_i$ is satisfied. This is the condition of being *DRC*. If $n = \sum_{i=1}^{M} c_i$, all of the machines in the production line are staffed by one worker, and the movement of a worker is not allowed due to the condition $0 \le n_i(t) \le c_i$. In this case, both the worker and the machine act as if they are one single resource. Since the unit penalty

cost function that we developed is embedded to our simulation model which is generic in terms of the worker flexibility (i.e., if $n = \sum_{i=1}^{M} c_i$, flexibility is not allowed, if $n < \sum_{i=1}^{M} c_i$ flexibility is allowed), we validated the simulation model of the hypothetical CONWIP line with saturation under the condition $n = \sum_{i=1}^{M} c_i$ which can be evaluated by *CQN* analytically. We use n_i to denote this stochastic process.

E(n$_i$): The expected number of workers attended to WS_i. If the condition $n = \sum_{i=1}^{M} c_i$ which was given above is satisfied, then, it can be written as $E(n_i) = c_i$ for $i = 1,\ldots,M$. If the worker flexibility is allowed, the minimum possible value of *E(n$_i$)* is equal to the expected number of jobs at S_{ij}'s since the job, the machine and the worker are tied to each other. Since the expected number of job at S_{ij}'s is $c_i \cdot \rho_i$, the condition can be written as $E(n_i) \geq c_i \cdot \rho_i$, and then, the range for possible values of *E(n$_i$)* is $c_i \geq E(n_i) \geq c_i \cdot \rho_i$. In the case of $n < \sum_{i=1}^{M} c_i$, it should be expected that the best flexibility policy provides the equality $E(n_i) = c_i \cdot \rho_i$ for all $i = 1,\ldots,M$. This equality represents the fully utilized workers, so the difference between *E(n$_i$)* and $c_i \cdot \rho_i$ is caused by inefficiency of the flexibility policy in saturated demand case. The *PR* has this property for given *n* under the conditions $c_i = n$ (i.e., ample capacity case) for all $i = 1,\ldots,M$, $K \geq n$, and the system is CONWIP controlled serial line under saturated demand. The *PR* (Oyen et al., 2001) is optimum in terms of the level of *WIP* and *TR*. It provides maximum possible *TR* (or, minimum theoretical cycle time) for given *n* with minimum *WIP*.

3.2.2 Notation and Definitions for Unit Penalty Cost

u$_{mc}$: The cost of each identical parallel machine per unit time. It is assumed that this unit cost includes overhead costs, such as depreciation and insurance. The operating costs are not included, since the operating costs are dominated by the

depreciation and the insurance in most cases. Depreciation and insurance costs are fixed costs in nature for a given period, however, they can be represented by per unit time basis. Therefore, these costs occur whether the line is operating or not. We also assumed that the u_{mc} is same for all of the parallel machines at the line.

u_{wc}: The worker cost per unit time. It is assumed that all of the workers are identical, and the cost rates are equal.

u_{cc}: The cost of one periodic control. This term has not been considered yet in the literature. We use this cost to differentiate the policies in terms of the number of controls realized for a given period. Through these periodic controls, decision is made on whether it is necessary to transfer a worker or not. It must be noted that the number of controls for a given period is not same as the number of transfers because there may be blockages of workers. If the flexibility policy requires no specific calculation, such as in the case of *PR*, u_{cc} is assumed to be 0.

P_i: The monetary value of one unit of *WIP* processed at workstation i, P_0=0 (Since q_1 is not included as *WIP*, we assumed that the monetary value of one unit *WIP* at raw material warehouse is equal to zero. Furthermore, the material cost is same for all jobs due to the reason that the line produces a single class of jobs. It has just additive effect in our unit penalty cost function). Since the value is added to the job through the line with respect to the investment, we did not use arbitrarily P_i's. We assigned the production costs to the inventory as the monetary value of one unit of *WIP*.

F: The inventory carrying cost fraction per unit time. It consists of both r and s where, r represents the fraction of opportunity cost, and s represents the fraction of physical inventory carrying cost.

H_i: The inventory carrying cost at workstation i per unit time per unit *WIP*. It can be written as $H_i = F \cdot P_i$.

IC_i: The total inventory carrying cost at workstation i per job produced in the line.

TC: The unit penalty cost.

TMC: The total machine penalty cost per job produced in the line.

TWC: The total worker penalty cost per job produced in the line.

TCC: The total control penalty cost per job produced in the line.

TIC: The total inventory penalty cost per job produced in the line.

3.3 Development of Unit Penalty Cost Function

The unit penalty cost (*TC*) consists of four components; 1. The total machine penalty cost per job (*TMC*). 2. The total worker penalty cost per job (*TWC*). 3. The total control penalty cost per job (*TCC*). 4. The total inventory penalty cost per job (*TIC*). The general form of the unit penalty cost can be written as;

$$TC = TMC + TWC + TCC + TIC \tag{3.1}$$

The total machine penalty cost per job produced in the line can be given as;

$$TMC = \left[\sum_{i=1}^{M} c_i\right] \cdot u_{mc} \cdot \frac{1}{TR} \tag{3.2}$$

In this equation, $\sum_{i=1}^{M} c_i$ represents the total number of machines in the line. The total worker penalty cost per job produced in the line can be represented by the Equation (3.3).

$$TWC = n \cdot u_{wc} \cdot \frac{1}{TR} \tag{3.3}$$

The total control penalty cost per job is given in Equation (3.4). It must be noted that, $1/T_{pc}$ is the control rate per period.

$$TCC = \frac{1}{T_{pc}} \cdot u_{cc} \cdot \frac{1}{TR} \tag{3.4}$$

The total inventory penalty cost per job is the summation of inventory holding costs at each workstation;

$$TIC = \sum_{i=1}^{M} IC_i \tag{3.5}$$

$$IC_i = H_i \cdot E(N_i) \cdot \frac{1}{TR} \tag{3.6}$$

We know that $H_i = F \cdot P_i$, then, replacing H_i in Equation (3.6) leads to the following equations;

$$TIC = \sum_{i=1}^{M} F \cdot P_i \cdot E(N_i) \cdot \frac{1}{TR} \tag{3.7}$$

$$TIC = \frac{F}{TR} \cdot \sum_{i=1}^{M} P_i \cdot E(N_i) \tag{3.8}$$

The value added to one unit of WIP being hold at the first workstation, P_1, can be calculated using the following equation;

$$P_1 = \frac{1}{\mu_1} \cdot \left(c_1 \cdot u_{mc} + E(n_1) \cdot u_{wc} + \frac{1}{T_{pc}} \cdot u_{cc} \right) \tag{3.9}$$

This equation says that the cost of processing one unit of *WIP* at WS_1 is equal to the production cost per job, and this cost is equal to the monetary value of one unit of *WIP* at WS_1. Since the costs are added incrementally, the following equation gives the recursive relation to calculate the value of the unit inventory at the downstream WS_i's.

$$P_{i+1} = P_i + \frac{1}{\mu_{i+1}} \cdot \left(c_{i+1} \cdot u_{mc} + E(n_{i+1}) \cdot u_{wc} + \frac{1}{T_{pc}} \cdot u_{cc} \right) \tag{3.10}$$

Equation (3.10) states that the monetary value of one unit of *WIP* at WS_{i+1} is equal to the value added to the unit inventory at WS_i plus the value added to the unit inventory at WS_{i+1}. Replacing P_i in Equation (3.8) with Equation (3.10), we simply obtain the following equation;

$$TIC = \frac{F}{TR} \cdot \sum_{i=0}^{M-1} \left[\left(P_i + \frac{1}{\mu_{i+1}} \cdot \left(c_{i+1} \cdot u_{mc} + E(n_{i+1}) \cdot u_{wc} + \frac{1}{T_{pc}} \cdot u_{cc} \right) \right) \cdot E(N_{i+1}) \right] \quad (3.11)$$

According to the formulations given so far, the unit penalty cost function can be represented by summing up Equation (3.2), Equation (3.3), Equation (3.4), and Equation (3.11). This sum is used to compare the performance of the alternative flexibility policies under different operating conditions.

The unit penalty cost function includes three important terms to be calculated. These are $1/TR$, $E(n_i)$, and $E(N_i)$. The others are just parameters which are used for defining the characteristics of the problem that we are dealing with. Whereas $1/TR$, and $E(N_i)$ are depending on the characteristics of *CQN*, $E(n_i)$ is depending on the flexibility policy which is applied. $1/TR$, and $E(N_i)$ can be calculated by applying classical *CQN* analytical solution methods when the system is not *DRC*. However, $E(n_i)$ is not tractable in terms of *CQN*. Therefore, the simulation model of CONWIP line will be validated under the condition $n = \sum_{i=1}^{M} c_i$, or simply, $E(n_i) = c_i$ (i.e. the flexibility is not allowed). So, the general unit penalty cost function, which is obtained by summing up Equation (3.2), Equation (3.3), Equation (3.4), and Equation (3.11), should be converted to a new form to be able to use *CQN* results. For validation, the unit penalty cost function is used under the following special condition.

Let us assume the case given below;

$$\left.\begin{matrix} E(n_i)=1 \\ c_i=1 \end{matrix}\right\}, i=1,\ldots,M$$

Based on the special case given above, if there are M WS_i's in the line, then the number of total machines and the workers are equal to M (i.e., $n=\sum_{i=1}^{M} c_i = M$). Under this special condition, the components of the unit penalty cost function become,

$$TMC = M \cdot u_{mc} \cdot \frac{1}{TR} \tag{3.12}$$

$$TWC = M \cdot u_{wc} \cdot \frac{1}{TR} \tag{3.13}$$

$$TCC = \frac{1}{T_{pc}} \cdot u_{cc} \cdot \frac{1}{TR} \tag{3.14}$$

Since the flexibility is not allowed uder this special case, the TCC should be zero. In this special case, the T_{pc} can also be thought as having an infinite value. If the control period is infinite, it means that there will not be any control. This condition can be represented as $\underset{T_{pc}\to\infty}{Lim}\frac{1}{T_{pc}}=0$, then the TCC becomes zero. For concreteness, we will continue as if the TCC is not equal to zero. If the Equation (3.11) is simplified by adding our special conditions, then the TIC becomes,

$$TIC = \frac{F}{TR} \cdot \sum_{i=0}^{M-1}\left[\left(P_i + \frac{1}{\mu_{i+1}} \cdot \left(u_{mc} + u_{wc} + \frac{1}{T_{pc}} \cdot u_{cc}\right)\right) \cdot E(N_{i+1})\right] \tag{3.15}$$

If we expand the recursive Equation (3.10) for the first two terms (i.e., i=0, and i=1) under the special condition, we obtain the following equations;

$$P_{i+1} = P_i + \frac{1}{\mu_{i+1}} \cdot \left(u_{mc} + u_{wc} + \frac{1}{T_{pc}} \cdot u_{cc} \right) \tag{3.16}$$

$$P_1 = \underbrace{P_0}_{0} + \frac{1}{\mu_1} \cdot \left(u_{mc} + u_{wc} + \frac{1}{T_{pc}} \cdot u_{cc} \right) = \frac{1}{\mu_1} \cdot \left(u_{mc} + u_{wc} + \frac{1}{T_{pc}} \cdot u_{cc} \right)$$

$$P_2 = P_1 + \frac{1}{\mu_2} \cdot \left(u_{mc} + u_{wc} + \frac{1}{T_{pc}} \cdot u_{cc} \right) = \left(\frac{1}{\mu_1} + \frac{1}{\mu_2} \right) \cdot \left(u_{mc} + u_{wc} + \frac{1}{T_{pc}} \cdot u_{cc} \right)$$

We can easily generalize the recursive relation for P_i's given above as;

$$P_i = \left(u_{mc} + u_{wc} + \frac{1}{T_{pc}} \cdot u_{cc} \right) \cdot \sum_{j=1}^{i} \frac{1}{\mu_j} \tag{3.17}$$

The term $\sum_{j=1}^{i} \frac{1}{\mu_j}$ in Equation (3.17) can be defined as the total workload capacity as it was given earlier. If we refer this term as WL_i, then, P_i becomes;

$$P_i = \left(u_{mc} + u_{wc} + \frac{1}{T_{pc}} \cdot u_{cc} \right) \cdot WL_i \tag{3.18}$$

It must be noted that if c_i's are not equal to one and the system is not *DRC*, then WL_i is defined as;

$$WL_i = \sum_{j=1}^{i} \frac{c_j}{\mu_j} \tag{3.19}$$

Since the flexibility is not considered under these special conditions, Equation (3.18) can be reduced to,

$$P_i = WL_i \cdot \left(u_{mc} + u_{wc} \right) \tag{3.20}$$

If Equation (3.20) is used, then the *TIC* function becomes;

$$TIC = \frac{F}{TR} \cdot (u_{mc} + u_{wc}) \cdot \sum_{i=1}^{M} WL_i \cdot E(N_i) \tag{3.21}$$

Lastly, all of the cost components for our special condition can be written as follows;

$$TC = \frac{M}{TR} \cdot (u_{mc} + u_{wc}) + \frac{F}{TR} \cdot (u_{mc} + u_{wc}) \cdot \sum_{i=1}^{M} WL_i \cdot E(N_i) \tag{3.22}$$

If Equation (3.22) is simplified, the following general unit penalty cost function can be obtained when the system is not *DRC* (i.e., machines are fully staffed);

$$TC = \left(\frac{u_{mc} + u_{wc}}{TR} \right) \cdot \left(M + F \cdot \sum_{i=1}^{M} WL_i \cdot E(N_i) \right) \tag{3.23}$$

The Equation (3.23) gives the *TC* under the condition $n = \sum_{i=1}^{M} c_i = M$, while $WL_i = \sum_{j=1}^{i} \frac{1}{\mu_j}$.

The Equation (3.24) which is given below represents the *TC* under the condition $\sum_{i=1}^{M} c_i = n$, while $WL_i = \sum_{j=1}^{i} \frac{c_j}{\mu_j}$.

$$TC = \left(\frac{u_{mc} + u_{wc}}{TR} \right) \cdot \left(\sum_{i=1}^{M} c_i + F \cdot \sum_{i=1}^{M} WL_i \cdot E(N_i) \right) \tag{3.24}$$

The general form of the unit penalty cost is Equation (3.24) when the worker flexibility is not allowed. Equation (3.23) is the special case of Equation (3.24). It can be obtained by setting all c_i's to one. The Equation (3.24) implies that the unit

penalty cost depends on two important terms which were mentioned before; The throughput rate (i.e., *TR*) of the system and the average *WIP* level at each workstation (i.e., $E(N_i)$). Since this equation is valid for the CONWIP systems which are not *DRC*, these terms can be calculated by applying *CQN* formulas.

The throughput rate of a *CQN* with *M* serially connected workstations is given by the following equation (Hopp & Spearman, 2001);

$$TR(K) = \frac{K}{WL_M \cdot \mu + K - 1} \cdot \mu \tag{3.25}$$

The Equation (3.25) is the result of Mean Value Analysis (*MVA*) under the following special conditions (Hopp & Spearman, 2001);

- Equal mean service times (i.e., the line is perfectly balanced),
- Single machine at each workstation (i.e., no parallel machines),
- The distribution of service time is exponential with the mean of $1/\mu$.

If these conditions are not held, Equation (3.25) will be invalid. For example, if there is single machine at each workstation, the service time distributions are not exponential, or the mean service times are not equal, the *MVA* should be used. The *MVA* is an algorithm introduced firstly by Reiser et al. (1980). However, the *MVA* does not cover the cases that parallel machines exist. When parallel machines exist, Marginal Distribution Analysis (*MDA*) should be used (Buzacott & Shanthikumar, 1993). Furthermore, Equation (3.25) is valid just for the *CQN* that we are interested in this thesis, however, the *MVA* and *MDA* are valid for general *CQN*'s which is called Single-Class Closed Jackson Queuing Networks (Buzacott & Shanthikumar, 1993). Since these algorithms require the evaluation of a set of recursive equations, and there is no closed form formulas related to these algorithms, the simulation model of CONWIP line will be validated under the special operating conditions given above. The other performance measures of related *CQN* can be calculated easily by applying Little's Law (i.e., $L = \lambda \cdot W$, where, *L*: The long term average of

total number of jobs in the system, λ:The mean arrival rate, W: The average time that a job spends in the system, or the average flow time).

Equation (3.25) gives the throughput rate of a CONWIP line for given K. As it was mentioned in Section 3.2.1, WL_M represents the total workload capacity of first M workstations and it is equal to $\sum_{i=1}^{M} \frac{1}{\mu_i} = \frac{M}{\mu}$. If the expression $\frac{M}{\mu}$ is replaced with WL_M in Equation (3.25), the following equation is obtained;

$$TR(K) = \frac{K}{M + K - 1} \cdot \mu \tag{3.26}$$

It must be noted in Equation (3.26) that if the level of K is set to the infinity, the resultant system has the throughput rate of μ. This means that increasing the level of K, increases the throughput rate of the system, and the upper bound on the throughput rate is μ. However, in practice, it is impossible to operate the system with high levels of K due to the limits on place and due to the inventory holding cost. Therefore, in these case, the increase in the throughput rate of the system can be provided just by additional capacities (i.e., increasing c_i's) for low levels of K.

Since the line is perfectly balanced, the K jobs are distributed equally to the workstations, then, each sub system consisting of q_i and S_{ij} will have $\frac{K}{M}$ number of jobs in total (we used the term "sub system" due to the reason that our workstations consist of S_{ij} and q_{i+1}. However, this makes no difference because all ρ_i's and $E(nq_i)$'s are equal throughout the line).

The utilizations of machines can be calculated using Equation (3.26). As it was given in Section 3.2.1, the utilization of a machine is given by $\frac{\lambda}{c_i \cdot \mu_i}$ providing that $\rho_i < 1$, for all $i = 1,\ldots,M$. Since each workstation has single machine, and the service times are equal, the utilization of each machine in the system can be

calculated by $\frac{\lambda}{\mu}$. The arrival rate of this closed system is equal to *TR*. Therefore, the utilization of each machine when there are *K* jobs in the system can be represented by the Equation (3.27).

$$\rho_i = \frac{K}{M+K-1} \text{ for all } i = 1,\ldots,M \tag{3.27}$$

This equation states that increasing the level of *K*, increases the utilization of the machines. Since each sub system consisting of q_i and S_{ij} has $\frac{K}{M}$ number of jobs, the expected number of jobs waiting in q_i is given as follows;

$$E(nq_i) = \frac{K}{M} - \frac{K}{M+K-1} \text{ for all } i = 1,\ldots,M \tag{3.28}$$

By using Little's Law given before, the expected time that a job spends in the sub system can be calculated as follows;

$$W_i = \frac{M+K-1}{M \cdot \mu} \text{ for all } i = 1,\ldots,M \tag{3.29}$$

The Equations from (3.25) to (3.29) are related to the closed system which consists of serially connected workstations controlled by CONWIP mechanism providing that each workstation has single machine with equal exponential service time distributions. Moreover, the worker flexibility is not allowed because each machine has its own dedicated worker. Based on these equations, the proposed unit penalty cost function given in Equation (3.23) is converted to Equation (3.30) by replacing *TR* and $E(N_i)$ with related equations.

$$TC = (u_{mc} + u_{wc}) \cdot \left(\frac{M+K-1}{K \cdot \mu}\right) \cdot \left(M + F \cdot \left(\frac{K}{M \cdot \mu} \cdot \sum_{i=1}^{M-1} i + \frac{K}{\mu \cdot (M+K-1)} \cdot \sum_{i=1}^{M} i\right)\right) \tag{3.30}$$

As a summary, two different unit penalty cost functions are developed in this section. The first one which consists of Equation (3.2), Equation (3.3), Equation (3.4), and Equation (3.11) will be used to compare the alternative flexibility policies under different operating conditions (e.g., balanced line, unbalanced line). Since this function is developed under the condition that the system is *DRC*, the terms $1/TR$, $E(N_i)$, and $E(n_i)$ can not be calculated by using *CQN* analytical solution methods. So, we employed the simulation to estimate the *TC* when the CONWIP line is *DRC*. The second unit penalty cost function (i.e., Equation (3.30)) is developed under the condition that the system is not *DRC*. In this case, there are analytical *CQN* results for the terms $1/TR$, $E(N_i)$, and $E(n_i)$. So, this function can be used for validating the simulation model of the CONWIP line with M serially connected workstations each having one machine assuming that the mean service times are equal and the service time distributions are exponential. Equation (3.30) is used in next chapter for a hypothetical CONWIP line to validate the simulation results. In following section, a new worker flexibility policy for a CONWIP line having parallel machines is explained. It must be noted that this rule is developed to be used under the condition that the demand is saturated.

3.4 A New Worker Flexibility Policy for CONWIP Lines

As it was mentioned before, a typical problem in *DRC* system is how to assign the workers dynamically to the machines. This issue is investigated under the topic of dynamic worker assignment heuristics in *DRC* literature. These rules include the decisions on when and where to move the workers. So, these rules are also called as when-where rule pairs as it was given in chapter one.

The most widely used "when" rules in the literature are the centralized and decentralized rules. The centralized rule implies that the workers are eligible for transfer after the completion of each job at their current workstations. The centralized rule provides the maximum worker flexibility because the worker is available for transfer each time the job is completed. However, if the worker transfer delays are significant, the effective worker capacity decreases dramatically due to the reason

that applying this rule increases the number of transfers. The decentralized rule implies that the workers are eligible for transfer when there is no job waiting to be processed at their current workstations. This rule is used especially when the transfer times are significant. Since the centralized and decentralized rules are easy to apply, the most attention was given to these rules. Moreover, there are other types of "when" rules in various studies (Nelson, 1967, Fryer, 1974, Gunther, 1979, and Treleven, 1987). In our worker flexibility policy, the timing of a worker transfer depends on the value of T_{pc}, which is the length of the periodic control. In other words, the state of the system is controlled periodically. If the value of T_{pc} is short relative to processing times, it implies more frequent controls to change the place of a worker. If the value of T_{pc} is long relative to machines processing times, the performance opportunity provided by the worker flexibility tends to decrease. By changing the value of T_{pc}, we have a control on the number of state evaluations, that is, the number of possible worker transfers. Since the unit penalty cost function introduced earlier has a cost term which depends on the number of state evaluations (i.e., controls), the less number of controls implies lower costs. So, there is a trade-off between the cost of more evaluations of current system status and the performance improvement obtained.

There are also a lot of "where" rules widely used in *DRC* literature. However, their effect on the shop performance has not been reported as strong as "when" rules (Treleven & Elvers, 1985). The typical "where" rules are *FISFS*, *LNQ*, *EDD*, etc. It must be noted that since the majority of studies focused on the job shops, the "where" rules are designed according to the typical performance measures used in job shops. So, most of the "where" rules are not proper to use in a CONWIP line. The worker flexibility policy proposed in this thesis uses the queue length information to transfer the workers.

3.4.1 The Proposed Policy for CONWIP lines

Besides the terms and symbols introduced in Section 3.2, we used the following terms in developing the worker flexibility policy which is referred as Periodic

Control (*PC*) in following chapters;

t: Time.

totalcap: Total number of machines in the system.

total: Total number of workers.

$v_1,\ldots,v_8$: The control variables used in the loops.

minindex: The index of the workstation which has the minimum number of jobs in its queue at that control point.

maxindex: The index of the workstation which has the maximum number of jobs in its queue at that control point.

load(M): *load*(1),...,*load*(*M*). The number of jobs in $q_1,\ldots,q_M$.

workload(M,2): Used in keeping the workstation indexes in the first column and the number of jobs at related workstations in the second column.

minload(M,2): Used in sorting the queue lengths from minimum to maximum. It keeps the sorted queue lengths in the second column, and their related workstation indexes in the first column.

maxload(M,2): Used in keeping the sorted (i.e., from maximum to minimum) queue lengths in the second column, and their related workstation indexes in the first column.

dummy1: Used in sorting algorithm.

dummy2: Used in sorting algorithm.

The algorithm developed to implement the worker flexibility consists of five different parts;

- Initializing of the system.
- Assigning the time to the next control point.
- Sorting the queue lengths from minimum to maximum.

- Sorting the queue lengths from maximum to minimum.
- Transferring the workers from one workstation to another.

We developed the following algorithm in implementing the proposed worker flexibility policy;

/* Initialization of the system */

$t = 0$;

$totalcap = 0$;

$total = n$;

$i = 1$;

while $i <= M$

$n_i(t) = 0$;

$totalcap = totalcap + c_i$;

$i = i + 1$;

endwhile

$i = 1$;

while $i <= M$

while $(c_i > n_i(t))$.**and.**$(total > 0)$

$n_i(t) = n_i(t) + 1$;

$total = total - 1$;

endwhile

$i = i + 1$;

endwhile

/* Assigning the time to the next control point */

$t = t + T_{pc}$; **10**

/* Sorting the queue lengths from minimum to maximum */

$v_1 = 1$;

while $v_1 <= M$

$minindex = 0$;

maxindex = 0 ;

load(v_1) = $nq_{v1}(t)$;

workload(v_1,1) = v_1 ;

workload(v_1,2) = *load*(v_1) ;

minload(v_1,1) = *workload*(v_1,1) ;

minload(v_1,2) = *workload*(v_1,2) ;

$v_1 = v_1 + 1$;

endwhile

$v_2 = 1$;

while $v_2 <= (M - 1)$

$v_3 = v_2 + 1$;

while $v_3 <= M$

if *minload*(v_2,2) > *minload*(v_3,2)

dummy1 = *minload*(v_2,1) ;

dummy2 = *minload*(v_2,2) ;

minload(v_2,1) = *minload*(v_3,1) ;

minload(v_2,2) = *minload*(v_3,2) ;

minload(v_3,1) = *dummy1* ;

minload(v_3,2) = *dummy2* ;

$v_3 = v_3 + 1$;

else

$v_3 = v_3 + 1$;

endwhile

$v_2 = v_2 + 1$;

endwhile

/* Sorting the queue lengths from maximum to minimum */

$v_4 = 1$;

while $v_4 <= M$

maxload(v_4,1) = *minload*($M - v_4 + 1$,1) ;

maxload(v_4,2) = *minload*($M - v_4 + 1$,2) ;

$v_4 = v_4 + 1$;

endwhile

/* The algorithm which transfers the workers from one workstation to another */

$v_5 = 1$;

$minindex = minload(v_5,1)$;

if $nr_{minindex}(t) < n_{minindex}(t)$

$v_6 = 1$;

$maxindex = maxload(v_6,1)$; **20**

if $c_{maxindex} > n_{maxindex}(t)$

if $load(maxindex) > load(minindex)$

$n_{minindex}(t) = n_{minindex}(t) - 1$;

$n_{maxindex}(t) = n_{maxindex}(t) + 1$;

go to 10;

else

if $v_6 < M$ **30**

$v_6 = v_6 + 1$;

go to 20 ;

else

go to 40 ;

else

go to 30 ;

else

if $v_5 < M$ **40**

$v_5 = v_5 + 1$;

else

$v_7 = 1$;

if $v_7 <= M$ **50**

if $n_{v7}(t) = = 0$

$v_8 = 1$;

if $v_8 <= M$ **60**

if $nr_{v8}(t) < n_{v8}(t)$

```
                    maxindex = v7 ;
                    minindex = v8 ;
                else
                    v8 = v8 +1 ;
                    go to 60 ;
            else
                v7 = v7 +1 ; .....70
                go to 50 ;
        else
            go to 70 ;
    else
        go to 10 ;
```

The first part of the algorithm initializes the number of workers at each workstation. The workers are distributed to the workstations according to the number of machines at each workstation at time t=0. For example, if the line has five workers with five workstations, and c_i=2, for all i=1,...,5, the algorithm is initialized as $n_1(0)$=2, $n_2(0)$=2, $n_3(0)$=1, $n_4(0)$=0, and $n_5(0)$=0 by filling up the maximum capacities as possible as in order from the beginning of the line to the end.

In second part of the algorithm, the time t is advanced to the next control point in time. If T_{pc} is very short in comparison to the processing times, the number of state evaluations increases. So, the state of the system is controlled more frequently to allow more worker transfers. However, if T_{pc} is considerably long, the number of possible replacements decreases which in turn cause a decrease in possible performance improvement. This type of modelling provides more general flexibility of the workers in comparison to typical centralized and decentralized rules. The study of Gunther (Gunther, 1979) also uses time information to transfer the workers. In his study, a worker is allowed to move to another workstation when there is no job in current workstation. He may also move to another station at any time t if his job has been completed, there are jobs in queue at other workstations, and t-$t_0 \geq d$. Here t_0 is the arrival time of a worker at his current workstation and d is a parametric

weight. The difference between our proposed timing mechanism and Gunther's is that we do not use the arrival time of the individual workers. We evaluate the state of the whole system after each T_{pc} time unit.

The third part of the algorithm uses the information on queue lenghts. Each time the current state is controlled, the system status is evaluated and the workstations are put in an increasing order with respect to their queue lengths. At the end of this part, the array called *minload* includes the workstation indexes and the queue lengths in increasing order. For example, if the queue lengths are 4, 5, 2, and 3 (M=4) at the time of control, the arrays mentioned take the following values;

$$load_{4\times1} = \begin{bmatrix} 4 \\ 5 \\ 2 \\ 3 \end{bmatrix}, \quad workload_{4\times2} = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 2 \\ 4 & 3 \end{bmatrix}, \quad minload_{4\times2} = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 2 \\ 4 & 3 \end{bmatrix}$$

After sorting the workstations *minload* takes the form,

$$minload_{4\times2} = \begin{bmatrix} 3 & 2 \\ 4 & 3 \\ 1 & 4 \\ 2 & 5 \end{bmatrix}$$

The fourth part of this algorithm uses the sorted *minload* to obtain *maxload* given below.

$$maxload_{4\times2} = \begin{bmatrix} 2 & 5 \\ 1 & 4 \\ 4 & 3 \\ 3 & 2 \end{bmatrix}$$

The last part of the algorithm is the part that transfers the workers. Since the system that we are dealing with is a CONWIP line with parallel identical machines,

this algorithm was developed to increase the effective worker capacity of this system. In general, the worker transfer algorithm works as follows;

- The worker is transferred from a workstation which having the minimum queue length to the workstation having the maximum queue length at the control point providing that,

  - the workstation which has the minimum queue length has at least one idle worker,

  - the workstation which has the maximum queue length is not fully staffed, and

  - the number of jobs waiting in the workstation which has the minimum queue length is lower than the number of jobs waiting in the workstation which has the maximum queue length.

