SERIAL PRODUCTION LINES UNDER PULL ENVIRONMENT

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Erdem Eskigün

September, 1996

SERIAL PRODUCTION LINES UNDER PULL ENVIRONMENT

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Erdem Eskigün

September, 1996

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Cemal Dinçer (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Selim Aktürk

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Erdal Erel

Approved for the Institute of Engineering and Sciences:

Prof. Mehmet Baray
Director of Institute of Engineering and Science

ABSTRACT

SERIAL PRODUCTION LINES UNDER PULL ENVIRONMENT

Erdem Eskigün
M.S. in Industrial Engineering
Supervisor: Assoc. Prof. Cemal Dinçer
September, 1996

The aim of this thesis is to evaluate the performance of transfer lines under pull environment. In this study, we have modelled a two-stage transfer line separated by finite buffers by using Markov-Chain representation. Experiments with different parameters were carried out to evaluate the different performance measures of the system. A two-stage simulation model was also constructed by using the package PromodelPC 2.0. Since two-stage models are limited to analyze most practical systems in manufacturing, we have extended it to an N-stage model. From the experiments performed for both two-stage and N-stage models, we have observed that we can decrease the in-process inventory substantially by just exchanging the positions of the machines in the system. Assembly-disassembly systems under pull environment have also been modelled by using simulation.

Key words: Pull Systems, Transfer Lines, Assembly-Disassembly Systems, Simulation

ÖZET

ÇEKME ORTAMLARINDA SERİ ÜRETİM SİSTEMLERİ

Erdem Eskigün
Endüstri Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Doç. Dr. Cemal Dinçer
Eylül, 1996

Bu tez çalışmasının amacı çekme ortamında seri üretim sistemlerinin performanslarını incelemektir. Bu çalışmada, kapasiteli ara stoklarla ayrılmış iki aşamalı seri üretim sistemi Markov-Zinciri gösterimi kullanılarak analitik olarak modellendi. Sistemin farklı performans ölçülerini görmek için değişik parametreler kullanılarak deneyler yapıldı. Ayrıca, bu sistemin simülasyon modeli, PromodelPC paketi kullanılarak oluşturuldu. İki aşamalı modeller pratik hayatta çok küçük modeller olarak kabul edildiği için, N aşamalı bir modele ihtiyaç duyuldu. Hem iki aşamalı, hem de N aşamalı modeller için yapılan deneyler sonunda görüldü ki, sistemdeki toplam ara stok, sadece makinaların sıraları değiştirilerek önemli seviyede azaltılabilir. Ayrıca diğer bir çalışma olarak, çekme ortamında montaj-demontaj sistemleri simülasyon yaklaşımı kullanılarak modellendi.

Anahtar sözcükler: Çekme Sistemleri, Seri Üretim Sistemleri, Montaj-Demontaj Sistemleri, Benzetim

To my family and to my all friends

ACKNOWLEDGEMENT

I am indebted to Assoc. Prof. Cemal Dinçer for his invaluable guidance, encouragement and above all, for the enthusiasm which he inspired on me during this study.

I am also indebted to Assist. Prof. Selim Aktürk and Assoc. Prof. Erdal Erel for showing keen interest to the subject matter and accepting to read and review this thesis.

I would like to thank to Mehmet Bayındır, Aydın Selçuk, Aziz İbrahim Sağlam, Nasuhi Yurt, Mehmet Orhan, Aliseydi Toy, Ali Bıçak, Ertuğrul Uysal, Ali İhsan Çağlayan and Ali Resul Usül for their friendship, valuable comments and support.

I would also like to thank to my classmates M. Bayram Yıldırım, Rıza Ceran, Murat Aksu, Siraceddin Önen, Abdullah Daşcı and Mustafa Karakul for their friendship, help and patience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Just-in-Time (JIT) is a philosophy grown out of the Japanese approach to organizing manufacturing operations. Although originally intended as a means of moving material through a plant, the core of JIT philosophy is to eliminate waste. Waste includes poorly timed movement of material through the plant, defective parts and poor scheduling of parts deliveries.

Inventory and material flow systems are classified as either push or pull systems. A push system is one in which decisions concerning how material will flow through the system are made centrally. Based on these decisions, material is produced and pushed to the next level of the system. A typical push system is Material Requirement Planning (MRP). In JIT approach, each manufacturing work center pulls the required parts from the supplying work center or supplier when parts are required. This procedure ensures that the only work-in-process (WIP) inventory is the one required for current manufacturing.

The layouts of the manufacturing systems are designed in different ways in different environments. Some of them are batch processing, job-shop, mass production lines or transfer lines, group technology etc.

Transfer lines are very important in mass production systems. In a transfer line, parts or workpieces enter the first machine and an operation takes place.

The parts are then moved to the next machine if it is available or to the buffer storage in between machines 1 and 2 if space is available. The parts are then processed on machine 2, then on machine 3, and so forth. Due to high capital investment needed for a transfer line, great care should be taken in its design so as to optimize the system output. One way to increase the system output is to provide buffer storage between successive stages of transfer lines. Therefore, there has been considerable interest shown in modeling and analyzing the effect of buffer storage on the performance of transfer lines.

1.1 Outline of the Thesis

The aim of this thesis is to see the performance of transfer lines in the pull environment. Although the literature dealing with transfer lines is abundant, most of them consider the models under push environment. We reconsider the transfer lines under pull policy.

First, the description and detailed explanations of transfer lines and pull systems are given in Chapter 2. The literature review of both transfer lines and pull systems are also provided in that chapter. Chapter 3 is devoted to analytical modeling of pull systems. In that chapter, two-stage models both with buffer size of one and with general buffer sizes are constructed and solved by using different techniques. The simulation approach to model two-stage system is given in Chapter 4. A validity analysis between two-stage analytical model and two-stage simulation model is also performed in that chapter. This two-stage model is extended into 20-stage model to see the behaviour of N-stage models. In Chapter 5, an extension model of transfer lines called assembly-disassembly systems under pull environment is constructed. The algorithm of the simulation model is also given in that chapter.

Chapter 2

Review of Serial Production Lines

In this chapter, transfer lines and pull systems are discussed. The most well known method of the pull systems, Kanban method, is introduced as well. The literature review of both transfer lines and pull systems are also given in this chapter.

2.1 Transfer Lines

A transfer line is a manufacturing system which is usually used in mass production systems. It is defined as a collection of linear machines (M_1, M_2,.....,M_k) separated by buffer storages (B_1, B_2 , ..., B_{k-1}). Parts flow from outside the system to M_1 then to B1 then to M2, and so forth until it reaches M_k after which it leaves. Figure 2.1 depicts a four-machine transfer line separated by three buffers. The rectangles represent machines and the ovals represent buffers.

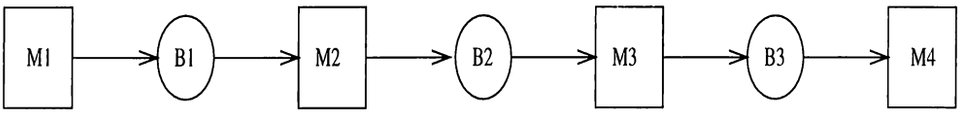

Figure 2.1: Four-Machine Transfer Line

Machines' behaviours are not perfectly predictable. All machines require unpredictable, or predictable but not constant amount of time to complete their operations. Furthermore, all machines eventually fail and their repair times and the time between failures are not also perfectly predictable. This irregularity has the potential for disrupting the operations of not only adjacent machines but also machines further away. Therefore, buffer storages are used to reduce this potential.

When a failure occurs, or when a machine takes an exceptionally long time to complete an operation, the level in the adjacent upstream storage may rise. If the disruption persist long enough that storage fills up and forces the upstream machine to stop processing parts. Such a forced-down machine is said to be blocked. Similarly, the level of adjacent downstream storage may fall during a failure, as the downstream machines drain its contents. If the failure persists long enough, the adjacent downstream storage is depleted and the downstream machine stops processing parts. Such a forced down machine is called starved or idle.

Interstage buffer storages partially decouple adjacent machines. Since machine failures are inevitable, buffer storages are used to reduce the effects of the failures of machines on the operations of other machines. When the buffer storages are full or empty, the decoupling effect can not take place. When the downstream buffer is full, upstream machine can not send a part to the buffer and when the upstream buffer is empty, downstrem machine can not take a part from the buffer when it needs to process. By supplying high capacity buffer storages, probability of storages being empty or full are reduced. However, the disadvantage of high capacity buffer storages is the amount of average in-process inventory. As buffer sizes increase, more work-in process (WIP) inventory accumulates between processing stages.

Work-in process, or WIP is undesirable. Gershwin[10] states the drawbacks of WIP inventory as :

i) *it costs money to create, but as long as it sits in buffers, it generates no revenue.*

ii) *the average lead time is proportional to the average amount of inventory*

iii) *inventory in a factory or a warehouse are vulnerable to damage or shrinkage. The more items and the more time they spend, the more vulnerable they are.*

iv) *the space and the material handling equipment needed for inventory costs money.*

An in-process inventory can affect the production rate if the line is close to balanced. If the line is not balanced, even infinite buffers will have very small effects on the production rate. The size of the buffers should also be calculated as the number of parts one nearby machine could make while another nearby machine is down[10].

The factors that limit the production rate of a serial line are unreliable machines and lack of synchronization. Unreliable machines cause the line to stop the production. All other machines wait the unreliable machines to be repaired. To synchronize the system, frequency and duration of unsynchronized events may be reduced or in-process buffers may be installed. The place and the size of an in-process buffer is a design problem and it is very important to increase the system performance and to decrease the number of in-process inventory.

Another feature of the transfer lines is the variability. There are few papers dealing with variability of production lines[10]. Almost all of them calculate the steady-state performance measures. However, variability of the system is very important. A stable system with low production rate may be preferred to an unstable system with high production rate. In other words, the less variability in the system, the more desirable the system is.

A common technique while modeling the transfer lines is **Markov-Chain** representation. In this technique, all states of the system and their transition probabilities are defined. While obtaining the steady-state probabilities, different methods are used. Another technique is **decomposition** method. In this method, line is decomposed into two-stages which are the small representations of the exact model. When all these methods are ineffective in modeling the transfer lines, **simulation** is used. Simulation is a powerful technique used in modeling and also in experiments done for different parameters of the system.

2.1.1 Difficulties of the Markov-Chain Approach

Markov-Chain models of the transfer lines are difficult to treat because of their large state spaces and their in-decomposability.

The Markov-Chain representation of a k-machine line with $k-1$ buffers has M distinct states [10], where

$$M = 2^k \prod_{i=0}^{k-1} N_i + 1$$

and N_i is the size of buffer storage B_i.

These models are not decomposable that is, portions of the system can not be treated as though they are isolated from other portions. Approximate decompositions are derived, but no exact decomposition exists.

2.2 Inventory Pull System

The traditional method of moving material is to push the required material down onto the shop floor, according to the production plan. A Just in Time approach converts traditional shop floor control from a push system to a pull system for controlling inventories of materials and subassemblies.

In the push system, the finished-products (and customer orders) create computer planned orders within MRP, calculated to meet the requirement of the forecasts, orders, and associated subassemblies and components. These orders are converted into firm, planned orders that authorize the purchase of components and determine planning capacity requirement. When the order is loaded to the shop floor it becomes a work order. At this point, components are allocated from inventory to meet the needs of this new work order. When components move onto the shop floor, a pick list is printed, parts are picked from component stores, and all parts are moved to the shop floor ready to be manufactured. This traditional approach is called a "push" system, because materials are pushed onto the shop floor based on the production plan as it is seen from Figure 2.2.

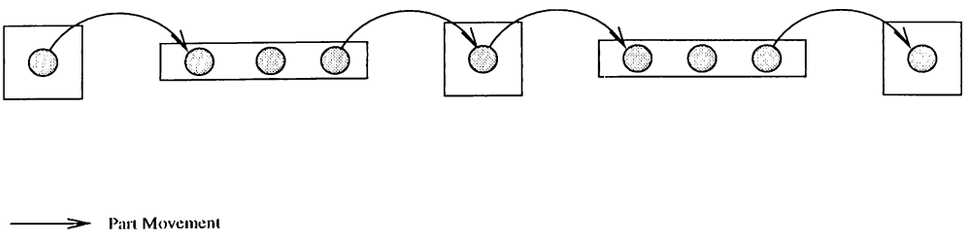

Figure 2.2: Push Production System

A Just in Time approach uses a "pull" system. A customer order appears at the end of the production line, and required materials are issued to the shop floor as the operator request. The requesting of materials can be done in a number of different ways, including the famous Kanban Method.

2.2.1 The Pull System

A pure pull system is initiated by a customer order. The supervisor on the final assembly line or at the final manufacturing station will "pull" the subassemblies required to complete the order from upstream workcells. The upstream workcells will pull subassemblies and components from lower level cells, the warehouse or suppliers (see Figure 2.3). For pull systems to work

cleanly and precisely, very short manufacturing times and uniform demands to the factory are required. In practice, this approach needs to be modified when there is a wide variety of products or when production cycle times must be longer.

One widely used practice is to maintain small inventories of subassemblies and components on the shop floor in front of a work center. Subassemblies are stored in standard containers that always contain a specific quantity. When a downstream work center requires a subassembly, the container is moved from the producing cell to the user cell. The user cell will return the empty container to the producing cell, which then makes more of the subassembly to fill the returned container. Theoretically a container quantity will be equivalent to the amount used by the downstream work center; in reality, a little safety stock is also provided.

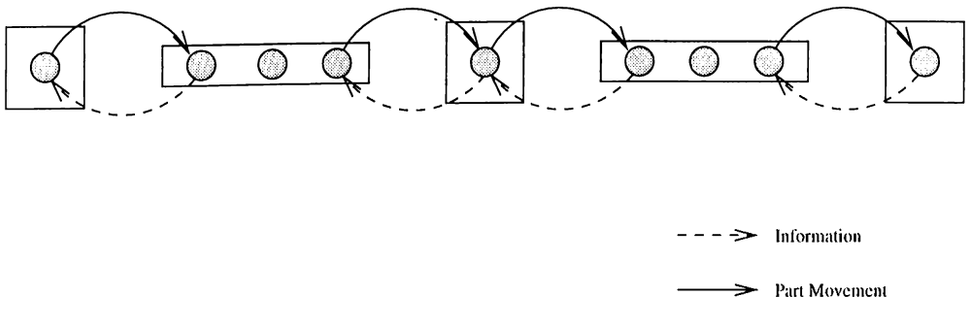

Figure 2.3: Pull Production System

2.2.2 Kanbans and Signals

A number of methods can signal the movement of parts and subassemblies to a downstream work center. The most well known method is the use of Kanban cards. This method, developed originally by Toyota as a part of its quest for efficient manufacturing, requires the use of cards that are passed from one work cell to another. The word Kanban is Japanese for card or ticket.

There are two varieties of the Kanban method: one card or two card. The two-card method has a production card and a move card. The move card authorizes the transfer of one container of components from the supplying work center to the using work center. The production card authorizes the supplying work center to make another container of components.

The one-card system can be used in less complex manufacturing environments such as when there is one type of product to manufacture in a serial production line. In this case the Kanban card acts as both a move card and a production card. The Kanban travels with a container from the supplying work center to the using work center.

When more components are required, the Kanban is sent back to the supplying work center; at that point, the Kanban authorizes both the moving of material and the production of a new container quantity.

An interesting and useful phenomenon of a Kanban system is that WIP inventory level can be directly controlled through Kanbans. Work-in-process inventory is directly related to the number of Kanbans on the floor.

Some more sophisticated companies make use of electronic Kanbans. This technique does not require any cards as such because any information required to pull material from the prior cell or the warehouse is provided electronically. The operator has a small computer terminal in the cell, enters the part number of the component or subassembly required, and triggers the supplying location to move the material.

2.2.3 Advantages of Pull Systems

Advantages of Pull systems can be summarized from Maskell [18] as:

i) One benefit of a demand-pull method is that shop floor systems are very much simplified. Dispatch lists, input/output reports, and detailed production activity reports are no longer required to control the daily

production process. On the other hand, the medium and longer term planning underlying the simplicity of many Just in Time techniques includes a great deal of planning and attention to detail. The availability of sufficient production resources, the purchase order of raw material and components, production quality, shop floor layout and container quantities must all be right for a pull system to be effective.

ii) Another result of using demand pull as the primary production control method is that more products are produced. There is an intangible but very strong relationship between excess WIP and poor productivity. When the shop floor is full of large batches of semi completed assemblies, operators may be confused as to what they should be working on and when.

iii) The big advantage of a pull system is that only the material required to make products is moved onto the shop floor. There is no excess or unneeded inventory; no waste. Pulling material answers the fundamental questions of when to manufacture a subassembly or buy a component and how much to make. Pulling does not rely on forecasts, run times or batch quantities.

iv) Despite the fact that some work centers are operating at less than full rate, the amount of product produced is higher than when each workcenter is loaded 100% through a push system. This rate is sometimes spoken as a "drum beat" of manufacturing because material flows through the factory at a constant velocity; there is a predictable rhythm to the process resulting in a larger volume of finished products.

2.2.4 Disadvantages of Pull Systems

In addition to requiring short lead times and predictable (stable) demand, a pull system requires that there are no conflicting priorities within the production schedule. Just-in-Time manufacturers employ a number of

techniques to alleviate priority conflicts. A key to the resolution of conflicts is flexibility of people and flexibility of machinery.

In the short term, the problem of priorities has to be overcome by keeping additional inventory of assemblies produced by the affected work center.

Shortly :

i) A Pull system requires short lead times and small production lots, which require fast setups.

ii) A Pull system requires flexibility of people and equipment so that changing requirements can be accommodated.

iii) A Pull system can theoretically be used with any kind of shop floor layout, but is most effective in conjunction with a cellular type of layout.

iv) A Pull system requires total quality assuarance because a quality failure at one cell can stop the entire production process.

v) A Pull system requires preventive maintenance rather than the maintenance after machine break down.

2.3 Literature Review

2.3.1 Automatic Transfer Lines

Automatic transfer lines play an important role in mass production systems. An automatic transfer line is defined as a number of automated machines integrated into one system by a common automated transfer mechanism. Workpieces (parts) enter the system through the first machine and after being processed by all the machines in a sequential order they leave the system through the last machine. Transfer of parts from one machine to the next

are synchronized to take place at the same time. In a synchronized transfer line, the failure of a single machine can cause the stoppage of the whole system resulting in the loss of production. This problem, however, can partially be solved if buffer storages are provided between the machines.

In the literature, there are lots of works of transfer lines. They are different in the assumptions of models, such as reliable or unreliable machines; machines with finite buffers or machines with infinite buffers; distributions of the downtimes, cycle times, repair times etc. However, they have a common feature of that all systems are modeled under the Push policy. There are few papers dealing with automatic transfer lines under Pull policy.

The two-stage transfer lines are solved by Markov-Chain easily. When the number of stages is increased, the number of states increases greatly and the solution with Markov Chain becomes very difficult. Some approximation techniques to solve N-stage transfer lines are developed and reported in the literature.

Shantikumar and Tien[26] have analyzed two-stage synchronized automatic transfer line where workpiece transfer at all stages is synchronized to occur at the same time. The model is represented as Markov-Chain. However, the method used to solve this Markov-Chain is different from solving the set of equations

$$xP = x \tag{2.1}$$

$$xe = 1 \tag{2.2}$$

where x is steady-state probability vector of Markov-Chain. The algorithm to solve the Markov-Chain representation of the model is given in the paper. It is also suggested that the idea of providing buffer storage is to minimize the second stage idleness and to reduce the probability of first stage being blocked.

Analysis of some extensions of this model using an efficient interpolation approximation is discussed in Ignall and Silver[14] and using simulation is described by Ho et al.[12, 13]. Okamura and Yamashina[22] have considered the same two stage transfer line except they assumed that units are scrapped

whenever a machine processing it fails.

Jafari and Shantikumar[15] have modeled N-stage transfer lines as Markov-Chain.

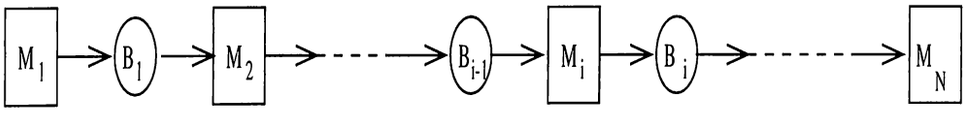

Figure 2.4: N-Stage Transfer Line

N-stage $(M_1, B_1, ...M_{N-1}, B_{N-1}, M_N)$ transfer lines (see Figure 2.4) are modeled as Markov-Chain by observing its state just after each transfer of parts. Because of the large state space, an approximate model called Decomposition-Aggregation Method is used. In this method,

First two-stage system is represented as (M_1, B_1, M_2)

Second two-stage system is represented as (RM_1, B_2, M_3) where RM_1 is the equivalent stage which replaces (M_1, B_1, M_2)

Third two-stage system is represented as (RM_2, B_3, M_4) where RM_2 is the equivalent stage which replaces $(M_1, B_1, M_2, B_2, M_3)$ or (RM_1, B_2, M_3).

In general, jth two stage system (for $j = 1...N-1$) is represented as (RM_{j-1}, B_j, M_{j+1}) where RM_{j-1} is defined as equivalent stage which replaces $(M_1, B_1, ...M_{j-1}, B_j, M_j)$ for $j = 2...N-1$.

This problem is solved through a series of iterative steps. The algorithm is given in the paper of Jafari and Shantikumar[15].

A similar model with no scrapping has been analysized by Sevastyanov [24], Buzacott[3, 4] , Sheskin[27], Gershwin[11]. The model of a multistage transfer line with scrapping has been considered by Shanthikumar[25].

Jafari and Shantikumar[16] have modeled 2-stage transfer lines with general

uptime and downtime distributions. It is assumed that the distribution of uptime and downtimes of each stage are of phase type. Using the phase structure of the underlying distributions, the system is modeled as a Markov-Chain.

Conway et al.[6] investigates the behaviour of asyncronous lines and explores the distribution and quantity of work in process (WIP) inventory that accumulates. Simple, generic production systems are studied to gain insight into the behaviour of more complex systems. The paper is concluded with the suggestion that buffers between workstations increase system capacity, but with sharply diminishing returns. Positions as well as capacity of buffers are important.

2.3.2 Pull Systems

Muckstadt and Tayur[20, 21] have shown the difference between Traditional Kanban Control System (TKCS) and Constant-Work-in-Process (CONWIP) type control system. If the serial production line consists of M machines arranged in a series of (or in tandem), these M machines are partitioned into N cells. If all the M machines are in the same cell, we have a CONWIP type control system; if on the other hand, there are a total of M cells, each cell containing exactly one machine, then we have a Traditional Kanban Control System (TKCS). In other words, if $N(M_1,....M_2)(C_1,....,C_N)(B)$ denotes a serial production line with N cells, M_i machines in cell i, C_i white kanbans in cell i, $i = 1...N$, and B colored cards that circulate throughout the shop floor, CONWIP is a $1/(M)/(C)/(.)$ system, and TKCS is a $M/(1,...,1)/(C_1,....,C_M)/(.)$ system. The difference between the model that we will construct and these two types of inventory control systems defined above is that the mechanism of the two control systems mentioned above is pull between cells and push within a cell. However, in our model that we will construct, mechanism is pull for only the last cell and the other cells operates according to the requests made in this last cell.

Spearman and Zazanis[28] examined the behaviour of push and pull production systems in an attempt to explain the apparent superior performance of pull systems. Three conjectures are considered in the paper; that pull systems have less congestion; that pull systems are inherently easier to control; and that the benefits of a pull environment owe more to the fact that WIP is bounded than to the practice of pulling everywhere. Moreover, a control strategy that has push and pull characteristics is identified and it is shown that this hybrid system outperforms both pure push and pure pull systems. Corbett[7] focuses exclusively on model based approaches in studying pull systems. Eventhough analytical models such as linear programming formulations or queing approximations exist, it is concluded that the inherent complexity of pull systems makes simulation an essential tool in studying them. Slobodow[29] studied the implementation of inventory pull (kanban) systems with simulation modeling as a means to quantitatively design such a system. The simulation results confirmed the feasibility of the pull system. The pull system caused an acceptably low amount of starvation in the machining lines and with this system, plant-wide inventory was reduced by 18%. Galbraith et al.[9] solves the problem of providing decision support for the transition from a traditional push production system to a pull system design.

Meral and Erkip[19] proposed a design methodology which addresses the design decisions involved in the design of an ideal JIT production line operating in an uncertain environment. Durmusoglu[8] describes the performance of the pull and push systems in a celluar manufacturing environment through an industrial application case. Ramudhin and Rochette[23] evaluated the performance of a JIT production system in the assembly department of an electronic equipment manufacturer where production is highly unstable. Then a pull type production system tailored to the specific needs of the problem at hand was developed. Lingayat and Mittenthal[17] suggested that the performance of a pull system deteriorates rapidly under non-ideal conditions, suggesting that a JIT system shouldn't be blindly implemented. An incremental approach based on order release was suggested as a step towards implementing JIT manufacturing.

Askin, Mitwasi and Goldberg[2] developed a stochastic model to determine the number of kanbans in multi item JIT systems. They also used simulation to check the accuracy of the model. Wang and Wang[30] developed a procedure to decompose a complex JIT system into a number of station pairs and they developed an algorithm to determine the optimal number of kanban for each pair of stations. Andijani and Clark[1] proposed different rules for allocating kanbans to maximize system performance for a pull system considering both throughput rate and WIP levels. Co and Jacopson[5] developed a recursive function which computes the number of kanbans required for all stages of serial production system. This function is used to compute a lower bound on the expected fill rate for an arbitrary kanban assignment.

As it is seen from the above survey that transfer lines and pull systems are the concepts that were usually worked on separately. There are few works about the transfer lines and pull systems together. However, these works are also based on different assumptions from the ones used in our model. In addition, constructing the Markov Chain model of this combined model is also a very good contribution to the literature.. In the following chapters of this study, we will see the details of our analytical and simulation model.

Chapter 3

Modeling of the Pull Systems

The Markov-Chain model of the two-stage transfer line with capacitated buffers under pull environment are constructed in this chapter. Although, most of the assumptions of the model with buffer sizes of 1 and those of the model with buffer sizes of more than two are the same, there are some small differences which may change the performance of the system.

3.1 Description of Model 1

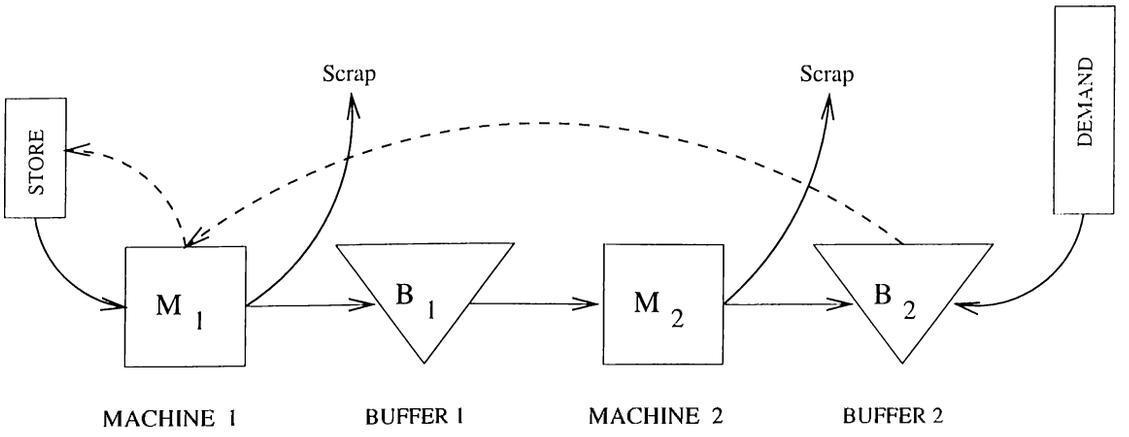

Figure 3.1: Pull Production System

A simple two-stage pull production system is depicted in Figure 3.1 and the assumptions of the model constructed are given below.

Assumptions :

- System is an automatic transfer line with fixed cycle time.
- System is observed after the transfer of each unit
- Demand arrives at the beginning of each cycle according to Poisson distribution with parameter λ.
- When the demand arrives, if there is an item in the last buffer, it takes that item, if there is no item in the last buffer, demand is not satisfied (no backlogging).
- All machine processing times are constant.
- There is no shortage of material on the store.
- Buffer capacities are one (i.e., $N_1 = 1$ and $N_2 = 1$).
- A machine breaks down only when it is processing an item.
- Machine i breaks down according to geometric distribution with parameter p_i $(i = 1, 2)$

$$P\{X_i = x\} = (1 - p_i)^{x-1} p_i \tag{3.1}$$

where X_i is the number of cycle times for machine i from the time that machine is repaired to the time that machine breaks down as it is seen from Figure 3.2. Here, U represents the uptime period and D represents the downtime period of the machine.

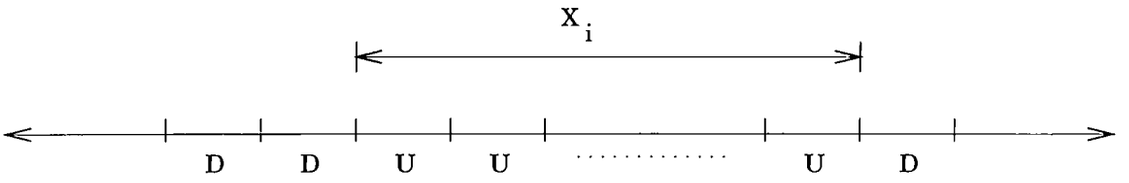

Figure 3.2: Time horizon for the states of an up system

- Machine i is repaired according to geometric distribution with parameter q_i $(i = 1, 2)$

$$P\{Y_i\} = (1 - q_i)^{y_i - 1} q_i \tag{3.2}$$

where Y_i is the number of cycle times in which machine is down (see Figure 3.3.)

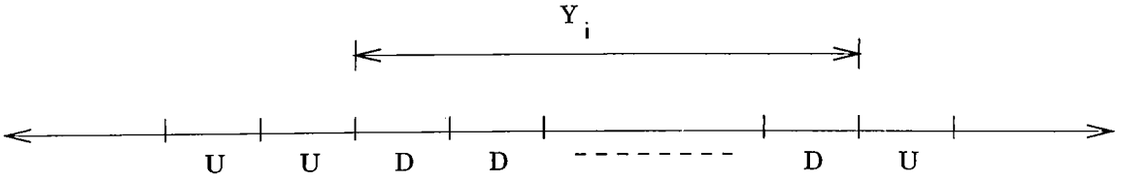

Figure 3.3: Time horizon for the states of a down system

- When the machine i is repaired, the item on that machine is scrapped with probability r_i ($i = 1, 2$).

Parameters : .

c : cycle time

p_i : probability that machine i breaks down during the cycle given that it was operative at the end of the previous cycle.

r_i : probability that part j will be scrapped when the machine i breaks down and then repaired.

$1 - r_i$: probability that part j will be completed on machine i after the machine i is repaired

$1/q_i$: mean of the repair time

q_i : probability that repair is completed in a cycle given that machine i ($i = 1, 2$) broke down or was under repair during the previous cycle (at the end of the previous cycle)

d : probability of one or more demand arrives

$$d = 1 - e^{-\lambda} \tag{3.3}$$

This system works in the pull environment. When one or more demand arrives, one of the demands is satisfied if there is a part in the last buffer and the remaining demands are lost. When a part is removed through the demand the last buffer becomes empty and production of a new part is instructed for machine 1. Machine 1 gets this instruction and asks a part from the store. When a new part comes to the machine 1, processing on that part starts. During that process, machine may break down. If it breaks down, a repair stage starts. Machine stays at the repair stage according to a geometric distribution. After the machine is repaired, the part on that machine might be scrapped or it is sent to downstream buffer. In the latter case, if the downstream buffer is full, machine becomes blocked. If a downstream machine can not find a part to process on the upstream buffer, it waits as idle.