- If the first and the second conditions are satisfied but the third one is not, the workstation which has the next maximum queue length is selected as target workstation, and the last two conditions are evaluated again. If the first condition is satisfied but the second one is not, in this case, the previous rule is applied again. This loop continues until the workstation has the last maximum (i.e., minimum) queue length. If the transfer of a worker can not be achieved at the end of the loop, the workstation which has the next minimum queue length is selected, and all three conditions are evaluated again. This rule is valid for the case in which the first condition given above is not satisfied. So, the algorithm scans for the first possible transfer that satisfies the three conditions.

- If a worker can not be transferred according to the rules above, the algorithm searches for the first workstation which is not staffed. This workstation is staffed by the first workstation which has at least one idle worker. If still there is no possible transfers, the state of the system is not changed. So, no worker is transferred until the next control point.

The study of Treleven (Treleven, 1987) uses the minimum and maximum queue lengths as in this study. The worker assignment rules in his study are called as "Pull rules", and they are used for determining both the time and the place of the transfers.

However, the proposed flexibility policy in this thesis uses T_{pc} for timing of the transfer, and uses the minimum/maximum queue lengths for the place of the transfer. Furthermore, our methodology searches for all possible transfers.

The advantages of this policy can be listed as follows; 1. The policy evaluates all the possible transfer alternatives. 2. Since only one worker may be replaced at the control point, this policy may limit the crowd running around the line by increasing the efficiency of the line. 3. The CONWIP level may be set to higher levels to use wider control intervals. The less number of controls results the less number of worker movements. If the movement times are not negligible due to ineffective layout, this property may provide better performance measure in terms of throughput rate. 4. This policy gives the flexibility to the manager to set the control parameters for reaching a predetermined production target. 5. By setting the parameters to the extreme cases, such as very short control periods relative to the processing times or high level of CONWIP, or both, the manager may observe how the performance of the line can be affected.

In this section, a new worker flexibility policy for CONWIP controlled lines with identical parallel machines was given. The policy was developed especially for the closed systems. The performance of this flexibility policy was compared with the performance of three alternative policies in next chapter. The comparison criteria were the mean cycle time and the unit penalty cost.

3.5 A Simple Methodology to Minimize the Unit Penalty Cost

As it was given in previous chapter, if the CONWIP line is not *DRC* and the demand is saturated, a typical problem is to minimize the level of CONWIP subject to a given throughput rate (Gstettner & Kuhn, 1996), (Dar-el et al., 1999). This problem can be represented in our terms and notation as follows;

Min K

S.T.

$TR \geq TR_{lim}$

where,

TR_{lim}: The target throughput rate

The solution procedure to this problem requires the evaluation of different levels of CONWIP, that is, K. If the CONWIP system with saturated demand consists of two resources (i.e., the system consists of machines, and machines are operated by workers), the throughput rate of the system can be increased by;

- increasing the level of CONWIP,
- adding equal number of both workers and parallel machines to the system,
- adding just machines to the system,

In first case, increasing the level of CONWIP (when the identical parallel machines exist at the workstations or not) increases the throughput rate of the system. So, it is clear that there is a practical limit on the level of CONWIP. The second case represents the increase in the capacity of the system. However, since the number of machines and workers are considered as equal, the system is not *DRC*. The last case represents the problem that we dealt with in this thesis. Adding just machines to the system requires the use of flexible workers. It must be noted that adding just workers has no effect on the capacity of the system.

Implementing the worker flexibility policy requires that the number of machines be greater than the number of workers. So, the decision variables of the problem are c_i's, K, and T_{pc}. This problem may be represented mathematically as follows;

Min TC

S.T.

$1/TR \leq 1/TR_{lim}$

$K \leq K_{lim}$

Where,

$1/TR_{lim}$: The target cycle time

K_{lim} : The maximum level of K allowed in the system

Based on the mathematical representation given above, it is assumed that the line consists of M workstations each having one machine and each machine is operated by one worker (i.e., $n=M$) initially. Our main goal is to find out the capacity structure (i.e., the number of machines at each workstation) of the line and the combination of K and T_{pc} that minimize the unit penalty cost to reach the target cycle time of $1/TR_{lim}$. Therefore, we suggest a two-stage methodology. The objective in first stage is to find out the capacity structure of the line, and the objective in second stage is to determine the optimum levels of K and T_{pc}. So, the following algorithm represents the methodology that we developed for minimizing the unit penalty cost;

The First Stage of the Methodology

Step 0. Set $c_i=1$ for all $i=1,\ldots,M$

Set $n=M$

Set $K_0=n+1$

Set $K=K_0$

Set $T_{pc}=T_{pc0}$

Step 1. Find $\max\{1/(\mu_i.c_i)$: $i=1,\ldots,M\}$ which is the most downstream WS_i.

Step 2. Add one machine to the workstation which is found in *Step 1*.

Step 3. Simulate the new system for the current K and T_{pc}.

Step 4. Is the cycle time of the new system is lower than $1/TR_{lim}$?

YES, Go to *Step 7*.

NO, Set $K=K_{lim}$ and Go to *Step 5*.

Step 5. Simulate the new system for the current K and T_{pc}.

Step 6. Is the cycle time of the new system is lower than $1/TR_{lim}$?

YES, Go to *Step 7*.

NO, Set $K=K_0$ and Go to *Step 1*.

The Second Stage of the Methodology

Step 7. Apply *RSM* to optimize the levels of K and T_{pc}.

As mentioned before, the first stage of the methodology finds the capacity structure. This is realized by sequentially adding one more machine to the bottleneck workstation. The sequential nature of the machine adding process allow us to obtain the minimum number of additional machines to reach the target cycle time (i.e., less additional investment). After completing the first stage, the *RSM* can be applied to minimize the unit penalty cost.

3.5.1 The Steps of the Methodology

This section explains the steps of the proposed methodology given above.

Step 0: Since it is assumed that the original CONWIP system consists of M workstations each having one machine, and each machine is operated by one worker (i.e., no flexibility), all c_i's are set to one, the number of workers is set equal to the number of workstations (i.e., $n=M$). The K_0 represents the least number of jobs in the CONWIP system for a given n so that the *PC* policy can be applied. By setting $K_0=n+1$, it is provided that there is at least one job waiting in queue when all the workers are busy. The term T_{pc0} represents the initial value of T_{pc}. Based on the pilot studies we observed that the lower the value of T_{pc} the lower the estimated mean cycle time. Therefore, in setting the initial value of T_{pc}, we assumed that it is 20% of the theoretical minimum cycle time. The theoretical cycle time of the system can be calculated by using Equation (4.2). For example, let us assume that the CONWIP line has four workstations and the mean service times are 5 minutes (i.e., balanced line). This system should be initialized as $c_1=c_2=c_3=c_4=1$ machine, $n=4$ workers, $K_0=4+1=5$ jobs, and $T_{pc0}=1$ minute.

Step 1, Step 2: In these two steps, the bottleneck workstation having the maximum mean service time is identified. First, the value of $1/(\mu_i.c_i)$ for all $i=1,\ldots,M$ is calculated and the workstation which has the maximum value of $1/(\mu_i.c_i)$ is identified as the bottleneck workstation. It is important to identify the bottleneck workstation because it points out the station where adding more capacity will enhance the performance of the line. If the bottleneck workstation is not unique,

selecting the most downstream one as a bottleneck workstation seems a better choice since additional capacity at the most downstream workstation results in lower inventory cost (see Equation (3.10) and Equation (3.11)). For example, let us assume the sample case which was given in the previous step. According to the parameters of the hypothetical system given in the previous step, all of the workstations have equal chance to become the bottleneck. The value of $1/(\mu_i.c_i)$'s are equal to 0.2 for all i=1,...,4. Since the bottleneck is not unique, a new machine is added to the most downstream workstation, and the new capacity structure becomes *1112* (i.e., c_1=c_2=c_3=1, and c_4=2). In the following steps, the bottleneck workstation for this capacity structure becomes the third one, and so on. If the line is not balanced due to different mean service times (i.e., 4, 5, 3, and 3 minutes), then, the new machines are added as follows; *1111*, *1211*, *1212*, *1222*, and so on.

***Step 3, Step 4, Step 5, Step 6*:** After identifying the bottleneck workstation and adding one machine to that workstation, the new CONWIP system is simulated for the current levels of K, T_{pc}, and n. Under the worker flexibility policy, *PC*, if the cycle time is estimated as lower than the target level, $1/TR_{lim}$, the first stage of the algorithm is completed, and the *RSM* is applied under the final capacity structure. If the estimated cycle time is not lower than the target level, then, the value of K is set to K_{lim} and the system is simulated again. By setting the value of K to K_{lim}, we want to see whether the *PC* policy can reach the target cycle time under this capacity structure or not. If the target cycle time can be reached, the algorithm stops and the *RSM* is applied, otherwise, the control is sent back to *Step 1* and this process continues until the target cycle time is reached.

***Step 7*:** In this step, the final capacity structure which was found in first stage of the methodology is used. It must be noted that the objective in earlier steps was to determine the operating conditions under which the target cycle time is reached with minimum number of machines. Hence, we wanted to have sound initial conditions to start the *RSM*, so that the optimum values of K and T_{pc} which lead to minimum unit penalty cost can be reached with less number of iterations. The *RSM* requires the use of Design of Experiments and Regression Analysis, and as mentioned before, the

implementation of *RSM* requires to carry out the following studies;

- Two control factors which are K and T_{pc} are considered in this *RSM* study. The performance measure is the unit penalty cost (*TC*). We employ the 2^2 full factorial design with center points to estimate the regression coefficients.
- To apply the steepest descent method, initially, we set K to a low value and T_{pc} to a high value and we standardize the factor levels to -1 (low value) and +1 (high value) in order to compare the factor effects by relative importance.
- After executing the simulation model at each design point, the least square estimation (*LSE*) procedure is applied to fit the response to a first-order model. The residuals are examined to have a confidence on the assumptions of the *LSE*.
- The significances of the model parameters and the lack-of-fit are checked. If the first-order model is not adequate in representing the simulated response then, either more terms (i.e., interaction term) are added to the first-order model or a second-order model is fit.
- If the first-order model is found to be adequately representing the simulated response, the optimization process is started. The steepest descent method is used to produce a path which results in a maximum decrease in response. The experimental runs are carried out along this path until the point in which the improvement in response eventually disappears is reached. A new experiment is conducted in the neighborhood of this point to fit a first-order model again, then, same procedure which was explained is followed. This process is continued until the new design has the interaction or curvature effects. If it is so, the second-order model is fit. To fit the second-order model to the simulation responses, the Face Center points are added to the previous design.
- After fitting a second-order model, the lack-of-fit is checked. If there is no lack-of-fit, the canonical analysis is applied to find out the optimum levels of K and T_{pc}. Since the unit penalty cost also depends on the cycle time, the optimum levels of these factors will satisfy the condition $1/TR \leq 1/TR_{lim}$ most probably. However, if this condition is not satisfied, the level of K is increased by one unit, and the

model is executed under this condition. This process continues until the target cycle time is reached providing that $K \leq K_{lim}$.

The application of this methodology to a hypothetical case is given in chapter four.

In summary, in this chapter, the structure of the unit penalty cost, the proposed worker flexibility policy, and the two-stage simulation optimization methodology to minimize the unit penalty cost were given. The experimental studies carried out to compare the performance of the proposed policy with alternative policies under different operating conditions and the results of implementing the two-stage simulation optimization methodology are presented in next chapter.

CHAPTER FOUR
EXPERIMENTAL STUDIES

4.1 Introduction

The experimental studies carried out can be placed into three groups;

- The comparisons of alternative flexibility policies with the proposed flexibility policy on a hypothetical CONWIP line under different operating conditions.
- Evaluating the effects of changing the factor levels of CONWIP (K), and periodic control length (T_{pc}) on mean cycle time and the unit penalty cost.
- The application of two-stage simulation optimization methodology on a hypothetical CONWIP controlled manufacturing line operating under the saturated demand.

The first group of experimental studies compares the performance of the proposed flexibility policy with the alternative flexibility policies under different operating conditions. The alternative policies are "Pick and Run" (*PR*), "Decentralized-*LNQ*" (WW_0), and "Centralized Control-*LNQ*" (WW_K). The *PR* policy gives the minimum cycle time (i.e., maximum throughput rate) under the assumptions given in previous chapter. Therefore, the *PR* policy can be used as a benchmark. The other two policies are widely used in *DRC* Job Shops. However, their performances are not known in CONWIP controlled manufacturing lines operating under the saturated demand. The proposed flexibility policy (*PC*) was compared with these three policies based on mean cycle time and the unit penalty cost. These comparison studies were carried out under different operating conditions to observe the effects of changing operating conditions on performance of these policies. The second group of experimental studies involves evaluating the effects of changing the factor levels of K and T_{pc} on

performance of the proposed policy. The third group of the experimental studies employs the two-stage simulation optimization methodology to minimize the unit penalty cost.

These experimental studies were carried out using a hypothetical serial manufacturing line which was previously introduced in Section 3.1 (see Figure 3.1). The main features of this system are;

- The system consists of four workstations in tandem configuration. The production and inventory control are provided by CONWIP and the demand is saturated. The level of CONWIP is K.
- Each workstation consists of identical parallel machines. The number of parallel machines at workstation "i" is denoted as c_i for all i=1,...,4. Each workstation has its input and output buffers. The output buffer of workstation "i" serves as the input buffer of workstation "i+1". Since the demand is saturated, there is no output buffer at the last workstation.
- The availability of raw material in front of first workstation is unlimited.
- Machines are not subject to breakdowns.
- Each identical machine in a workstation can be operated if and only if a worker is available. The total number of machines in the line is greater than the total number of workers.
- The system is capable of producing just one type of job.
- The jobs are transferred between the workstations asynchronously. There is no time delay for the transfer of jobs.
- The jobs are processed at the workstations with respect to *FISFS*.
- Any information, such as CONWIP card, is transmitted instantly.
- The workers are homogenous.
- The workers are full cross-trained.
- The jobs are noncollaborative.

- The preemption of job is not allowed.

4.2 The Experiment Plan and the Research Objectives

The first group of experimental studies was carried out in eight phases;

Phase 1: Development of the simulation models. In this phase, the CONWIP line without flexibility, the *PR*, WW_0, WW_K, and *PC* policies with CONWIP line were modeled using ARENA2.2. The simulation model of CONWIP line was validated based on the unit penalty cost. The simulation models are seen in Appendices. The research objective in this phase can be summarized as follows;

To develop a generic test bed. So, we can easily change the factors and their levels, such as the number of workers, the number of workstations, the number of machines, etc. without additional modeling effort during the experimental studies.

Phase 2: Satisfaction of the optimality condition of the *PR* policy. As it was mentioned before, the optimality conditions of the *PR* policy were $c_i = n$ for all $i = 1,\ldots,M$, and $K \geq n$ for a given n. Besides these conditions, setting K equal n guarantees that the *PR* policy reaches optimum with respect to both *TR* and *E(N)*. It gives the maximum throughput rate with minimum average *WIP* level of K by providing the equality $E(n_i) = c_i \rho_i$ given in Section 3.2.1. This equality represents the fully utilized workers, so the difference between $E(n_i)$ and $c_i \cdot \rho_i$ is caused by inefficiency of the flexibility policy in saturated demand case. It must be noted that since the number of identical parallel machines in a workstation is sufficient to assign all the workers in a single workstation, there will be no worker blockages. In this experiment, it was assumed that the hypothetical system was perfectly balanced with respect to mean service times. The research objective in this phase was;

To show how effective the alternative policies are under the optimality condition of the *PR* policy. Since the optimality condition of the *PR* policy requires $n \cdot M$ machines (i.e., there are M workstations each having c_i identical machines,

and $c_i = n$ for all $i = 1,\ldots,M$. So, the total number of machines is equal to $n \cdot M$) for a given n, this system can be thought as *DRC*. The WW_0 and WW_K are the most well known and efficient *DRC* dynamic worker assignment heuristics. However, their performances are not known in CONWIP controlled serial lines under the saturated demand. Therefore, besides the proposed policy, these two policies were also compared with the optimum *PR* policy.

Phase 3: Dissatisfaction of the optimality condition of the *PR* policy. In this experiment, it was assumed that the values of c_i's are less than n. Therefore, there may be worker blockages which may decrease the performances of the policies. The line is perfectly balanced with respect to mean service times. The research objective in this experiment was as follows;

To evaluate the effects of worker blockages on performances of the flexibility policies in a perfectly balanced line. Since the difference between $E(n_i)$ and $c_i \cdot \rho_i$ is caused by inefficiency of the flexibility policy as it was mentioned before, it is expected that these inefficiencies will increase in the case of worker blockages. In this experiment, we wanted to observe how efficient the policies were to overcome these blockages in a perfectly balanced line.

Phase 4: Unbalanced line due to differences in workstation capacities. This experiment was carried out under the conditions that the optimality condition of the *PR* will be violated and the manufacturing line will not be perfectly balanced with respect to c_i's. Furthermore, we assumed that the mean service time at each workstation is equal and the values of c_i's will differ. The research objective of this experiment was as follows;

To show how the policies compensate imbalances caused by inequalities in the capacities of workstations in the line. In this experiment, we wanted to observe which policy uses the additional capacity more effectively.

Phase 5: Unbalanced line due to differences in mean service times. Like in earlier phase, in this experiment, we also assumed that the optimality condition will be violated. Moreover, we assumed that the values of c_i's will remain constant while the values of μ_i's differ. The research objective in this experiment was as follows;

To show how the policies compensate imbalances caused by inequalities in the mean service times. In other words, we investigated the effect of bottleneck workstation(s) on the performances of the policies.

Phase 6: Increasing the total number of workers. The research objective in this experiment was as follows;

To show the effects of increasing the total number of workers on system performance. In this experiment, we investigated how sensitive the alternative worker flexibility policies were against increases in total number of workers.

Phase 7: Increasing the number of workstations. The research objective in this experiment was as follows;

To show which policy works better on lines having much more workstations. This experiment was conducted in order to generalize the simulation results obtained in earlier phases.

Phase 8: Including the worker transfer delays. The research objective in this experiment was as follows;

To show which policy is more sensitive to the transfer times of the workers. In the experiments which were given so far, the transfer times were negligible. However, if there is a time delay in replacing the workers, it might be important to observe how the performances of the policies are affected by this.

The second group of experimental studies involved the evaluation of the effects of the length of control period (T_{pc}) on the performance of proposed flexibility policy. Therefore, T_{pc}, and its interaction with K was investigated in this experiment. The research objective was as follows;

To show the relation between T_{pc} and K. In this experiment, we wanted to gain more insight into the behavior of the simulation model by screening for two control factors (i.e., T_{pc} and K), which had the greatest impact on the simulation response.

The third group of experimental studies involved the application of a two-stage simulation optimization methodology to minimize the unit penalty cost introduced earlier. The research objective was as follows;

To show how the unit penalty cost can be minimized by applying the two-stage simulation optimization methodology. In the first stage, the various sets of experiments are carried out to determine the minimum number of machines at each workstation (i.e., the capacity structure of the system) until a predetermined cycle time is reached. This capacity structure is used in the second stage to find out the levels of K and T_{pc} which minimize the unit penalty cost by applying the *RSM*.

4.3 The Comparisons of Worker Flexibility Policies

We first developed the simulation model of a CONWIP controlled serial line without considering the worker flexibility. In this case, workers are not allowed to move between the workstations, that is, each worker is assigned to a specific workstation. This type of production line can be thought as a kind of *CQN*. In *CQN*'s, the total number of entities in the system remains constant. The analytical results for *CQN*'s were used for validating the simulation model of the CONWIP controlled serial line.

4.3.1 The CONWIP Controlled Serial Manufacturing Line (Phase 1)

As it was given in previous chapter, the simulation model of the CONWIP line was validated using the unit penalty cost function given in Section 3.3 (see Equation (3.30)).

It must be noted that this unit penalty cost function is valid under the following conditions;

$n = M = \sum_{i=1}^{M} c_i$, for all i=1,...,M.

All μ_i's are equal.

Service times are exponentially distributed.

After setting the values of M, u_{mc}, u_{wc}, μ, and F to 4 workstations, \$0.01/minute, \$0.01/minute, 0.2 unit/minute and 0.1% per unit inventory per minute, respectively, Equation (3.30) can be simplified to;

$$TC = \frac{1.2}{K} + 0.00075 \cdot K + 0.40725 \qquad (4.1)$$

The equation above is used for validating the results for TC because the unit penalty cost values are calculated directly in the simulation runs. The simulation model of CONWIP line was executed under the following parameters; M=4 workstations, n=4 workers, μ_i's are all 0.2 unit/minute, c_i's are all 1 unit, u_{mc}=u_{wc}=\$0.01/minute, F is 0.1% per unit inventory per minute. The levels of these parameters are the same as the levels of parameters which were used in developing Equation (4.1). It must be noted that if it is not stated otherwise, the same values will be used in other experiments too.

The experiment factor is K, the number of jobs circulating in the network. The levels of this factor are set to 1, 2, 3, 4, 5, 10, 20, 30, 40, 50, 100, 200. The model

was run until 20000 jobs are completed and the warmup period was accepted as 10000 minutes since the pilot runs showed that 10000-minute-warmup period is enough to reach the steady state for all levels of K. The number of replications is set to 20 for all cases and the variance reduction technique of Common Random Numbers was applied. The comparison of the simulation results and the analytical results with respect to cycle time and the associated costs can be seen in Table 4.1. The results in Table 4.1 are averages of 20 replications. The last two columns which are obtained by using Equation (3.26) and Equation (4.1) show the analytical results for the CONWIP controlled serial manufacturing line.

Table 4.1 The Simulation Results for CONWIP Line

K	TTC	TMC	TWC	TIC	$1/TR$	TC	$1/TR_A$	TC_A
1	32084.38	15992.24	15992.24	99.91	19.990	1.604	20.000	1.608
2	20104.37	9994.73	9994.73	114.92	12.494	1.006	12.500	1.009
3	16128.05	7999.04	7999.04	129.97	9.999	0.807	10.000	0.810
4	14135.66	6995.39	6995.39	144.88	8.744	0.706	8.750	0.710
5	12950.72	6395.42	6395.42	159.88	7.994	0.648	8.000	0.651
10	10629.79	5197.58	5197.58	234.63	6.497	0.531	6.500	0.535
20	9582.00	4599.17	4599.17	383.66	5.749	0.480	5.750	0.482
30	9330.35	4399.37	4399.37	531.61	5.499	0.467	5.500	0.470
40	9280.38	4300.49	4300.49	679.40	5.376	0.463	5.375	0.467
50	9310.88	4241.68	4241.68	827.52	5.302	0.466	5.300	0.469
100	9813.44	4122.48	4122.48	1568.4	5.153	0.490	5.150	0.494
200	11067.23	4065.90	4065.90	2935.4	5.082	0.554	5.075	0.563

As it is seen in Table 4.1, the more the level of K the less the cycle time (see column eight). However, regarding the TTC (i.e., the TTC is Total Penalty Cost and it is equal to the sum of all cost terms. The unit penalty cost, TC, is calculated by dividing TTC by the number of jobs produced), we observe a different behavior in the sense that it starts increasing at higher levels of K since the total inventory penalty cost (i.e., TIC) gets higher. It must be noted that, for this special case, the values of TMC and TWC are equal since the number of workers and machines, and their unit costs (i.e., u_{mc}, uw_c) are equal. Another important point here is that the TCC was not included in the table because the worker flexibility was not considered in developing the simulation model. To compare the simulation results with the analytical results, the Mean Absolute Percent Deviation ($MAPD$) was used;

$$MAPD_{TC} = \frac{\sum_{i=1}^{12} \left| \frac{TC_i - TC_{Ai}}{TC_{Ai}} \right|}{12} \cdot 100, \quad MAPD_{1/TR} = \frac{\sum_{i=1}^{12} \left| \frac{(1/TR_i) - (1/TR_{Ai})}{(1/TR_{Ai})} \right|}{12} \cdot 100$$

The $MAPD_{TC}$ and $MAPD_{1/TR}$ are calculated as 0.64%, 0.05%, respectively. We can state that these values are small enough to accept that the simulation model is valid with respect to both cycle time and unit penalty cost.

As it was mentioned above, the *TC* starts to increase if the level of *K* is greater than a specific level. To find out this level, Equation (4.1) can be used. Let us assume that *K* is continuous in the interval $[1,\infty)$. Since Equation (4.1) is convex in the interval, the minimum point can be calculated by $\frac{\partial(TC)}{\partial K} = \frac{-1.2}{K^2} + 0.00075 = 0$, then, *K*=40 jobs in the network. This point can be seen approximately in Figure 4.1, which depicts simulation results for *TC* vs. *K*.

As it is depicted in Figure 4.1, the *TC* keeps decreasing until the level of *K* is 40, from that point the values of TC starts increasing. This relation shows the trade-off between the inventory carrying cost and the production costs.

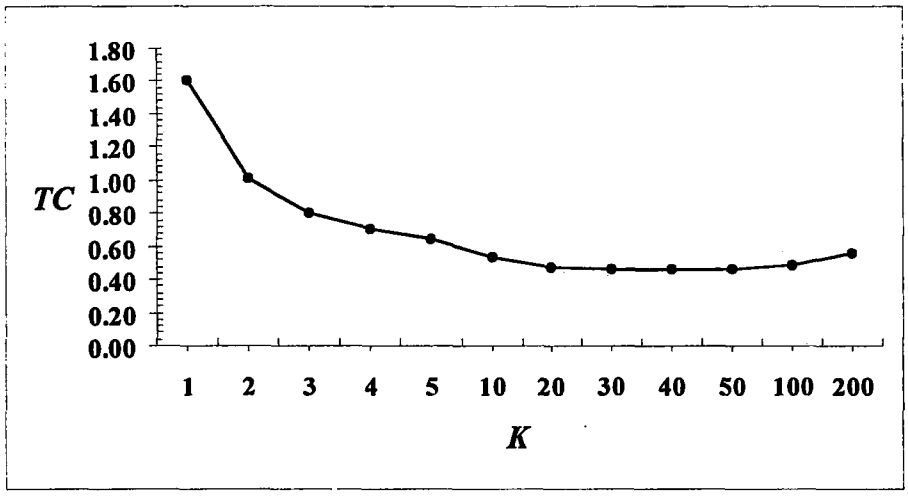

Figure 4.1 The Simulation Results for *TC* vs. *K*

Once we observed that the simulation model of the CONWIP controlled serial manufacturing line is valid, we embedded the alternative flexibility policies into this model. In next section, the alternative policies were evaluated under the optimality condition of the *PR* policy.

4.3.2 Satisfaction of the Optimality Condition of *PR* Policy (Phase 2)

As it was mentioned before, the optimality conditions of the *PR* are $c_i = n$ for all $i = 1,\ldots,M$, and $K \geq n$ for a given n. Under these conditions, and additionally, providing that $K = n$, the *PR* policy is optimum with respect to *TR* and *E(N)*. It gives maximum throughput rate with minimum average *WIP* level of K by providing the equality $E(n_i) = c_i \cdot \rho_i$ given in Section 3.2.1. The proposed flexibility policy (*PC*) was compared with the following alternative policies;

***PR*:** Since there are no space and machine limitations, each worker picks a job from the beginning of the line, then, performs the related operations at each workstation. Having completed all operations in a job, the workers goes back to the beginning of the line and picks another job and so on. The workers move with jobs.

***WW_0*:** The workers are eligible for transfer to another workstation only when there is not any job in their workstations (i.e., nq_i=0). If they are eligible for transfers, they go to the workstation which has the longest queue.

***WW_K*:** The workers are eligible for transfer to another workstation each time they finish the jobs they are assigned for (i.e., $nq_i \leq K$). When they are eligible for transfers, they go to the workstation which has the longest queue.

***WW_q*:** This policy is the general case of both WW_0 and WW_K. It was only used in the comparative studies under the *PR* optimality condition. The workers are eligible for transfer to another workstation only when there is maximum of "q" jobs in their workstations (i.e., $nq_i \leq q$). If they are eligible for transfers, they go to the workstation which has the longest queue.

Since the *PR* policy does not require any decision to take, such as when and where to go, it was not penalized by u_{cc}. The other policies require making a decision on transfer time and transfer place. Therefore, we used u_{cc} in evaluating other policies.

In this phase, two experiments are carried out. The first experiment is conducted for comparing the *PR*, WW_0, WW_K and *PC* under the optimality conditions. The second experiment is conducted for evaluating the effects of "*q*", which is the parameter of the policy WW_q, on performance of the system. The following section gives the results of the first experiment.

4.3.2.1 The Comparisons of the *PR*, WW_0, WW_K, and *PC* Policies

The simulation model of the CONWIP line under flexible workers, and optimality conditions was executed under the following parameters; M=4 workstations, n=4 workers (these four workers are at the first workstation at the beginning of the simulation runs), μ_i's are all 0.2 unit/minute, c_i's are all 4 units, u_{mc}=u_{wc}=\$0.01/minute, u_{cc}= \$0.01/control, F=0.1% per unit inventory per minute, and T_{pc}=1 minute. It must be noted that the T_{pc} is the control parameter of the proposed policy. For the experiments in first group of studies, its value was selected arbitrarily. In second group of experiments, we changed the value of T_{pc}.

There are two factors in this experiment. These are the flexibility policies, and *K*. The factor levels for policy type are *PR*, WW_0, WW_K, and *PC*. The factor levels of *K* are 5, 10, 20, 30, 40, 50, 100, 200. Since all the policies but the *PR* use the queue length information to change the place of the workers, we should have at least n+1 jobs in the network. However, the *PR* can be evaluated for the values of *K* providing the condition that *K* is less than *n*. Therefore, in experimenting with the *PR* policy, we included the levels of *K* levels of 1, 2, 3, 4. The experiment consists of 4*8=32 design points. The model was run until 20000 jobs are produced and the warmup period was considered as 10000 minutes. The number of replications was 20 for all design points and the variance reduction technique of Common Random Numbers was applied. The performance measures are the mean cycle time, the unit penalty

cost and mean utilizations. Table 4.2. and Table 4.3. give the simulation results for the *PR* policy. The last column in Table 4.2 represents the analytical results when all c_i's are equal to one and n is equal to four.