3.1.1 Notation

States:

U : represents the state of being operative (up) for a machine
D : represents the state of being down or under repair for a machine
B : represents the state of being blocked for a machine
I : represents the state of being idle for a machine

Any state of the system (or model) is represented in the four tuples of (S_1, N_1, S_2, N_2) where

S_1 represents the state of machine 1 and takes the values of $\{U, D, B\}$
N_1 represents the state of buffer 1 and takes the values of $\{0, 1\}$
S_2 represents the state of machine 2 and takes the values of $\{U, D, B, I\}$
N_2 represents the state of buffer 2 and takes the values of $\{0, 1\}$

To represent the Markov-Chain model of the system, all states of the model must be obtained. If the states are written in blocks, it becomes very easy to obtain a transition probability matrix in block-diagonal form. The block diagonal matrix saves time in computation and it is also easier to write a computer program to solve this Markov-Chain representation.

We describe the blocks of the states in three different blocks for the buffer size of one. They are described as follows :

X_I : represents the set of states where the state of machine 2 is idle.
X_0 : represents the set of states where the state of buffer 1 is zero
X_1 : represents the set of states where the state of buffer 1 is one

If all states are shown in their respective blocks,

$X_I = \{(U,0,I,0),(D,0,I,0),(U,0,I,1),(D,0,I,1)\}$

$X_0 = \{[(U,0,U,0),(U,0,D,0),(D,0,U,0),(D,0,D,0)],$

$[(U,0,U,1),(U,0,D,1),(D,0,U,1),(D,0,D,1),(U,0,B,1),(D,0,B,1)]\}$

$X_1 = \{[(U,1,U,0),(U,1,D,0),(D,1,U,0),(D,1,D,0),(B,1,U,0),(B,1,D,0)],$

$[(U,1,U,1),(U,1,D,1),(D,1,U,1),(D,1,D,1),(U,1,B,1),(D,1,B,1),$

$(B,1,B,1),(B,1,U,1),(B,1,D,1)]\}$

Block Diagonal form of the probability transition matrix is shown in Figure 3.4. When writing the states in blocks, the order given above is preserved.

A few examples are given below to represent the computation of getting the probability of going to a state from another state.

Example 1 :

Prob$\{(U,0,D,1) \Rightarrow (D,0,I,0)\}$ = Prob{machine 1 breaks down} * Prob{machine 2 is repaired}*Prob{the part is scrapped after machine 2 is repaired}*Prob{one or more demand arrives}

$= p_1 * q_2 * r_2 * d$

Example 2:

Prob$\{(U,1,U,1) \Rightarrow (B,1,B,1)\}$ = Prob{machine 1 does not break down}*Prob{machine 2 does not break down}*Prob{no demand arrives}

$= (1-p_1) * (1-p_2) * (1-d)$

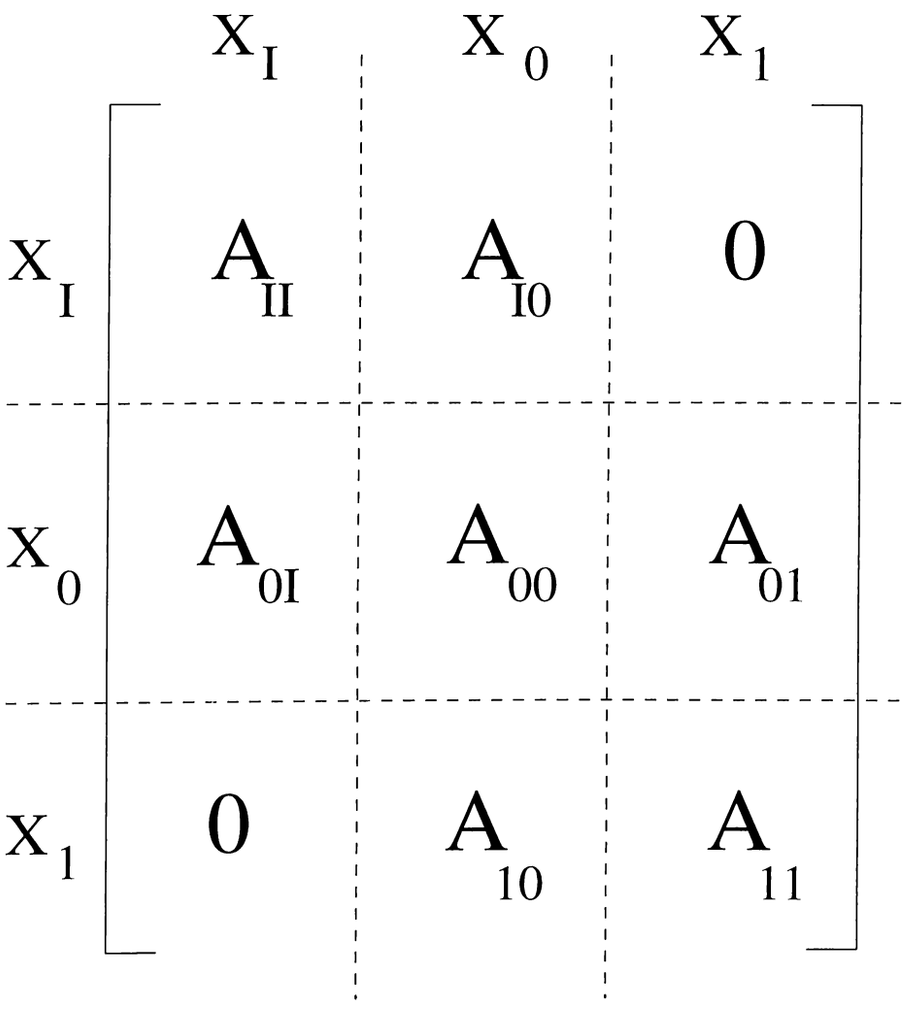

Figure 3.4: Block diagonal form of the Markov Chain Model for $N = 1$

Example 3:

Prob$\{(D, 1, D, 1) \Rightarrow (U, 1, U, 1)\}$ = Prob{machine 1 is repaired}*Prob{the part on machine 1 is not scrapped}*Prob{machine 2 is repaired}*Prob{(the part on machine 2 is scrapped and no demand arrives) or (the part on machine 2 is not scrapped and one or more demand arrives}

$= q_1 * (1 - r_1) * [r_2 * (1 - d) + (1 - r_2) * d]$

Two block probability matrices are shown below to get a better idea about the model description as a Markov-Chain.

Example 4:

Probability transition matrix of going from the states of X_I to the states of X_I is shown below.

$$A_{II} = \begin{bmatrix} 0 & p_1 & 0 & 0 \\ q_1 r_1 & (1-q_1) & 0 & 0 \\ 0 & dp_1 & 0 & (1-d)p_1 \\ q_1 r_1 d & (1-q_1)d & q_1 r_1 (1-d) & (1-q_1)(1-d) \end{bmatrix}$$

Example 5 :

As another example, the probability transition matrix of going from the states of X_0 to the states of X_I is shown below.

$$A_{01} = \begin{bmatrix} 0 & 0 & 0 & p_1(p_2) \\ 0 & p_1q_2r_2 & 0 & p_1q_2(1-r_2) \\ 0 & 0 & 0 & (1-q_1)(1-p_2) \\ q_1r_1q_2r_2 & (1-q_1)q_2r_2 & q_1r_1q_2(1-r_2) & (1-q1)q_2(1-r_2) \\ 0 & 0 & 0 & p_1(1-p2)d \\ 0 & p_1q_2r_2d & 0 & p_1q_2[(1-r_2)d+r_2(1-d)] \\ 0 & 0 & q_1r_1(1-p_2)d & (1-q_1)(1-p_2)d \\ q_1r_1q_2r_2d & (1-q_1)q_2r_2d & q_1r_1q_2r_2(1-d) & (1-q_1)q_2[r_2(1-d)+(1-r_2)d] \\ 0 & 0 & 0 & p_1d \\ 0 & 0 & q_1r_1d & (1-q_1)d \end{bmatrix}$$

3.1.2 Model Solution

This Markov-Chain model is solved by using the set of equations

$$xP = x \tag{3.4}$$

$$xe = 1 \tag{3.5}$$

where x is the steady-state probabilities of each state of the model and e is the unit vector. Since the number of states is small for the buffer size of one, there is no difficulty in solving this equation set.

We have written a program in the package Maple to solve this set of equations. Using computer program makes the life easy since you can change

parameters at the initial step, compute the steady-state probabilities of the states and then you can compute the performances of the system by using these probabilities.

The algorithm of the program is provided below. Since the buffer size is one and state space is small, transition probabilities are entered directly, there is no need to generate it automatically.

Algorithm :

Step 1 : Generate A_{II}, A_{I0}, A_{I1} block matrices by directly writing the transition probabilities.

Step 2 : Generate A_{0I}, A_{00}, A_{01} block matrices by directly writing the transition probabilities.

Step 3 : Generate A_{1I}, A_{10}, A_{11} block matrices by directly writing the transition probabilities.

Step 4 : Combine all block-matrices sequentially (i.e., A_{II}, A_{I0},A_{I1}, A_{0I}, A_{00},A_{01}, A_{1I},A_{10},A_{11}) to obtain a block-diagonal matrix P.

Step 5 : Solve the set of equations 3.4 and 3.5 by using

$$x(P - I) = 0$$

$$xe = 1$$

Experiments :

By using different parameters, we have designed an experiment to see the relations between parameters of the model and to compute the performance measures. The table of parameters used in the experiment is shown below :

Run#	p_1	p_2	q_1	q_2	r_1	r_2	λ_D	d
1	0.1	0.1	0.95	0.95	0.1	0.1	1	0.632
2	0.01	0.01	0.70	0.70	0.01	0.01	1	0.632
3	0.1	0.1	0.70	0.70	0.1	0.1	1	0.632
4	0.01	0.01	0.95	0.95	0.01	0.01	1	0.632
5	0.01	0.01	0.95	0.01	0.01	0.01	4	0.98

Table 3.1: Parameters used in Experiment 1

By using these parameters, we have computed different performance measures. The performance measures used and their formulas are given below:

a) a = Percentage of Demand satisfied = Prob { last buffer size is one }

b) b = Expected numbers in the buffers = 1 * Prob {no. of parts on the buffers is one} + 2* Prob {no of parts on the buffer is two }

c) c_1= Prob {Machine 1 is blocked } = Total probability of the states that machine 1 is in B state

c_2 =Prob {Machine 2 is blocked } = Total probability of the states that machine 2 is in B state.

d) d = Prob{machine 2 is idle} =Total probability of states that machine 2 is in I state.

e) e_1 = Percentage of scrapping on machine 1 = r_1 * Prob {machine 1 is down }

e_2 = Percentage of scrapping on machine 2 = r_2 * Prob {machine 2 is down}

f) f_1 = Percentage of uptime for machine 1 = Total probability of the states that machine 1 is in U state.

f_2 = Percentage of uptime for machine 2 = Total probability of the states that machine 2 is in U state.

The results are shown in the table below :

Run#	a	b	c_1	c_2	d	e_1	e_2	f_1	f_2
1	0.9464	1.8467	0.3241	0.3234	$7.4 * 10^{-3}$	$6.45 * 10^{-3}$	$6.38 * 10^{-3}$	0.6115	0.6055
2	0.99	1.977	0.3636	0.3623	0.0014	$8.9 * 10^{-5}$	$8.8 * 10^{-5}$	0.62	0.6272
3	0.9201	1.7867	0.3221	0.3088	0.0203	$8.5 * 10^{-3}$	$8.4 * 10^{-3}$	0.5933	0.5873
4	0.9916	1.8771	0.3066	0.3604	$6.3 * 10^{-3}$	$6.6 * 10^{-5}$	$6.5 * 10^{-5}$	0.6273	0.6267
5	0.9173	1.1123	$5.2 * 10^{-3}$	0.0168	0.0747	$9.48 * 10^{-4}$	$9.4 * 10^{-5}$	0.899	0.899

Table 3.2: Results for Experiment 1

In an another experiment, the parameters and the results for the performance measures explained before are shown in Table 3.3. and Table 3.4. respectively.

Run#	p_1	p_2	q_1	q_2	r_1	r_2	λ_D
1	0.01	0.1	0.7	0.95	0.01	0.1	1
2	0.01	0.1	0.7	0.95	0.01	0.1	4
3	0.01	0.1	0.7	0.95	0.01	0.1	8
4	0.1	0.01	0.95	0.70	0.1	0.01	1
5	0.1	0.01	0.95	0.70	0.1	0.01	4
6	0.1	0.01	0.95	0.70	0.1	0.01	4
7	0.001	0.15	0.70	0.90	0.90	0.05	1
8	0.001	0.15	0.70	0.90	0.90	0.05	4
9	0.001	0.15	0.70	0.90	0.90	0.05	8
10	0.15	0.001	0.90	0.70	0.05	0.90	1
11	0.15	0.001	0.90	0.70	0.05	0.90	4
12	0.15	0.001	0.90	0.70	0.05	0.90	8

Table 3.3: Parameters used in Experiment 2

Run#	a	b	c_1	c_2	d	e_1	e_2	f_1	f_2
1	0.9525	1.9397	0.3833	0.3268	$1.3 * 10^{-3}$	$7.7 * 10^{-5}$	$6.39 * 10^{-3}$	0.608	0.608
2	0.8957	1.8238	0.1005	0.0161	$3.7 * 10^{-3}$	$1.26 * 10^{-4}$	$9.24 * 10^{-3}$	0.8867	0.8868
3	0.8921	1.805	0.0862	0.002	$4.1 * 10^{-3}$	$1.28 * 10^{-4}$	$8.71 * 10^{-3}$	0.9009	0.8931
4	0.9869	1.8571	0.3035	0.3574	$9.76 * 10^{-3}$	$6.6 * 10^{-3}$	$8.82 * 10^{-5}$	0.6421	0.6237
5	0.9044	1.104	0.0081	0.0164	0.082	$8.99 * 10^{-3}$	$1.25 * 10^{-4}$	0.897	0.8882
6	0.8921	09776	0.0039	0.0002	0.0948	$9.74 * 10^{-3}$	$1.27 * 10^{-4}$	0.9019	0.8920
7	0.9306	1.928	0.4058	0.3076	$3.01 * 10^{-4}$	$6.48 * 10^{-4}$	$4.92 * 10^{-3}$	0.5931	0.5927
8	0.8545	1.847	0.1541	0.0144	$8.6 * 10^{-4}$	$1.04 * 10-3$	$7.02 * 10^{-3}$	0.8445	0.8438
9	0.8499	1.8476	0.1417	0.0002	$8.7 * 10^{-4}$	$1.01 * 10^{-3}$	$7.12 * 10^{-3}$	0.8569	0.8557
10	0.9834	1.774	0.2667	0.3557	0.0197	$5.22 * 10^{-3}$	$6.64 * 10^{-4}$	0.6268	0.6233
11	0.8657	0.9497	$1.75 * 10^{-3}$	0.014	0.1345	$7.12 * 10^{-3}$	$9.88 * 10^{-4}$	0.8552	0.849
12	0.8498	0.8561	$2.94 * 10^{-4}$	$1.8 * 10^{-4}$	0.1479	$7.12 * 10^{-3}$	$1.05 * 10^{-3}$	0.8568	0.8504

Table 3.4: Results for Experiment 2

In the following figures, each bar represents the value of the chosen performance measure for a specific run. For example, the number [1,4] above the bars show that the first bar under this number shows the value for Run#1 and the second bar under this number shows the value for Run#4.

In Figure 3.5, we see that when we put more reliable machine with $p2$=0.01 and $r2$=0.01 but with $q2$=0.7 (i.e., waits in repair state more) at the end of the line in case 1 (ie., Run # 1-4) and with p=0.001 and r=0.9 and q=0.7 in case 2 (ie., Run # 7-10) , the percentage of demand satisfied increases. This is because of the fact that when demand arrives, the last machine is found more operative and sending finished parts to the buffer in that case. However, there is an interesting result that, when the demand increases greatly with respect to the buffer sizes, in our case d=8 and N=1, there is no difference in percentage of demand satisfied.

When we again put the more reliable machine at the end of the line, the expected total number of parts in the buffers decreases as it is seen in Figure 3.6. If we increase demand greatly, such as d=8, the difference between these two cases becomes more apperant. This is because of the fact that when the more reliable machine is in the upstream and the other machine is in the downstream, upstream machine becomes blocked quickly and the parts on buffer 1 piles up. This increases the average number in the buffers.

As we have said before, when the more reliable machine is put at the downstream and the other machine at the upstream, percentage of upstream

machine's blockage decreases (see Figure 3.7) and percentage of downstream machine's blockage increases (see Figure 3.8) since it is more reliable and rapid.

When the more reliable machine is in the downstream, the percentage of idleness increases sharply, since the unreliable machine can not send enough parts to the downstream buffer and makes the downstream machine idle (Figure 3.9).

The scrapping on Machine 1 and Machine 2 increases when demand increases (see Figure 3.10), but the slope of the figure decreases and levels at a fixed value. This may be because of the fact that, the machines reach up the full capacities and at the full capacity there is a fixed scrap percentage. So, at very large demands, the scrap percentage hits its limit.

When we exchange the machines, the percentage of uptime does not change very much for the same machine (see Figure 3.11).The percentage of uptime does not differ greatly, when demands reach high values, such as d=4 or d=8.

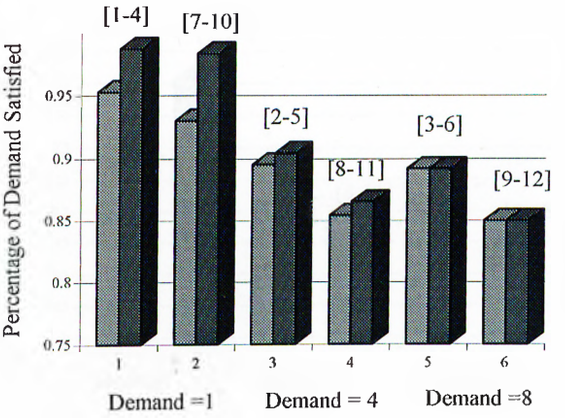

Figure 3.5: Percentage of demand satisfied versus Demand for different parameters and for N=1

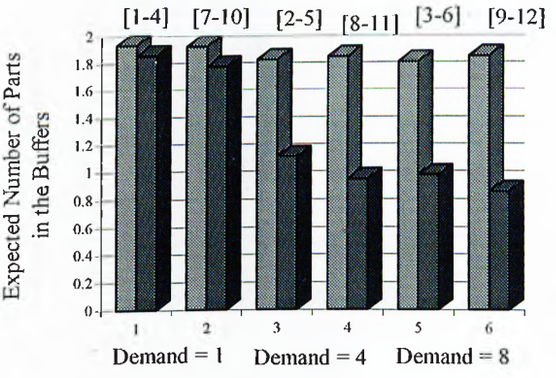

Figure 3.6 : Expected number of parts versus Demand for different parameters and for N=1

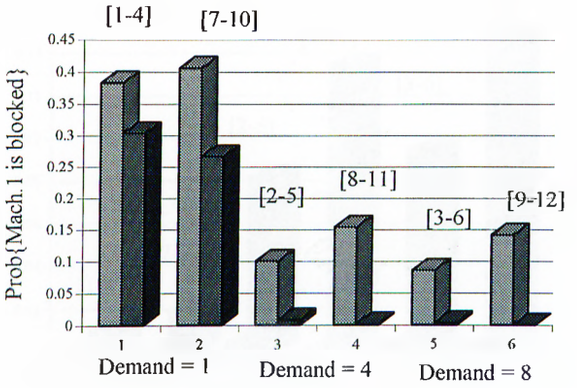

Figure 3.7 : Blockage of Mach.1 versus Demand for different parameters and for N=1

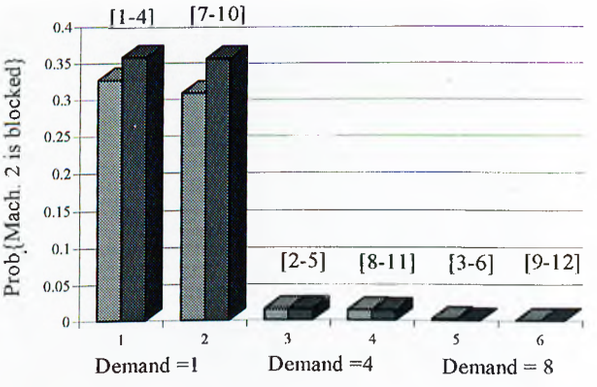

Figure 3.8 : Blockage of Mach.2 versus Demand for different parameters and for N=1

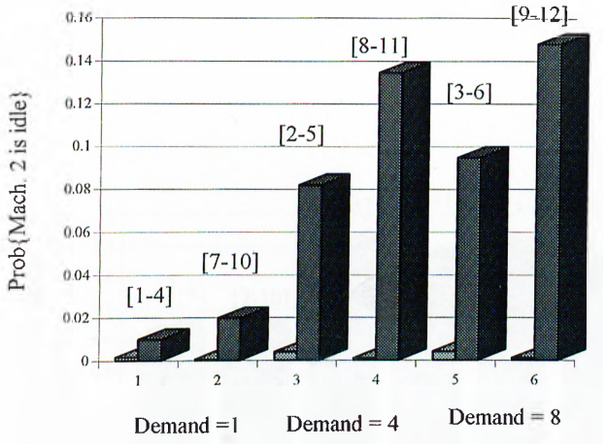

Figure 3.9 : Idleness for Mach.1 versus Demand for different parameters and for N=1

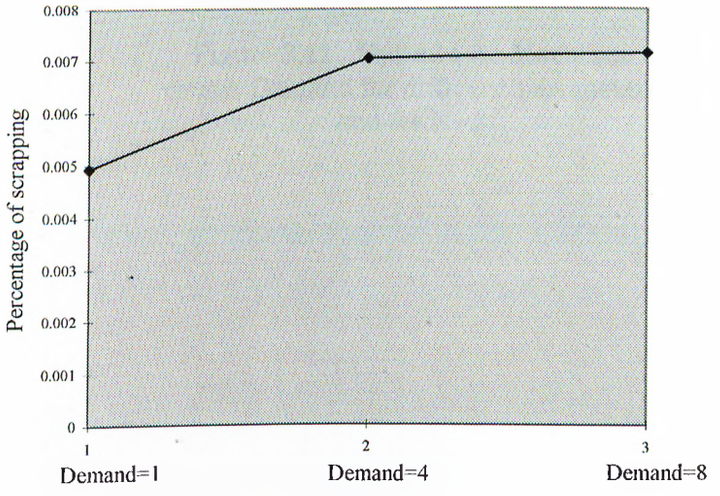

Figure 3.10 : Scrapping versus Demand

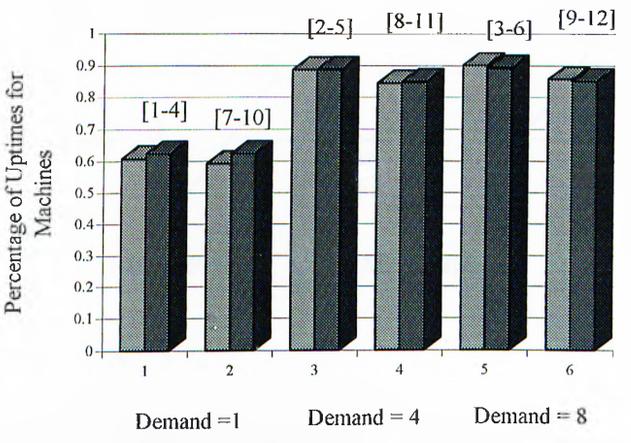

Figure 3.11 : Uptimes for Machines versus Demand for different parameters and for N=1

3.2 Two Stage Model with Buffer Capacities of $N(N \geq 2)$

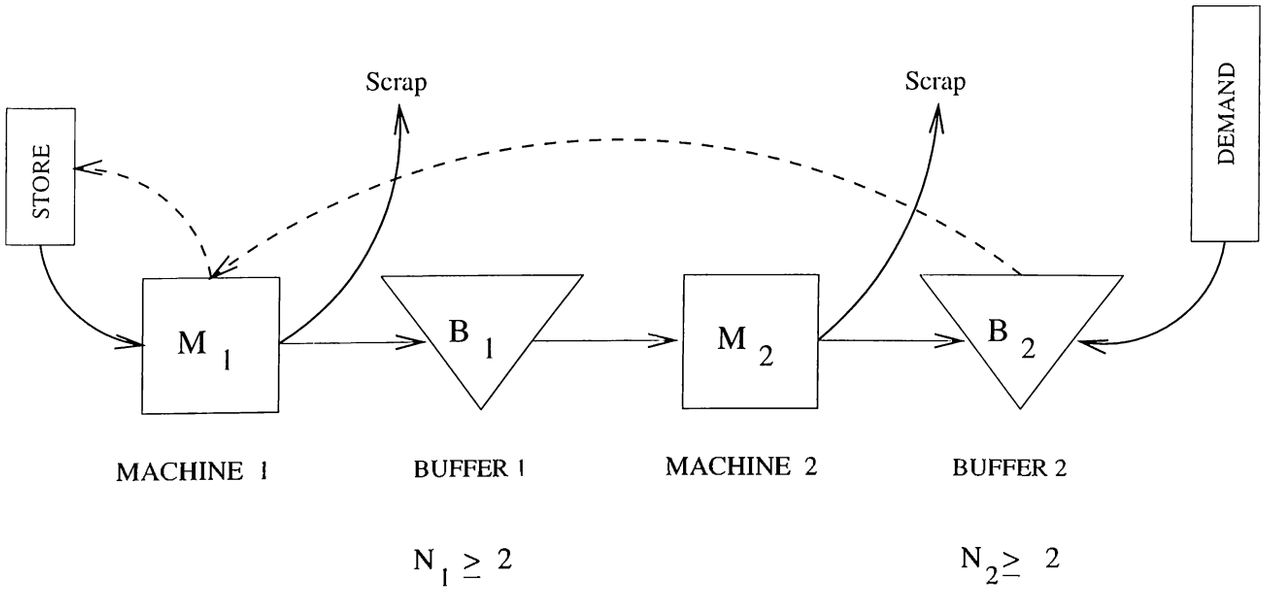

Figure 3.12: Two Stage Model with buffer capacities of $N \geq 2$

Most of the assumptions of this two-stage model are same with the those of our previous two-stage model. However, there are some differences in the assumptions which are given below :

- Buffer sizes are general ($N \geq 2$)
- Machine 1 can be in idle state because when the last buffer is full and machine 2 is up at the end of a cycle time, machine 2 becomes blocked and a new part can not be released from the store to the machine 1. Therefore, machine 1 waits idle until a command is sent from machine 2 when the blockage is removed.
- Number of states differ greatly. If N represents the buffer size,

  X_{I35_1} (set of states where the state of machine 1 is idle) has $N+1$ states

  X_{I_2} (set of states where the state of machine 2 is idle) has $2^*(N+1)$ states

X_0 has 4*$(N+1)$+2 states
$X_i (1 \leq i \leq N-1)$ has 4*$(N+1)$+2 states
X_N has 6*$(N+1)$+3 states

- d_k $(0 \leq k \leq N)$ represents the probability of k demand arrives

$$d_k = Prob\{k \ \ demand \ \ arrives\} = (\lambda/k!) * e^{-\lambda_D} \quad (0 \leq k \leq N-1)$$

$$d_N = 1 - \sum_{k=0}^{N-1} d_k$$

To represent the states, we give an example of the state blocks for the buffer sizes of two.

$X_{I_1} = \{(I,0,B,2),(I,1,B,2),(I,2,B,2)\}$

$X_{I_2} = \{(U,0,I,0),(D,0,I,0),(U,0,I,1),(D,0,I,1),(U,0,I,2),(D,0,I,2)\}$

$X_0 = \{[(U,0,U,0),(U,0,D,0),(D,0,U,0),(D,0,D,0)],$
$[(U,0,U,1),(U,0,D,1),(D,0,U,1),(D,0,D,1)],$
$[(U,0,U,2),(U,0,D,2),(D,0,U,2),(D,0,D,2),(U,0,B,2),(D,0,B,2)]\}$

$X_1 = \{[(U,1,U,0),(U,1,D,0),(D,1,U,0),(D,1,D,0)],$
$[(U,1,U,1),(U,1,D,1),(D,1,U,1),(D,1,D,1)],$
$[(U,1,U,2),(U,1,D,2),(D,1,U,2),(D,1,D,2),(U,1,B,2),(D,1,B,2)\ \}$

$X_2 = \{[(U,2,U,0),(U,2,D,0),(D,2,U,0),(D,2,D,0),(B,2,U,0),(B,2,D,0)],$
$[(U,2,U,1),(U,2,D,1),(D,2,U,1),(D,2,D,1),(B,2,U,1),(B,2,D,1)],$
$[(U,2,U,2),(U,2,D,2),(D,2,U,2),(D,2,D,2),(U,2,B,2),(D,2,B,2),$
$(B,2,B,2),(B,2,U,2),(B,2,D,2)\ \}$

The block diagonal form of the probability transition matrix is shown in Figure 3.13. If we removed the first row and the first column of the blocks, we would obtain a pure block-diagonal matrix.