Table 4.2 The Results for the *PR* Policy with respect to Cost Function and Mean Cycle Time

K	*1/TR*	*TTC*	*TC*	*TMC*	*TWC*	*TIC*	$1/TR_A$
1	19.990	80233.51	4.012	63968.95	15992.24	272.32	20.000
2	9.999	40262.26	2.012	31997.79	7999.45	265.03	12.500
3	6.667	26924.70	1.346	21333.71	5333.43	257.56	10.000
4	5.000	20248.95	1.013	15999.16	3999.79	250.00	8.750
5	5.000	20248.95	1.013	15999.16	3999.79	250.00	8.000
10	5.000	20248.95	1.013	15999.16	3999.79	250.00	6.500
20	5.000	20248.95	1.013	15999.16	3999.79	250.00	5.750
30	5.000	20248.95	1.013	15999.16	3999.79	250.00	5.500
40	5.000	20248.95	1.013	15999.16	3999.79	250.00	5.375
50	5.000	20248.95	1.013	15999.16	3999.79	250.00	5.300
100	5.000	20248.95	1.013	15999.16	3999.79	250.00	5.150
200	5.000	20248.95	1.013	15999.16	3999.79	250.00	5.075

As it is seen in Table 4.2, the increase in the level of K decreases the cycle time of the system. However, at the level of K=4, the theoretical minimum cycle time for given n is reached. Theoretical minimum cycle time for this system can be calculated as follows;

$$\frac{1}{TR_{\min}} = \frac{\sum_{i=1}^{M} \frac{1}{\mu_i}}{n} \tag{4.2}$$

Therefore, allowing more than four jobs in the system increases the average flow time of a job, and these additional jobs wait just in q_1. Since q_1 is not included in the system as we mentioned in Section 4.1, the *WIP* consists of only the jobs being processed, so the number of jobs in the system is equal to 4 when $K \geq 4$. This result is consistent with the values of *TIC*. It must be noted that increasing the level of K more than 4 does not have any effect on *TIC*, and the cycle time is at its theoretical minimum level. Another point to not is that although the cycle time for K=1 are the

same in both cases represented in Table 4.1 and Table 4.2, the *TIC*'s are not equal (i.e., *TIC* in Table 4.1 is \$99.91, *TIC* in Table 4.2 is \$272.32). Since we assign production costs (i.e., direct labor and overheads. See Equation (3.10), (3.11)) to the inventory, the value of the unit inventory in the workstation with c_i=1 is not equal to the value of the unit inventory in the workstation with c_i=4. Therefore, the difference in *TIC*'s represents the difference in the investments made for generating inventory when the throughput rates are equal in both systems. In addition to these, the values of the *TC* are greater than the results for *TC* in the previous experiment at all levels of *K* because we have 12 additional machines for n=4. At this point, the questions "Is it possible to reach maximum throughput rate with less *TC*?", or "Is it possible to use an acceptable throughput rate in terms of less *TC*?" can be asked. This issue is investigated in following sections. Moreover, with respect to throughput rate, this flexibility policy produces better results than analytical results. It also compensates the uncertainties due to the stochastic processing time even if the line is long term balanced (i.e., equal processing time distributions). It must be noted that in the analytical results the time-weighted average worker capacities of the workstations is equal to one due to the reason that the movement of the workers between the workstations is not allowed. Therefore, the effective capacities of the workstations are not changed during the process. Another important performance measure is the worker utilization. As it was given before, the policy which provides the fully utilized workers should be the best policy. Table 4.3 gives the results for utilization.

Table 4.3 The Utilization Results for the *PR* Policy

K	$E(n_1)$	$c_1 \cdot \rho_1$	$E(n_2)$	$c_2 \cdot \rho_2$	$E(n_3)$	$c_3 \cdot \rho_3$	$E(n_4)$	$c_4 \cdot \rho_4$
1	3.2503	0.2503	0.2501	0.2501	0.2500	0.2500	0.2497	0.2497
2	2.5005	0.5005	0.5000	0.5000	0.5000	0.5000	0.4995	0.4995
3	1.7507	0.7507	0.7502	0.7502	0.7500	0.7500	0.7491	0.7491
4	1.0006	1.0006	1.0002	1.0002	1.0001	1.0001	0.9991	0.9991
5	1.0006	1.0006	1.0002	1.0002	1.0001	1.0001	0.9991	0.9991
10	1.0006	1.0006	1.0002	1.0002	1.0001	1.0001	0.9991	0.9991
20	1.0006	1.0006	1.0002	1.0002	1.0001	1.0001	0.9991	0.9991
30	1.0006	1.0006	1.0002	1.0002	1.0001	1.0001	0.9991	0.9991
40	1.0006	1.0006	1.0002	1.0002	1.0001	1.0001	0.9991	0.9991
50	1.0006	1.0006	1.0002	1.0002	1.0001	1.0001	0.9991	0.9991
100	1.0006	1.0006	1.0002	1.0002	1.0001	1.0001	0.9991	0.9991
200	1.0006	1.0006	1.0002	1.0002	1.0001	1.0001	0.9991	0.9991

The most important result in this table is that the time-weighted average worker capacities generated at workstations are equal to the expected number of jobs at that workstation for $K \geq 4$. So, it can be said that the equality of $E(n_i) = c_i \cdot \rho_i$ is held for the condition $K \geq 4$. Therefore, the workers do not wait for jobs, and the workers are not in blocked state. Except for with respect to the average flow time of a job, the same results are obtained when $K \geq 4$. Table 4.4 and Table 4.5 give the experimental results for the WW_0 policy.

Table 4.4 The Results for the WW_0 Policy with respect to Cost Function and Mean Cycle Time

K	*1/TR*	*TTC*	*TC*	*TMC*	*TWC*	*TCC*	*TIC*	$1/TR_A$
5	5.000	20920.62	1.046	15999.23	3999.81	599.57	322.01	8.000
10	5.000	20746.02	1.038	15999.56	3999.89	251.40	495.17	6.500
20	5.000	21027.46	1.053	15999.99	4000.00	152.60	874.87	5.750
30	4.999	21353.40	1.068	15997.49	3999.37	105.61	1250.92	5.500
40	4.998	21697.07	1.084	15992.85	3998.21	79.82	1626.19	5.375
50	4.997	22049.70	1.103	15988.69	3997.17	63.94	1999.91	5.300
100	4.995	23886.83	1.194	15985.08	3996.27	31.98	3873.50	5.150
200	4.997	27630.03	1.382	15990.53	3997.63	16.00	7625.87	5.075

In general, under the condition that c_i's are all n, this policy works in the same way as the *PR* does. However, the only difference is that a group of workers processes the jobs in batch size of K and the workers are not tied to jobs when they are moving. They work together until the queue is empty. Therefore, the level of *WIP* is greater than the level of *WIP* when the *PR* is implemented. As it is seen in Table 4.4., the *TCC* column was added due to the reason that the WW_0 policy requires to make a decision on when and where to go. The increase in the level of K causes a decrease in *TCC* because the number of cases that the queues are empty decreases. In addition to these, the cycle time of the system is also at the level of theoretical minimum for all levels of K. However, unlike the *PR* policy, the cost of *WIP* increases due to high levels of *WIP*. Table 4.5 represents the results for utilization.

Table 4.5 The Utilization Results for the WW_0 Policy

K	$E(n_1)$	$c_1 \cdot \rho_1$	$E(n_2)$	$c_2 \cdot \rho_2$	$E(n_3)$	$c_3 \cdot \rho_3$	$E(n_4)$	$c_4 \cdot \rho_4$
5	1.0006	1.0006	1.0002	1.0002	1.0001	1.0001	0.9991	0.9991
10	1.0006	1.0006	1.0003	1.0003	1.0001	1.0001	0.9990	0.9990
20	1.0003	1.0003	1.0002	1.0002	1.0004	1.0004	0.9991	0.9991
30	1.0002	1.0002	1.0002	1.0002	1.0004	1.0004	0.9992	0.9992
40	1.0001	1.0001	1.0001	1.0001	1.0004	1.0004	0.9995	0.9995
50	1.0000	1.0000	1.0000	1.0000	1.0003	1.0003	0.9997	0.9997
100	0.9993	0.9993	1.0001	1.0001	1.0006	1.0006	1.0000	1.0000
200	0.9990	0.9990	1.0007	1.0007	1.0006	1.0006	0.9997	0.9997

As it can be seen in table, the WW_0 policy is as effective as the *PR* policy with respect to utilization of workers. It provides fully utilized workers. Table 4.6 and Table 4.7 give the experimental results for the WW_K policy.

Table 4.6 The Results for the WW_K Policy with respect to Cost Function and Mean Cycle Time

K	$1/TR$	TTC	TC	TMC	TWC	TCC	TIC	$1/TR_A$
5	5.000	21132.57	1.057	15999.23	3999.81	799.98	333.54	8.000
10	5.000	21364.55	1.069	15999.16	3999.79	799.98	565.63	6.500
20	5.000	21741.85	1.087	15999.25	3999.81	799.99	942.80	5.750
30	5.000	22234.72	1.113	15999.00	3999.75	799.99	1435.98	5.500
40	5.000	22612.30	1.132	15999.24	3999.81	800.00	1813.25	5.375
50	5.000	22612.30	1.132	15999.24	3999.81	800.00	1813.25	5.300
100	5.000	25223.07	1.262	15998.95	3999.74	799.98	4424.40	5.150
200	5.000	29575.49	1.481	15999.24	3999.81	799.98	8776.46	5.075

As it can be seen in Table 4.6, the values of *TMC* and *TWC* are approximately the same for all three policies because these costs are linearly dependent on the cycle time. If the cycle times and unit costs of machines and workers are the same, the *TMC*'s and *TWC*'s of all the policies will be the same. In this case, the only difference will be in *TCC* and *TIC*. It must be noted that the *TCC*'s of WW_0 and WW_K are not zero but the *TCC* of *PR* is. Under the WW_K policy, since the state of the system is controlled each time that a worker finishes his job, the number of decision epochs reaches its maximum, so the values of *TCC* in this table are the maximum control cost. Additionally, the difference between the *TIC* costs of WW_0 and WW_K policy for the same level of K is caused by the difference in *TCC*'s. Since the *TCC* is

thought as overhead, the increase in overhead cost increases the value of unit inventory at a workstation. As a summary, the *TC* of the policy WW_K is greater than the *TC* of policy the WW_0 for all levels of *K* when the cycle times are equal, and the reason is that the *TCC* of WW_K is always greater than the *TCC* of WW_0 (the increase in *TCC* also causes the increase in *TIC* for the same level of *K*). Table 4.7 represents the utilizations of workers under the WW_K policy.

Table 4.7 The Utilization Results for the WW_K Policy

K	$E(n_1)$	$c_1 \cdot \rho_1$	$E(n_2)$	$c_2 \cdot \rho_2$	$E(n_3)$	$c_3 \cdot \rho_3$	$E(n_4)$	$c_4 \cdot \rho_4$
5	1.0006	1.0006	1.0002	1.0002	1.0001	1.0001	0.9991	0.9991
10	1.0006	1.0006	1.0002	1.0002	1.0001	1.0001	0.9991	0.9991
20	1.0006	1.0006	1.0002	1.0002	1.0001	1.0001	0.9991	0.9991
30	1.0006	1.0006	1.0002	1.0002	1.0001	1.0001	0.9991	0.9991
40	1.0006	1.0006	1.0003	1.0003	1.0001	1.0001	0.9990	0.9990
50	1.0007	1.0007	1.0002	1.0002	1.0000	1.0000	0.9991	0.9991
100	1.0008	1.0008	1.0002	1.0002	1.0000	1.0000	0.9991	0.9991
200	1.0007	1.0007	1.0003	1.0003	1.0000	1.0000	0.9991	0.9991

As it is seen in Table 4.7, the WW_K policy satisfies the condition $E(n_i) = c_i \cdot \rho_i$ for all $i = 1,\ldots,M$. It means that the WW_K is as effective as the *PR* and WW_0 policy with respect to utilization of workers. It provides the fully utilized workers. Table 4.8 and Table 4.9 give the experimental results for the proposed policy which we denote as the *PC*. It must be noted that the value of T_{pc} is selected arbitrarily and is set to the level of one minute.

Table 4.8 The Results for the *PC* Policy with respect to Cost Function and Mean Cycle Time

K	*1/TR*	*TTC*	*TC*	*TMC*	*TWC*	*TCC*	*TIC*	$1/TR_A$
5	5.435	23190.20	1.160	17391.09	4347.77	1086.94	364.40	8.000
10	5.210	22469.99	1.123	16671.74	4167.93	1041.98	588.34	6.500
20	5.128	22580.79	1.130	16409.50	4102.37	1025.59	1043.32	5.750
30	5.088	22864.56	1.143	16281.85	4070.46	1017.61	1494.63	5.500
40	5.065	23217.83	1.162	16207.83	4051.96	1012.99	1945.05	5.375
50	5.050	23602.33	1.181	16159.07	4039.77	1009.94	2393.55	5.300
100	5.023	25739.51	1.287	16073.44	4018.36	1004.59	4643.12	5.150
200	5.012	30194.45	1.511	16036.50	4009.13	1002.28	9146.54	5.075

Since the system state is controlled periodically (i.e., T_{pc}=1 minute in this experiment, and the decision epochs do not occur on time when a job is finished at a workstation) under the proposed policy, its performance with respect to the mean cycle time is not as good as those of other policies. However, we expect that decreasing T_{pc} will decrease the cycle time. If the value T_{pc} is selected as very short time unit (i.e., the control of system state occurs at time $t+\Delta T_{pc}$), then the system will be controlled at almost in real time (e.g., we simulated our model with T_{pc}=0.1 minute and K=5 units for 10 replications, and we reached 5.036 minutes of cycle time at TC of \$1.554 per unit). As it is seen in Table 4.8., since the arbitrarily selected valu of T_{pc} causes a great number of control points, the TCC of the PC policy is estimated as much greater than the TCC of the WW_K policy, so the unit penalty cost of the PC is greater than the TC's of all the policies for all levels of K. Table 4.9 shows the utilizations.

Table 4.9 The Utilization Results for the *PC* Policy

K	$E(n_1)$	$c_1 \cdot \rho_1$	$E(n_2)$	$c_2 \cdot \rho_2$	$E(n_3)$	$c_3 \cdot \rho_3$	$E(n_4)$	$c_4 \cdot \rho_4$
5	0.9946	0.9208	1.0043	0.9200	0.9812	0.9198	1.0198	0.9190
10	1.0000	0.9604	1.0001	0.9598	0.9999	0.9597	1.0000	0.9588
20	1.0006	0.9757	1.0004	0.9753	1.0000	0.9751	0.9991	0.9740
30	1.0004	0.9830	1.0002	0.9827	1.0003	0.9828	0.9991	0.9817
40	1.0002	0.9869	0.9999	0.9867	1.0004	0.9871	0.9995	0.9861
50	1.0000	0.9893	0.9999	0.9892	1.0003	0.9896	0.9999	0.9891
100	0.9997	0.9943	1.0001	0.9947	1.0004	0.9951	0.9998	0.9944
200	0.9994	0.9967	1.0006	0.9979	1.0004	0.9978	0.9996	0.9969

In Table 4.9, it is shown that the PC policy does not satisfy the condition $E(n_i) = c_i \cdot \rho_i$ for all $i = 1,\ldots,M$. As we mentioned before, $E(n_i)$ can take the values between $c_i \geq E(n_i) \geq c_i \cdot \rho_i$. If $E(n_i)$ is greater than $c_i \cdot \rho_i$, it means that the policy is not effective in transferring the workers to the workstations where the workers are needed.

The comparison of policies under the optimality conditions of the PR policy consists of two sets of experiments. The first set of experiments was comparison of the PR, WW_0, WW_K, PC policies with respect to the mean cycle time and the unit

penalty cost. The second set of experiments involved evaluating the effects of "q", which is the parameter of policy WW_q, on performance of the policy with respect to the percentage of K. The following figure depicts the performances of the policies with respect to the unit penalty cost.

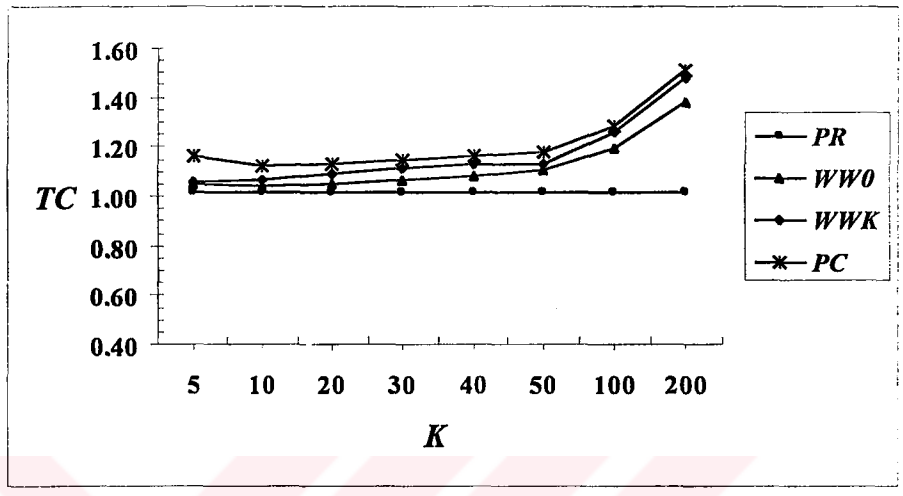

Figure 4.2 The Comparisons of Unit Penalty Costs

The results obtained from these comparative studies with respect to both the mean cycle time and the unit penalty cost can be summarized as follows;

- As it was stated in Oyen's study (Oyen et al., 2001), the *PR* is optimum with respect to the mean cycle time under the *WIP* level of $K=n$. It also gives the minimum unit penalty cost in comparison to other policies. The *TC* of the *PR* policy is independent of the level of K when $K \geq n$.
- The WW_0 is optimum (i.e., at least it is optimum for our data set. Theoretical aspects were not considered, however, we believe that it works well under the optimality conditions of *PR*) with respect to the mean cycle time under the *WIP* level of $K=n+1$. However, the *TC* of WW_0 policy is greater than the *TC* of *PR* policy. This difference is caused by the increase in *TCC* (the *TCC* of *PR* policy was assumed zero) and in *TIC*.
- The WW_K satisfies the minimum theoretical cycle time for the system having $K=n+1$, so it produces an optimum solution. However, it leads to a greater *TCC* in

comparison to WW_0 policy. This difference results in greater *TIC* for the same level of *K*.

- The *PC* is the worst policy in comparison to the *PR*, WW_0 and WW_K, with respect to both cycle time and unit penalty cost. It does not guarantee the minimum cycle time. However, decreasing T_{pc} effects the cycle time significantly. Moreover, this policy causes the greatest *TC*.

We can summarize the results of these experimental studies as follows;

"The WW_0 and WW_K, which are the most widely used heuristics in DRC job shops, are well suited to the serial lines having CONWIP control under the saturated demand. Under the optimality condition of PR, they result in the same cycle time performance in comparison to the PR policy with one more additional job (i.e., K=n+1) in the system".

"The PC policy is not found to be as effective as the WW_0 and WW_K policies under the optimality condition of the PR. However, it has a potential to produce near minimum cycle times at K=n+1 provided that the value of T_{pc} is set to the small values".

The next section gives the evaluation of the WW_q policy under the optimality condition of the *PR*.

4.3.2.2 The Evaluation of WW_q

As it was mentioned earlier, the WW_q represents the general case of the WW_0, and WW_K. The WW_q is general because when the value of q is set to zero, the resultant policy will be the WW_0, when the value of q is set to K, the resultant policy will be the WW_K. So, the WW_0 and WW_K are two extreme cases of WW_q (i.e., decentralized vs. centralized). This concept was stated first in (Fryer, 1974). In WW_q policy, the workers are eligible for transfer to another workstation only when there is maximum of "q" jobs in their workstations (i.e., $nq_i \leq q$). If the workers are eligible for transfer,

they go to the workstation which has the longest queue. In this experiment, the effects of the level of q on performance of the policy was investigated. The levels of q are defined as percentages of the level of K.

The experiment plan consists of two factors; These are q and K. The factor levels of q were set to 0% (i.e.,WW_0), 20%, 40%, 60%, 80%, and 100% (i.e.,WW_K) of K. The factor levels of K were selected as 5, 10, 20, and 30 jobs. So, the levels of q when K=5, K=10, K=20, and K=30 are set to 0, 1, 2, 3, 4, and 5 units; 0, 2, 4, 6, 8, and 10 units; 0, 4, 8, 12, 16, and 20 units; and 0, 6, 12, 18, 24, and 30 units, respectively. The total number of design points is 4*6=24. All other model parameters such as the run length, the warmup period, and the number of replications are the same as in previous experiments. Table 4.10 represents the results of this experiment.

Table 4.10 The Results for the WW_q Policy with respect to *TC* and 1/*TR*

K	*% of K*	*Q*	*1/TR*	*TTC*	*TC*	*TMC*	*TWC*	*TCC*	*TIC*
5	0%	0	5.000	20920.62	1.046	15999.23	3999.81	599.57	322.01
	20%	1	5.000	21132.57	1.057	15999.23	3999.81	799.98	333.54
	40%	2	5.000	21132.57	1.057	15999.23	3999.81	799.98	333.54
	60%	3	5.000	21132.57	1.057	15999.23	3999.81	799.98	333.54
	80%	4	5.000	21132.57	1.057	15999.23	3999.81	799.98	333.54
	100%	5	5.000	21132.57	1.057	15999.23	3999.81	799.98	333.54
10	0%	0	5.000	20746.02	1.038	15999.56	3999.89	251.40	495.17
	20%	2	5.000	21309.85	1.066	15999.32	3999.83	749.93	560.78
	40%	4	5.000	21364.55	1.069	15999.16	3999.79	799.98	565.63
	60%	6	5.000	21364.55	1.069	15999.16	3999.79	799.98	565.63
	80%	8	5.000	21364.55	1.069	15999.16	3999.79	799.98	565.63
	100%	10	5.000	21364.55	1.069	15999.16	3999.79	799.98	565.63
20	0%	0	5.000	21027.46	1.053	15999.99	4000.00	152.60	874.87
	20%	4	5.000	21567.50	1.079	15999.20	3999.80	650.09	918.41
	40%	8	5.000	21741.85	1.087	15999.25	3999.81	799.99	942.80
	60%	12	5.000	21741.85	1.087	15999.25	3999.81	799.99	942.80
	80%	16	5.000	21741.85	1.087	15999.25	3999.81	799.99	942.80
	100%	20	5.000	21741.85	1.087	15999.25	3999.81	799.99	942.80
30	0%	0	4.999	21353.40	1.068	15997.49	3999.37	105.61	1250.92
	20%	6	5.000	21785.69	1.090	15999.09	3999.77	453.01	1333.82
	40%	12	5.000	22234.58	1.113	15998.91	3999.73	799.98	1435.96
	60%	18	5.000	22234.72	1.113	15999.00	3999.75	799.99	1435.98
	80%	24	5.000	22234.72	1.113	15999.00	3999.75	799.99	1435.98
	100%	30	5.000	22234.72	1.113	15999.00	3999.75	799.99	1435.98

As it can be seen from Table 4.10, the theoretical minimum cycle time of 5 minutes was reached for all the levels of q, so the *TMC* and *TWC* are approximately the same for all design points. It must be noted that the *TC*'s for a given level of K are the same if the level of q is greater than a specific level. This specific level is 40% of the level of K. Setting q to the higher levels which are greater than this level has no effect on *TC* due to the reason that the *TCC* and *TIC* are not sensitive to changes in the level of q (i.e., for the levels of q greater than 40% of K). However, as it can be seen in the table, the *TCC* is an increasing function of the level of q in the interval 0% and 40% of K. In this interval, it is possible to obtain theoretical minimum cycle time leading to lower *TC*. Figure 4.3 represents the relation between the levels of q as a percentage of K, and *TCC*.

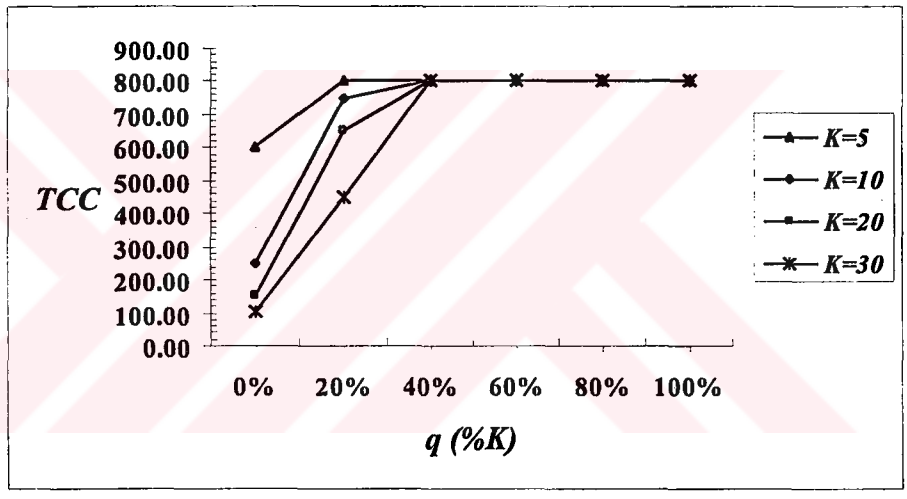

Figure 4.3 The Relation Between *TCC* and q

As it is depicted in Figure 4.3, increasing the level of K decreases the *TCC* within the interval 0% and 40% of K. We also analyzed the relation between q and the sum of *TCC* and *TIC*. The results are depicted in Figure 4.4.

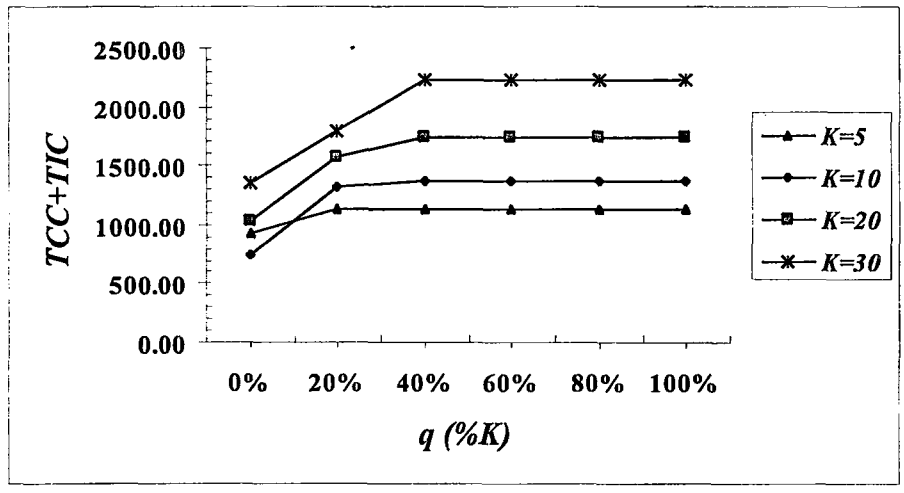

Figure 4.4 The Relation Between *TCC+TIC* and *q*

Since the differences in *TC*'s are originated from just the differences in *TCC* and *TIC*, we included only the sum of the *TCC* and *TIC* in this Figure. The levels of *q* which are greater than the specific level of *q* (i.e., 40% of *K*) have no effect on *TCC+TIC* for a given *K*. Based on this experiment we can conclude that;

- We have a control on *TCC* within the interval 0% and 40% of *K*. Morever, providing that cycle times are equal, decreasing the level of *q* decreases the unit penalty cost within this interval.

This property is important when the worker transfer times are significant. Since the number of worker transfers is a function of the control points over time, it is proper to use the level of *q* which is less than the 40% of *K* in this production environment. As last word, we can state that,

"Any policy whose performance with respect to TCC is less than the performance of the WW_0 has potential to lead the lower values of TC, provided that the cycle times and K's are equal".

In Section 4.3.2, the flexibility policies were compared under the optimality condition of *PR*. The most critical point in implementing the *PR* policy is the assumption of $c_i = n$ for all $i = 1,\ldots,M$. This is not a realistic assumption in today's

competitive firms. In most cases, it is impossible to place all the workers in a same workstation. Moreover, it will not be economical to provide a sufficient number of equipment or machine for all the workers at each workstation. In reality, the ample capacity is not an applicable concept. For example, let us assume that there is a CONWIP controlled serial manufacturing line consisting of 10 workstations each having one machine, and we are operating this line with 10 workers (i.e. no flexibility). The optimality condition requires having 10 parallel machines at each workstation, and setting the value of K equal or more than 10 to reach theoretical minimum cycle time for this line. Indeed, it will not be practical to supply 90 additional machines or equipment to reach the theoretical minimum cycle time. At this point, two important questions should be asked;

- Is there any flexibility policy which provides the theoretical minimum cycle time for a given system at an additional investment cost which is less than what implementing the *PR* policy for?
- Is it optimum to operate the system under the theoretical cycle time with respect to the *TC*?

In following section, the policies were compared under the violation of the optimality condition of the *PR* policy and also the answers for these questions are given.

4.3.3 Dissatisfaction of the Optimality Condition of the *PR* Policy (Phase 3)

In this experiment, the values of c_i's are set to less than n. This implies that all the workers can not be assigned to a single workstation which is having more than one but less than n machines. Therefore, there may be worker blockages which may decrease the performances of the policies. In previous experiments, the capacities of the workstations were not constraints with respect to n, so there were not any worker blockages. However, in this experiment, the workers may be blocked due to the lack of a machine. Therefore, the *PR*, WW_0, and WW_K policies require some modifications to handle this blockage problem. We call the modified version of the *PR* as

Generalized Pick and Run (*GPR*). The *GPR* works as follows; If the worker does not come across with a blockage, the worker takes the job from the beginning of the line, processes the job at each workstation, and finishes the job at the last workstation, then he takes a new job from the beginning of the line (i.e., same as *PR*). However, if there is a blockage (i.e. each machine at the downstream workstation is staffed by one worker), the worker sends the job to the input buffer of downstream workstation, then, takes a new job from the input buffer of his current workstation, if there is one. Otherwise, he remains idle at his workstation. The modified versions of the WW_0 and WW_K also work in the same way. If the worker can not be transferred to the workstation which has the longest queue due to the reason that the workstation with longest queue is fully staffed, the worker stays at his current workstation, so no worker transfer is made as in the study of Treleven (Treleven & Elvers,1985). It must be noted that the proposed policy, *PC*, was designed to overcome these blockages.

One of the important assumptions that we made while modifying the policies is that the workers are distributed to the workstations with respect to the workstation capacities at the beginning of the simulation run. For example, if the line has five workers with five workstations, and c_i=2, for all i=1,...,5, the simulation model begins with $n_1(0)$=2, $n_2(0)$=2, $n_3(0)$=1, $n_4(0)$=0, and $n_5(0)$=0.

This experiment aims at showing the effects of c_i's (i.e., when they are not set to the level of n) on performance of the policies. The line is perfectly balanced as in the case of previous experiment with respect to mean service times, and c_i's.

We have three factors in this experiment; The type of the flexibility policy, K, and c_i's. The flexibility policies are *GPR*, WW_0, WW_K, and *PC*. The levels of K were set to the levels of 5, 10, 20, 30, 40, 50, 100, and 200 jobs. The levels of c_i's are set to two, and three machines. There are total of 4*8*2=64 design points in this experiment. All of the system and simulation model parameters are the same as in previous experiments. Table 4.11 represents the results of this experiment when c_i's are set to two machines.

Table 4.11 The Simulation Results for c_i=2

Policy	c_i=2, for all i=1,...,M								
	K	*1/TR*	*TTC*	*TC*	*TMC*	*TWC*	*TCC*	*TIC*	$1/TR_A$
GPR	5	7.241	17601.01	0.881	11585.42	5792.71	0.00	222.89	8.000
	10	6.171	15147.99	0.757	9873.95	4936.98	0.00	337.06	6.500
	20	5.597	13994.62	0.700	8955.49	4477.75	0.00	561.38	5.750
	30	5.401	13743.49	0.688	8641.37	4320.68	0.00	781.45	5.500
	40	5.300	13736.03	0.688	8479.41	4239.70	0.00	1016.93	5.375
	50	5.241	13792.72	0.689	8385.44	4192.72	0.00	1214.55	5.300
	100	5.130	14612.78	0.732	8208.24	4104.12	0.00	2300.41	5.150
	200	5.081	16261.35	0.813	8130.41	4065.21	0.00	4065.74	5.075
WW_0	5	5.898	14861.25	0.743	9436.96	4718.48	487.79	218.02	8.000
	10	6.192	15408.67	0.770	9906.87	4953.44	193.53	354.84	6.500
	20	6.562	16519.84	0.826	10498.83	5249.42	104.87	666.72	5.750
	30	6.641	16988.56	0.850	10625.56	5312.78	74.90	975.33	5.500
	40	6.685	17391.29	0.870	10696.07	5348.03	59.11	1288.07	5.375
	50	6.706	17741.00	0.888	10730.04	5365.02	49.68	1596.26	5.300
	100	6.771	19411.17	0.972	10833.05	5416.53	29.25	3132.34	5.150
	200	6.819	22637.02	1.133	10909.55	5454.77	17.60	6255.10	5.075
WW_K	5	5.787	14920.36	0.747	9258.54	4629.27	799.99	232.56	8.000
	10	5.375	14079.58	0.704	8599.62	4299.81	799.98	380.17	6.500
	20	5.190	13902.72	0.696	8304.09	4152.05	799.98	646.60	5.750
	30	5.137	14080.04	0.703	8219.29	4109.64	799.97	951.14	5.500
	40	5.116	14301.44	0.715	8185.40	4092.70	800.03	1223.31	5.375
	50	5.093	14546.19	0.728	8149.13	4074.56	799.98	1522.52	5.300
	100	5.048	15849.67	0.792	8076.23	4038.12	799.95	2935.37	5.150
	200	5.024	18642.28	0.933	8038.95	4019.48	799.98	5783.87	5.075
PC	5	5.548	14670.50	0.733	8876.54	4438.27	1109.57	246.13	8.000
	10	5.160	13803.20	0.691	8255.47	4127.74	1031.93	388.06	6.500
	20	5.066	13859.13	0.693	8106.15	4053.07	1013.27	686.64	5.750
	30	5.041	14093.34	0.704	8066.05	4033.02	1008.26	986.01	5.500
	40	5.029	14360.41	0.719	8046.54	4023.27	1005.81	1284.80	5.375
	50	5.023	14645.28	0.733	8037.24	4018.62	1004.65	1584.77	5.300
	100	5.012	16116.99	0.806	8019.86	4009.93	1002.48	3084.72	5.150
	200	5.012	19179.00	0.960	8019.79	4009.90	1002.47	6146.85	5.075

As it is seen in Table 4.11, increasing the level of K causes decreases in cycle times in all policies, but in WW_0. Furthermore, the performance of WW_0 is worse than that of the analytical case (i.e., c_i's are all one) where $K \geq 20$. So, the WW_0 policy can be said that it does not work under the conditions given. Since the system studied is a closed one, waiting for an empty input buffer to decide on where to send a worker can not be a flexibility alternative when all c_i's are two. However, the performances of other policies are not affected by the worker blockages as in the case of WW_0. The

comparative cycle times and unit penalty costs are represented in Figure 4.5, and Figure 4.6, respectively.

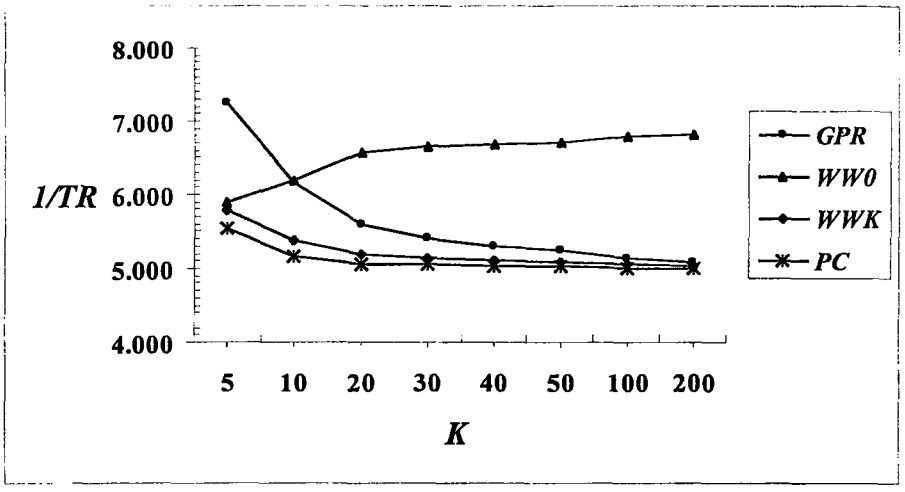

Figure 4.5 The Comparisons of Cycle Times (c_i=2)

As it is depicted in Figure 4.5, in comparison to the other policies, the cycle time performance of the *PC* (i.e., 1/*TR* in the table) is superior at all levels of *K*. The performance of WW_K is also better than those of the *GPR* and WW_0 with respect to the cycle time. However, it must be noted that the differences in performances between the *PC* and other policies are greater when the level of *K* is relatively low. Increasing the level of *K* compensates these differences between the policies. Figure 4.6 represents the differences of policies based on the unit penalty cost.

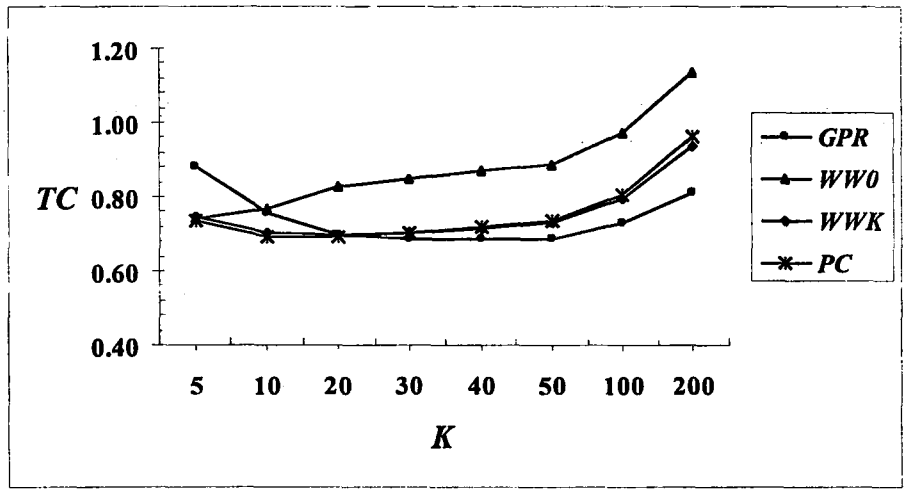

Figure 4.6 The Comparisons of Unit Penalty Costs (c_i=2)

It can be seen in Figure 4.6 that the *PC* and WW_K have approximately the same unit penalty costs at all levels of K. However, it must be noted that the *PC* produces low cycle times at the same cost. Since the *TCC* was not included in *GPR*, it generates the lowest *TC* for the levels of K approximately greater than 25 units. Table 4.12 shows the results of this experiment when the c_i's are set to three parallel machines.

Table 4.12 The Simulation Results for c_i=3

Policy	c_i=3, for all i=1,...,M								
	K	*1/TR*	*TTC*	*TC*	*TMC*	*TWC*	*TCC*	*TIC*	$1/TR_A$
GPR	5	6.084	19729.99	0.986	14600.38	4866.79	0.00	262.81	8.000
	10	5.591	18303.92	0.916	13418.74	4472.91	0.00	412.27	6.500
	20	5.309	18648.15	0.934	12741.58	4247.19	0.00	859.36	5.750
	30	5.201	17627.08	0.881	12482.89	4160.96	0.00	983.23	5.500
	40	5.155	17771.66	0.889	12370.77	4123.59	0.00	1277.30	5.375
	50	5.136	17906.31	0.896	12325.63	4108.54	0.00	1472.15	5.300
	100	5.099	18296.09	0.914	12236.27	4078.76	0.00	1981.06	5.150
	200	5.098	18324.02	0.915	12235.14	4078.38	0.00	2010.50	5.075
WW_0	5	5.085	17124.63	0.856	12203.10	4067.70	587.78	266.05	8.000
	10	5.400	17931.39	0.897	12959.22	4319.74	228.89	423.54	6.500
	20	5.795	19461.48	0.973	13907.91	4635.97	121.90	795.70	5.750
	30	5.899	20123.73	1.007	14158.24	4719.41	82.75	1163.33	5.500
	40	5.946	20618.96	1.032	14270.55	4756.85	62.61	1528.95	5.375
	50	5.980	21083.37	1.054	14352.83	4784.27	50.39	1895.88	5.300
	100	6.042	23086.48	1.155	14501.87	4833.96	25.43	3725.23	5.150
	200	6.045	26718.68	1.336	14509.19	4836.40	12.73	7360.35	5.075
WW_K	5	5.076	17322.06	0.867	12183.16	4061.05	799.99	277.86	8.000
	10	5.021	17336.02	0.867	12050.73	4016.91	799.97	468.41	6.500
	20	5.002	17588.38	0.880	12004.54	4001.52	799.98	782.33	5.750
	30	5.000	17988.19	0.900	12000.60	4000.20	799.99	1187.40	5.500
	40	5.000	18301.97	0.915	11999.63	3999.88	800.00	1502.46	5.375
	50	5.000	18706.95	0.935	11999.45	3999.82	799.99	1907.69	5.300
	100	5.000	20461.91	1.023	11998.97	3999.66	799.97	3663.32	5.150
	200	5.000	24064.60	1.203	11999.53	3999.84	799.99	7265.24	5.075
PC	5	5.432	18773.43	0.940	13037.55	4345.85	1086.46	303.57	8.000
	10	5.177	18087.56	0.904	12423.73	4141.24	1035.31	487.28	6.500
	20	5.089	18166.14	0.909	12214.04	4071.35	1017.84	862.92	5.750
	30	5.059	18438.57	0.923	12141.28	4047.09	1011.77	1238.43	5.500
	40	5.044	18764.64	0.939	12106.33	4035.44	1008.86	1614.01	5.375
	50	5.035	19108.03	0.955	12084.43	4028.14	1007.03	1988.42	5.300
	100	5.016	20917.02	1.046	12038.11	4012.70	1003.17	3863.03	5.150
	200	5.008	24642.13	1.232	12018.65	4006.22	1001.55	7615.71	5.075

The most important result in this table is that the WW_K policy works well under the condition that all c_i's are equal to three. Furthermore, it produces near optimum cycle times when the level of K is greater than 30 units (i.e., we called it near optimum due to the reason that we did not prove it theoretically). The WW_0 shows the same behaviour as in the previous case. Another important result here is that the PC does not work as well as the WW_K under these conditions. Figure 4.7 compares the policies with respect to the mean cycle time when all c_i's are equal to three.

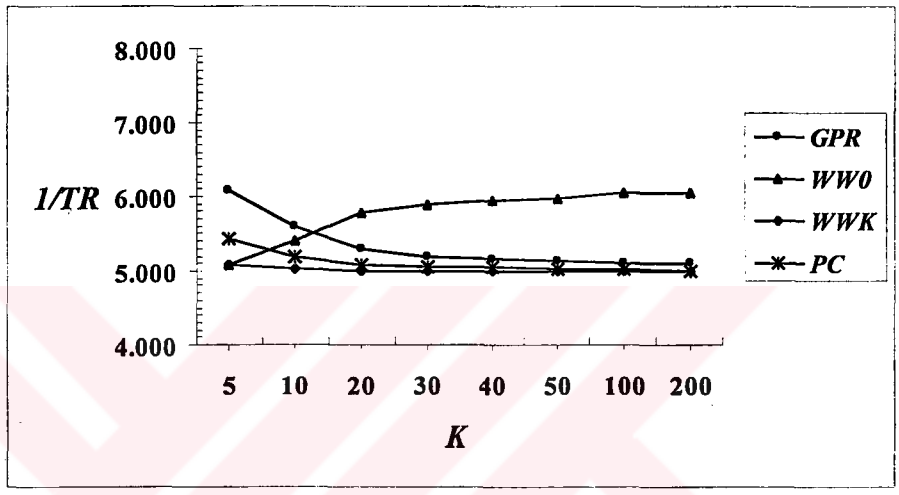

Figure 4.7 The Comparisons of Cycle Times (c_i=3)

As it was depicted in Figure 4.7, the WW_K can reach the theoretical minimum cycle time when the level of K is greater than 30. It seems that the PC is not a good candidate under the conditions provided. However, it must be noted that the PC has another control parameter called T_{pc}. We know that decreasing T_{pc} causes decrease in cycle time at higher level of unit penalty costs. The results with respect to TC's can be summarized in Figure 4.8.

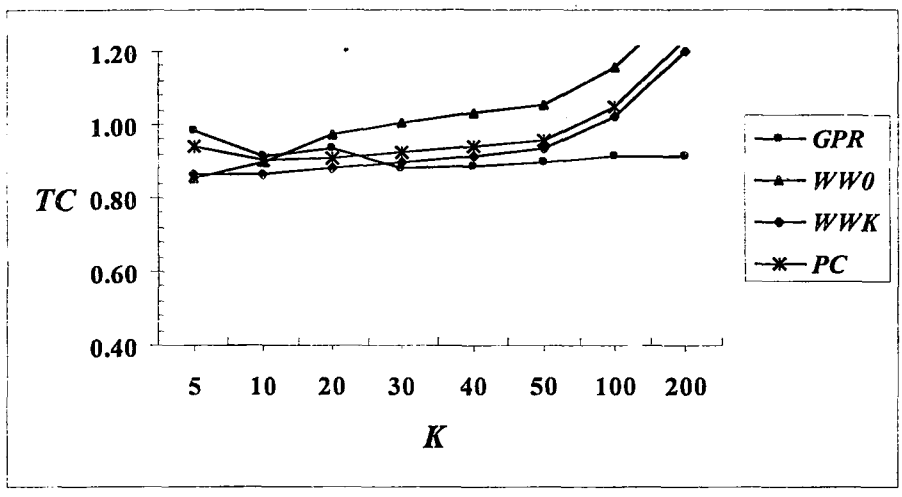

Figure 4.8 The Comparisons of Unit Penalty Costs (c_i=3)

As it is depicted in Figure 4.8, the WW_K has the lowest levels of *TC* values for low levels of *K*. The lowest *TC*'s are obtained by the *GPR* policy when the level of *K* is more than 30 units due to the reason that the *GPR* does not include the *TCC*.

The results of this experiment can be summarized as follows;

- The WW_0 does not work under the violation of *PR* optimality condition even if the line is balanced with respect to both mean service times and c_i's. Although it works well when all c_i's are equal to *n*, the worker blockages increase the cycle time of the system operated under the WW_0 policy even if the level of *K* is increased.
- The difference in the performance of different policies becomes less visible when the level of *K* is increased. Likewise, the same behavior is observed when the number of machines is increased.
- If the c_i's are far away from the optimality condition of *PR* policy, the *PC* works better than the WW_K policy. If the c_i's are near to the optimality condition of the *PR*, the WW_K works better than the *PC*.
- The WW_K has capability to produce near optimum cycle times at lower level of additional investment in comparison to the *GPR* for a given level of *n* and *K*.

In summary;

"Even though the number of machines added and the level of K are kept low, the PC policy has more potential to produce low cycle times than the GPR and WW_K in CONWIP controlled serial manufacturing lines under the saturated demand".

In Sections 4.3.2 and 4.3.3, the policies were compared in perfectly balanced CONWIP controlled lines. To be able to generalize the insights gained from these experiments, the policies need to be compared under different operating conditions other than the conditions given so far. For instance, a balanced serial line can be turned into an imbalanced line by adding machines, or changing the mean service times. In following two sections, this issue is analyzed.

4.3.4 Unbalanced Line due to Differences in Workstation Capacities (Phase 4)

As it was mentioned in Section 4.2, this experiment was carried out under the violation of the optimality condition of the *PR* policy. Moreover, the manufacturing line was not perfectly balanced with respect to the c_i's. In this experiment, the equal mean service times but unequal values of c_i's were used.

There were two factors in this experiment; the policy type, and c_i's. The policy type has four levels as in the previous experiments; *GPR*, WW_0, WW_K, and *PC*. The levels of c_i's were determined as follows; Since we were interested in unbalancing the line by adding more capacity, first, we selected three balanced lines. These are c_i=1, c_i=2, and c_i=3, for all i=1,...,4. These lines were denoted as *1111*, *2222*, and *3333*. Each number represents the number of parallel machines at WS_i. For example, the first number on the left represents the number of parallel machines at WS_1, etc. The factor level of *4444* was not included in this experiment due to the reason that the *PR* policy produces theoretical minimum cycle times with the minimum level of *K*, so, adding one more capacity to the case *4444* does not have any practical meaning. Second, one more machine was added to these balanced lines. We also considered the issue of where to place the additional machine to see whether the

policies are sensitive to the location of additional machine or not. As a result, the following factor levels for c_i's were obtained; *1112*, *1121*, *1211*, *2111*, *2223*, *2232*, *2322*, *3222*, *3334*, *3343*, *3433*, *4333*. We have total of 4*12=48 design points in this experiment. The level of K was fixed to 20 units because the two competing policies, the WW_K and the *PC*, have approximately similar results with respect to the mean cycle time and unit penalty cost at this level. All of the other system and simulation model parameters are the same as in previous experiments. Table 4.13 shows the simulation results for the mean cycle time and the unit penalty cost.

Table 4.13 The Effects of Additional Capacity on 1/*TR* and *TC*

Policies / c_i	*GPR*		WW_0		WW_K		*PC*	
	1/TR	*TC*	*1/TR*	*TC*	*1/TR*	*TC*	*1/TR*	*TC*
1112	7.366	0.679	6.176	0.580	5.996	0.603	5.671	0.589
1121	7.365	0.688	6.153	0.581	5.981	0.604	5.668	0.596
1211	7.361	0.696	6.169	0.588	5.981	0.610	5.668	0.601
2111	7.365	0.703	6.157	0.591	5.984	0.616	5.666	0.606
2223	6.977	0.932	6.592	0.895	5.155	0.742	5.071	0.743
2232	6.983	0.940	6.584	0.895	5.140	0.741	5.071	0.748
2322	6.980	0.951	6.603	0.900	5.144	0.743	5.071	0.747
3222	6.966	0.959	6.596	0.906	5.152	0.747	5.070	0.749
3334	6.667	1.147	5.752	1.023	5.001	0.930	5.097	0.959
3343	6.667	1.147	5.736	1.021	5.001	0.931	5.096	0.964
3433	6.667	1.144	5.750	1.026	5.001	0.932	5.097	0.963
4333	6.666	1.207	5.739	1.028	5.001	0.933	5.096	0.964

As it is seen from Table 4.13, the place of additional capacity has no effect on the cycle time of the system. However, the value of *TC* is affected by the place of the additional capacity due to the relations given in Equation (3.16) and Equation (3.19). The *PC* is the most effective flexibility policy because it uses the additional capacity even if the number of additions is not much. For example, if one machine is added to the base case (i.e., all c_i's are one), the *PC* policy has potential to decrease mean cycle time, however, the other policies can not exploit the advantage of this additional capacity. The cycle time for the base case is equal to 5.750 minutes when the level of K is 20 units. The *PC* produces cycle time of 5.668 minutes at average. So, it can be said that each addition is utilized by the *PC* policy. It must be also noted that the differences observed between the *PC* and other policies was greater if the K and T_{pc} were set to the lower values (i.e., their current values are 20 units and 1

minute, respectively). Figure 4.9 represents the relation between the additional capacities and cycle time.

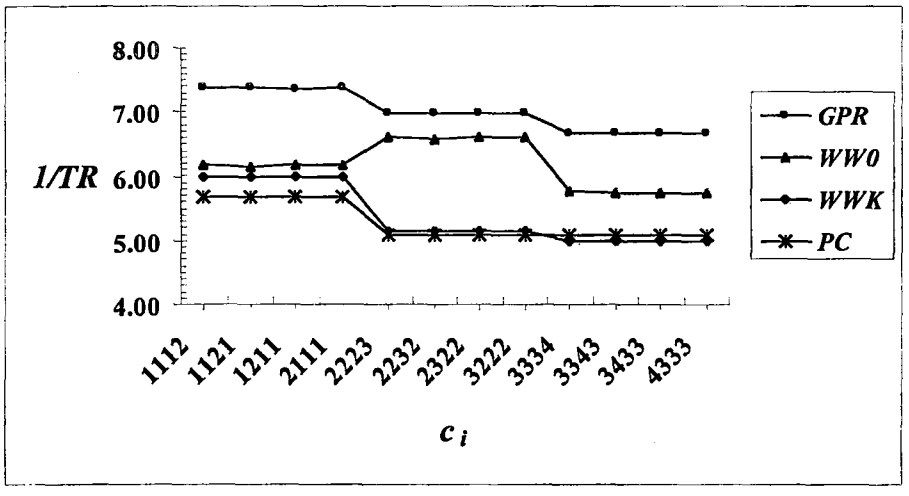

Figure 4.9 The Comparisons of Policies under Different Capacities

As it is depicted in Figure 4.9, the *PC* provides lower cycle times until one of the capacities is at least equal to four (e.g., *3334*). The difference between the performance of *PC* and WW_K decreases until this point, then, the WW_K starts to produce lower cycle times. This result is consistent with the 3rd property given in Section 4.3.3. The *GPR* and WW_0 do not work well under these conditions.

The results of this experiment can be summarized as follows;

- If the serial line is balanced and if there is a need for additional capacity, to add capacities starting from the end of the line to the beginning of the line leads to a decrease in *TC*.
- There is a specific capacity structure that the WW_K policy starts to produce lower cycle times than the *PC* policy. Furthermore, the differences observed between the performance of these two policies can be increased by setting K and T_{pc} to the lower values.

In summary,

"Sequentially adding one machine to the base case (i.e., 1111) starting from the end of the line to the beginning of the line may be a good starting point to optimize the system operated under the PC policy".

In following section, the effects of changing the mean service times on performance of the policies are analyzed.

4.3.5 Unbalanced Line due to Differences in Mean Service Times (Phase 5)

In this experiment, the values of c_i's are the same, they were set to two. However, the mean service times are not equal. The aim of this experiment was to show how the policies balance these differences in mean service times. Therefore, the mean service times are selected as one of the factors of the experiment. The levels of this factor were selected as follows; in the base case (i.e., mean service times are all 5 minutes) the total workload of the system was 20 minutes, and the resulted theoretical minimum cycle time was 5 minutes with respect to Equation (4.2). This total workload was changed by adding and subtracting 1, 2, 3, and 4 minutes to total workload of 20 minutes. So, the total workload pairs are 19-21, 18-22, 17-23, and 16-24 minutes. These total workloads were obtained by changing the mean service times of the workstations 1, 2, 3, and 4 consecutively. For example, the total workload pair "19-21" was obtained by setting the mean service time of the first workstation to 4 and 6 minutes, respectively. We denoted these factor levels as *4555*, and *6555* (the second pair is *5355*, and *5755*, so on). So, we have total of eight levels; *4555*, *6555*, *5355*, *5755*, *5525*, *5585*, *5551*, and *5559*. The other factor is the type of the policy. The factor levels are *GPR*, WW_0, WW_K, and *PC*. The *PR* policy was also included for benchmarking. The level of K was fixed to 20 units. All of the other system and model parameters are the same as in previous experiments. The following table represents the results of this experiment.

Table 4.14 The Effects of Changing Mean Service Times on 1/*TR* and *TC*

Policies	*GPR*		WW_0		WW_K		*PC*		*PR*	
$1/\mu_i$'s	*1/TR*	*TC*	*1/TR*	*TC*	*1/TR*	*TC*	*1/TR*	*TC*	$1/TR_{min}$	*TC*
4555	5.446	0.683	6.247	0.785	4.949	0.663	4.818	0.657	4.750	0.961
6555	6.096	0.755	6.897	0.870	5.454	0.731	5.318	0.730	5.250	1.064
5355	5.417	0.674	5.945	0.745	4.715	0.632	4.571	0.621	4.500	0.910
5755	6.900	0.859	7.285	0.923	5.767	0.772	5.570	0.767	5.500	1.115
5525	5.441	0.673	5.641	0.707	4.516	0.608	4.325	0.590	4.250	0.860
5585	7.761	0.980	7.686	0.975	6.103	0.815	5.823	0.800	5.750	1.167
5551	5.509	0.682	5.285	0.667	4.348	0.589	4.078	0.557	4.000	0.809
5559	8.572	1.102	8.101	1.019	6.419	0.851	6.077	0.837	6.000	1.218

As it was seen in Table 4.14, the *PC* policy produces lower cycle times in comparison to other policies. Furthermore, the results produced by the *PC* policy deviate from the minimum theoretical cycle time about 1.481%. It must be noted that the *PC* has minimum *TC*'s. The Figure 4.10 depicts the relation between the cycle time and the mean service times.

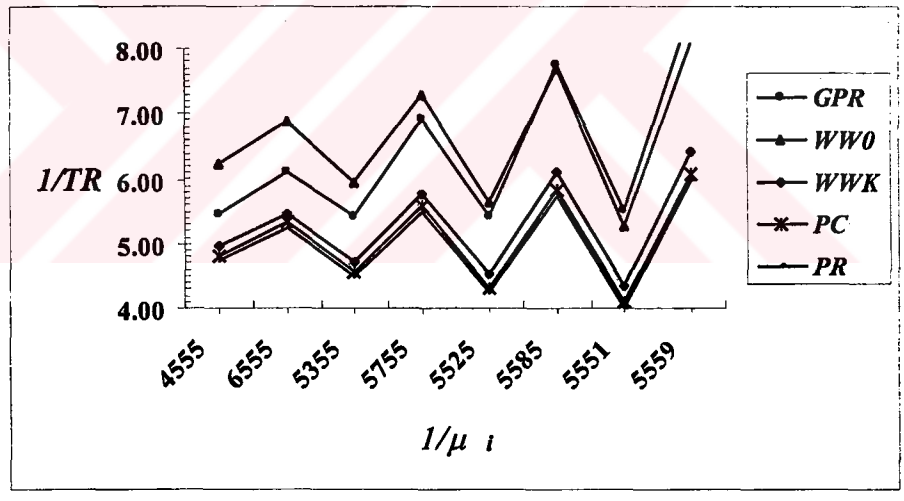

Figure 4.10 The Comparisons of Policies under Different Bottleneck Cases

As it is seen in the figure, the *PC* policy produces similar results with the *PR* policy. While the *PR* policy reaches the minimum theoretical cycle time by providing that $K=n=4$ and $c_i=4$ for all $i=1,\ldots,4$, the *PC* policy reaches near minimum cycle times with $K=20$, $n=4$, $c_i=2$ for all $i=1,\ldots,4$. The results of this experiment can be summarized as follows;

- Under different service times, the *PC* policy produces near minimum cycle times even if the line is not balanced, and the condition $c_i=n$ for all $i=1,\ldots,4$ is not satisfied.

In summary,

"*The PC is a better alternative flexibility policy even if the CONWIP controlled serial line with saturated demand is unbalanced*".

4.3.6 Increasing the Total Number of Workers (Phase 6)

The base system used in this experiment is the same as the system in Section 4.3.3 (i.e, all c_i's are equal to two, mean services times are 5 minutes). The only difference is the level of K which was fixed to 20 units. The experiment consists of two factors; The type of the flexibility policy, and the total number of workers in the system. There are four types of policies as in the previous experiments. The *PR* policy was included for benchmarking. The total number of workers has three levels; these are 5, 6, and 7. So, we have total of 4*3=12 design points. These levels were selected so that n satisfies the condition of $n < \sum_{i=1}^{M} c_i$ (i.e., the property of a *DRC* system). Since we have total of 8 machines in the system and the base case has 4 workers, n should be less than 8 and greater than 4. All of the other system and model parameters are the same as in previous experiments. The following table represents the results. The last column in Table 4.15 was calculated by using the Equation (4.2).

Table 4.15 The Effects of Increasing the Total Number of Workers

Policies	*GPR*		*WW*$_0$		*WW*$_K$		*PC*		*PR*
n	*1/TR*	*TC*	*1/TR*	*TC*	*1/TR*	*TC*	*1/TR*	*TC*	*1/TR*$_{min}$
5	4.523	0.613	5.073	0.694	4.287	0.628	4.057	0.598	4.000
6	3.810	0.557	3.993	0.590	3.692	0.587	3.410	0.539	3.333
7	3.290	0.516	3.335	0.531	3.265	0.559	3.048	0.514	2.857

As it is seen in the table, the *PC* policy produces lower cycle times and the unit penalty costs at each level of n. Figure 4.11 represents the relation between n and

cycle time.