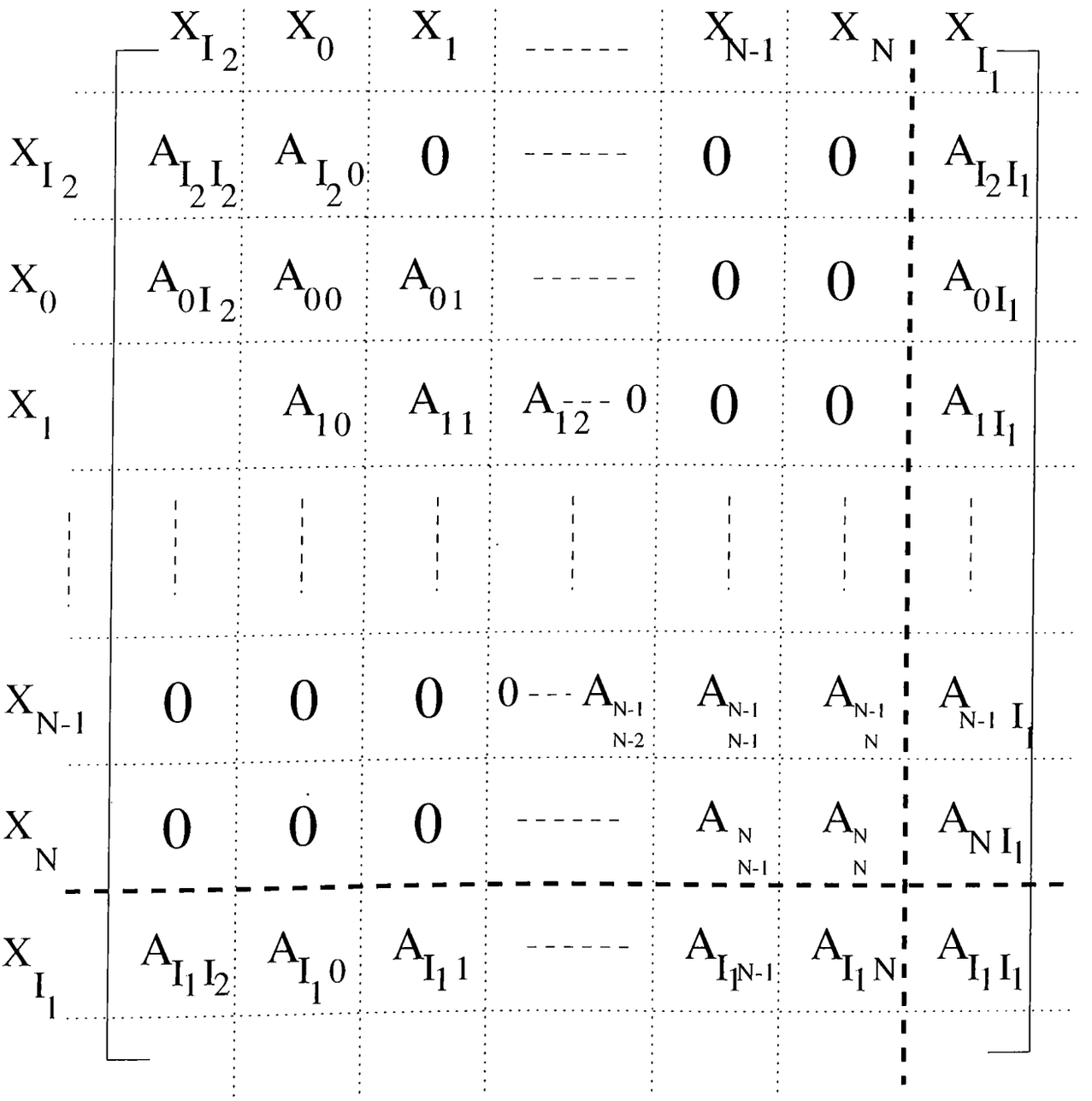

Figure 3.13: Block-Diagonal form of the Markov-Chain Model for buffer sizes of N

To ease computations, we also break each block matrix into sub blocks. This eases writing a flexible computer program for any buffer sizes. Let us give some examples to show these block features.

Example 6 :

	(I,0,B,2)	(I,1,B,2)	(I,2,B,2)
(U,1,U,0)	0	0	0
(U,1,D,0)	0	0	0
(D,1,U,0)	0	0	0
(D,1,D,0)	0	0	0
(U,1,U,1)	0	0	0
(U,1,D,1)	0	0	0
(D,1,U,1)	0	0	0
(D,1,D,1)	0	0	0
(U,1,U,2)	0	0	$(d_0 p_1 p_2)$
(U,1,D,2)	0	0	$(d_0 q_2(1-r_2)(1-p_2))$
(D,1,U,2)	0	$q_1 r_1(1-p_2)d_0$	$d_0 q_1(1-r_1)(1-p_2)$
(D,1,D,2)	0	$d_0 q_1 r_1 q_2(1-r_2)$	$(d_0 q_1(1-r_1)q_2(1-r_2))$
(U,1,B,2)	0	0	$d_0(1-p_1)$
(D,1,B,2)	0	$d_0 q_1 r_1$	$d_0 q_1(1-r_1)$

Table 3.5: Probability transition matrix from $X_1 \rightarrow X_{I_1}$ of the Two stage model with $N = 2$

	(U,0,I,0)	(D,0,I,0)	(U,0,I,1)	(D,0,I,1)	(U,0,I,2)	(D,0,I,2)
(U,0,I,0)	0	p_1	0	0	0	0
(D,0,I,0)	q_1r_1	(1-q_1)	0	0	0	0
(U,0,I,1)	0	$(d_1+d_2)p_1$	0	$p_1(\overline{d})$	0	0
(D,0,I,1)	$(d_1+d_2)q_1r_1)$	$(d_1+d_2)(1-q_1)$	$(\overline{d})q_1r_1$	$(\overline{d})(1-q_1)$	0	0
(U,0,I,2)	0	p_1d_2	0	p_1d_1	0	$p_1(\overline{d})$
(D,0,I,2)	$q_1r_1d_2$	$(1-q1)d_2$	$q_1r_1d_1$	$(1-q_1)d_1$	$q_1r_1(\overline{d})$	$(1-q_1)(\overline{d})$

Table 3.6: Probability transition matrix from $X_{I_2} \rightarrow X_{I_2}$ of the Two stage model with $N = 2$

where $(\overline{d})$ means $(1 - d_1 - d_2)$.

3.2.1 Model Solution

This model is again solved by using the set of equations

$$xP = x$$

$$xe = 1$$

However, since the model is more complex and the state space is large, block feature of the matrix is used when writing a computer program in Maple. The algorithm is given below :

Algorithm :

We use general names for the buffer sizes and the parameters for generating blocks. This makes the program flexible and adaptable to any buffer sizes and parameters.

Step 1 : Generate first row of the block diagonal matrix P

1.1. Generate $X_{I_2} \Rightarrow X_{I_2}, X_{I_2} \Rightarrow X_0, ...X_{I_2} \Rightarrow X_N, X_{I_2} \Rightarrow X_{I_1}$ blocks

1.2. Combine these block matrices to get first row block.

Step 2 : Generate second row of the block diagonal matrix.

2.1. Generate $X_0 \Rightarrow X_{I_2}, X_0 \Rightarrow X_0, ...X_0 \Rightarrow X_N, X_0 \Rightarrow X_{I_1}$ blocks

2.2. Combine these block matrices to get second row block.

Step 3 : Generate $i+2$ th row block $(i = 0..N-1)$ by using nested loops.

3.1. Generate $X_i \Rightarrow X_{I_1}, X_i \Rightarrow X_j$ where $(j = 0..N-1)$, $X_i \Rightarrow X_N$ and $X_i \Rightarrow X_{I_2}$ blocks.

3.2. : Combine these blocks to obtain $i+2$ th row for each i $(i = 0...N)$

Step 4 : Generate $N+2$ th row of the block diagonal matrix.

4.1. Generate $X_N \Rightarrow X_{I_2}, X_N \Rightarrow X_0, ...X_N \Rightarrow X_N, X_N \Rightarrow X_{I_1}$ blocks

4.2. Combine these block matrices to get $N+2$ th row block.

Step 5 : Generate $N+3$ th row of the block diagonal matrix P

5.1. Generate $X_{I_1} \Rightarrow X_{I_2}, X_{I_1} \Rightarrow X_0, ...X_{I_1} \Rightarrow X_N, X_{I_1} \Rightarrow X_{I_1}$ blocks

5.2.Combine these block matrices to get $N+2$ th row block.

Step 6 : Combine all row blocks into one transition matrix P.

Step 7 : Solve the set of equations

$$xP = x$$

$$xe = 1$$

by using

$$x(P - I) = 0$$

and

$$xe = 1$$

and by changing one column of P by e and obtain the steady state probability vector x.

Some Different Algorithms :

After getting the P matrix in the computer program, a different method or algorithm is utilized rather than the equation set of 3.4. and 3.5. to obtain the steady state vector x. Some of them are Jacobi, Gauss-Seidel and SOR. For the possibility of reducing the time to obtain x , we have written an extended program to the main program for the SOR algorithm. This algorithm is given below.

SOR : $(0 < w < 2)$

$$A = [(1/w) * D - L] - [((1-w)/w) * D + U] \Longrightarrow$$

$$[(1/w) * D - L] * x^{(k+1)} = [((1-w)/w) * D + U] * x^{(k)}$$

$$[x^{(k+1)} = w * D^{-1} * (L * x^{(k+1)} + U * x^{(k)}) + (1-w) * x^{(k)}$$

$$x_i{}^{(k+1)} = w * [(1/d_{ii}) * (\sum_{j<i} l_{ij} x_j{}^{(k+1)} + \sum_{j>i} u_{ij} x_j{}^{(k)}] \quad + \quad (1-w) * x_i{}^{(k)}$$

$$\text{for } (i = 1, 2, ...n)$$

where

D : represents diagonal matrix
L : represents lower part of the matrix
U : represents upper part of the matrix
$X_i{}^{(k)}$: ith row of the steady state vector at the k th step.
l_{ij} : element of the ith row and jth column of L
u_{ij} : element of the ith row and jth column of U

At the first step, assign initial value of

$$x^{(0)} = [\frac{1}{n}, ...\frac{1}{n}]$$

Stop when

$$\|x^{(k+1)} - x(k)\|_2 / \|x^{(k+1)}\|_2 < 10^{-7}$$

The extended program of SOR in Maple did not provide us the desired results concerning the time saving. If we had written that program in Fortran language, it would have worked more rapidly. Although we could not have good results, this application gave us an idea of applying different algorithms to the same problem.

Experiment 3:

The aim of this experiment is to see the performance of the system in the Pull environment under different parameters. The parameters used are repeated in Table 3.7. and the results obtained for each performance measure are given in Table 3.8 and 3.9. The notation used for the performance measures in Table 3.8 and 3.9 is also summarized below :

a_i : Average number of parts in buffer i , $i = 1, 2$
b_i : Prob{Machine i is blocked} , $i = 1, 2$
c_i : Prob{Machine i is starved (idle)}, $i = 1, 2$
d_i : Prob{Machine i is operative (up)}, $i = 1, 2$
e_i : Prob{Machine i is down}, $i = 1, 2$

Run#	p_1	p_2	q_1	q_2	r_1	r_2	λ_D
1	0.01	0.1	0.7	0.95	0.01	0.1	1
2	0.01	0.1	0.7	0.95	0.01	0.1	4
3	0.01	0.1	0.7	0.95	0.01	0.1	8
4	0.1	0.01	0.95	0.70	0.1	0.01	1
5	0.1	0.01	0.95	0.70	0.1	0.01	4
6	0.1	0.01	0.95	0.70	0.1	0.01	4
7	0.001	0.15	0.70	0.90	0.90	0.05	1
8	0.001	0.15	0.70	0.90	0.90	0.05	4
9	0.001	0.15	0.70	0.90	0.90	0.05	8
10	0.15	0.001	0.90	0.70	0.05	0.90	1
11	0.15	0.001	0.90	0.70	0.05	0.90	4
12	0.15	0.001	0.90	0.70	0.05	0.90	8

Table 3.7: Parameters used in Experiment 3

Run#	a_1	a_2	b_1	b_2	c_1
1	1.89	1.46	0.225	0.174	0.017
2	1.855	0.9084	0.0837	0.00026	0
3	1.8572	0.893	0.0835	0	0
4	0.3169	1.473	0.0058	0.179	0.162
5	0.1327	0.9083	0.00139	0.0002	0.0002
6	0.132	0.8924	0.0014	0	0
7	1.981	1.406	0.273	0.156	1.69810^{-3}
8	1.981	0.866	0.141	0.0002	0
9	1.98	0.85	0	0	0
10	0.1545	1.4212	0.0002	0.1643	0.1481
11	$7.12 * 10{-}3$	0.84	0	0.00018	0.00018
12	$7.97 * 10^{-3}$	0.84	0	0	0

Table 3.8: Results for Experiment 3

Run#	c_2	d_1	d_2	e_1	e_2
1	$8.2 * 10^{-4}$	0.744	0.744	$1.04 * 10^{-4}$	$4.3 * 10^{-3}$
2	$8.2 * 10^{-4}$	0.744	0.744	$1.04 * 10^{-4}$	$4.3 * 10^{-3}$
3	$1.43 * 10^{-3}$	0.902	0.902	1.27 * 10−4	$9.49 * 10^{-3}$
4	0.0649	0.7468	0.7419	$7.79 * 10^{-3}$	10−4
5	0.0926	0.899	,0.891	0.0094	1.24810^{-4}
6	0.0925	0.9021	0.8934	0.00949	$1.2 * 10^{-4}$
7	0	0.721	0.720	$5.4 * 10^{-3}$	$5.9 * 10^{-3}$
8	0.0001	0.855	0.848	$9.18 * 10^{-4}$	$7.05 * 10^{-3}$
9	0.0001	0.8568	0.8565	$8.55 * 10^{-4}$	7.13 * 10−3
10	0.1091	0.7278	0.7217	$5.75 * 10^{-3}$	$6.03 * 10^{-3}$
11	0.1464	0.8538	0.8489	$7.04 * 10^{-3}$	$9.18 * 10^{-4}$
12	0.1473	0.855	0.849	$7 * 10^{-3}$	$9.27 * 10^{-4}$

Table 3.9: Results for Experiment 3

The parameters used for this experiment are the same as those of Experiment 2 except for the buffer size which is two. The results obtained in this experiment are very similar to the ones obtained in Experiment 2 with respect to the chosen performance measures of the system. Although they are almost same, there are some minor differences between these two experiments.

For the performance measure of average number in buffer 1 and buffer 2, when we put the more reliable machine ($p = 0.01$) at the end (i.e., near the last buffer) we see that average number in buffer 1 decreases to 17% of its previous value in which reliable machine is at the beginning of the line when demand is one (See Figure 3.14). We also observe that average number in buffer 2 does not change (see Figure 3.15). Therefore, total number of parts decreases to half of its previous value in which reliable machine is at the beginning of the line. So, we see that total number of parts in buffers decreases when we put the more reliable machine at the end. Since Pull systems aim at reducing the Work in Process(WIP) inventory, we can reduce WIP by just changing the positions of the machines (i.e., by putting the more reliable machine at the end) in the Pull environments.

If we put the more reliable machine at the end, percentage of blockage for machine 1 decreases significantly, to its 2.5% value as it is seen from Figure 3.16. However, percentage of machine 1 idleness increases greatly(see Figure 3.18). So, the total non-operative(blockage and idleness) times in these two cases do not change very much. For machine 2, if we put the more reliable machine at the end, both blockage and idleness increase (see Figure 3.17 and Figure 3.19). However, if we sum the non-operative times for both machines in both cases, we see that non-operative times for the machines are the same. This means that we do not gain any benefit for the non-operative times by exchanging the positions of the machines in the Pull environment.

The percentage of machines' uptime does not change when we again exchange the positions of reliable and unreliable machines (see Figure 3.20 and Figure 3.21). The percentages of scrapping on machines of each type(reliable and unreliable) do not change either (see Figure 3.22 and Figure 3.23)

As a result, we can say that by just exchanging the positions of the machines we can decrease the average number of parts in the buffers without changing the other performance measures of the machines.

Another interesting result is that when we increase the demand, total average number of parts in buffers decreases sharply first and then levels off at a fixed value(see Figure 3.24). This may be because of that the system reaches an equilibrium point and is fully utilized for these parameters.

When we compare the results in Table 3.8 and the results in Table 3.4, the total average number of parts in the buffers for the buffer size of 1 may sometimes be slightly greater than that for buffer size of 2. This may be due to that in the second model, machines are found in the idle state more than machines in the first model because of the slight differences in the model assumptions.