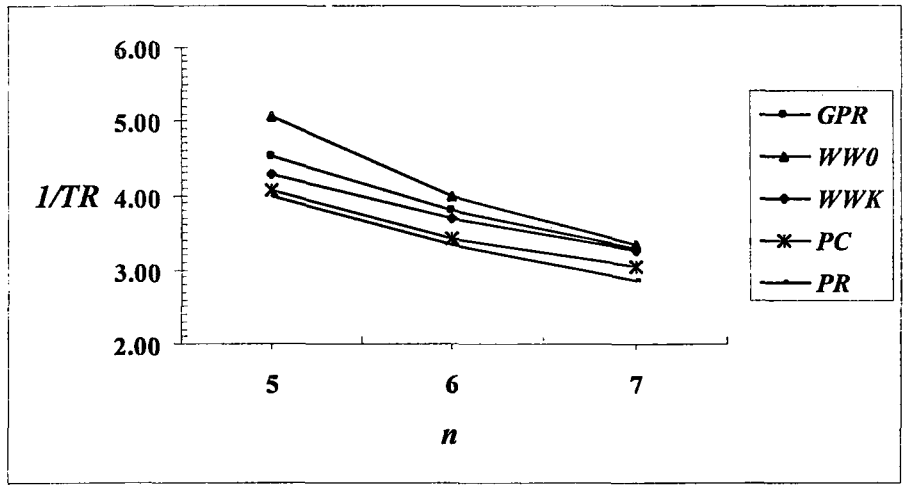

Figure 4.11 The Comparisons of Policies under Different *n*'s

As it was depicted in Figure 4.11, if this serial line is operated by the *PC* policy, increasing the total number of workers decreases the cycle time. Furthermore, these cycle times will be very close to the theoretical minimum cycle time in comparison to the other policies.

In following section, the performance of the policies are evaluated on lines having more than four workstations.

4.3.7 Increasing the Number of Workstations (Phase 7)

There are two factors in this experiment. These are the type of flexibility policy, and the number of workstations, there are four types of flexibility policies as in previous experiments, the *PR* policy was included for benchmarking. The factor levels for the number of workstations were set to 5, 6, 7, 8, 9, and 10, consecutively. The level of K was fixed to 20 units and the c_i's are all set to two. The mean service times are five minutes, and the number of workers is four. As it was mentioned earlier, the *PR* policy produces the minimum theoretical cycle time for a given n providing that all c_i's are equal to n and $K \geq n$. So, the *PR* policy was simulated by setting all c_i's to four and K to 20. The other parameters of the system and the

simulation model are the same as in previous experiments. Table 4.16 represents the results of this experiment. The cycle time of the system operated by the *PR* policy was calculated using Equation (4.2), the corresponding *TC* values were obtained from the simulation runs.

Table 4.16 The Effects of Increasing the Number of Workstations

Policies	*GPR*		WW_0		WW_K		*PC*		*PR*	
M	*1/TR*	*TC*	*1/TR*	*TC*	*1/TR*	$1/\mu_i$'s	*1/TR*	*TC*	$1/TR_{min}$	*TC*
5	7.161	1.046	8.976	1.318	6.474	1.007	6.335	1.003	6.250	1.517
6	8.784	1.468	11.402	1.914	7.753	1.374	7.602	1.367	7.500	2.122
7	10.468	1.969	13.858	2.616	9.021	1.793	8.875	1.787	8.750	2.829
8	12.205	2.552	16.317	3.423	10.311	2.263	10.152	2.263	10.000	3.638
9	13.979	3.217	18.805	4.341	11.651	2.814	11.428	2.793	11.250	4.549
10	15.796	3.968	21.257	5.354	12.952	3.414	12.705	3.379	12.500	5.560

It is clear that the most effective policy is the *PC* when the number of workstations gets larger. In comparison to the *PR* policy, furthermore, it produces near minimum cycle times at lower values of *TC*. This relation can be depicted in Figure 4.12.

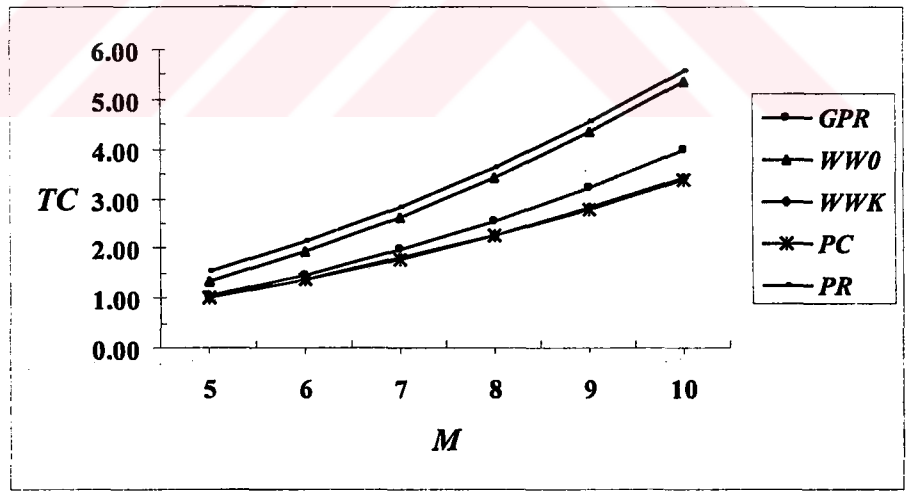

Figure 4.12 The Comparisons of Policies under Different *M*'s

As it can be easily seen in the figure, the unit penalty cost of the *PR* policy is higher in comparison to the unit penalty costs of other policies. Furthermore, increasing the number of workstations increases the differences in *TC*'s between the

policies. So, it can be said that it is possible to obtain near minimum cycle times at lower costs by applying the *PC* and WW_K policies on lines having more than four workstations. In next section, the effects of worker transfer delays on performance of the policies are evaluated.

4.3.8 Including the Worker Transfer Delays (Phase 8)

The worker transfer delay refers to any delay caused by the movement of the worker. This delay may include "the movement time to the target workstation", "the setup time to be able to start processing a job in a new workstation". It depends on the distances between the workstations, and the ability of the worker to adapt to his new job.

We assumed that the transfer delay depends on the distances between the workstations, and the time to transfer a worker from WS_i to WS_{i+1} is equal to *d* minutes for all i=1,...,*M*. For example, if the worker goes to 5th workstation from the 3rd workstation, the time may be calculated as (5-3)**d*. If the worker transfer delays are included, there is a loss in effective worker capacity of the system. If the number of movements increases, the effective number of workers decreases. Therefore, if transfer delays are not negligible in comparison to processing times, we should limit the number of movements. This limitation can be provided by setting the control periods to a larger value in the proposed policy. The WW_0 is the most widely used flexibility policy in *DRC* job shops when the transfer delays are not negligible.

This experiment consists of three factors. These are the type of policy, the level of *d*, and the number of machines at each workstation. The selected policies are the WW_0, WW_K, and *PC*. The factor levels of *d* are 0, 1, and 2 minutes (i.e., *d*=0 represents the insignificant transfer delays. The level of *d* was zero in the previous experiments). The last factor was evaluated in three levels. These are c_i=2, c_i=3, c_i=4 for all *i*=1,...,4. The combination of the second and the third factors are denoted as the numbers such as *20*, *21*, *22*, *30*, *31*, *32*, *40*, *41*, and *42*. The first digits of these numbers represent the number of machines at each workstation. The second digits

show the level of d. So, the total number of design points in this experiment was 3*9=27. The level of K was fixed to 20 jobs. The performance measure is the mean cycle time. All of the other parameters are the same as in previous experiments. Table 4.17 and Figure 4.13 show the results of this experiment.

Table 4.17 The Effect of Transfer Delay on Cycle Time

Policies / c_i vs. d	1/TR		
	WW_0	WW_K	PC
20	6.562	5.190	5.066
21	6.619	5.307	5.084
22	6.676	5.405	5.104
30	5.795	5.002	5.089
31	6.032	5.144	5.110
32	6.252	5.271	5.135
40	5.000	5.000	5.128
41	5.596	5.145	5.154
42	6.065	5.272	5.183

As it can be seen in the table, the PC is less sensitive to the worker transfer delays. Increase in the level of d causes an increase in cycle time in all policies. This relation can be seen in Figure 4.13.

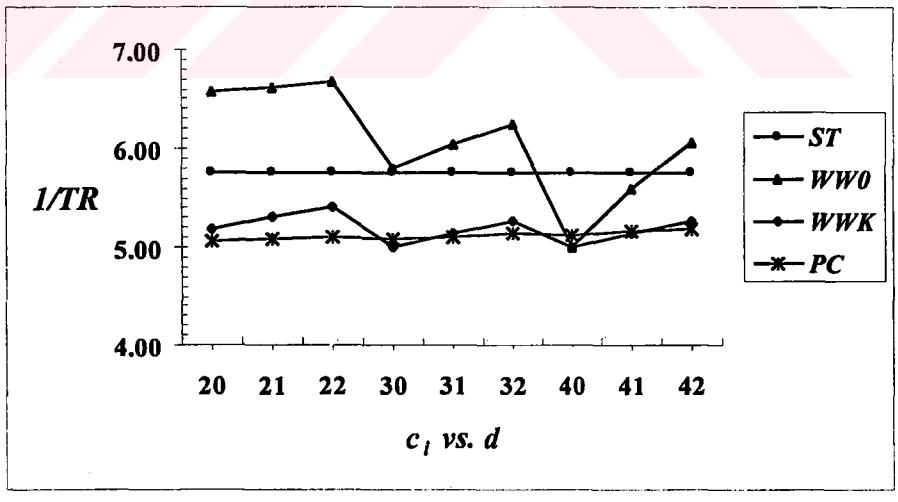

Figure 4.13 The Effect of Transfer Delay on Cycle Time

It must be noted that the analytical base case was included in Figure 4.13. The analytical base case represents the system in which all c_i's are equal to one. The cycle time of this base case can be calculated by using Equation (3.26). So, the cycle

time of this base case for K=20 units is 5.750 minutes. If the transfer delays are significant, the cycle time of any flexibility policy should be less than 5.750 minutes. This condition was satisfied by the *PC* and WW_K policies at each factor level. Moreover, the WW_0 policy can satisfy this condition when all c_i's are equal to four. However, increasing the level of d decreases the cycle time of the system operated by the WW_0 dramatically even if c_i's are all four. It is clear that the *PC* is the most robust policy against to the worker transfer delays.

In sections 4.3.6, 4.3.7, and 4.3.8, the policies were compared with respect to some critical factors such as the number of workers, the number of workstations and the existence of worker transfer delays. The results of these experiments can be summarized as follows;

- Increasing the number of the workers dedicated to the line decreases the differences observed between the performances of the policies.
- The *PC* and WW_K policies produce near minimum cycle times at lower *TC* on the lines having more workstations in comparison to the *PR*.
- The *PC* is the most robust policy against to the worker transfer delays in comparison to other policies.

In summary;

"*The PC is a better alternative flexibility policy for CONWIP controlled serial lines even if the line has more workstations or the worker transfer delays are significant. Furthermore, increasing the number of the workers does not change the effectiveness of the PC*".

In the first group of experimental studies carried out so far, the T_{pc} was selected arbitrarily and it was set to one minute. However, it was expected that the selection of the value of T_{pc} has a great effect on the performance of the *PC* policy. Therefore, in the following section, various sets of experiments are carried out to observe the effects of different levels of K and T_{pc} on system performance.

4.4 The Relation between the Levels of CONWIP (K) and the Periodic Control Length (T_{pc})

In this section, the effects of control periods on the unit penalty cost and the cycle time were analyzed. Under the *PC* policy, the state of the system is controlled periodically over time. These periods, which are denoted as T_{pc}, can be set to the different levels to apply different levels of flexibilities. If these periods are short relative to the processing times, more frequent controls can be applied to the system to change the places of the workers. If these durations are longer, the performance opportunity provided by the flexibility tends to decrease. In previous experiments, the T_{pc} was set to one minute arbitrarily. To see the effects of various control periods on performance of the policy, the following experiment was conducted;

There are two factors in this experiment; K and T_{pc}. The K has 8 levels which are set to 5, 10, 20, 30, 40, 50, 100, and 200 jobs in the system. The T_{pc} has 13 levels which are set to 0.1, 0.2, 0.5, 1, 2, 3, 4, 5, 10, 20, 30, 40, and 50 minutes. So, there are total of 8*13=104 design points in the experiment. The values of c_i's were set to two. Twenty replications were carried out at each design point. The results given in Table 4.18 and Table 4.19 are averages of these replications.

Table 4.18 The Effects of K and T_{pc} on Cycle Time

T_{pc}	K							
	5	**10**	**20**	**30**	**40**	**50**	**100**	**200**
0.1	5.254	5.036	5.009	5.004	5.003	5.002	5.000	5.003
0.2	5.284	5.049	5.015	5.009	5.006	5.005	5.001	5.010
0.5	5.378	5.089	5.033	5.021	5.015	5.011	5.003	5.010
1	5.548	5.160	5.066	5.041	5.029	5.023	5.012	5.012
2	5.923	5.311	5.136	5.087	5.064	5.049	5.023	5.013
3	6.346	5.472	5.212	5.137	5.101	5.079	5.038	5.024
4	6.817	5.650	5.292	5.189	5.141	5.110	5.054	5.031
5	7.330	5.843	5.380	5.245	5.180	5.143	5.069	5.034
10	9.613	7.194	5.899	5.569	5.417	5.325	5.158	5.078
20	12.362	9.212	7.288	6.364	5.983	5.767	5.363	5.167
30	14.260	10.129	8.454	7.282	6.593	6.232	5.587	5.270
40	16.779	10.697	8.873	8.008	7.211	6.706	5.802	5.387
50	20.156	11.113	8.898	8.443	7.764	7.183	6.016	5.494
ST	**7.994**	**6.497**	**5.749**	**5.499**	**5.376**	**5.302**	**5.153**	**5.082**

As it can be seen in Table 4.18, increasing the level of T_{pc} at each level of K increases the cycle time. It must be noted that the cycle time of the analytical base case at each level of K (i.e., all c_i's are one. We denoted it as ST at the last row of the table) is lower for some levels of T_{pc} (e.g., T_{pc}=10 for all levels of K but K=200). So, it can be said that the level of T_{pc} which is greater than 10 is not applicable to the system studied in this thesis. Another important point here is that if the T_{pc} is set to a lower value such as 0.1 minute, it is possible to obtain near minimum theoretical cycle times at lower levels of K. For example, when the T_{pc} is equal to 0.1 minute and K is equal to 10 units, the mean cycle time is estimated as 5.036 minutes. The deviation from the minimum theoretical cycle time is 0.72%. It must be noted that the PC policy reaches this cycle time with the condition that all c_i's are two and the level of K is 10 units. The cycle time of WW_K, which is the next best policy according to the previous experiments, was 5.375 minutes under these conditions (see Table 4.11). The relation between K and T_{pc} can be seen in Figure 4.14 more clearly.

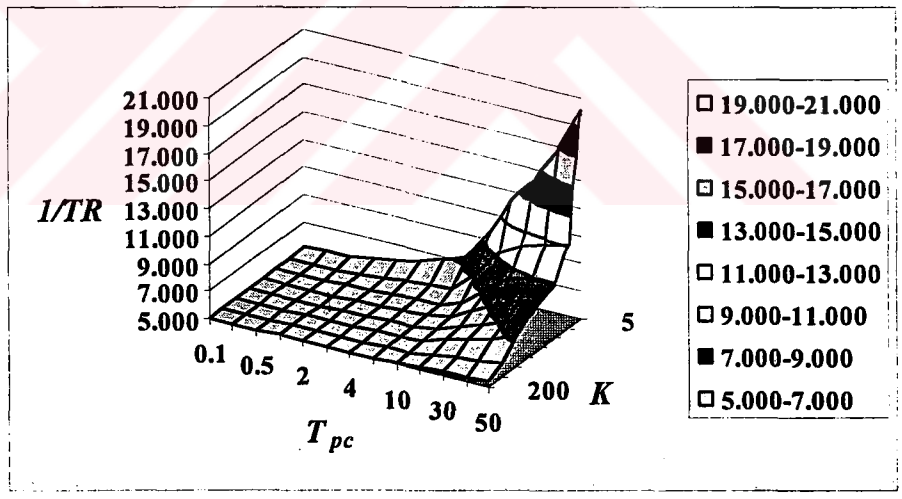

Figure 4.14 The Combined Effect of K and T_{pc} on Cycle Time

As it was stated before, if the level of K is increased, the differences observed between the performance of alternative policies become less visible. This result is also applicable to this experiment due to the reason that changing the level of T_{pc} generates different flexibility policies. So, if the level of K is relatively greater, the difference in cycle time for each level of T_{pc} is lower in comparison to low levels of

K. For example, the cycle time is not very sensitive to changes in T_{pc} when the level of K is 200 units as it is depicted in Figure 4.14. However, if K is equal to five units, the cycle time is more sensitive to the changes in T_{pc}. Table 4.19 represents the effects of T_{pc} on TC for different levels of K.

Table 4.19 The Effects of K and T_{pc} on TC

T_{pc}	K							
	5	**10**	**20**	**30**	**40**	**50**	**100**	**200**
0.1	1.194	1.171	1.213	1.261	1.308	1.358	1.601	2.092
0.2	0.922	0.896	0.921	0.950	0.979	1.009	1.158	1.464
0.5	0.769	0.737	0.749	0.764	0.783	0.801	0.892	1.464
1	0.733	0.691	0.693	0.704	0.719	0.733	0.806	0.960
2	0.751	0.681	0.673	0.679	0.690	0.701	0.763	0.893
3	0.793	0.691	0.672	0.676	0.683	0.693	0.749	0.871
4	0.847	0.710	0.679	0.678	0.683	0.691	0.745	0.864
5	0.907	0.730	0.684	0.681	0.684	0.692	0.744	0.859
10	1.176	0.890	0.743	0.715	0.709	0.710	0.747	0.853
20	1.503	1.130	0.911	0.812	0.778	0.763	0.772	0.861
30	1.729	1.240	1.053	0.924	0.855	0.822	0.798	0.874
40	2.032	1.308	1.101	1.012	0.929	0.883	0.829	0.886
50	2.437	1.356	1.103	1.064	0.995	0.939	0.860	0.902

The most important result in this table is that there is a specific combination of K and T_{pc} which minimizes the unit penalty cost at each column. For example, increasing the T_{pc} from 0.1 minute to 1 minute at K=5 causes a decrease in the unit penalty cost due to the increase in the cost of control (i.e.,TCC). However, increasing the T_{pc} from 1 minute to 50 minutes at K=5 results in an increase in the unit penalty cost due to the higher cycle times. This relation can be depicted in surface and contour plots. Figure 4.15 and Figure 4.16 represents the surface and contour plots of the data set which was given in Table 4.19.

As it was depicted in the Figure 4.14, there are two possible ways to decrease the cycle time of the system (i.e., c_1=c_2=c_3=c_4=2 machines, and n=4 workers) which is operated under the PC policy. These are to increase the level of K, and to decrease the level of T_{pc}. So, the minimum cycle time can be reached by setting the level of K to its maximum possible value and setting the level of T_{pc} to its minimum possible value. However, this combination does not result in the minimum value of TC, as it

is depicted in Figure 4.15 and Figure 4.16. It is clear that the combination which has the minimum unit penalty cost should be between 1 and 8 minutes for T_{pc} and 10 and 50 units for K. According to Table 4.19, the minimum unit penalty cost can be reached in the neighborhood of K=20 units and T_{pc}=3 minutes.

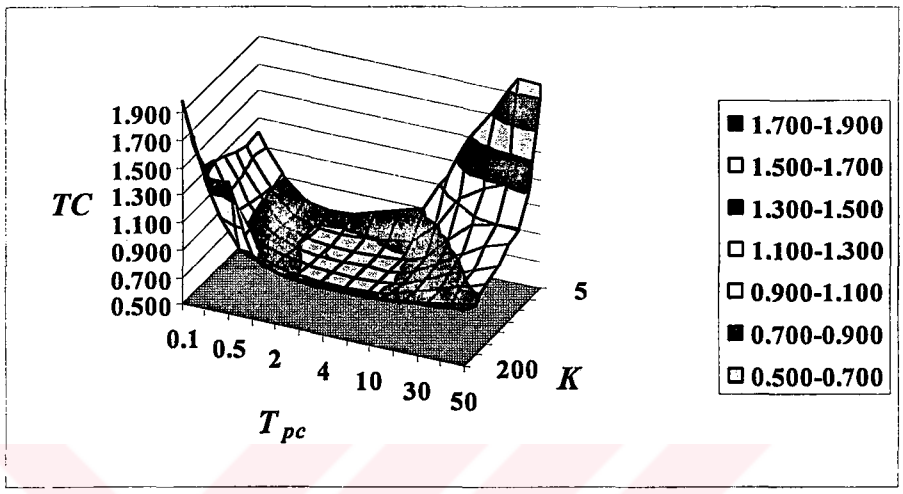

Figure 4.15 The Surface Plot of Data Set for *TC*

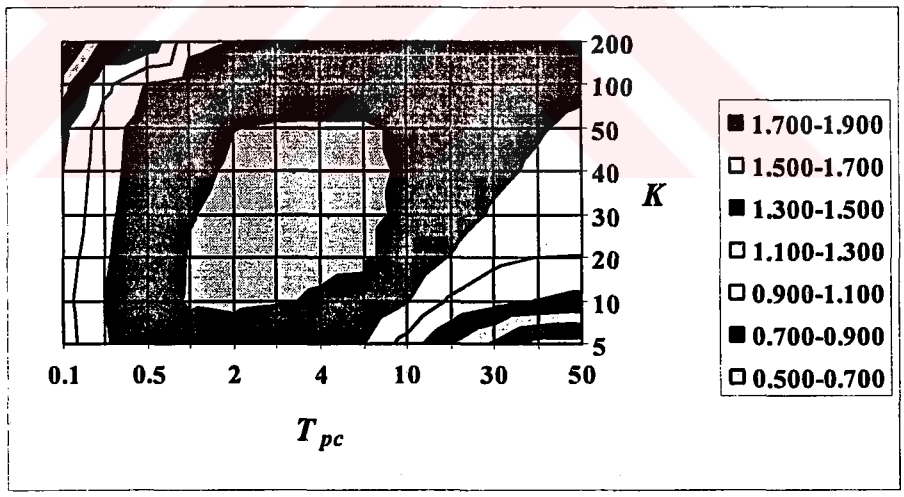

Figure 4.16 The Contour Plot of Data Set for *TC*

Based on this experiment, it can be easily said that The Response Surface Methodology can be used as a simulation optimization tool to find the optimum levels of K and T_{pc} for a given capacity structure to minimize the unit penalty cost. In following section, the application of two-stage simulation optimization methodology to the hypothetical case studied in thesis is given.

4.5 An Application of Two-Stage Simulation Optimization Methodology

As mentioned earlier, the third group of experimental studies involved the application of two-stage simulation optimization methodology to a hypothetical manufacturing system introduced in Section 4.1. The first stage of the methodology aims at finding the capacity structure which reaches the target cycle time with the minimum number of additional machines. The second stage aims at finding the optimum levels of K and T_{pc} which minimize the unit penalty cost.

The CONWIP controlled saturated line consists of four workstations each having one machine. Each machine is operated by one worker. The line is balanced with respect to the mean service times. The mean service time of each machine is exponentially distributed with the mean of 5 minutes. The physical shape and the size of the job produced in the line limits the number of jobs allowed in the system. So, the number of jobs in the system can not be greater than 25 units. Since this is a saturated line, there is no limit on the demand, so the greater the production capacity the better the performance of the line. The cycle time of the current system can be calculated by Equation (3.26).

$$TR(K) = \frac{K}{M + K - 1} \cdot \mu \qquad TR(25) = \frac{25}{4 + 25 - 1} \cdot 0.2 = 0.178571 \,\text{job/min}$$

The throughput rate of 0.178571 job/min is equal to the cycle time of 1/0.178571=5.6 minutes/job. So, the minimum possible cycle time of the current system is 5.6 minutes/job. Since the system has physical limit on the number of the jobs, it is impossible to decrease the cycle time without increasing the capacity (i.e., additional machines). The theoretical cycle time of this system is 5 minutes (Equation (4.2)). The target cycle time is 5.20 minutes/job. According to the data given above, the problem can be represented as follows;

Min TC

S.T.

$1/TR \leq 5.20$ minutes/job

$K \leq 25$ jobs

The application of two-stage simulation optimization methodology to this problem is as follows;

Step 0: Set $c_1=c_2=c_3=c_4=1$

Set $n=4$

Set $K_0=5$

Set $K=5$

Set $T_{pc}=1$

The algorithm was initialized with respect to the parameters of the sample case. Since the theoretical minimum cycle time is 5 minutes, T_{pc0} was set to 1 minute by applying 20% rule.

Step 1: $1/(\mu_1.c_1)=5/1=5$, $1/(\mu_2.c_2)=5/1=5$, $1/(\mu_3.c_3)=5/1=5$, $1/(\mu_4.c_4)=5/1=5$

Since all the $1/(\mu_i.c_i)$'s are equal to 5, the most downstream workstation should be selected as the bottleneck workstation.

Step 2: The old capacity structure : *1111*, $c_1=c_2=c_3=c_4=1$

The new capacity structure : *1112*, $c_1=c_2=c_3=1$, $c_4=2$

So, the fourth workstation is identified as a bottleneck workstation and a new machine is added to this workstation.

Step 3: The new system was simulated under the current levels of K (i.e., 5) and T_{pc} (i.e., 1). The capacity structure is *1112*. The simulation model was executed until 20000 jobs were completed. Twenty replications were carried out with the warmup period of 10000 minutes. The average cycle time and the average unit penalty cost were estimated as 7.517 minutes and \$0.762/job, respectively.

Step 4: Since the cycle time of this new system for K=5 is not lower than the target cycle time level (i.e., 7.517>5.20), the new system is simulated by setting the level of K to a maximum level possible. So, the level of K was set to 25 units.

Step 5: The new system is simulated under the new level of K (i.e., 25) and T_{pc} (i.e., 1). The capacity structure is *1112*. The average cycle time and the average unit penalty cost are estimated as 5.548 minutes and $0.581/job, respectively.

Step 6: Since the cycle time of this new system for K=25 is not lower than the target cycle time level (i.e., 5.548>5.20), it can be concluded that we can not reach the target cycle time with this new capacity structure. So, we return back to *Step 1*, and set the level of K to 5.

This process continues until the target cycle time is reached for a specific capacity structure and the level of K. As a result of applying these steps recursively, finally we obtained a cycle time which is less than the target cycle time and determined the final capacity structure as *2222*. By applying the *PC* policy for this system, we can reach the target cycle time of 5.20 minutes. Furthermore, it is possible to obtain the cycle time which deviates from the theoretical minimum cycle time by just approximately 1%. So, the first stage of the algorithm is completed. The results of the first stage of the algorithm are summarized in Table 4.20.

Table 4.20 The Summary Results of the First Stage of the Algorithm

Capacity Structure	K	T_{pc}	$1/(\mu_i.c_i)$				$1/TR$	TC	$1/TR_{lim}$
			i=1	i=2	i=3	i=4			
1111	5	-	5	5	5	5*	8.000	-	5.200
	25	-					5.600	-	5.200
1112	5	1	5	5	5*	2.5	7.517	0.762	5.200
	25	1					5.548	0.581	5.200
1122	5	1	5	5*	2.5	2.5	6.972	0.777	5.200
	25	1					5.428	0.621	5.200
1222	5	1	5*	2.5	2.5	2.5	6.335	0.772	5.200
	25	1					5.283	0.660	5.200
2222	5	1	2.5	2.5	2.5	2.5*	5.548	0.733	5.200
	25	1					5.051	0.699	5.200

As it can be seen in Table 4.20, the algorithm stops at the capacity structure *2222* due to the reason that the target cycle time is reached when K is 25. The stars in the table represent the bottleneck workstations. It must be noted that the capacity structure *1222* produces the cycle time of 5.283 minutes at lower cost, which is very close to the target level. So, we can establish the 95% confidence interval for the cycle time to see whether this interval includes the target cycle time of 5.200 minutes. The average and the standard deviation of the cycle time were estimated as 5.283, and 0.0319 minutes, respectively. The 95% confidence interval for the related data set is $5.283 \pm \frac{0.0319}{\sqrt{20}} \cdot t_{0.975,19}$. The tabulated value of $t_{0.975,19}$ is 2.093. So, the resultant confidence interval is (5.268, 5.298). Since this interval does not include the target cycle time, it can be concluded that we can not reach the target cycle time by using the capacity structure of *1222*. As a conclusion, we need at least 4 additional machines (i.e., one for each workstation) to reach the target cycle time for the system studied.

Although we reached the final capacity structure, it is not known whether it is optimum or not to operate this new system under the current combination of K=25 and T_{pc}=1. So, the second stage of the methodology (i.e., *Step 7*) involves the use of *RSM* to search the optimum levels of these parameters.

Step 7: The problem we dealt with has two decision variables (i.e., K and T_{pc}). The initial experiment region was selected arbitrarily as (10,20) units of K and (20,40) minutes of T_{pc}. In order to compare the factor effects by relative importance, the factor levels were standardized to -1 (low value) and +1 (high value) as follows;

$$X_1 = (K-(20+10)/2) / ((20-10)/2) = (K-15)/5,$$

$$X_2 = (T_{pc}-(20+40)/2) / ((40-20)/2) = (T_{pc}-30)/10$$

The 2^2 full factorial design with center point was used to explore the response surface within this initial experiment region. Table 4.21 shows the design of the first experiment with both coded and natural variables.

Table 4.21 The Initial Design for Fitting The First Order Model

Design Point	K	T_{pc}	X_1	X_2
1	20	40	+1	+1
2	20	20	+1	-1
3	10	40	-1	+1
4	10	20	-1	-1
5	15	30	0	0

We carried out 10 replications at each design point and fit the following first-order model to the simulation results; $TC = \beta_0 + \beta_1 X_1 + \beta_2 X_2$. Table 4.22 represents the simulation results for *TC*.

Table 4.22 The Simulation Results for *TC*

Replication No.	$TC(X_1,X_2)$				
	(+1,+1)	(+1,-1)	(-1,+1)	(-1,-1)	(0,0)
1	1.11	0.92	1.33	1.13	1.11
2	1.11	0.91	1.31	1.12	1.11
3	1.11	0.91	1.30	1.14	1.11
4	1.10	0.91	1.32	1.12	1.11
5	1.08	0.91	1.29	1.12	1.11
6	1.10	0.91	1.33	1.13	1.11
7	1.09	0.91	1.31	1.13	1.12
8	1.12	0.91	1.30	1.13	1.12
9	1.11	0.91	1.29	1.13	1.11
10	1.09	0.91	1.31	1.13	1.09

When we compare the results in this table with the results in Table 4.19, we can state that we are remote from the optimum levels of K and T_{pc} because there are other combinations of K and T_{pc} which produce lower values of *TC* as it is seen in Table 4.19. We used MINITAB R13.31 to estimate the following first-order polynomial model; $TC = 1.112 - 0.106X_1 + 0.093X_2$. Table 4.23 represents the results of ANOVA.