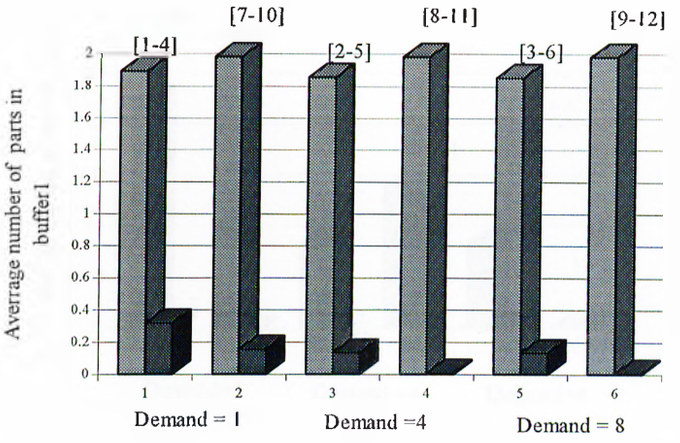

Figure 3.14 : Parts in buffer 1 versus Demand for different parameters and for N=2

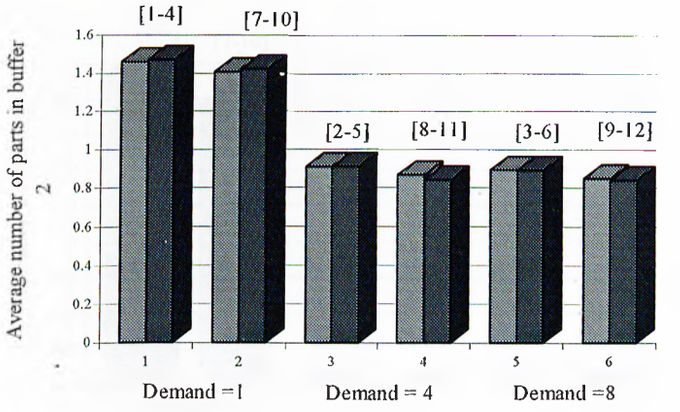

Figure 3.15 : Parts in buffer 2 versus Demand for different parameters and for N=2

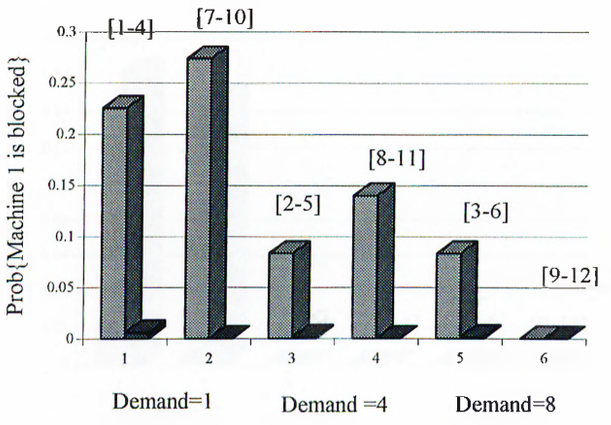

Figure 3.16 : Blockage of Mach.1 versus Demand for different parameters and for N=2

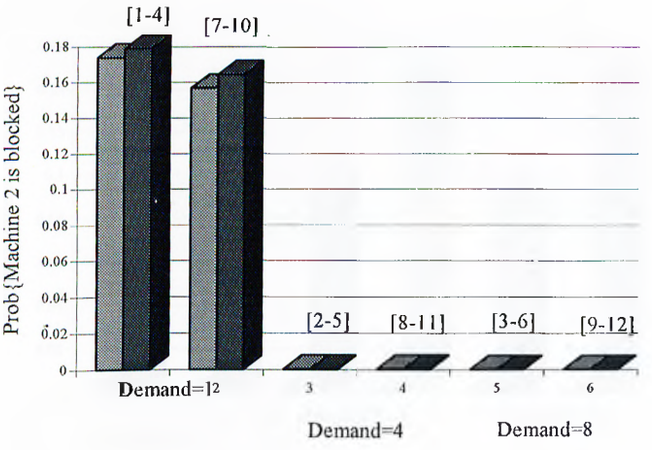

Figure 3.17 : Blockage of Mach.2 versus Demand for different parameters and for N=2

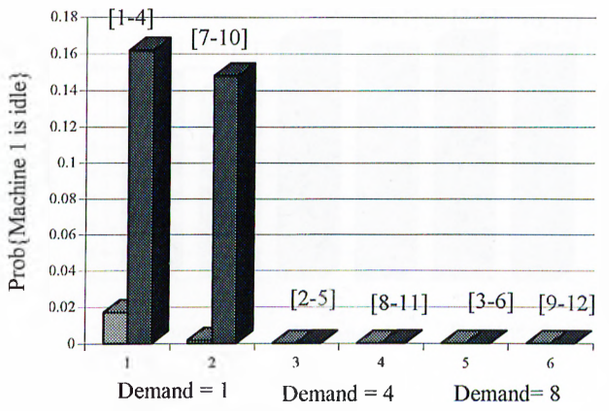

Figure 3.18 : Idleness of Mach.1 versus Demand for different parameters and for N=2

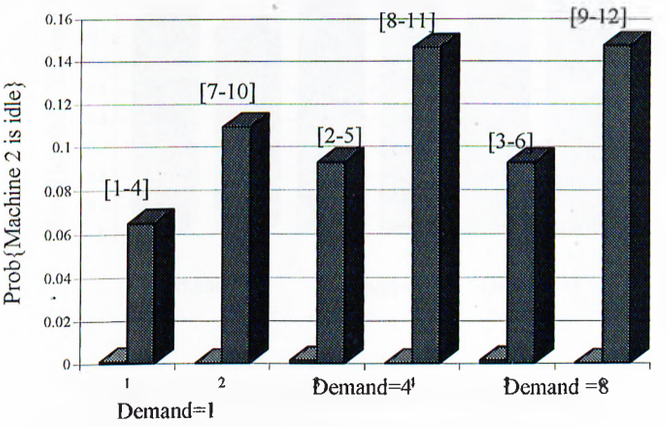

Figure 3.19 : Idleness of Mach.2 versus Demand for different parameters and for N=2

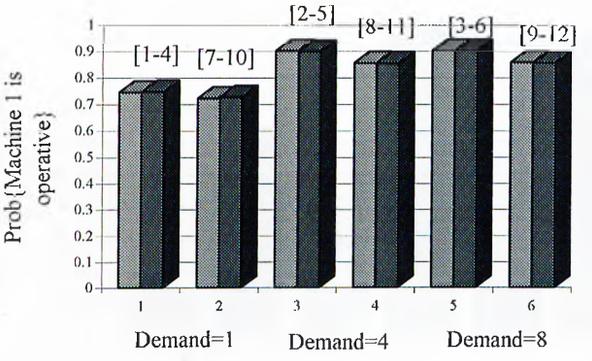

Figure 3.20 : Uptime for Mach.1 versus Demand for different parameters and for N=2

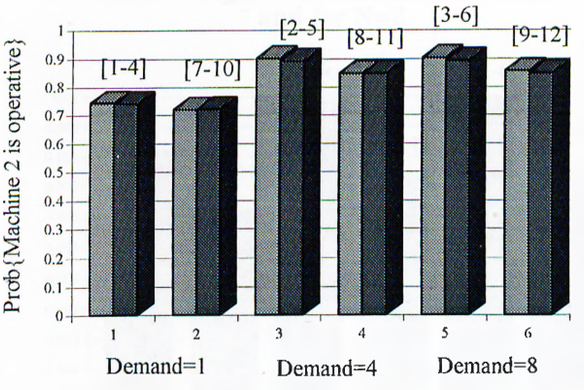

Figure 3.21 : Uptime for Mach.2 versus Demand for different parameters and for N=2

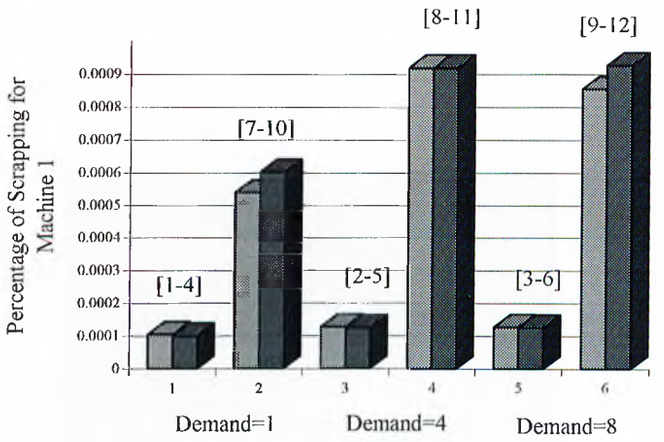

Figure 3.22 : Scrapping for Mach.1 versus Demand for different parameters and for N=2

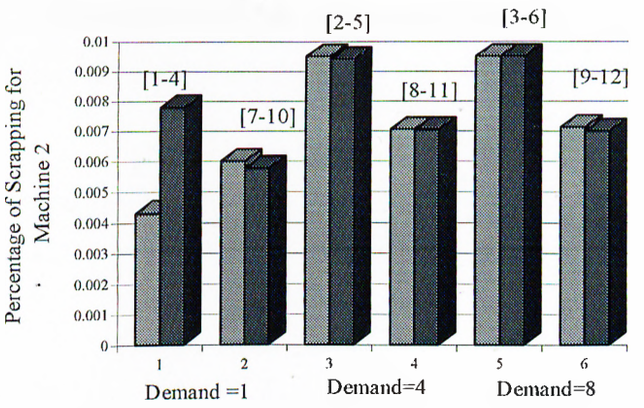

Figure 3.23 : Scrapping for Mach.2 versus Demand for different parameters and for N=2

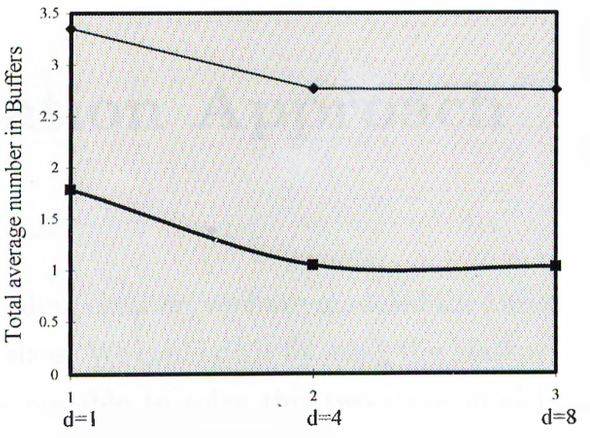

Figure 3.24 : Parts in buffers versus Demand

Chapter 4

Simulation Approach

In the preceding chapter, we have modeled the two-stage Pull systems with general buffer sizes. We could do it by using the Markov-Chain representations. Although it is possible to solve this two-stage model analytically, when the buffer size gets larger it becomes very time consuming to solve the model. When the number of stages increases, analytic approach becomes inefficient since it becomes too difficult to model and to solve the system.

The simulation approach to solve this N-Stage Pull system helps us in that case. We just model the system in a simulation environment and obtain the results we need.

4.1 Two-Stage Model

To simulate this two-stage model, we have used the package PromodelPC 2.0. Since Promodel uses modular approach in an interactive fashion, it is very easy to model the system.

The assumptions and the mechanics of the system is the same as those of the analytical model. However, since simulation is a dynamic environment we have used the random number generators and the associated distributions.

4.1.1 Algorithm of the Program

The model development in Promodel is conducted in modules. Therefore, the algorithm of our simulation program for two-stage Pull systems are explained in modules.

Initialization Logic

- Generate geometric distributions for the uptime periods for each machine.
- Put each value of these geometric distributions to arrays
- Generate geometric distributions for the downtime or under repair period for each machine.
- Put each value of these geometric distributions to arrays.

Locations

- Construct machines, buffers, stores and demand arrival places
- Define usage downtimes for each machine. Usage downtime means that machine breaks down according to how much time that machine has been operative.

Entities

- Construct the unit processed and the demand arrived

Path Networks

- Construct a path between each two places on which unit processed moves.

- Define an interface (a location) for each path.

Processing

Processing is divided into two parts for each location, Process part and the Routing part.

- Process Part : Define all processes (waiting, manufacturing, splitting, joining, definitions etc.) for each location.
- Routing Part : After the unit finishes the operation on each cycle, it must move to another location. The locations and the rules according to which part moves (probabilities) are defined in routing part. The scrapping probability is also defined in this part.

Variables

- All global variables are defined in the variables module.

Subroutines

- The procedures that are called from other modules are defined in the Subroutine module.

4.1.2 Generation of Geometric Distribution

Consider the geometric distribution with *pmf*

$$p(x) = p(1-p)^{x-1} \quad x = 0, 1, 2...$$

and with *cdf*

$$F(x) = 1 - (1-p)^{x+1} \;\; ; \; x = 0, 1, 2...$$

X can be generated by using

$$X = 1 + \lceil \frac{ln(1-R)}{ln(1-p)} - 1 \rceil$$

where R is the (0,1) uniform random variable and $\lceil . \rceil$ is the round-up function.

4.1.3 Validity of the Model

To check the validity of the model, we have run it 450 hours with 50 hours warmup period in five replications. While choosing the warmup period, we have considered the minimum error between the analytical model and the simulation model. The parameters used are the same as in Experiment 3. The performance measures and the notations used for them in the tables are summarized below just to remind them.

a_1 : Average number of parts in buffer 1
a_2 : Average number of parts in buffer 2
b_1 : Probability of Machine 1 is blocked
b_2 : Probability of Machine 2 is blocked
c_1 : Probability of Machine 1 is idle
c_2 : Probability of Machine 2 is idle
d_1 : Percentage of uptime for Machine 1
d_2 : Percentage of uptime for Machine 2
e_1 : Percentage of scrapping for Machine 1
e_2 : Percentage of scrapping for Machine 2

The parameters used for this experiment and the results are shown in Table 4.1, Table 4.2, Table 4.3 and Table 4.4.

As we can see from these tables, the simulation model is almost the exact representation of the analytic model. We could get better results by increasing the simulation run time period and the number of replications. However, these results seem satisfactory for practical purposes.

Run#	p_1	p_2	q_1	q_2	r_1	r_2	λ_D
1	0.01	0.1	0.7	0.95	0.01	0.1	1
2	0.01	0.1	0.7	0.95	0.01	0.1	4
3	0.01	0.1	0.7	0.95	0.01	0.1	8
4	0.1	0.01	0.95	0.70	0.1	0.01	1
5	0.1	0.01	0.95	0.70	0.1	0.01	4
6	0.1	0.01	0.95	0.70	0.1	0.01	4
7	0.001	0.15	0.70	0.90	0.90	0.05	1
8	0.001	0.15	0.70	0.90	0.90	0.05	4
9	0.001	0.15	0.70	0.90	0.90	0.05	8
10	0.15	0.001	0.90	0.70	0.05	0.90	1
11	0.15	0.001	0.90	0.70	0.05	0.90	4
12	0.15	0.001	0.90	0.70	0.05	0.90	8

Table 4.1: Parameters used in Experiment 4

Run#	$a_1(An.)$	$a_1(Sim.)$	$a_2(An.)$	$a_2(Sim.)$	$b_1(An.)$	$b_1(Sim.)$	$b_2(An.)$	$b_2(Sim.)$
1	1.89	1.88	1.46	1.45	0.225	0.2232	0.174	0.1753
2	1.855	1.87	0.9084	0.768	0.0837	0.077	0.00026	0.0002
3	1.8572	1.879	0.893	0.832	0.0835	0.0745	0	0
4	0.3169	0.3556	1.473	1.4823	0.0058	0.001	0.179	0.1885
5	0.1327	0.075	0.9083	0.6534	0.00139	0.00416	0.0002	0.0002
6	0.132	0.1377	0.8924	0.8629	0.0014	0.0014	0	0
7	1.981	1.98	1.406	1.41	0.2736	0.2758	0.1567	0.1663
8	1.98	1.98	0.866	0.728	0.1406	0.1584	0.0002	0.0002
9	1.98	1.98	0.85	0.87	0.1410	0.1358	0	0
10	0.545	0.1630	1.4212	1.406	0.0002	0	0.1643	0.1666
11	$7.12 * 10^{-3}$	$3.6 * 10^{-2}$	0.84	0.84	0	0.0068	0.0018	0.0004
12	$7.97 * 10^{-3}$	$6.6 * 10^{-3}$	0.84	0.88	0	0	$\cdot 0$	0

Table 4.2: Results for Experiment 4

Run#	$c_1(An.)$	$c_1(Sim.)$	$c_2(An.)$	$c_2(Sim.)$	$d_1(An.)$	$d_1(Sim.)$
1	0.017	0.0155	$8.12 * 10^{-4}$	$2.9 * 10^{-3}$	0.744	0.7491
2	0	0	$1.42 * 10^{-3}$	0.015	0.901	0.898
3	0	0	$1.43 * 10^{-3}$	$1.2 * 10^{-3}$	0.9024	0.9147
4	0.162	0.1727	0.0649	0.0631	0.7468	0.7416
5	0.0002	0	0.0926	0.1023	0.899	0.8982
6	0	0	0.0925	0.0864	0.9021	0.909
7	$1.69 * 10^{-3}$	$1.04 * 10^{-3}$	0	0	0.7213	0.7220
8	0	0	0.0001	0.0002	0.8558	0.8405
9	0	0	0.0001	0	0.8568	0.8633
10	0.1481	0.1526	0.1091	0.1097	0.7278	0.7270
11	0.00018	0	0.1464	0.1439	0.8538	0.8539
12	0	0	0.1473	0.14574	0.855	0.8583

Table 4.3: Results for Experiment 4

Run#	$e_1(An.)$	$e_1(Sim.)$	$e_2(An.)$	$e_2(Sim.)$	$d_2(An.)$	$d_2(Sim.)$
1	$1.04 * 10^{-4}$	$1.2 * 10^{-4}$	$4.3 * 10^{-3}$	$7.27 * 10^{-3}$	0.744	0.7489
2	$1.27 * 10^{-4}$	$2.33 * 10^{-4}$	$9.49 * 10^{-3}$	$8.59 * 10^{-3}$	0.901	0.898
3	$1.27 * 10^{-4}$	$1.06 * 10^{-4}$	$9.49 * 10^{-3}$	$8.35 * 10^{-3}$	0.9024	0.9147
4	$7.79 * 10^{-3}$	$6.88 * 10^{-3}$	10^{-4}	$1.31 * 10^{-4}$	0.7419	0.7352
5	$9.4 * 10^{-3}$	$10 * 10^{-3}$	$1.24 * 10^{-4}$	$8.33 * 10^{-5}$	0.8912	0.8890
6	0.00949	0.0088	$1.2 * 10^{-4}$	$1.2 * 10^{-4}$	0.8934	0.902
7	$5.4 * 10^{-3}$	$9.36 * 10^{-4}$	$5.99 * 10^{-3}$	$5.62 * 10^{-3}$	0.7207	0.7210
8	$9.18 * 10^{-4}$	$9.36 * 10^{-4}$	$7.05 * 10^{-3}$	$7.98 * 10^{-3}$	0.8488	0.8399
9	$8.55 * 10^{-4}$	$7.47 * 10^{-4}$	$7.13 * 10^{-3}$	$6.87 * 10^{-3}$	0.8565	0.8625
10	$5.75 * 10^{-3}$	$6.01 * 10^{-3}$	$6.03 * 10^{-4}$	$1.125 * 10^{-3}$	0.7217	0.7222
11	$7.04 * 10^{-3}$	$6.95 * 10^{-3}$	$9.18 * 10^{-4}$	$7.3 * 10^{-3}$	0.8489	0.8475
12	$7 * 10^{-3}$	$7.08 * 10^{-3}$	$9.27 * 10^{-4}$	$6.3 * 10^{-4}$	0.849	0.8535

Table 4.4: Results for Experiment 4

4.2 N-Stage Simulation Model

After modeling the two-stage Pull system and obtaining satisfactory results with small errors between the analytical model and the simulation model, we have extended this model to a 20-Stage model. We have thought that a 20-Stage model becomes a good representative for the general N-Stage models.

The assumptions and the constructions for each location (machine, buffer, demand arrival etc.) are the same as the those of two-stage model. The only difference is the number of machines and the number of buffers. In this case, there are 20 machines and 20 buffers.

In the construction of this 20-Stage model, each machine had a scrap place. The aim for this is to see the percentage of scrap for each machine. In reality, a scrap place may be used by more than one workplace or machine simultaneously.

Experiment 5 :

We have done some experiments in the analytical and two-stage simulation model. Now, we carry another experiment with this extended model by using different parameters. The aim of this experiment is to see the performance measures of transfer lines in the pull environments.

We have again run the model 450 hours with 50 hours warmup period and with five replications. Actually, the warmup period needed to stabilize this extended model must be longer than the previous model. However, since we have worked on a PC to get the runs, speed and the memory limitations of PC environment prevented us to experiment with longer runs. In spite of all these drawbacks, we have obtained logical and satisfactory results when compared with two-stage model. In this experiment, we have only played with the parameters of machines 1,5,10,15 and 20. Therefore, the appropriate parameter setting are shown in Table 4.5 by considering this fact.

Run#	p_1	p_5	p_{10}	p_{15}	p_{20}	q_1	q_5	q_{10}	q_{15}	q_{20}	r_1	r_5	r_{10}	r_{15}	r_{20}	λ_D
1	0.01	0.01	0.01	0.01	0.01	0.95	0.95	0.95	0.95	0.95	0.01	0.01	0.01	0.01	0.01	1
2	0.01	0.1	0.1	0.1	0.01	0.95	0.7	0.7	0.7	0.95	0.01	0.1	0.1	0.1	0.01	1
3	0.1	0.01	0.01	0.01	0.01	0.7	0.95	0.95	0.95	0.95	0.1	0.01	0.01	0.01	0.01	1
4	0.01	0.01	0.01	0.01	0.1	0.95	0.95	0.95	0.95	0.7	0.01	0.01	0.01	0.01	0.1	1
5	0.1	0.1	0.1	0.1	0.1	0.7	0.7	0.7	0.7	0.7	0.1	0.1	0.1	0.1	0.1	1
6	0.01	0.01	0.01	0.01	0.01	0.95	0.95	0.95	0.95	0.95	0.01	0.01	0.01	0.01	0.01	2
7	0.01	0.1	0.1	0.1	0.01	0.95	0.7	0.7	0.7	0.95	0.01	0.1	0.1	0.1	0.01	2
8	0.1	0.01	0.01	0.01	0.01	0.7	0.95	0.95	0.95	0.95	0.1	0.01	0.01	0.01	0.01	2
9	0.01	0.01	0.01	0.01	0.1	0.95	0.95	0.95	0.95	0.7	0.01	0.01	0.01	0.01	0.1	2
10	0.1	0.1	0.1	0.1	0.1	0.7	0.7	0.7	0.7	0.7	0.1	0.1	0.1	0.1	0.1	2
11	0.01	0.01	0.01	0.01	0.01	0.95	0.95	0.95	0.95	0.95	0.01	0.01	0.01	0.01	0.01	4
12	0.01	0.1	0.1	0.1	0.01	0.95	0.7	0.7	0.7	0.95	0.01	0.1	0.1	0.1	0.01	4
13	0.1	0.01	0.01	0.01	0.01	0.7	0.95	0.95	0.95	0.95	0.1	0.01	0.01	0.01	0.01	4
14	0.01	0.01	0.01	0.01	0.1	0.95	0.95	0.95	0.95	0.7	0.01	0.01	0.01	0.01	0.1	4
15	0.1	0.1	0.1	0.1	0.1	0.7	0.7	0.7	0.7	0.7	0.1	0.1	0.1	0.1	0.1	4

Table 4.5: Parameters used in Experiment 5

In the two-stage models (both analytical and simulation models), when we increased the demand, total average number of parts in buffers decreased and leveled at a fixed value. In our 20-Stage simulation model, we see that when we increase the demand size total average number of parts in the buffers increase (see Figure 4.2). Although this result at first seems to contradict with the previous two models, in fact it does not. This result is mainly due to the number of stages in the system, in other words, when the number of stages increase, full utilizations for the machines are reached up at greater demand sizes. Therefore, since the machines are not fully utilized, they increase the number of parts in the buffers when demand size increases.

When all machines are highly reliable (i.e., $p_i = 0.01$ and $q_i = 0.95$, $i = 1, ..., 20$), the percentages of blockages for machines show interesting behaviours. As it is seen in Figure 4.3, percentage of blockages for machines increases while reaching the end of the line when the demand size is one. For the other demand sizes (i.e., $d = 2$ and $d = 4$), the behaviour deviates slightly but it is almost the same.

Another interesting result was obtained when machine numbers 5, 10 and 15 are unreliable and demand size is one. When we move from the first machine to the last machine, the blockage oscillates (see Figure 4.4). When we reach

to the unreliable machine, it goes down and when we move forward from the unreliable machine it goes up. This result may be due to that unreliable machines behave as bottleneck machines. When the demand size is small, inaddition to unreliable machines, demand also becomes bottleneck. Since as it is seen from Figure 4.4, blockage increases significantly at the last buffer. For the large demand sizes, the same types of oscillations are obtained, but with small magnitudes.

When the machines 1, 5, 10, 15, 20 are unreliable, number of parts in the buffers oscillates too(see Figure 4.5.). While reaching to the downstream buffer of unreliable machine, the number of parts in the buffers increase and at the downstream buffer of unreliable machine it decreases sharply. After this buffer, number of parts in buffers again begins to increase. This result may be again due to the bottleneck behaviour of the unreliable machines.

When we exchange the first and last machines (i.e., we put the unreliable machine at the end), we see that percentage of uptime increases 2% contrary to the two-stage model(see Figure 4.6.). Although uptime percentages increase for the machines, total average number of parts in the buffers increases 46% (see Figure 4.7.). Therefore, putting the unreliable machine at the beginning and the reliable machine at the end is again more desirable.

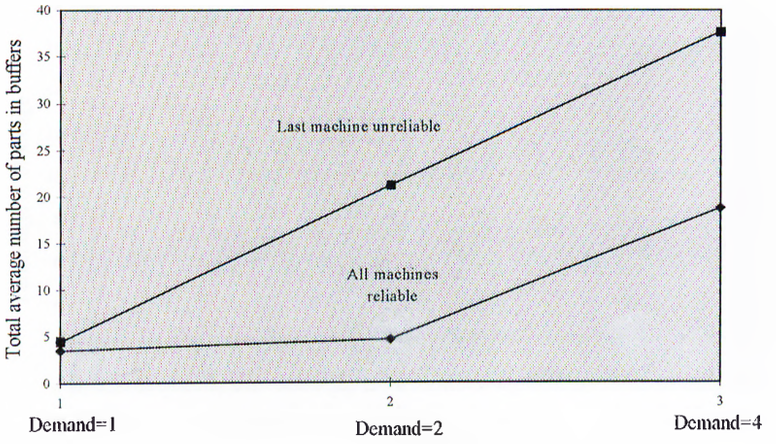

Figure 4.1 : Parts in buffers versus Demand for K=20

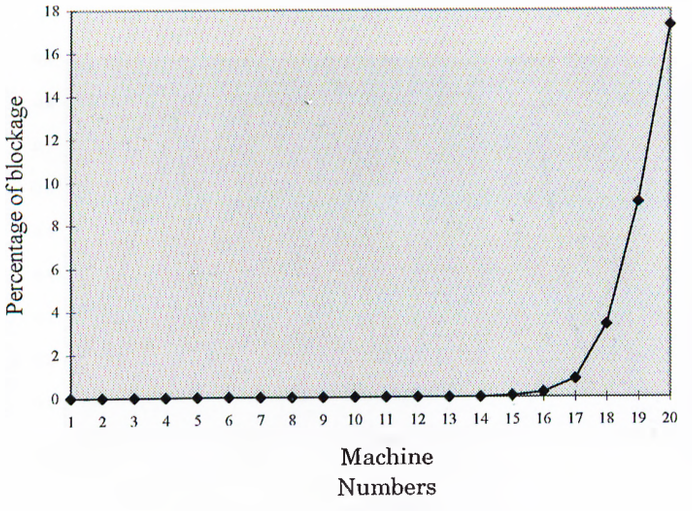

Figure 4.2 : Blockage versus Machine Numbers

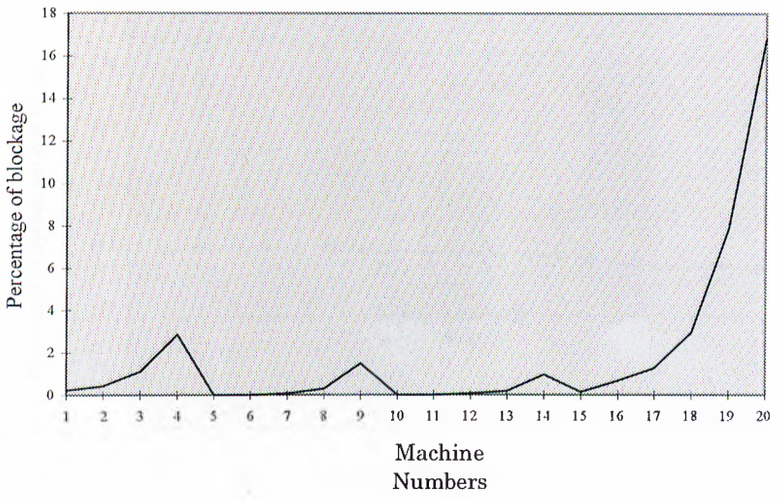

Figure 4.3 : Blockage versus Machine numbers

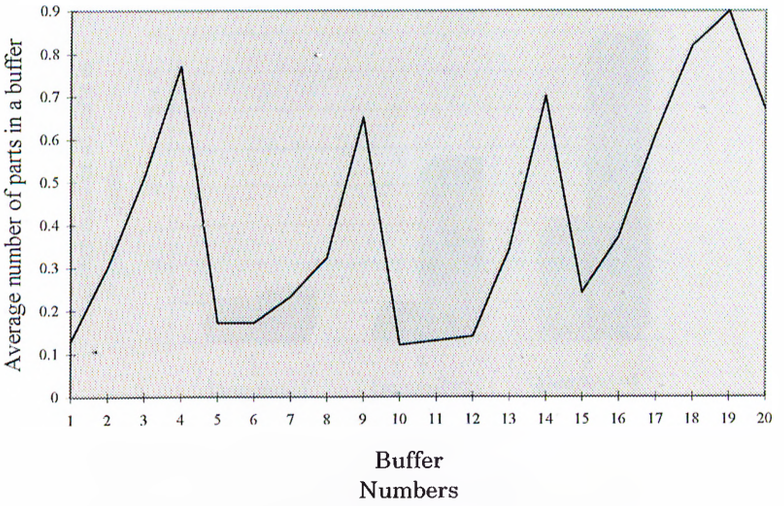

Figure 4.4 : Parts in buffers versus Buffer numbers

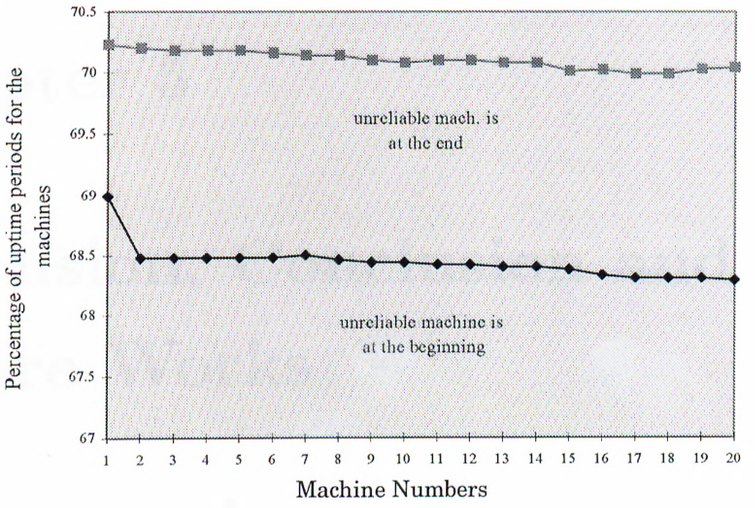

Figure 4.5 : Percentage of Uptimes versus Machine Numbers

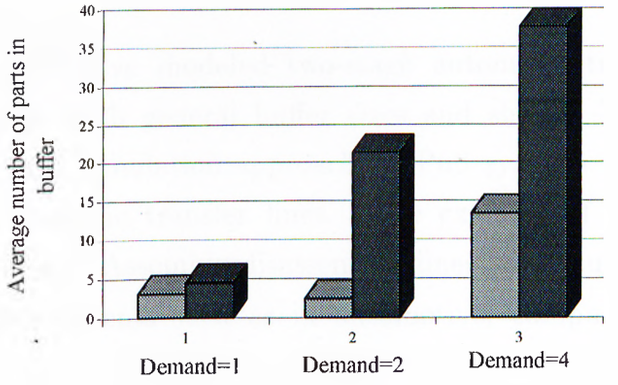

Figure 4.6 : Parts in buffers versus Demand for different parameters and for K=20

Chapter 5

Extension, Conclusion and Future Works

5.1 Extensions

So far, we have modeled two-stage automatic transfer line in the Pull environment with general buffer sizes and showed how to solve it by using analytic and simulation approaches. Pull systems are very effectively used in the automatic transfer lines. One example of these is assembly-disassembly lines. Assembly-disassembly lines are usually used in mass production environments such as in automotive, computer and electronics industries.