As it can be seen in Table 4.23, the regression coefficients are significant at α=0.05. The overall *LOF* including both the interaction and pure quadratic effects is insignificant at α=0.05. The R^2 and the R^2_{adj} are 99.43% and 99.40%. Table 24. represents the *t*-tests on all of the coefficients to show the significances of the individual regression coefficients.

Table 4.23 ANOVA for the First Order Model

Source of Variation		Sum of Squares	Degrees of Freedom	Mean Squares	*F*	*P*
Regression		0.795400	2	0.397700	4063.457	0.000
Residual		0.004600	47	9.7872E-5		
LOF	Interaction	0.000250	1	0.000250	2.616	0.113
	Quadratic	0.000050	1	0.000050	0.523	0.473
	Pure Error	0.004300	45	9.5555E-5		
Total		0.800000	49			

Table 4.24 The Test on the Individual Regression Coefficients

Predictor	Coefficients	Standard Error	*t*-ratio	*P*
β_0	1.11200	0.001399	794.804	0.000
β_1	-0.10600	0.001564	-67.765	0.000
β_2	0.09300	0.001564	59.454	0.000

As it can be seen in Table 4.24, the individual regression coefficients are significant at α=0.05. So, this first-order model can be used to apply the steepest descent method to improve the performance of the system with respect to *TC*. The steepest descent method is a procedure for moving sequentially along the path in the direction of the maximum decrease in response. To move away from the center point along the path of steepest descent, we would move 0.106 units in the X_1 direction for every -0.093 units in the X_2 direction. It must be noted that the path of steepest descent is the opposite sign of the gradient vector of the first-order model. The gradient vector of the model is found by taking the derivation of *TC* with subject to X_1 and X_2. The resultant gradient vector for our first-order model is $\nabla TC = [-0.106, 0.093]$. The opposite signs will be used for steepest descent method. So, the path of steepest descent has the same direction with the vector $-\nabla TC = [0.106, -0.093]$. The path of steepest descent passes through the center point and has a slope $-0.093/0.106 = -0.877$. Any combination of X_1 and X_2 that satisfies the slope given above can be used to move on the same path. However, we prefer to select the step size of 1 unit of *K* in natural form. Since the one unit of change in X_1 is equal to the 5 units of change in *K* according to the initial experiment, the change of X_1 should be 1/5=0.2 if one unit of change in *K* is required. So, the change in X_2 should be $0.2 \times (-0.877) = -0.1754 \approx -0.175$ unit. The step sizes for X_1 and X_2 can be

denoted as ΔX_1 and ΔX_2, respectively. As a conclusion, we will move away from the center point of the initial design by ΔX_1=0.2 and ΔX_2=-0.175. For example, since the center point of the initial design is (0,0), the first point in the path of steepest descent will be $(0+0.2,0-0.175)=(0.2,-0.175)$. The new point in natural variables of K and T_{pc} will be $(15+1,30-10\times 0.175)=(16,28.25)$. Following, we simulate the model for K=16 units and T_{pc}=28.25 minutes. We expect that the TC at this new point will be lower than TC at the center point of the initial design. This process will continue until when there is no further improvement on TC. We carried out 10 replications at each point and applied the variance reduction technique of Common Random Numbers. The simulation model was executed until 20000 jobs were completed. The warmup period was 10000 minutes. Table 4.25 represents the points on steepest descent path and their related TC values. The cycle time of the system at each point was also included in the table to show the changes in the cycle time along the path of steepest descent.

Table 4.25 The Steepest Descent Method

No.	New Point	X_1	X_2	K	T_{pc}	TC	$1/TR$
	Center	0	0	15	30	1.11	8.998
	Δ	0.2	-0.175	1	-1.75	-	-
1	Center +Δ	0.200	-0.175	16	28.25	1.09	8.792
2	Center +2Δ	0.400	-0.350	17	26.50	1.07	8.616
3	Center +3Δ	0.600	-0.525	18	24.75	1.02	8.225
4	Center +4Δ	0.800	-0.700	19	23.00	0.98	7.851
5	Center +5Δ	1.000	-0.875	20	21.25	0.93	7.460
6	Center +6Δ	1.200	-1.050	21	19.50	0.89	7.082
7	Center +7Δ	1.400	-1.225	22	17.75	0.85	6.719
8	Center +8Δ	1.600	-1.400	23	16.00	0.81	6.415
9	Center +9Δ	1.800	-1.575	24	14.25	0.78	6.152
10	Center +10Δ	2.000	-1.750	25	12.50	0.75	5.932
11	Center +11Δ	2.200	-1.925	26	10.75	0.73	5.731
12	Center +12Δ	2.400	-2.100	27	9.00	0.71	5.559
13	Center +13Δ	2.600	-2.275	28	7.25	0.69	5.408
14	Center +14Δ	2.800	-2.450	29	5.50	0.68	5.280
15	Center +15Δ	3.000	-2.625	30	3.75	0.68	5.174
16	Center +16Δ	3.200	-2.800	31	2.00	0.68	5.081
17	Center +17Δ	3.400	-2.975	32	0.25	0.89	5.007

The results in Table 4.25 suggest that the minimum point was reached at 16th iteration. The values of K and T_{pc} were 31 units and 2 minutes at this point and the unit penalty cost is \$0.68/job. Since the TC starts to increase after this point, we decided to carry out a new set of experiment around this point. In other words, the design point (31, 2) was considered as center point in new experimental plan. It must be noted that the new level of K (i.e., 31) is greater than K_{lim} (i.e., 25) which was given before. So, the K_{lim} is included in the new design region.

The new design region of the experiment was selected as (15,47) units of K and (0.5,3.5) minutes of T_{pc}. Table 4.26 shows the new design of experiment with both coded and natural variables.

Table 4.26 The New Design for Fitting The First Order Model

Design Point	K	T_{pc}	X_1	X_2
1	47	3.5	+1	+1
2	47	0.5	+1	-1
3	15	3.5	-1	+1
4	15	0.5	-1	-1
5	31	2	0	0

Based on this new plan, we carried out five replications at each design point to fit a first-order model. All of the other system and model parameters are the same as in previous experiment. Table 4.27 represents the results of the simulation.

Table 4.27 The Simulation Results (After Steepest Descent)

Replication No.	TC (X_1,X_2)				
	(+1,+1)	(+1,-1)	(-1,+1)	(-1,-1)	(0,0)
1	0.69	0.80	0.69	0.75	0.68
2	0.69	0.79	0.68	0.74	0.68
3	0.69	0.79	0.68	0.74	0.68
4	0.69	0.79	0.68	0.74	0.68
5	0.68	0.79	0.68	0.74	0.68

Based on the data set given in Table 4.27, the estimated first-order model was $TC = 0.7168 + 0.014X_1 - 0.041X_2$. Table 4.28 represents the results of ANOVA.

Table 4.28 ANOVA for the First Order Model (After Steepest Descent)

Source of Variation		Sum of Squares	Degrees of Freedom	Mean Squares	*F*	*P*
Regression		0.037540	2	0.018770	36.857	0.000
Residual		0.011204	22	0.000509		
LOF	Interaction	0.002420	1	0.002420	151.250	0.000
	Quadratic	0.008464	1	0.008464	529.000	0.000
	Pure Error	0.000320	20	1.6000E-5		
Total		0.048744	24			

As it can be seen in table, the regression is significant. However, there is a *LOF* in the model. The interaction and the quadratic effects are statistically significant at $\alpha = 0.05$. The R^2 and the R^2_{adj} are estimated as 77.02% and 74.93%. So, it can be said that the first-order model is not adequately representing the response in the new design region. Table 4.29 represents the *t*-tests on all of the coefficients to show the significances of the individual regression coefficients.

Table 4.29 The Test on the Individual Regression Coefficients

Predictor	Coefficients	Standard Error	*t*-ratio	*P*
β_0	0.71680	0.004513	158.816	0.000
β_1	0.01400	0.005046	2.774	0.011
β_2	-0.04100	0.005046	-8.125	0.000

As it can be seen in Table 4.29, the individual regression coefficients are statistically significant at $\alpha = 0.05$. However, as it was mentioned above, the model has a *LOF*. The curvature and interaction effects are significant. So, at this stage, the following second-order model is fit to the simulation results;

$$TC = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \beta_4 X_1^2 + \beta_5 X_2^2$$

The face center points were added to the previous design to estimate the second-order model given above. The resultant design is called as the Face Centered Central Composite Design. The face center points are (+1,0), (-1,0), (0,+1), and (0,-1). The previous design (i.e., the 2^2 factorial design with center points) was augmented by adding the face center points. Table 4.30 shows the face center points of this experiment with both coded and natural variables.

Table 4.30 The Additional Face Center Points

Design Point	K	T_{pc}	X_1	X_2
1	47	2	+1	0
2	15	2	-1	0
3	31	3.5	0	+1
4	31	0.5	0	-1

Five replications were carried out at each design point given in Table 4.30. All of the other system and model parameters are the same as in previous experiment. Table 4.31 represents the results of simulation.

Table 4.31 The Simulation Results using Face Center Points

Replication No.	$TC(X_1,X_2)$			
	(+1,0)	(-1,0)	(0,+1)	(0,-1)
1	0.70	0.68	0.68	0.77
2	0.70	0.67	0.67	0.76
3	0.70	0.67	0.68	0.76
4	0.70	0.67	0.68	0.77
5	0.69	0.67	0.67	0.76

By applying the *LSE* procedure to the data set given in Table 4.27 and Table 4.31, the following regression equation was estimated;

$$TC = 0.67956 + 0.01367X_1 - 0.042X_2 - 0.011X_1X_2 + 0.00567X_1^2 + 0.04067X_2^2$$

Table 4.32 and Table 4.33, represent the ANOVA for the regression and the significances of the individual regression coefficients, respectively.

Table 4.32 ANOVA for the Second Order Model

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Squares	*F*	*P*
Regression	0.077802	5	0.015560	769.256	0.000
Residual	0.000789	39	2.0228E-05		
LOF	0.000069	3	2.2963E-05	1.148	0.343
Pure Error	0.000720	36	2.0000E-05		
Total	0.078591	44			

As it can be seen in the table, the regression is significant and there is no *LOF* in the model. The R^2 and the R^2_{adj} are estimated as 99.00% and 98.87%. So, it can be said that the second-order model is well suited to our data.

Table 4.33 The Tests on the Individual Regression Coefficients

Predictor	Coefficients	Standard Error	*t*-ratio	*P*
β_0	0.67956	0.001499	453.284	0.000
β_1	0.01367	0.000821	16.644	0.000
β_2	-0.04200	0.000821	-51.149	0.000
β_3	-0.01100	0.001006	-10.938	0.000
β_4	0.00567	0.001422	3.984	0.000
β_5	0.04067	0.001422	28.593	0.000

As it can be seen in Table 4.33, all of the regression coefficients are statistically significant. So, this second-order model can be used for finding the levels of K and T_{pc} which minimizes the unit penalty cost. Before we use this second-order model for optimization, we need to validate it using randomly chosen design points in given design region. Ten points were selected randomly, and the results from the regression equation were compared with the results from the simulation model. The number of replications at each randomly selected point was 10. The comparison criterion was *MAPD*. The results of this study can be seen in Table 4.34.

Table 4.34 The Validation of the Second Order Regression Equation

K	T_{pc}	X_1	X_2	TC_{Reg}	TC_{Sim}	*APD*
20	1	-0.6875	-0.6667	0.714	0.692	3.179%
30	1	-0.0625	-0.6667	0.724	0.703	2.987%
40	1	0.5625	-0.6667	0.739	0.719	2.782%
18	1.5	-0.8125	-0.3333	0.688	0.677	1.625%
35	1.5	0.2500	-0.3333	0.703	0.691	1.737%
16	2.5	-0.9375	0.3333	0.666	0.671	0.745%
43	2.5	0.7500	0.3333	0.681	0.689	1.161%
20	3	-0.6875	0.6667	0.668	0.671	0.447%
30	3	-0.0625	0.6667	0.669	0.674	0.742%
40	3	0.5625	0.6667	0.675	0.682	1.026%

The *APD* Column in Table 4.34 represents the absolute percent deviations. It is calculated as $|TC_{Reg}-TC_{Sim}|/TC_{Sim}$. The *MAPD* is average of the numbers in *APD* column. The *MAPD* value of this model is 1.643% This level of *MAPD* can be

accepted as satisfactory and stated that this second-order model can be safely employed for optimization. The Contour Plot of the second-order regression equation for the design region can be seen in Figure 4.17.

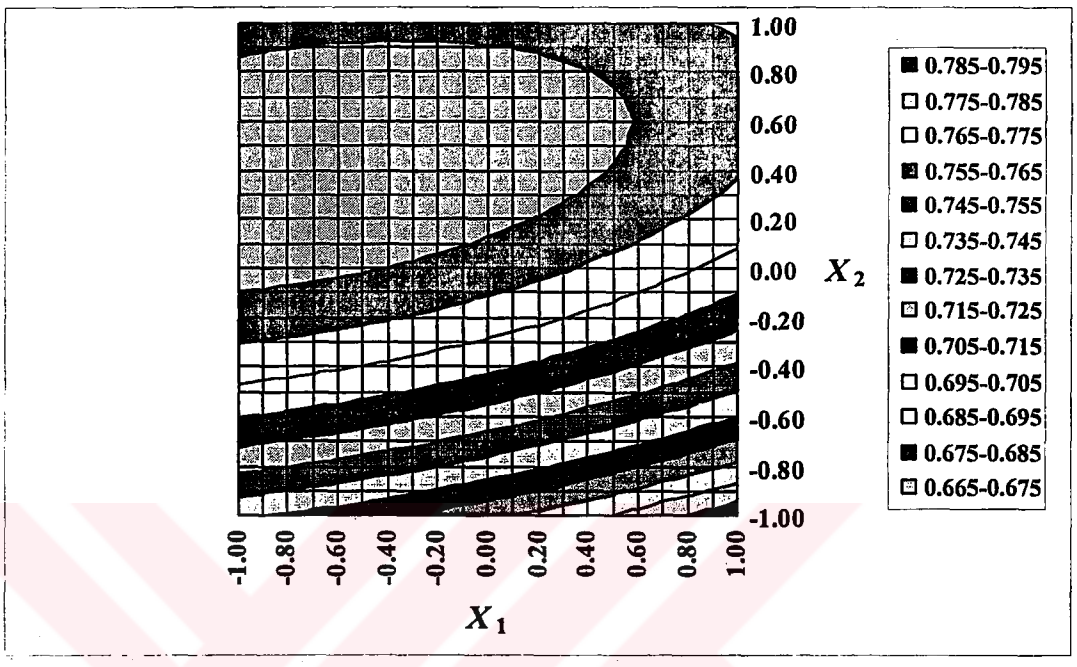

Figure 4.17 The Contour Plot of the Second Order Model

As it can be seen from the contour plot, the minimum point is not very remote from the center of the design. Most probably, the optimum levels of X_1 and X_2 are around the neighborhood of the point (-0.80,0.40). The location of the stationary point can be calculated by applying canonical analysis;

Let the second-order model be

$$TC = \beta_0 + X'b + X'BX$$

where,

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, b = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 0.01367 \\ -0.042 \end{bmatrix}, B = \begin{bmatrix} \beta_4 & \beta_3/2 \\ \beta_3/2 & \beta_5 \end{bmatrix} = \begin{bmatrix} 0.00567 & -0.0055 \\ -0.0055 & 0.04067 \end{bmatrix}$$

At the stationary point,

$\frac{\partial TC}{\partial X} = b + 2BX = 0$, then, $X_0 = -0.5B^{-1}b$. So, the location of the stationary point is,

$$X_0 = -0.5 \cdot \begin{bmatrix} 0.00567 & -0.0055 \\ 0.0055 & 0.04067 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 0.01367 \\ -0.042 \end{bmatrix} = \begin{bmatrix} -0.811 \\ 0.407 \end{bmatrix}$$

The stationary point for the second-order model is X_1=-0.811, and X_2=0.407. These values should be converted to their natural values. The natural values are $K = 31 - 0.811 \times 16 = 18.024 \approx 18$ units, and $T_{pc} = 2 + 0.407 \times 1.5 = 2.611$ minutes. The unit penalty cost at this optimum point is \$0.665/job with respect to the second-order model. Moreover, we carried out 20 replications at the optimum level of K and T_{pc} and estimated the cycle time and the unit penalty cost as 5.204 minutes and \$0.671/job, respectively. The *APD* for *TC* is $|0.665 - 0.671| / 0.671 = 0.894\%$. So, the deviation between the result of the estimated model and the results of simulation at the optimum point is lower than one percent. This low deviation shows the effectiveness of the polynomial regression metamodel.

It must be noted that the location of the stationary point (i.e., 18-2.611) is consistent with the results of the studies presented in Section 4.4 (see Table 4.19 and Figure 4.16). In those studies, it was stated that the minimum unit penalty cost can be reached in the neighborhood of K=20 jobs and T_{pc}=3 minutes.

As it was mentioned earlier, our objective in all these experiments was to obtain a cycle time which is lower than the target cycle time, $1/TR_{lim}$, (i.e., 5.20 minutes) So, lastly we wanted to evaluate whether the difference between the estimated cycle time and the target cycle time is statistically significant or not. The 95% confidence interval for the cycle time results was estimated as $5.204 \pm \frac{0.0186}{\sqrt{20}} \cdot t_{0.975,19}$ (i.e.,5.195-5.213). Since this confidence interval includes the target cycle time of

5.20 minutes/job at α=0.05, we can state that this CONWIP line will not operate above a given target cycle time under these optimum levels.

CHAPTER FIVE
CONCLUSION AND FUTURE RESEARCH

5.1 Conclusion

The main contribution of this thesis was to propose a worker flexibility policy for a CONWIP controlled serial manufacturing line producing a single class job under the saturated demand. The primary issue in developing this worker flexibility policy was to decide on the length of periodic controls for evaluating the current system status. To determine the optimum level of this decision variable, we developed a two-stage simulation optimization methodology and used a cost-based performance measure in implementing this procedure. Hence, another contribution of this thesis was to develop a unit penalty cost function.

To evaluate the effectiveness of this worker flexibility policy, we carried out the experimental studies in three groups. During the first group of studies, we developed the simulation model of a hypothetical serial CONWIP line using ARENA 2.2, and compared the performance of this new policy (*PC*), with other flexibility policies reported in the literature; WW_0, WW_K, and *PR*. The performance criteria in these comparison studies were the mean cycle time and the unit penalty cost. These first set of experiments were conducted under the following operating conditions;

- Satisfaction of the optimality condition of the *PR* policy.
- Dissatisfaction of the optimality condition of the *PR* policy.
- Unbalanced line due to differences in workstation capacities.
- Unbalanced line due to differences in service times.

- Increasing the number of workers.
- Increasing the number of workstations.
- Including the worker transfer delays.

When the optimality condition of the *PR* policy was satisfied, as expected, the performance of *PR* policy at each level of K was the best in both of the performance measures. The proposed policy had the worst performance with respect to both criteria.

When the optimality condition of *PR* was not satisfied, the WW_0 policy turned to be inappropriate to use because increasing the level of K resulted in increase in mean cycle time. Another result was that the difference in the performance of different policies (i.e., cycle time) became less and less when the level of K was increased. A similar behavior was observed when the number of machines was increased. In comparison to the *PR* policy, the WW_K policy produced near optimum cycle times at lower level of additional investment. And lastly, in comparison to the WW_K policy, the proposed flexibility policy produced lower cycle times at lower level of additional investment.

The results of the comparisons studies given so far are valid under the assumption that the line is balanced in terms of both the mean service times and the number of machines. To evaluate the performances of the policies under the conditions that the additional capacities were used and the workload of bottleneck station was shared, first, we allowed differences in workstation capacities and made the line unbalanced. The most effective policy under the condition that more capacity was added was found to be the proposed policy. Second, we allowed differences in service times to make the line unbalanced. The results of this experiment were similar to the results of the first one. So, the proposed policy outperformed other policies in sharing the workload of the bottleneck.

When the total number of workers was increased, the proposed policy produced lower cycle times than the other policies. However, the differences in cycle times

were low for high levels of staffing. When the number of workstations was increased, the proposed policy and the WW_K policy produced near minimum cycle times at low levels of unit penalty cost in comparison to the *PR* policy. Likewise, when the worker transfer delays were included, the most robust policy was the proposed policy.

The main goal of the experimental studies in the second group was to get more insight into the behaviour of the simulation model by screening for two control factors (i.e., the level of CONWIP, K and the periodic control length, T_{pc}) which had the greatest impact on the simulation response (i.e., the mean cycle time and the unit penalty cost). The results of these experiments can be summarized as follows;

- Irrespective of the level of K, any increase in T_{pc} leads to an increase in cycle time.
- High levels of K and low levels of T_{pc} produce low cycle times.
- The minimum cycle time and the minimum unit penalty cost can be reached at different levels of K and T_{pc}.

The last group of experimental studies involved applying a two-stage simulation optimization methodology to minimize the unit penalty cost under a given cycle time.

In the light of the second group of experiment results, we gained confidence that the *RSM*-based simulation optimization methodology can be effectively used to determine the optimum levels of K and T_{pc}. We applied the *RSM*-based simulation optimization strategy in two stages. In the first stage, we carried out various sets of experiments to determine the minimum number of machines at each workstation until a predetermined cycle time is reached. As a result of these studies, we determined the capacity structure of the system as *2222*, which means that the number of machines at each workstation is two. In the second stage of the methodology, the *RSM* was applied to find the optimum levels of K and T_{pc}. First, we fit a first-order model to the simulation generated cost-based performance measure

and applied the steepest descent method to find the direction in which our performance measure decreases most rapidly. We moved along this path until the first-order model is inadequate. Later, the following second-order model was fit to the data and found to be adequately representing the true response function;

$$TC = 0.67956 + 0.01367X_1 - 0.042X_2 - 0.011X_1X_2 + 0.00567X_1^2 + 0.04067X_2^2$$

Moreover, the validation studies also supported the adequacy of this fitted model. Next, we applied canonical analysis to determine the optimum levels of K and T_{pc}, which were 18 jobs, and 2.611 minutes.

Based on the experimental results given so far, the proposed policy, *PC*, outperformed other alternatives with respect to both the mean cycle time and the unit penalty cost especially in systems having fewer number of machines. In other words, in comparison to the *PR* policy, the proposed policy resulted in near minimum cycle time at much lower cost since it required less additional capacity. So, if the capacity of a CONWIP system under the saturated demand can be increased by adding parallel machines or equipment, the *PC* policy has quite potential to improve the performance of the line. Moreover, since the *PC* policy evaluates the system status periodically, we have a direct control on the throughput rate of the system. By alternating the length of the periodic control, the throughput rate of the system can be set to a desired production target.

5.2 Future Research

In this thesis, it was assumed that the workers were homogenous and full cross trained. The worker flexibility policies for CONWIP lines having heterogeneous and partial cross trained workers are still a research gap. The proposed policy may be extended by adding worker efficiency information and allowing partial cross trained workers.

Another assumption in this thesis was the demand saturation. We saw just one study applying the *DRC* worker assignment rule pairs in a pull environment with nonsaturated demand (Horng & Cochran, 2001). However, this study is very limited in scope. Therefore, another research gap is the use of flexible workers in a pull environment with nonsaturated demand. Different pull mechanisms such as, KANBAN, CONWIP, and Hybrid may be modified to allow the use of flexible workers.

Another research gap in *DRC* systems is the use of multi-level worker flexibility which was introduced first by Felan and Fry (2001). The worker flexibility combination which minimizes the cost of training may be a good candidate area.

REFERENCES

Altıok, T. (1997). Performance Analysis of Manufacturing Systems. New York: Springer-Verlag.

Arora J.S. (1989). Introduction to Optimum Design, McGraw Hill.

Askin, R.G. & Standridge, C.R. (1993). Modeling and Analysis of Manufacturing Systems. Singapore: John Wiley & Sons.

Askin, Ronald G. & Huang, Yuanshu. (2001). Forming Effective Worker Teams for Cellular Manufacturing. International Journal of Production Research, 39, (11), pp.2431-2451.

Ayhan, H. & Wortman, M.A. (1999). An Approximation for Computing the Throughput of Closed Assembly-Type Queuing Networks. European Journal of Operational Research, 112, (1), pp.107-121.

Azadivar, F. (1999). Simulation Optimization Methodologies. In Proceedings of Winter Simulation Conference, pp.198-204.

Bartholdi III J.J.& Einstein D.D. (1996). A Production Line that Balances Itself. Operations Research, 44, (1), pp.21-34.

Bartholdi III J.J.& Einstein D.D. (1998). A Self Organizing Scheme for Sharing Work. Technical Report, Georgia Institute of Technology.

Bartholdi III J.J., Bunimovich L.A. & Einstein D.D. (1999). Dynamics of 2- and 3-Worker Bucket Brigade Production Lines. Operations Research, 17, (3), pp.488-

491.

Bartholdi III J.J., Einstein D.D. & Foley R.D., (2000), Performance of Bucket Brigades When Work is Stochastic. Technical Report, Georgia Institute of Technology.

Beach, R., Muhlemann, A.P., Price, D.H.R., Paterson, A. & Sharp, J.A. (2000). A Review of Manufacturing Flexibility. European Journal of Operational Research, 122, pp.41-57.

Bischak D.P. (1996). Performance of a Manufacturing Module with Moving Workers. IIE Transactions, 28, pp.723-733.

Bobrowski, Paul M. & Park, P. Sungchil. (1989). Work Release Strategies in a Dual Resource Constrained Job Shop. OMEGA, 17, (2), pp.177-188.

Bobrowski, Paul M. & Park, P. Sungchil. (1993). An Evaluation of Labor Assignment Rules When Workers are not Perfectly Interchangeable. Journal Of Operations Management, 11, (3), pp.257-268.

Bonvik, A.M., Couch, C.E. & Gershwin, S.B. (1997). Comparison of Production Line Control Mechanisms. International Journal of Production Research, 35, (3), pp.789-804.

Box G.E.P. & Draper N.R. (1987). Emprical Model Building and Response Surfaces, John Wiley & Sons, Inc.

Buzacott, J.A. & Shanthikumar, J.G. (1993). Stochastic Models of Manufacturing Systems. New Jersey: Prentice Hall.

Carson Y. & Maria A. (1997). Simulation Optimization: Methods and Applications. In Proceedings of Winter Simulation Conference, pp.118-126.

Chen, Houn-Gee. (1995). Heuristics for Operator Scheduling in Group Technology Cells. Computers and Operations Research, 22, (3), pp.261-276.

Conway, R., Maxwell W., McClain, J.O. & Thomas, L.J. (1988). The Role of Work-in-Process Inventory in Serial Production Lines. Operations Research, 36, (2), pp.229-241.

Dallery, Y. & Liberopoulos G. (2000). Extended Kanban Control System: Combining Kanban and Base Stock. IIE Transactions, 32, 369-386.

Dar-El, E.M., Herer, Y.T. & Masin, M. (1999). CONWIP-Based Production Lines with Multiple Bottlenecks: Performance and Design Implications. IIE Transactions, 31, (2), pp.99-111.

Diamond W. J. (1989). Practical Experiment Designs, Van Nostrand Reinhold, Competetive Manufacturing Series.

Donohue J.M., Houck E.C. & Myers R.H. (1992). Simulation Designs for Quadratic Response Surface Models in the Presence of Model Misspecification. Management Science, 38, (12), pp.1765-1791.

Donohue J.M., Houck E.C. & Myers R.H. (1995). Simulation Designs for the Estimation of Quadratic Response Gradients in the Presence of Model Misspecification. Management Science, 41, (2), pp.244-262.

Draper N.R. (1981). Applied Regression Analysis, John Wiley & Sons, Inc.

Duenyas, I. & Hopp, W.J. (1992). CONWIP Assembly with Deterministic Processing and Random Outages. IIE Transactions, 24, (4), pp.97-109.

Duenyas, I., Hopp, W.J. & Spearman, M.L. (1993). Characterizing the Output Process of a CONWIP Line with Deterministic Processing and Random Outages. IIE Transactions, 39, (8), pp.975-988.

Duri, C., Frein, Y. & Lee, H.S. (2000). Performance Evaluation and Design of a CONWIP System with Inspections. International Journal of Production Economics, 64, (1-3), pp.219-229.

Elvers, Douglas A. & Treleven, Mark D. (1985). Job Shop vs. Hybrid Flow Shop Routing in a Dual Resource Constrained System. Decision Sciences, 16, (2), pp.213-222.

Felan III, J. T., Fry, Timothy D. & Philipoom, Patric R. (1993). Labour Flexibility and Staffing Levels in a Dual Resource Constrained Job Shop. International Journal of Production Research, 31, (10), pp.2487-2506.

Felan III, J. T., & Fry, Timothy D. (2001). Multi-Level Heterogenous Worker Flexibility in a DRC Job Shop. International Journal of Production Research, 39, (14), pp.3041-3059.

Fredendall, L.D., Melnyk, S.A. & Ragatz, G. (1996). Information and Scheduling in a Dual Resource Constrained Job Shop. International Journal of Production Research, 34, (10), pp.2783-2802.

Frein Y., Di Mascolo, M. & Dallery Y. (1995). On the Design of Generalized Kanban Control Systems. International Journal of Operations and Production Management, 15, (9), pp.158-184.

Fry, Timothy D., Kher, Hemant V. & Malhotra, Manoj K. (1995). Managing Worker Flexibility and Attrition in Dual Resource Constrained Job Shops. International Journal of Production Research, 33, (8), pp.2163-2179.

Fryer, John S. (1973). Operating Policies in Multiechelon Dual Constrained Job Shops. Management Science, 19, (9), pp.1001-1002.

Fryer, John S. (1974). Labor Flexibility in Multiechelon Dual Constrained Job Shops. Management Science, 20, pp.1073-1080.

Fryer, John S. (1975). Effects of Shop Size and Labor Flexibility in Labor and Machine Limited Production Systems. Management Science, 21, (5), pp.507-515.

Fu, M.C. (1994). Optimization via Simulation: A Review. Annals of Operations Research, 53, pp.199-247.

Gaury, E.G.A., Pierreval, H. & Kleijnen, P.C. (2000). An Evolutionary Approach to Select a Pull System among Kanban, CONWIP, Hybrid. Journal of Intelligent Manufacturing, 11, (2), pp.157-167.

Gel, E.S. (1999). Stochastic Models of Workforce Agility in Production Systems. Ph.D. Dissertation, North Western University.

Gel, E.S., Hopp, W.J. & Oyen M.P.V. (2000). Factors Affecting Opportunity of Worksharing as a Dynamic Line Balancing Mechanism, Technical Report, Arizona State University.

Golany, B., Dar-El, E.M. & Zeev, N. (1999). Controlling Shop Floor Operations in a Multi-Family, Multi-Cell Manufacturing Environment through Constant Work-in-Process. IIE Transactions, 31, (8), pp.771-781.

Groover, M.P. (2001). Automation, Production Systems, and Computer-Integrated Manufacturing. Englewood Cliffs, New Jersey: Prentice-Hall.

Gstettner, S. & Kuhn H. (1996). Analysis of Production Control Systems Kanban and CONWIP. International Journal of Production Research, 34, (11), pp.3253-3273.

Gunther, Richard E. (1979). Server Transfer Delays in A Dual Resource Constrained Parallel Queuing System. Management Science, 25, (12), pp.1245-1257.

Gunther, Richard E. (1981). Dual Response Paralel Queues with Server Transfer and Information Access Delays. Decision Sciences, 12, (1), pp.97-111.

Hardin R.H. (1992). A new Approach to the Construction of Optimal Designs. Technical Report, AT&T Bell Laboratories.

Hazra, J., Schweitzer, P.J. & Seidman, A. (1999). Analyzing Closed Kanban Controlled Assembly Systems by Iterative Aggregation-Disaggregation. Computers and Operations Research, 26, (10-11), pp.1015-1039.

Hogg ,G. L., Phillips, D.T. & Maggard, M.J. (1975). GERTS QR: A Model for Multi-Resource Constrained Queuing Systems, Part I:Concepts, Notation, and Examples. AIIE Transactions, 7, (2), pp.89-99.

Hogg ,G. L., Phillips, D.T. & Maggard, M.J. (1977).Parallel-Channel, Dual-Resource Constrained Queuing Systems with Heterogenous Resources. AIIE Transactions, 9, (4), pp.352-362.

Hood S.J. (1993). Response Surface Methodology and Its Application in Simulation. In Proceedings of Winter Simulation Conference, pp.115-122.

Hopp, W.J. & Spearman, M.L. (2001). Factory Physics. New York: Mc Graw Hill.

Horng, H.-C. & Cochran, J.K. (2001). Project Surface Regions: A Decision Support Methodology for Multitasking Worker Assignment in JIT Systems. Computers and Industrial Engineering, 39, (1-2), pp.159-171.

Hottenstein, Michael P. & Bowman, Sherry A. (1998). Cross-Training and Worker Flexibility: A Review of DRC System Research. The Journal of High Technology Management Research, 9, (2), pp.157-174.

Huang, M., Wang, D. & Ip, W.H. (1998). Simulation Study of CONWIP for a Cold Rolling Plant. International Journal of Production Economics, 54, (3), pp.257-266.

Hussey J.R., Myers, R.H. & Houck E.C. (1987). Pseudorandom Number Assignment in Quadratic Response Surface Designs. IIE Transactions, 19, (4), pp.395-403.

Ip, W.H., Yung, K.L., Huang, M. & Wang, D. (2002). A CONWIP Model for FMS Control. Journal of Intelligent Manufacturing, 13, (2), pp.109-117.

Isakow, S.A. & Golany, B. (2003). Managing Multi-Project Environments through Constant Work-in-Process. International Journal of Project Management, 21, (1), pp.9-18.

Jensen, John B. (2000). The Impact of Resource Flexibility and Staffing Decisions on Cellular and Departmental Shop Performance. European Journal of Operational Research, 127, (2), pp.279-296.

Joshi S., Sherali H. & Tew J. D. (1998). An Enhanced Responce Surface Methodology (RSM) Algorithm Using Gradient Deflection and Second-Order Search Strategies. Computers and Operation Research, 25, (7-8), pp.531-541.

Karaesmen, F. & Dallery, Y. (2000). A Performance Comparison of Pull Type Control Mechanisms for Multi-Stage Manufacturing. International Journal of Production Economics, 68, pp.59-71.

Kher, Hemant V. & Fry Timothy D. (2001). Labour Flexibility and Assignment Policies in a Job Shop having Incommensurable Objectives. International Journal of Production Research, 39, (11), pp.2295-2311.

Kher, Hemant V. & Malhotra, Manoj K. (1994). Acuiring and Operationalizing Worker Flexibility in Dual Resource Constrained Job Shops with Worker Transfer Delays and Learning Losses. OMEGA, 22, (5), pp.521-533.

Kher, Hemant V., Malhotra, Manoj K., Philipoom, Patrick R. & Fry Timothy D. (1999). Modeling Simultaneous Worker Learning and Forgetting in Dual Resource Constrained Systems. European Journal of Operational Research, 115, (1), pp.158-172.

Kher, Hemant V. (2000). Examination of Worker Assignment and dispatching Rules for Managing Vital Customer Priorities in Dual Resource Constrained Job Shop Environments. Computers and Operations Research, 27, (6), pp.525-537.

Kleijnen J.P.C. (1992). Regression Metamodels for Simulation with Common Random Numbers: Comparison of Validation Tests and Confidence Intervals. Management Science, 38,(8), pp.1164-1185.

Kleijnen, J.P.C. (1994). Sensitivity Analysis and Optimization of Simulation Models. In Proceedings of the European Simulation Symposium, pp.3-7.

Kleijnen, J.P.C. (1995). Theory and Methodology-Verification And Validation Of Simulation Models. European Journal Of Operational Research, 82, 145-162.

Kleijnen, J.P.C. (1998). Experimental Design for Sensitivity Analysis, Optimization, and Validation of Simulation Models. Handbook of Simulation, Ed. Jerry Banks, pp.173-223.

Kleijnen J.P.C. & Sargent R.G. (2000). A Methodology for Fitting And Validating Metamodels in Simulation. European Journal of Operational Research, 120, pp.14-29.

Koste, L.L. & Malhotra Manoj K. (1999). A Theoretical Framework for Analyzing the Dimensions of Manufacturing Flexibility. Journal of Operations Management, 18, (1), pp.75-93.

Kwon C. & Tew J.D. (1994). Strategies for Combining Antithetic Variates and Control Variates in Designed Simulation Experiments. Management Science, 40, (8), pp.1021-1033.

Law, A.M., & Kelton, W.D. (1991). Simulation Modeling & Analysis. Singapore: McGraw-Hill.

Liao, C.-J. & Lin H.-T. (1998). A Case Study in A Dual Resource Constrained Job Shop. International Journal of Production Research, 36, (11), pp.3095-3111.

Liberopoulos, G. & Dallery, Y. (2002). Base Stock versus WIP cap in Single-Stage Make-to-Stock Production-Inventory Systems. IIE Transactions, 34, pp.627-636.

Malhotra, Manoj K., Fry, Timothy D., Kher, Hemant V. & Donohue, Joan M. (1993) The Impact of Learning and Labor Attrition on Worker Flexibility in Dual Resource Constrained Job Shops. Decision Sciences, 24, (3), pp.641-663.

Malhotra, Manoj K. & Kher, Hemant V. (1994). An Evaluation of Worker Assignment Policies in Dual Resource Constrained Job shops with Heterogenous

Resources and Worker Transfer Delays. International Journal of Production Research, 32, (5), pp.1087-1103.

Marek R.P., Elkins, D.A. & Smith, D.R. (2001). Understanding the Fundamentals of Kanban and CONWIP Pull Systems Using Simulation. In Proceedings of Winter Simulation Conference, 2, pp.921-929.

McClain J.O., Thomas L.J. & Sox C. (1992). On the Fly Line Balancing with very little WIP. International Journal of Production Economics, 27, pp.283-289.

Montgomery D.C. (1991). Design and Analysis of Experiments. Singapore: John Wiley & Sons, Inc.

Morrice D. J. & Schruben L.W. (1993). Simulation Factor Screening Using Harmonic Analysis. Management Science, 39, (12), pp.1459-1476.

Morris, John S. & Tersine, Richard J. (1994). A Simulation comparison of Process And Cellular Layouts in A Dual Resource Constrained Environment. Computers and Industrial Engineering, 26, (4), pp.733-741.

Muckstadt J.A. & Tayur S.R. (1995a). A Comparison of Alternative Kanban Control Mechanisms: Background and Structural Results. IIE Transactions, 27, pp.140-150.

Muckstadt J.A. & Tayur S.R. (1995b). A Comparison of Alternative Kanban Control Mechanisms: Experimental Results. IIE Transactions, 27, pp.151-161.

Murugabaskar S. & Huang W.V. (1993) Simulation Analysis with Group Screening. Computers and Industrial Engineering, 25, pp.25-28.

Myers R.H. & Montgomery D.C. (1995). Response Surface Methodology: Process and Product Optimization Using Designed Experiments, John Wiley & Sons, Inc.

Nelson, R.T. (1967). Labor and Machine Limited Production Systems. Management Science, 13, (9), pp.648-671.

Nelson, R.T. (1970). A Simulation of Labor Efficiency and Centralized Assignment in a Production Model. Management Science, 17, (2), pp.97-106.

Nozari A., Arnold S. F. & Pegden C.D. (1987) Statistical Analysis for Use with The Schruben and Margolin Correlation Induction Strategies. Operations Research, 35, (1), pp.127-138.

Ostolaza J., McClain J.& Thomas J. (1990). The Use of Dynamic Assembly Line Balancing to Improve Throughput, Journal Of Manufacturing Operations Management, 3, pp.105-133.

Ovalle, O.R. & Marquez, A.C. (2003). Exploring the Utilization of A CONWIP System for Supply Chain Management. International Journal of Production Economics, 83, (2), pp.195-215.

Oyen, Mark P., Gel, Esma G.S. & Hopp, Wallace J. (2001). Performance Opportunity for Workforce Agility in Collaborative and Noncollaborative Work Systems. IIE Transactions, 33, (9), pp.761-777.

Papadopoulos, H.T. & Heavey, C. (1996). Queuing Theory in Manufacturing Systems Analysis and Design: A Classification of Models for Production and Transfer Lines. European Journal of Operational Research, 92, pp.1-27.

Park, C.S. (2002). Contemporary Engineering Economics, New Jersey: Prentice Hall.

Park, P. Sungchil & Bobrowski, Paul M. (1989). Job Release and Labor Flexibility in a Dual Resource Constrained Job Shop. Journal Of Operations Management, 8,(3), pp.230-249.

Paternina, C.D. & Das, T.K. (2001). Intelligent Dynamic Control Policies for Serial Production Lines. IIE Transactions, 33, (1), pp.65-77.

Pflug, G.C. (1996). Optimization of Stochastic Models: The Interface between Simulation and Optimization, Kluwer Academics Publishers.

Rardin R.L. (1998). Optimization in Operations Research, Prentice Hall.

Reiser, M. & Lavenberg, S.S. (1980). Mean Value Analysis of Closed Multi-Chain Queuing Networks. Journal of the ACM, 27, (2), pp.313-322.

Ryan, S.M., Baynat, B. & Choobineh, F.F. (2000). Determining Inventory Levels in a CONWIP Controlled Job Shop. IIE Transactions, 32, (2), pp.105-114.

Safizadeh, M.H. (1990). Optimization in Simulation: Current Issues and the Future Outlook. Naval Research Logistics, 37, pp.807-825.

Schultz, Kenneth L., McClain, John O. & Thomas, Joseph L. (2003). Overcoming the Dark Side of Worker Flexibility. Journal of Operations Management, 21,(1), pp.81-92.

Song W.T. & Su C.C. (1996.) An Extension of the Multiple-Blocks Strategy on Estimating Simulation Metamodels. IIE Transactions, 28, pp.511-519.

Song W.T. & Su C.C. (1998). A Three-class Variance Swapping Technique for Simulation Experiments. Operations Research Letters, 23, pp.63-70.

Spearman, M.L., Woodruff, D.L. & Hopp W.J. (1990). CONWIP: A Pull Alternative to Kanban. International Journal of Production Research, 28, (5), pp.879-894.

Suresh, Nallan C. & Gaalman, Gerard J. C. (2000). Performance Evaluation of Cellular Layouts: Extension to DRC System Contexts. International Journal of Production Research, 38, (17), pp.4393-4402.

Swisher, J.R., Hyden, P.D., Jacobson, S.H. & Schruben, L.W. (2000). A Survey of Simulation Optimization Techniques and Procedures, In Proceedings of Winter Simulation Conference, pp.119-128.

Tew J.D. (1992). Using Central Composite Designs in Simulation Experiments. In Proceedings Winter Simulation Conference, pp.529-538.

Tew J.D. & Wilson J.R. (1994). Estimating Simulation Metamodels Using Combined Correlated Based Variance Reduction Techniques. IIE Transactions, 26, (3), pp.2-16.

Treleven, Mark D. & Elvers, Douglas A. (1985). An Investigation of Labor Assignment Rules in a Dual-Constrained Job Shop. Journal of Operations Management, 6, (1), pp.51-67.

Treleven, Mark D. (1987). The Timing of Labor Transfers in Dual Resource-Constrained Systems: Push vs. Pull Rules. Decision Sciences, 18,(1), pp.73-88.

Treleven, Mark D. (1988). A Comparison of Flow and Queue Time Variances in Machine Limited Versus Dual Resource Constrained Systems, IIE Transactions, 20, (1), pp.63-67.

Treleven, Mark D. (1989) A Review of the Dual Resource Constrained System Research., IIE Transactions, 21, (3), pp.279-287.

Wang, H. (1997). Approximate MVA Algorithms for Solving Queuing Network Models. M.Sc. Thesis, University of Toronto.

Weeks, James K. & Fryer, John S. (1976). A Simulation study of Operating Policies In a Hypothetical Dual Constrained Job Shop. Management Science, 22, (12), pp.1362-1371.

Wisner, J.D. (1991). The Effect of Labor Relearning on Scheduling Policies in a Job Shop. Ph.D. Dissertation, Arizona State University.

Wisner, J.D. (1995). A Review of the Order Release Policy Research. International Journal of Operations and Production Management, 15, (6), pp.25-40.

Youssef Y.A., Beauchamp Y. & Thomas M. (1994). Comparison of a Full Factorial Experiment to Fractional And Taguchi Designs in a Lathe Dry Turning Operation. Computers and Industrial Engineering, 27, pp.59-62.

Zavadlav E., McClain J.O. & Thomas L.J. (1996). Self Buffering, Self Balancing, Self Flushing Production Lines. Management Science, 42, (8), pp.1151-1164.

Zhang Q., Vonderembse M.A. & Lim, Jeen Su. (2003). Manufacturing Flexibility: Defining and Analyzing Relationships among Competence, Capability, and Customer Satisfaction. Journal of Operations Management, 21, (2), pp.173-191.

APPENDICES

Appendix A1. The MOD. File of ARENA2.2 for the Standard CONWIP Line

```
0$              CREATE,         conwip,0:,1;
1$              ASSIGN:         m=1:MARK(time);
6$              ROUTE:          0.0,m+1;
7$              STATION,        2-5;
2$              QUEUE,          m-1:MARK(time1);
3$              SEIZE,          1:
                                m-1,1;
10$             TALLY:          m-1,int(time1),1;
4$              DELAY:          processtime(m-1):MARK(time2);
16$             TALLY:          m+9,int(time2),1;
8$              BRANCH,         1:
                                If,m<(nos+1),5$,Yes:
                                Else,11$,Yes;
5$              RELEASE:        m-1,1:NEXT(6$);
11$             RELEASE:        m-1,1;
9$              TALLY:          21,int(time),1;
13$             BRANCH,         1:
                                If,nc(c1)==(quantity-1),15$,Yes:
                                Else,12$,Yes;
15$             ASSIGN:         cycletime=(tnow-warmup)/quantity;
14$             WRITE,          file1,"%8.5f %6.4f %6.4f %6.4f %6.4f
                                %6.4f %6.4f %6.4f %6.4f %6.4f %6.4f
                                %6.4f %6.4f\n":
                                cycletime,
                                davg(1),
                                davg(2),
                                davg(3),
                                davg(4),
                                davg(5),
                                davg(6),
                                davg(7),
                                davg(8),
                                davg(9),
                                davg(10),
                                davg(11),
                                davg(12);
20$             WRITE,          file2,"%8.5f %8.5f %8.5f %8.5f %8.5f
                                %8.5f %8.5f %8.5f %8.5f %8.5f\n":
                                tavg(1),
                                tavg(2),
                                tavg(3),
                                tavg(4),

                                tavg(11),
                                tavg(12),
```

```
                                tavg(13),
                              . tavg(14),
                                tavg(21);
12$             COUNT:          c1,1:NEXT(1$);
17$             CREATE,         1,0:,1;
18$             ASSIGN:         mc(c1)=quantity;
19$             DISPOSE;
```

Appendix A2. The EXP. File of ARENA2.2 for the Standard CONWIP Line

```
ATTRIBUTES:     1,time:
                2,time1:
                3,time2;
FILES:          1,file1,"c:\belgelerim\closednormal1.txt",
                Sequential(),Free Format,Error,No,Hold:
                2,file2,"c:\belgelerim\closednormal2.txt",
                Sequential(),Free Format,Error,No,Hold;
VARIABLES:      warmup,100000:
                cycletime:
                conwip,10:
                nos,4:
                quantity,200000;
SEEDS:          1,,Common:
                2,,Common:
                3,,Common:
                4,,Common:
                6,,Common:
                7,,Common;
QUEUES:         1,q1,FirstInFirstOut:
                2,q2,FirstInFirstOut:
                3,q3,FirstInFirstOut:
                4,q4,FirstInFirstOut;
RESOURCES:      1,r1,Capacity(4,):
                2,r2,Capacity(4,):
                3,r3,Capacity(4,):
                4,r4,Capacity(4,);
STATIONS:       1,s1:
                2,s2:
                3,s3:
                4,s4:
                5,s5;
COUNTERS:       1,c1,,Replicate;
TALLIES:        21,t1;
DSTATS:         1,nq(1):
                2,mr(1):
                3,nr(1):
                4,nq(2):
                5,mr(2):
                6,nr(2):
                7,nq(3):
                8,mr(3):
                9,nr(3):
                10,nq(4):
                11,mr(4):
                12,nr(4);
REPLICATE,      10,0.0,,Yes,Yes,100000;
EXPRESSIONS:    1,processtime(4),expo(5,1),expo(5,2),expo(5,3),
                expo(5,4);
```

Appendix B1. The MOD. File of ARENA2.2 for the *PR* Policy

```
0$              CREATE,         1,0.000000001:,1;
1$              DUPLICATE:      1,3$;
2$              ASSIGN:         wip=wip+1:
                                m=1:MARK(time);
13$             ROUTE:          0.0,m+1;
3$              BRANCH,         1:
                                If,wip<=(conwip-1),1$,Yes:
                                Else,5$,Yes;
5$              QUEUE,          1;
4$              WAIT:           signal1,1:NEXT(1$);
14$             STATION,        2-100;
6$              QUEUE,          m:MARK(time1);
7$              SEIZE,          1:
                                m-1,1;
17$             TALLY:          m-1,int(time1),1;
8$              DELAY:          processtime(m-1);
15$             BRANCH,         1:
                                If,m<(nos+1),18$,Yes:
                                Else,19$,Yes;
18$             ASSIGN:         mr(m-1)=mr(m-1)-1:
                                mr(m)=mr(m)+1;
9$              RELEASE:        m-1,1:NEXT(13$);
19$             ASSIGN:         mr(m-1)=mr(m-1)-1:
                                mr(1)=mr(1)+1:
                                wip=wip-1;
26$             RELEASE:        m-1,1;
12$             BRANCH,         1:
                                If,wip<=conwip,10$,Yes:
                                Else,16$,Yes;
10$             SIGNAL:         signal1,1;
16$             TALLY:          m,int(time),1;
27$             COUNT:          c1,1;
11$             DISPOSE;
20$             CREATE,         1,0:,1;
23$             WHILE:          index<=nos;
21$             ASSIGN:         mr(index)=0:
                                index=index+1;
24$             ENDWHILE;
25$             ASSIGN:         mr(1)=n;
22$             DISPOSE;
```

Appendix B2. The EXP. File of ARENA2.2 for the *PR* Policy

```
ATTRIBUTES:     1,signal1:
                2,time:
                3,time1;
VARIABLES:      1,index,1:
                2,conwip,10:
                3,n,2:
                4,nos,2:
                5,wip;
SEEDS:          1,,Common:
                2,,Common;
QUEUES:         100,qset,FirstInFirstOut;
RESOURCES:      100,rset,Capacity(1,);
STATIONS:       100,sset;
COUNTERS:       1,c1,10000,Replicate;
TALLIES:        100,t1;
DSTATS:         1,nq(2):
                2,mr(1):
                3,nr(1):
                4,nq(3):
                5,mr(2):
                6,nr(2):
                7,nq(4):
                8,mr(3):
                9,nr(3);
REPLICATE,      10,0.0,,Yes,Yes,5000;
EXPRESSIONS:    1,processtime(2),expo(5,1),expo(5,2);
```

Appendix C1. The MOD. File of ARENA2.2 for the WW_0 Policy

```
0$              CREATE,         conwip,0.00000000002:,1;
2$              ASSIGN:         m=1:MARK(time);
7$              ROUTE:          0,m+1;
8$              STATION,        2-11;
3$              QUEUE,          m-1:MARK(time1);
4$              SEIZE,          1:
                                m-1,1;
11$             TALLY:          m-1,int(time1),1;
5$              DELAY:          processtime(m-1):MARK(time2);
17$             TALLY:          m+9,int(time2),1;
9$              BRANCH,         1:
                                If,m<(nos+1),control,Yes:
                                Else,73$,Yes;
control         BRANCH,         1:
                                If,nq(m-1)<=q,62$,Yes:
                                Else,back,Yes;
62$             ASSIGN:         v1=1:
                                totalcon=totalcon+1;
50$             WHILE:          v1<=nos;
51$             ASSIGN:         minindex=0:
                                maxindex=0:
                                lower=0:
                                upper=0:
                                transfertime=0:
                                randoms=0:
                                load(v1)=nq(v1):
                                workload(v1,1)=v1:
                                workload(v1,2)=load(v1):
                                minload(v1,1)=workload(v1,1):
                                minload(v1,2)=workload(v1,2):
                                v1=v1+1;
52$             ENDWHILE;
63$             ASSIGN:         v2=1;
53$             WHILE:          v2<=(nos-1);
54$             ASSIGN:         v3=v2+1;
55$             WHILE:          v3<=nos;
60$             BRANCH,         1:
                                If,minload(v2,2)>minload(v3,2),56$,Yes:
                                Else,61$,Yes;
56$             ASSIGN:         depo1=minload(v2,1):
                                depo2=minload(v2,2):
                                minload(v2,1)=minload(v3,1):
                                minload(v2,2)=minload(v3,2):
                                minload(v3,1)=depo1:
                                minload(v3,2)=depo2:
                                v3=v3+1;
57$             ENDWHILE;
58$             ASSIGN:         v2=v2+1;
59$             ENDWHILE;
66$             ASSIGN:         v4=1;
67$             WHILE:          v4<=nos;
68$             ASSIGN:         maxload(v4,1)=minload(nos-v4+1,1):
                                maxload(v4,2)=minload(nos-v4+1,2):
                                v4=v4+1;
69$             ENDWHILE;
```

```
64$             ASSIGN:         v5=1;
65$             ASSIGN:         maxindex=maxload(v5,1):
                                lower=((maxindex-flex)<=0)*1+
                                ((maxindex-flex)>0)*(maxindex-flex):
                                upper=((maxindex+flex)>=nos)*nos+
                                ((maxindex+flex)<nos)*(maxindex+flex);
70$             BRANCH,         1:
                                If,((m-1)>=lower).and.
                                ((m-1)<=upper),71$,Yes:
                                Else,back,Yes;
71$             BRANCH,         1:
                                If,maxindex==(m-1),back,Yes:
                                Else,72$,Yes;
back            BRANCH,         1:
                                If,end==0,6$,Yes:
                                Else,12$,Yes;
6$              RELEASE:        m-1,1:NEXT(7$);
12$             RELEASE:        m-1,1;
74$             ASSIGN:         end=0;
1$              DUPLICATE:      1,2$;
10$             TALLY:          21,int(time),1;
15$             BRANCH,         1:
                                If,nc(c1)==(quantity-1),16$,Yes:
                                Else,14$,Yes;
16$             ASSIGN:         cycletime=(tnow-warmup)/quantity:
                                col=n*ulc*(tnow-warmup):
                                com=totalcap*umc*(tnow-warmup):
                                coc=((totalcap>nos).and.(totalcap>n))*
                                totalcon*ucc:
                                acoc=coc/(tnow-warmup):
                                h(1)=tavg(11)*(capacity(1)*umc+
                                davg(11)*ulc+acoc):
                                v12=2;
18$             WHILE:          v12<=nos;
19$             ASSIGN:         h(v12)=h(v12-1)+tavg(v12+10)*
                                (capacity(v12)*umc+davg(v12+10)*
                                ulc+acoc):
                                v12=v12+1;
20$             ENDWHILE;
39$             ASSIGN:         v12=1;
21$             WHILE:          v12<=(nos-1);
22$             ASSIGN:         costwip(v12)=h(v12)*(davg(v12+20)+
                                davg(v12+1))*(tnow-warmup)*fraction:
                                totalcostwip=totalcostwip+costwip(v12):
                                v12=v12+1;
23$             ENDWHILE;
76$             ASSIGN:         costwip(v12)=h(v12)*davg(v12+20)*
                                (tnow-warmup)*fraction:
                                totalcostwip=totalcostwip+costwip(v12):
                                tc=com+col+coc+totalcostwip:
                                avgcost=tc/quantity;
75$             WRITE,          file1,"%8.3f %10.2f %10.2f %10.2f
                                %10.2f %10.2f %10.2f\n":
                                cycletime,
                                tc,
                                avgcost,
                                com,
                                col,
                                coc,
```

```
                                totalcostwip;
77$             WRITE,          file2,"%f %f %6.4f %6.4f %6.4f %6.4f
                                %6.4f %6.4f %6.4f %6.4f\n":
                                totalcon,
                                nc(c2),
                                davg(11),
                                davg(21),
                                davg(12),
                                davg(22),
                                davg(13),
                                davg(23),
                                davg(14),
                                davg(24);
78$             WRITE,          file3,"%7.4f %7.4f %7.4f %7.4f %7.4f
                                %7.4f %7.4f %7.4f %8.4f %8.4f %8.4f
                                %8.4f %8.4f\n":
                                davg(1),
                                davg(2),
                                davg(3),
                                davg(4),
                                tavg(1),
                                tavg(2),
                                tavg(3),
                                tavg(4),
                                tavg(11),
                                tavg(12),
                                tavg(13),
                                tavg(14),
                                tavg(21);
14$             COUNT:          c1,1;
13$             DISPOSE;
72$             BRANCH,         1:
                                If,mr(maxindex)<capacity(maxindex),
                                46$,Yes:
                                Else,back,Yes;
46$             BRANCH,         1:
                                If,onpath(maxindex)==0,42$,Yes:
                                Else,back,Yes;
42$             ASSIGN:         mr(m-1)=mr(m-1)-1:
                                transfertime=abs(maxindex-m-1)*
                                unittime;
45$             COUNT:          c2,1;
47$             DUPLICATE:      1,49$:NEXT(back);
49$             ASSIGN:         onpath(maxindex)=onpath(maxindex)+1;
44$             DELAY:          transfertime;
43$             ASSIGN:         onpath(maxindex)=onpath(maxindex)-1:
                                mr(maxindex)=mr(maxindex)+1;
48$             DISPOSE;
61$             ASSIGN:         v3=v3+1:NEXT(55$);
73$             ASSIGN:         end=1:NEXT(control);
24$             CREATE,         1,0.00000000001:,1;
25$             ASSIGN:         mc(c1)=quantity:
                                controlbatchsize=(totalcap>n).and.
                                (totalcap>nos);
26$             DISPOSE;
27$             CREATE,         1,0:,1;
30$             WHILE:          index<=nos;
28$             ASSIGN:         mr(index)=0:
                                totalcap=totalcap+capacity(index):
```

```
                                index=index+1;
31$             ENDWHILE;
32$             ASSIGN:         index=1:
                                total=n;
33$             WHILE:          index<=nos;
34$             WHILE:          (capacity(index)>mr(index)).and.
                                (total>0);
35$             ASSIGN:         mr(index)=mr(index)+1:
                                total=total-1;
36$             ENDWHILE;
38$             ASSIGN:         index=index+1;
37$             ENDWHILE;
29$             DISPOSE;
40$             CREATE,         1,warmup:,1;
41$             ASSIGN:         totalcon=0:NEXT(26$);
```

Appendix C2. The EXP. File of ARENA2.2 for the WW_0 Policy

```
ATTRIBUTES:     1,time:
                2,time1:
                3,time2:
                4,maxindex:
                5,minindex:
                6,transfertime:
                7,lower:
                8,upper:
                9,randoms:
                workload(5,2):
                load(5),:
                end:
                minload(5,2):
                maxload(5,2),;
FILES:          1,file1,"d:\sampletc.txt",Sequential(),Free
                Format,Error,No,Hold:
                2,file2,"d:\sampleuti.txt",Sequential(),Free
                Format,Error,No,Hold:
                3,file3,"d:\samplesure.txt",Sequential(),Free
                Format,Error,No,Hold;
VARIABLES:      ucc,0.01:
                v4:
                tc,:
                v5:
                warmup,10000:
                v12:
                unittime,0:
                v6:
                v7:
                totalcostwip:
                h(5):
                coc:
                v8:
                capacity(5),2,2,2,2:
                onpath(5):
                umc,0.01:
                acoc:
                cycletime:
                control,1:
                ulc,0.01:
                totalcap:
                flex,3:
                avgcost:
                totalcon:
                costwip(5):
                n,4:
                controlbatchsize:
                conwip,20:
                q,0:
                col:
                fraction,0.001:
                com:
                nos,4:
                v1:
                index,1:
```

```
                quantity,20000:
                v2:
                depo1:
                total:
                v3:
                depo2;
SEEDS:          1,,Common:
                2,,Common:
                3,,Common:
                4,,Common:
                5,,Common;
QUEUES:         11,qset,FirstInFirstOut;
RESOURCES:      10,rset,Capacity(1,);


STATIONS:       11,sset;
COUNTERS:       1,c1,,Replicate:
                2,c2,,Replicate:
                3,c3,,Replicate:
                4,c4,,Replicate:
                6,c6,,Replicate;
TALLIES:        21,t1;
DSTATS:         1,nq(1):
                2,nq(2):
                3,nq(3):
                4,nq(4):
                5,nq(5):
                6,nq(6):
                7,nq(7):
                8,nq(8):
                9,nq(9):
                10,nq(10):
                11,mr(1):
                12,mr(2):
                13,mr(3):
                14,mr(4):
                15,mr(5):
                16,mr(6):
                17,mr(7):
                18,mr(8):
                19,mr(9):
                20,mr(10):
                21,nr(1):
                22,nr(2):
                23,nr(3):
                24,nr(4):
                25,nr(5):
                26,nr(6):
                27,nr(7):
                28,nr(8):
                29,nr(9):
                30,nr(10);
OUTPUTS:        1,h(1):
                2,h(2):
                3,h(3):
                4,h(4):
                6,totalcostwip:
                7,costwip(1):
                8,costwip(2):
```

```
                9,costwip(3):
                10,costwip(4);

REPLICATE,      20,0.0,,Yes,Yes,10000;
EXPRESSIONS:    1,processtime(10),expo(5,1),expo(5,2),expo(5,3),
                expo(5,4);
```