The model we have constructed is a small representation of assembly-disassembly lines in the Pull environments. We have again used capacitated buffers to increase the system performance. The representation of the model can be seen in Figure 5.1. Description of the system together with limiting assumptions are provided in succeeding section.

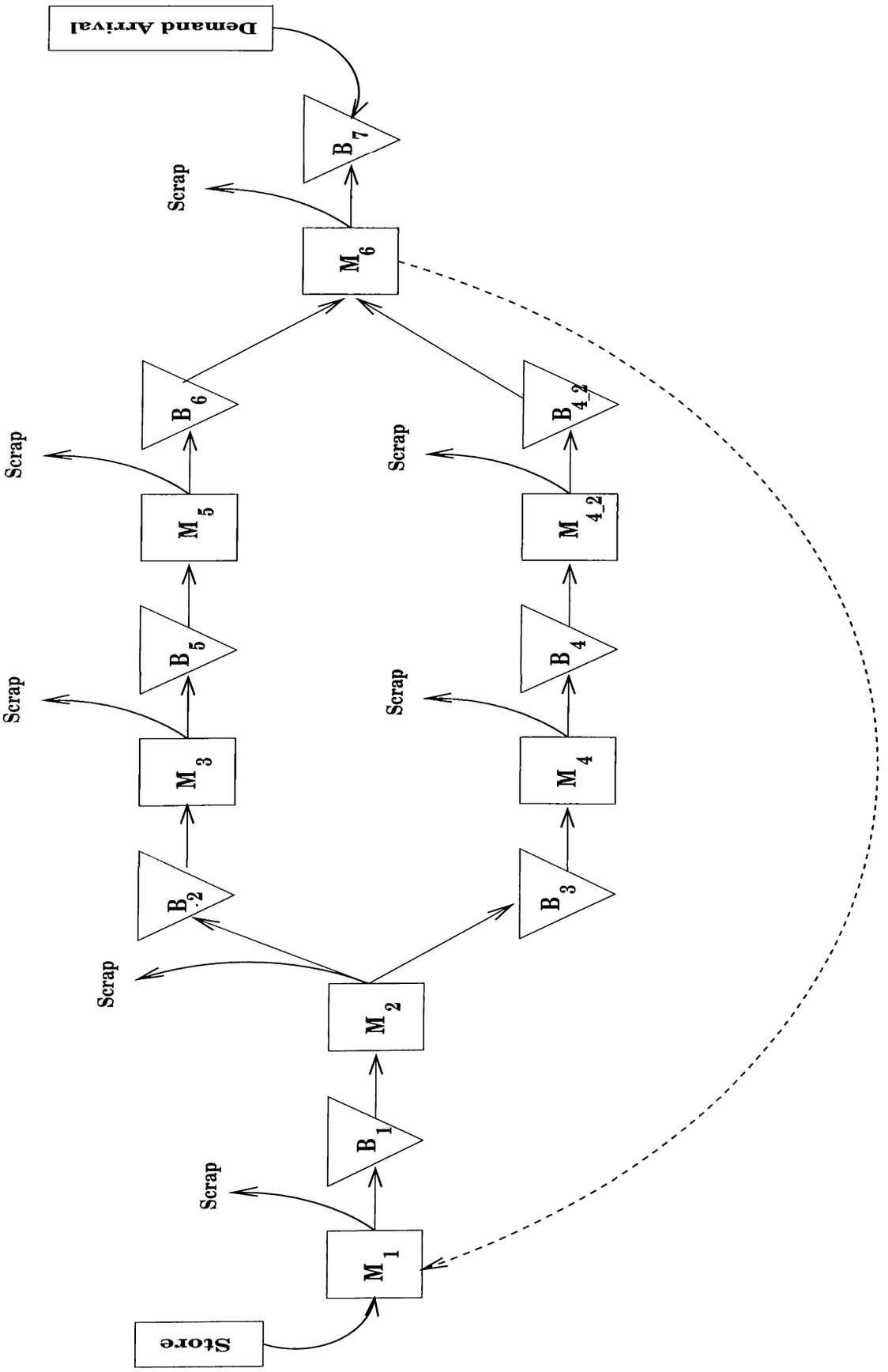

Figure 5.1: Assembly-Disassembly Systems in the Pull Environment

5.1.1 Description of the Model

Since the system works in Pull environment production process starts with the demand arrival. When the demand arrives, if there is a part in the last buffer, it is satisfied; if there is no available part in the last buffer, demand is lost. In other words, backlogging is not permitted. When the demand takes the part in the last buffer, last machine sends a command to the first machine to produce a new part. When the first machine gets this command, it starts to produce the part and sends it to buffer 1 after finishing the operation. Machine 2 gets a part from buffer 1 and processes it. After finishing the production on Machine 2, the part is disassembled into two parts. One goes to buffer 2 and the other goes to buffer 3. Each part moves on machines and buffers in different but parallel lines up to machine 6. At machine 6, one part from each line is taken and assembled into a new part. This new part is processed on machine 6 and sent to buffer 7. With a new demand arrival, cycle starts over again.

Assumptions :

- Each machine uptime is geometrically distributed with parameter p_i and each machine downtime or repair time is geometrically distributed with parameter q_i, $i = 1...6$
- Each machine scraps the material after the downtime period with probability r_i, $i = 1...6$
- Demand arrives according to Poisson distribution with parameter λ_D
- Buffers are capacitated (i.e., N is finite)
- There is no shortage of material.
- No backlogging is permitted.
- Parts are processed according to Pull systems' rules.

- Parts are disassembled into two parts which have the same properties with the original part.
- Parts are assembled into one part which has the same properties with the assembled parts.
- System is synchronized

5.1.2 Simulation Model

We have modeled this system in PromodelPC 2.0. Since Promodel is a modular program, we again constructed the model with modules. The algorithm of the simulation model we have constructed is given below.

5.1.3 Algorithm

Initilization Logic

- Generate geometric distributions for the uptime periods of each machine and put them into arrays
- Generate geometric distributions for the downtime periods of each machine and put them into arrays

Locations

- Construct each location (buffers, machines,store and demand arrival)
- Define usage downtime for each machine

Entities

- Construct each entity(unit, demand, subassemblies and assembled part)

Path Networks

- Construct a path between each location so that part can move on this path
- Define interfaces for each path

Processing

- Define processing on machines i, i=1,3,4,5
- Define processing and then splitting at machine 2
- Define assembling and then processing at machine 6
- Define routings from location to location

Arrivals

- Define each demand arrival

Variables

- Define all global variables used in the other modules of the program

Arrays

- Construct an array for each distribution

Subroutines

- Define subroutines used in the processing part of the program.

5.2 Conclusion

The major contribution of this study to the literature is to reconsider the transfer lines under pull environment. Two-stage analytical model of this system is also provided in this study. Furthermore, experiments with different parameters were carried out to see the performance measures of the model. One of the important results of these experiments is that more reliable machines are pushed to the end of the line to decrease the WIP inventory level in the system. The main aim of the pull systems is to decrease the in-process inventory. Therefore, by just exchanging the positions of the machines on the line a significant decrease in WIP level can be obtained.

Since the two-stage transfer lines rarely arise in practical applications, a generalized N-stage model is needed to study the behaviour of the real life systems. Because it is difficult to treat N-stage analytical models, a simulation approach is utilized to model and to analyze the system. After performing experiments with different parameters, we have observed that WIP levels in the buffers between the reliable machines fluctuate. When the total number of parts in the buffers are considered, it is observed that the more reliable machines must be laid at the end of the line to decrease the total WIP level in the system.

Another contribution of this study is to model the assembly-disassembly systems under pull environment by using the simulation approach. The

modeling and the analysis of complicated practical systems are effectively realized through the use of powerful tools such as simulation.

5.3 Future Works

There are dozens of papers in the literature dealing with transfer lines. However, papers dealing with transfer lines under pull environment are very few. Reconsideration and remodeling of the transfer lines previously worked on can be a future work. General behaviour in the literature is to model two or three stage transfer lines analytically and to extend these models to N-stage models by using simulation or approximation methods such as decomposition. This is because of the large state space of the Markov-Chain representation of the models. However, N-stage systems can be modeled analytically and the Markov-Chain representation of these models can be solved by writing effective computer programs in a high level language such as C++ and Fortran.

Another feature of the literature is that most of the papers deal with steady-state performance measures. However, variability is very important in manufacturing. A study about the variability of transfer lines under pull environment will be a good contribution to the literature.

Pull systems are applied not only in transfer lines or mass production lines but also in batch processing , cell structure and even in job-shop environments. Analytical and simulation modeling of these different environments under pull policy can be made and the comparisons between these environments under pull policy can be done.

A push version of our model can also be constructed and comparisons between measures can be made.

The limiting assumptions on the systems can be relaxed and more realistic systems can be modeled. These new assumptions might be as follows :

- System is not synchronized.
- Backlogging is allowed.
- Reworking on the parts is allowed.
- Different pull policies may be used. For example, the machine at the end of the line sends command to directly its upstream machine.
- Zero or infinite buffer sizes can be used.
- State space of the machines can be enlarged

Bibliography

[1] Andijani, A. and Clark, G.M., "Kanban allocation to serial production lines in a stocastic environment", *Just in Time Manufacturing Systems, Elsevier Science Publishers B.V., Satir, A., Editor*, 1991

[2] Askin, R.G. and Mitwasi, M.G., Goldberg, J.B., "Determining the number of kanbans in multiitem Just in Time systems", *Just in Time Manufacturing Systems, Elsevier Science Publishers B.V., Satir, A., Editor*, 1991

[3] Buzacott, J.A., "Automatic transfer lines with buffer stocks" , *Int. J. Prod. Res.*, 1967 vol.5 183-200

[4] Buzacott, J.A., "Methods of reliability analysis of production systems subject to breakdowns", *Operations Research and Reliability , D. Grouchko, Editor, Gordon and Breach*, 1967 211-232

[5] Co, H.C. and Jacopson, S.H., "The kanban problem in serial just in time production systems", *IIE Transactions*, 1994 vol.26 76-85

[6] Conway, R., Maxwell, J.M., Thomas, L.J., "The role of work-in process inventory in serial production lines", *Operations Research*, 1988 vol.36 229-241

[7] Corbett, C. and Yucesan, E., "Modeling just-in-time systems", *Winter Simulation Conference Proceedings, Publ. by IEEE*, 1993 819-827

[8] Durmusoglu, M.B., "Comparison of push and pull systems in a celluar manufacturing environment", *Just in Time Manufacturing Systems, Elsevier Science Publishers B.V., Satir, A., Editor*, 1991

[9] Galbraith, L., Miller, W.A. and Suresh, S., "Identifications of metrics and transition functions of pull system implementation", *Computer Integrated Manufacturing Systems*, 1993 vol.6 117-124

[10] Gershwin, S.B., "Manufacturing Systems Engineering", Prentice Hall., 1994

[11] Gershwin, S.B., "Efficent decomposition method for the approximate evaluation of production line with finite storage space", *Proceedings of 6 th international conference on the analysis and Optimizations of Systems, Spring-Verlag Lecture Note Series on Control and Information Sciences*, 1984 63

[12] Ho. Y.C. , Eyler, M.A., and Chien, T.T., "Gradient technique for general buffer storage design in a production line", *International J. of Prod. Res.*, 1979 vol.17 557-580

[13] Ho. Y.C. , Eyler, M.A., and Chien, T.T., "A New approach to determine parameter sensitivities on transfer lines", *Harvard Division of Applied Sciences, Cambridge, Mass.*, 1979

[14] Ignall, E. and Silver, A., "The output of a two-stage system with unreliable machines and limited storage", *AIIE Transaction*, 1977 vol.9 183-188

[15] Jafari, M.A. and Shantikumar, J.G., "An approximate model of multistage automatic transfer lines with possible scrapping of workpieces", *IIE Transactions*, 1987 vol.19 252-265

[16] Jafari, M.A. and Shantikumar, J.G., "Exact and approximate solutions to two-stage transfer lines with general uptime and downtime distributions", *IIE Transactions*, 1987 vol.19 412-420

[17] Lingayat, S., O'Keefe, R.M. and Mittenthal, J., "Order release mechanisms : A step towards implementing Just in Time manufacturing", *Just*

in Time Manufacturing Systems, Elsevier Science Publishers B.V., Satir, A., Editor, 1991

[18] Maskell, B.H., "Just in Time, Implementing the New Strategy", Hitchcock Publishing Co., 1989

[19] Meral, S. and Erkip, N., "Design and analysis of a Just in Time production line", *Just in Time Manufacturing Systems, Elsevier Science Publishers B.V., Satir, A., Editor*, 1991

[20] Muckstadt, J.A. and Tayur, S.R., "A comparison of alternative kanban control mechanisms. I.Background and structural results", *IIE Transactions*, 1995 vol.27 140-150

[21] Muckstadt, J.A. and Tayur, S.R., "A comparison of alternative kanban control mechanisms. II.Experimental results", *IIE Transactions*, 1995 vol.27 151-161

[22] Okamura, K. and Yamashina, H., "Analysis of the effect of buffer storage capacity in transfer lines systems", *AIIE Transaction*, 1977 vol.9 127-135

[23] Ramudhin, A. and Rochette, R., "Just in Time practices in an unstable environment : A simulation study", *Just in Time Manufacturing Systems, Elsevier Science Publishers B.V., Satir, A., Editor*, 1991

[24] Sevastyanov, B.A., "Influence of storage bin capacity on the average standstill time of production line", *Theory of Probability and Its Applications*, 1967 vol.7 429-438

[25] Shanthikumar, J.G., "On the production capacity of automatic transfer lines with unlimited buffer space", *AIIE Transactions*, 1980 vol.12 273-274

[26] Shantikumar, J.G. and Tien, C.C., "An algorithmic solution to two-stage transfer lines with possible scrapping of units", *Management Science*, 1983 vol.29 1069-1084

[27] Sheskin,T.J. , "Allocation of interstage storage along an automatic transfer line", *AIIE Transactions*, 1976 vol.8 146-152

[28] Spearman, M.L. and Zazanis, M.A., "Push and pull production systems, issues and comparisons", *Operations Research*, 1992 vol.40 521-532

[29] Slobodow, B.L., "Simulation of a plant-wide inventory pull system", *Winter Simulation Conference Proceedings, Publ. by IEEE*, 1993 904-907

[30] Wang, H. and Wang, H.B., "Decomposition and optimal design of kanban systems", *Just in Time Manufacturing Systems, Satir A. (Editor), Elsevier Science Publishers B.V.*, 1991

VITA

Erdem Eskigün was born on September 22, 1972 in Pazarcık, K.Maraş, Turkey. He received his high school education at Pazarcık Lisesi, K.Maraş, Turkey. He has graduated from the Department of Industrial Engineering, Boğaziçi University, in 1994. In October 1994, he joined to the Department of Industrial Engineering at Bilkent University for his graduate study. From that time to the present, he worked with Assoc. Prof. Cemal Dinçer at the same department.