Appendix D1. The MOD. File of ARENA2.2 for the WW_K Policy

```
0$              CREATE,         conwip,0.00000000002:,1;
2$              ASSIGN:         m=1:MARK(time);
7$              ROUTE:          0,m+1;
8$              STATION,        2-11;
3$              QUEUE,          m-1:MARK(time1);
4$              SEIZE,          1:
                                m-1,1;
11$             TALLY:          m-1,int(time1),1;
5$              DELAY:          processtime(m-1):MARK(time2);
17$             TALLY:          m+9,int(time2),1;
9$              BRANCH,         1:
                                If,m<(nos+1),control,Yes:
                                Else,73$,Yes;
control         BRANCH,         1:
                                If,nq(m-1)<=q,62$,Yes:
                                Else,back,Yes;
62$             ASSIGN:         v1=1:
                                totalcon=totalcon+1;
50$             WHILE:          v1<=nos;
51$             ASSIGN:         minindex=0:
                                maxindex=0:
                                lower=0:
                                upper=0:
                                transfertime=0:
                                randoms=0:
                                load(v1)=nq(v1):
                                workload(v1,1)=v1:
                                workload(v1,2)=load(v1):
                                minload(v1,1)=workload(v1,1):
                                minload(v1,2)=workload(v1,2):
                                v1=v1+1;
52$             ENDWHILE;
63$             ASSIGN:         v2=1;
53$             WHILE:          v2<=(nos-1);
54$             ASSIGN:         v3=v2+1;
55$             WHILE:          v3<=nos;
60$             BRANCH,         1:
                                If,minload(v2,2)>minload(v3,2),56$,Yes:
                                Else,61$,Yes;
56$             ASSIGN:         depo1=minload(v2,1):
                                depo2=minload(v2,2):
                                minload(v2,1)=minload(v3,1):
                                minload(v2,2)=minload(v3,2):
                                minload(v3,1)=depo1:
                                minload(v3,2)=depo2:
                                v3=v3+1;
57$             ENDWHILE;
58$             ASSIGN:         v2=v2+1;
59$             ENDWHILE;
66$             ASSIGN:         v4=1;
67$             WHILE:          v4<=nos;
68$             ASSIGN:         maxload(v4,1)=minload(nos-v4+1,1):
                                maxload(v4,2)=minload(nos-v4+1,2):
                                v4=v4+1;
69$             ENDWHILE;
```

```
64$             ASSIGN:         v5=1;
65$             ASSIGN:         maxindex=maxload(v5,1):
                                lower=((maxindex-flex)<=0)*1+
                                ((maxindex-flex)>0)*(maxindex-flex):
                                upper=((maxindex+flex)>=nos)*nos+
                                ((maxindex+flex)<nos)*(maxindex+flex);
70$             BRANCH,         1:
                                If,((m-1)>=lower).and.
                                ((m-1)<=upper),71$,Yes:
                                Else,back,Yes;
71$             BRANCH,         1:
                                If,maxindex==(m-1),back,Yes:
                                Else,72$,Yes;
back            BRANCH,         1:
                                If,end==0,6$,Yes:
                                Else,12$,Yes;
6$              RELEASE:        m-1,1:NEXT(7$);
12$             RELEASE:        m-1,1;
74$             ASSIGN:         end=0;
1$              DUPLICATE:      1,2$;
10$             TALLY:          21,int(time),1;
15$             BRANCH,         1:
                                If,nc(c1)==(quantity-1),16$,Yes:
                                Else,14$,Yes;
16$             ASSIGN:         cycletime=(tnow-warmup)/quantity:
                                col=n*ulc*(tnow-warmup):
                                com=totalcap*umc*(tnow-warmup):
                                coc=((totalcap>nos).and.(totalcap>n))*
                                totalcon*ucc:
                                acoc=coc/(tnow-warmup):
                                h(1)=tavg(11)*(capacity(1)*umc+
                                davg(11)*ulc+acoc):
                                v12=2;
18$             WHILE:          v12<=nos;
19$             ASSIGN:         h(v12)=h(v12-1)+tavg(v12+10)*
                                (capacity(v12)*umc+davg(v12+10)*
                                ulc+acoc):
                                v12=v12+1;
20$             ENDWHILE;
39$             ASSIGN:         v12=1;
21$             WHILE:          v12<=(nos-1);
22$             ASSIGN:         costwip(v12)=h(v12)*(davg(v12+20)+
                                davg(v12+1))*(tnow-warmup)*fraction:
                                totalcostwip=totalcostwip+costwip(v12):
                                v12=v12+1;
23$             ENDWHILE;
76$             ASSIGN:         costwip(v12)=h(v12)*davg(v12+20)*
                                (tnow-warmup)*fraction:
                                totalcostwip=totalcostwip+costwip(v12):
                                tc=com+col+coc+totalcostwip:
                                avgcost=tc/quantity;
75$             WRITE,          file1,"%8.3f %10.2f %10.2f %10.2f
                                %10.2f %10.2f %10.2f\n":
                                cycletime,
                                tc,
                                avgcost,
                                com,
                                col,
                                coc,
```

```
                                totalcostwip;
77$             WRITE,          file2,"%f %f %6.4f %6.4f %6.4f %6.4f
                                %6.4f %6.4f %6.4f %6.4f\n":
                                totalcon,
                                nc(c2),
                                davg(11),
                                davg(21),
                                davg(12),
                                davg(22),
                                davg(13),
                                davg(23),
                                davg(14),
                                davg(24);
78$             WRITE,          file3,"%7.4f %7.4f %7.4f %7.4f %7.4f
                                %7.4f %7.4f %7.4f %8.4f %8.4f %8.4f
                                %8.4f %8.4f\n":
                                davg(1),
                                davg(2),
                                davg(3),
                                davg(4),
                                tavg(1),
                                tavg(2),
                                tavg(3),
                                tavg(4),
                                tavg(11),
                                tavg(12),
                                tavg(13),
                                tavg(14),
                                tavg(21);
14$             COUNT:          c1,1;
13$             DISPOSE;
72$             BRANCH,         1:
                                If,mr(maxindex)<capacity(maxindex),
                                46$,Yes:
                                Else,back,Yes;
46$             BRANCH,         1:
                                If,onpath(maxindex)==0,42$,Yes:
                                Else,back,Yes;
42$             ASSIGN:         mr(m-1)=mr(m-1)-1:
                                transfertime=abs(maxindex-m-1)*
                                unittime;
45$             COUNT:          c2,1;
47$             DUPLICATE:      1,49$:NEXT(back);
49$             ASSIGN:         onpath(maxindex)=onpath(maxindex)+1;
44$             DELAY:          transfertime;
43$             ASSIGN:         onpath(maxindex)=onpath(maxindex)-1:
                                mr(maxindex)=mr(maxindex)+1;
48$             DISPOSE;
61$             ASSIGN:         v3=v3+1:NEXT(55$);
73$             ASSIGN:         end=1:NEXT(control);
24$             CREATE,         1,0.00000000001:,1;
25$             ASSIGN:         mc(c1)=quantity:
                                controlbatchsize=(totalcap>n).and.
                                (totalcap>nos);
26$             DISPOSE;
27$             CREATE,         1,0:,1;
30$             WHILE:          index<=nos;
28$             ASSIGN:         mr(index)=0:
                                totalcap=totalcap+capacity(index):
```

```
                                index=index+1;
31$             ENDWHILE;
32$             ASSIGN:         index=1:
                                total=n;
33$             WHILE:          index<=nos;
34$             WHILE:          (capacity(index)>mr(index)).and.
                                (total>0);
35$             ASSIGN:         mr(index)=mr(index)+1:
                                total=total-1;
36$             ENDWHILE;
38$             ASSIGN:         index=index+1;
37$             ENDWHILE;
29$             DISPOSE;
40$             CREATE,         1,warmup:,1;
41$             ASSIGN:         totalcon=0:NEXT(26$);
```

Appendix D2. The EXP. File of ARENA2.2 for the WW_K Policy

```
ATTRIBUTES:    1,time:
               2,time1:
               3,time2:
               4,maxindex:
               5,minindex:
               6,transfertime:
               7,lower:
               8,upper:
               9,randoms:
               workload(5,2):
               load(5),:
               end:
               minload(5,2):
               maxload(5,2),;
FILES:         1,file1,"d:\sampletc.txt",Sequential(),Free
               Format,Error,No,Hold:
               2,file2,"d:\sampleuti.txt",Sequential(),Free
               Format,Error,No,Hold:
               3,file3,"d:\samplesure.txt",Sequential(),Free
               Format,Error,No,Hold;
VARIABLES:     ucc,0.01:
               v4:
               tc,:
               v5:
               warmup,10000:
               v12:
               unittime,0:
               v6:
               v7:
               totalcostwip:
               h(5):
               coc:
               v8:
               capacity(5),2,2,2,2:
               onpath(5):
               umc,0.01:
               acoc:
               cycletime:
               control,1:
               ulc,0.01:
               totalcap:
               flex,3:
               avgcost:
               totalcon:
               costwip(5):
               n,4:
               controlbatchsize:
               conwip,20:
               q,20:
               col:
               fraction,0.001:
               com:
               nos,4:
               v1:
               index,1:
```

```
                quantity,20000:
                v2:
                depo1:
                total:
                v3:
                depo2;
SEEDS:          1,,Common:
                2,,Common:
                3,,Common:
                4,,Common:
                5,,Common;
QUEUES:         11,qset,FirstInFirstOut;
RESOURCES:      10,rset,Capacity(1,);


STATIONS:       11,sset;
COUNTERS:       1,c1,,Replicate:
                2,c2,,Replicate:
                3,c3,,Replicate:
                4,c4,,Replicate:
                6,c6,,Replicate;
TALLIES:        21,t1;
DSTATS:         1,nq(1):
                2,nq(2):
                3,nq(3):
                4,nq(4):
                5,nq(5):
                6,nq(6):
                7,nq(7):
                8,nq(8):
                9,nq(9):
                10,nq(10):
                11,mr(1):
                12,mr(2):
                13,mr(3):
                14,mr(4):
                15,mr(5):
                16,mr(6):
                17,mr(7):
                18,mr(8):
                19,mr(9):
                20,mr(10):
                21,nr(1):
                22,nr(2):
                23,nr(3):
                24,nr(4):
                25,nr(5):
                26,nr(6):
                27,nr(7):
                28,nr(8):
                29,nr(9):
                30,nr(10);
OUTPUTS:        1,h(1):
                2,h(2):
                3,h(3):
                4,h(4):
                6,totalcostwip:
                7,costwip(1):
                8,costwip(2):
```

```
                9,costwip(3):
                10,costwip(4);

REPLICATE,      20,0.0,,Yes,Yes,10000;
EXPRESSIONS:    1,processtime(10),expo(5,1),expo(5,2),expo(5,3),
                expo(5,4);
```

Appendix E1. The MOD. File of ARENA2.2 for the *GPR* Policy

```
0$              CREATE,         conwip,0.00000000001:,1;
1$              ASSIGN:         m=1:MARK(time);
5$              ROUTE:          0.0,m+1;
6$              STATION,        2-5;
2$              QUEUE,          m-1:MARK(time1);
3$              SEIZE,          1:
                                m-1,1;
8$              TALLY:          m-1,int(time1),1;
4$              DELAY:          processtime(m-1):MARK(time2);
40$             TALLY:          m+9,int(time2),1;
46$             ASSIGN:         totalcon=totalcon+1;
7$              BRANCH,         1:
                                If,m<(nos+1),14$,Yes:
                                Else,15$,Yes;
14$             BRANCH,         1:
                                If,capacity(m)>mr(m),11,Yes:
                                Else,13,Yes;

11              BRANCH,         1:
                                If,onpath(m)==0,47$,Yes:
                                Else,13,Yes;
47$             ASSIGN:         mr(m-1)=mr(m-1)-1:
                                transfertime=unittime;
50$             COUNT:          c2,1;
51$             DUPLICATE:      1,53$:NEXT(13);
13              RELEASE:        m-1,1:NEXT(5$);
53$             ASSIGN:         onpath(m)=onpath(m)+1;
49$             DELAY:          transfertime;
48$             ASSIGN:         onpath(m)=onpath(m)-1:
                                mr(m)=mr(m)+1;
52$             DISPOSE;
15$             BRANCH,         1:
                                If,capacity(1)>mr(1),12,Yes:
                                Else,14,Yes;
12              BRANCH,         1:
                                If,onpath(m)==0,54$,Yes:
                                Else,14,Yes;
54$             ASSIGN:         mr(m-1)=mr(m-1)-1:
                                transfertime=abs(m-2)*unittime;
57$             COUNT:          c2,1;
58$             DUPLICATE:      1,60$:NEXT(14);
14              RELEASE:        m-1,1;
23$             DUPLICATE:      1,1$;
24$             TALLY:          21,int(time),1;
27$             BRANCH,         1:
                                If,nc(c1)==(quantity-1),29$,Yes:
                                Else,26$,Yes;
29$             ASSIGN:         cycletime=(tnow-warmup)/quantity:
                                col=n*ulc*(tnow-warmup):
                                com=totalcap*umc*(tnow-warmup):
                                coc=((totalcap>nos).and.(totalcap>n))*
                                totalcon*ucc:
                                acoc=coc/(tnow-warmup):
                                h(1)=tavg(11)*(capacity(1)*umc+
                                davg(11)*ulc):
```

```
                                v12=2;
30$             WHILE:          v12<=nos;
31$             ASSIGN:         h(v12)=h(v12-1)+tavg(v12+10)*
                                (capacity(v12)*umc+davg(v12+10)*ulc):
                                v12=v12+1;
32$             ENDWHILE;
37$             ASSIGN:         v12=1;
33$             WHILE:          v12<=(nos-1);
34$             ASSIGN:         costwip(v12)=h(v12)*(davg(v12+20)+
                                davg(v12+1))*(tnow-warmup)*fraction:
                                totalcostwip=totalcostwip+costwip(v12):
                                v12=v12+1;
35$             ENDWHILE;
36$             ASSIGN:         costwip(v12)=h(v12)*davg(v12+20)*
                                (tnow-warmup)*fraction:
                                totalcostwip=totalcostwip+costwip(v12):
                                tc=com+col+totalcostwip:
                                avgcost=tc/quantity;
28$             WRITE,          file1,"%8.3f %10.2f %10.2f %10.2f
                                %10.2f %10.2f %10.2f\n":
                                cycletime,
                                tc,
                                avgcost,
                                com,
                                col,
                                coc,
                                totalcostwip;
38$             WRITE,          file2,"%f %f %6.4f %6.4f %6.4f %6.4f
                                %6.4f %6.4f %6.4f %6.4f\n":
                                totalcon,
                                nc(c2),
                                davg(11),
                                davg(21),
                                davg(12),
                                davg(22),
                                davg(13),
                                davg(23),
                                davg(14),
                                davg(24);
39$             WRITE,          file3,"%7.4f %7.4f %7.4f %7.4f %7.4f
                                %7.4f %7.4f %7.4f %8.4f %8.4f %8.4f
                                %8.4f %8.4f\n":
                                davg(1),
                                davg(2),
                                davg(3),
                                davg(4),
                                tavg(1),
                                tavg(2),
                                tavg(3),
                                tavg(4),
                                tavg(11),
                                tavg(12),
                                tavg(13),
                                tavg(14),
                                tavg(21);
26$             COUNT:          c1,1;
25$             DISPOSE;
60$             ASSIGN:         onpath(m)=onpath(m)+1;
56$             DELAY:          transfertime;
```

```
55$          ASSIGN:        onpath(m)=onpath(m)-1:
                            mr(1)=mr(1)+1;
59$          DISPOSE;
9$           CREATE,        1,0:,1;
12$          WHILE:         index<=nos;
10$          ASSIGN:        mr(index)=0:
                            capacity(index)=capacity(index)*
                            (index<=nos):
                            totalcap=totalcap+capacity(index):
                            index=index+1;
13$          ENDWHILE;
16$          ASSIGN:        index=1:
                            total=n;
17$          WHILE:         index<=nos;
18$          WHILE:         (capacity(index)>mr(index)).and.
                            (total>0);
19$          ASSIGN:        mr(index)=mr(index)+1:
                            total=total-1;
20$          ENDWHILE;
22$          ASSIGN:        index=index+1;
21$          ENDWHILE;
11$          DISPOSE;
41$          CREATE,        1,0.00000000001:,1;
42$          ASSIGN:        mc(c1)=quantity;
43$          DISPOSE;
44$          CREATE,        1,warmup:,1;
45$          ASSIGN:        totalcon=0:NEXT(43$);
```

Appendix E2. The EXP. File of ARENA2.2 for the *GPR* Policy

```
ATTRIBUTES:    1,time:
               2,time1:
               3,time2:
               4,transfertime;
FILES:         1,file1,"d:\sampletc.txt",Sequential(),
               Free Format,Error,No,Hold:
               2,file2,"d:\sampleuti.txt",Sequential(),
               Free Format,Error,No,Hold:
               3,file3,"d:\samplesure.txt",Sequential(),
               Free Format,Error,No,Hold;
VARIABLES:     tc,:
               ucc,0.01:
               unittime,0:
               v12:
               warmup,10000:
               coc:
               h(5):
               totalcostwip:
               capacity(5),2,2,2,2:
               onpath(5):
               cycletime:
               acoc:
               umc,0.01:
               totalcap:
               ulc,0.01:
               totalcon:
               avgcost:
               n,4:
               costwip(5):
               conwip,20:
               controlbatchsize:
               col:
               nos,4:
               index,1:
               com:
               fraction,0.001:
               total:
               quantity,20000;
SEEDS:         1,,Common:
               2,,Common:
               3,,Common:
               4,,Common:
               5,,Common;
QUEUES:        11,qset,FirstInFirstOut;
RESOURCES:     10,rset,Capacity(1,);
STATIONS:      11,sset;
COUNTERS:      1,c1,,Replicate:
               2,c2,,Replicate:
               3,c3,,Replicate:
               4,c4,,Replicate:
               6,c6,,Replicate;
TALLIES:       21,t1;
DSTATS:        1,nq(1):
               2,nq(2):
               3,nq(3):
```

```
                4,nq(4):
                5,nq(5):
                6,nq(6):
                7,nq(7):
                8,nq(8):
                9,nq(9):
                10,nq(10):
                11,mr(1):
                12,mr(2):
                13,mr(3):
                14,mr(4):
                15,mr(5):
                16,mr(6):
                17,mr(7):
                18,mr(8):
                19,mr(9):
                20,mr(10):
                21,nr(1):
                22,nr(2):
                23,nr(3):
                24,nr(4):
                25,nr(5):
                26,nr(6):
                27,nr(7):
                28,nr(8):
                29,nr(9):
                30,nr(10);
OUTPUTS:        1,h(1):
                2,h(2):
                3,h(3):
                4,h(4):
                6,totalcostwip:
                7,costwip(1):
                8,costwip(2):
                9,costwip(3):
                10,costwip(4);
REPLICATE,      20,0.0,,Yes,Yes,10000;
EXPRESSIONS:    1,processtime(10),expo(5,1),expo(5,2),expo(5,3),
                expo(5,4);
```

Appendix F1. The MOD. File of ARENA2.2 for the *PC* Policy

```
0$              CREATE,         conwip,0.00000000002:,1;
2$              ASSIGN:         m=1:MARK(time);
7$              ROUTE:          0,m+1;
8$              STATION,        2-11;
3$              QUEUE,          m-1:MARK(time1);
4$              SEIZE,          1:
                                m-1,1;
11$             TALLY:          m-1,int(time1),1;
5$              DELAY:          processtime(m-1):MARK(time2);
17$             TALLY:          m+9,int(time2),1;
9$              BRANCH,         1:
                                If,m<(nos+1),control,Yes:
                                Else,73$,Yes;
control         BRANCH,         1:
                                If,nq(m-1)<=q,62$,Yes:
                                Else,back,Yes;
62$             ASSIGN:         v1=1:
                                totalcon=totalcon+1;
50$             WHILE:          v1<=nos;
51$             ASSIGN:         minindex=0:
                                maxindex=0:
                                lower=0:
                                upper=0:
                                transfertime=0:
                                randoms=0:
                                load(v1)=nq(v1):
                                workload(v1,1)=v1:
                                workload(v1,2)=load(v1):
                                minload(v1,1)=workload(v1,1):
                                minload(v1,2)=workload(v1,2):
                                v1=v1+1;
52$             ENDWHILE;
63$             ASSIGN:         v2=1;
53$             WHILE:          v2<=(nos-1);
54$             ASSIGN:         v3=v2+1;
55$             WHILE:          v3<=nos;
60$             BRANCH,         1:
                                If,minload(v2,2)>minload(v3,2),56$,Yes:
                                Else,61$,Yes;
56$             ASSIGN:         depo1=minload(v2,1):
                                depo2=minload(v2,2):
                                minload(v2,1)=minload(v3,1):
                                minload(v2,2)=minload(v3,2):
                                minload(v3,1)=depo1:
                                minload(v3,2)=depo2:
                                v3=v3+1;
57$             ENDWHILE;
58$             ASSIGN:         v2=v2+1;
59$             ENDWHILE;
66$             ASSIGN:         v4=1;
67$             WHILE:          v4<=nos;
68$             ASSIGN:         maxload(v4,1)=minload(nos-v4+1,1):
                                maxload(v4,2)=minload(nos-v4+1,2):
                                v4=v4+1;
69$             ENDWHILE;
```

```
64$             ASSIGN:         v5=1;
65$             ASSIGN:         maxindex=maxload(v5,1):
                                lower=((maxindex-flex)<=0)*1+
                                ((maxindex-flex)>0)*(maxindex-flex):
                                upper=((maxindex+flex)>=nos)*nos+
                                ((maxindex+flex)<nos)*(maxindex+flex);
70$             BRANCH,         1:
                                If,((m-1)>=lower).and.
                                ((m-1)<=upper),71$,Yes:
                                Else,back,Yes;
71$             BRANCH,         1:
                                If,maxindex==(m-1),back,Yes:
                                Else,72$,Yes;
back            BRANCH,         1:
                                If,end==0,6$,Yes:
                                Else,12$,Yes;
6$              RELEASE:        m-1,1:NEXT(7$);
12$             RELEASE:        m-1,1;
74$             ASSIGN:         end=0;
1$              DUPLICATE:      1,2$;
10$             TALLY:          21,int(time),1;
15$             BRANCH,         1:
                                If,nc(c1)==(quantity-1),16$,Yes:
                                Else,14$,Yes;
16$             ASSIGN:         cycletime=(tnow-warmup)/quantity:
                                col=n*ulc*(tnow-warmup):
                                com=totalcap*umc*(tnow-warmup):
                                coc=((totalcap>nos).and.(totalcap>n))*
                                totalcon*ucc:
                                acoc=coc/(tnow-warmup):
                                h(1)=tavg(11)*(capacity(1)*
                                umc+davg(11)*ulc+acoc):
                                v12=2;
18$             WHILE:          v12<=nos;
19$             ASSIGN:         h(v12)=h(v12-1)+tavg(v12+10)*
                                (capacity(v12)*umc+davg(v12+10)*
                                ulc+acoc):
                                v12=v12+1;
20$             ENDWHILE;
39$             ASSIGN:         v12=1;
21$             WHILE:          v12<=(nos-1);
22$             ASSIGN:         costwip(v12)=h(v12)*(davg(v12+20)+
                                davg(v12+1))*(tnow-warmup)*fraction:
                                totalcostwip=totalcostwip+costwip(v12):
                                v12=v12+1;
23$             ENDWHILE;
76$             ASSIGN:         costwip(v12)=h(v12)*davg(v12+20)*
                                (tnow-warmup)*fraction:
                                totalcostwip=totalcostwip+costwip(v12):
                                tc=com+col+coc+totalcostwip:
                                avgcost=tc/quantity;
75$             WRITE,          file1,"%8.3f %10.2f %10.2f %10.2f
                                %10.2f %10.2f %10.2f\n":
                                cycletime,
                                tc,
                                avgcost,
                                com,
                                col,
                                coc,
```

```
                                totalcostwip;
77$             WRITE,          file2,"%f %f %6.4f %6.4f %6.4f %6.4f
                                %6.4f %6.4f %6.4f %6.4f\n":
                                totalcon,
                                nc(c2),
                                davg(11),
                                davg(21),
                                davg(12),
                                davg(22),
                                davg(13),
                                davg(23),
                                davg(14),
                                davg(24);
78$             WRITE,          file3,"%7.4f %7.4f %7.4f %7.4f %7.4f
                                %7.4f %7.4f %7.4f %8.4f %8.4f %8.4f
                                %8.4f %8.4f\n":
                                davg(1),
                                davg(2),
                                davg(3),
                                davg(4),
                                tavg(1),
                                tavg(2),
                                tavg(3),
                                tavg(4),
                                tavg(11),
                                tavg(12),
                                tavg(13),
                                tavg(14),
                                tavg(21);
14$             COUNT:          c1,1;
13$             DISPOSE;
72$             BRANCH,         1:
                                if,mr(maxindex)<capacity(maxindex),
                                46$,Yes:
                                Else,back,Yes;
46$             BRANCH,         1:
                                If,onpath(maxindex)==0,42$,Yes:
                                Else,back,Yes;
42$             ASSIGN:         mr(m-1)=mr(m-1)-1:
                                transfertime=abs(maxindex-m-1)*
                                unittime;
45$             COUNT:          c2,1;
47$             DUPLICATE:      1,49$:NEXT(back);
49$             ASSIGN:         onpath(maxindex)=onpath(maxindex)+1;
44$             DELAY:          transfertime;
43$             ASSIGN:         onpath(maxindex)=onpath(maxindex)-1:
                                mr(maxindex)=mr(maxindex)+1;
48$             DISPOSE;
61$             ASSIGN:         v3=v3+1:NEXT(55$);
73$             ASSIGN:         end=1:NEXT(control);
24$             CREATE,         1,0.00000000001:,1;
25$             ASSIGN:         mc(c1)=quantity:
                                controlbatchsize=(totalcap>n).and.
                                (totalcap>nos);
26$             DISPOSE;
27$             CREATE,         1,0:,1;
30$             WHILE:          index<=nos;
28$             ASSIGN:         mr(index)=0:
                                totalcap=totalcap+capacity(index):
```

```
                                index=index+1;
31$             ENDWHILE;
32$             ASSIGN:         index=1:
                                total=n;
33$             WHILE:          index<=nos;
34$             WHILE:          (capacity(index)>mr(index)).and.
                                (total>0);
35$             ASSIGN:         mr(index)=mr(index)+1:
                                total=total-1;
36$             ENDWHILE;
38$             ASSIGN:         index=index+1;
37$             ENDWHILE;
29$             DISPOSE;
40$             CREATE,         1,warmup:,1;
41$             ASSIGN:         totalcon=0:NEXT(26$);
```

Appendix F2. The EXP. File of ARENA2.2 for the *PC* Policy

```
ATTRIBUTES:     1,time:
                2,time1:
                3,time2:
                4,maxindex:
                5,minindex:
                6,transfertime:
                7,lower:
                8,upper:
                9,randoms:
                workload(5,2):
                load(5),:
                minload(5,2):
                maxload(5,2),;
FILES:          1,file1,"d:\sampletc.txt",Sequential(),
                Free Format,Error,No,Hold:
                2,file2,"d:\sampleuti.txt",Sequential(),
                Free Format,Error,No,Hold:
                3,file3,"d:\samplesure.txt",Sequential(),
                Free Format,Error,No,Hold;
VARIABLES:      tc,:
                v4:
                ucc,0.01:
                v5:
                v6:
                unittime,0:
                v12:
                warmup,10000:
                v7:
                v8:
                coc:
                h(5):
                totalcostwip:
                onpath(5):
                capacity(5),2,2,2,2:
                control,1:
                cycletime:
                acoc:
                umc,0.01:
                flex,3:
                totalcap:
                ulc,0.01:
                totalcon:
                avgcost:
                n,4:
                costwip(5):
                conwip,20:
                controlbatchsize:
                col:
                index,1:
                v1:
                nos,4:
                com:
                fraction,0.001:
                total:
                depo1:
```

```
                v2:
                quantity,20000:
                depo2:
                v3;
SEEDS:          1,,Common:
                2,,Common:
                3,,Common:
                4,,Common:
                5,,Common;
QUEUES:         11,qset,FirstInFirstOut;
RESOURCES:      10,rset,Capacity(1,);
STATIONS:       11,sset;
COUNTERS:       1,c1,,Replicate:
                2,c2,,Replicate:
                3,c3,,Replicate:
                4,c4,,Replicate:
                6,c6,,Replicate;
TALLIES:        21,t1;
DSTATS:         1,nq(1):
                2,nq(2):
                3,nq(3):
                4,nq(4):
                5,nq(5):
                6,nq(6):
                7,nq(7):
                8,nq(8):
                9,nq(9):
                10,nq(10):
                11,mr(1):
                12,mr(2):
                13,mr(3):
                14,mr(4):
                15,mr(5):
                16,mr(6):
                17,mr(7):
                18,mr(8):
                19,mr(9):
                20,mr(10):
                21,nr(1):
                22,nr(2):
                23,nr(3):
                24,nr(4):
                25,nr(5):
                26,nr(6):
                27,nr(7):
                28,nr(8):
                29,nr(9):
                30,nr(10);
OUTPUTS:        1,h(1):
                2,h(2):
                3,h(3):
                6,totalcostwip:
                7,costwip(1):
                8,costwip(2):
                9,costwip(3):
                10,costwip(4);
REPLICATE,      20,0.0,,Yes,Yes,10000;
EXPRESSIONS:    1,processtime(10),expo(5,1),expo(5,2),expo(5,3),
                expo(5,4);
```