

**DOKUZ EYLÜL UNIVERSITY  
GRADUATE SCHOOL OF NATURAL AND APPLIED  
SCIENCES**

**SOLVING BUFFER ALLOCATION PROBLEM IN  
PRODUCTION LINES USING TABU SEARCH  
BASED APPROACHES**

by  
**Leyla DEMİR**

**April, 2011  
İZMİR**

# **SOLVING BUFFER ALLOCATION PROBLEM IN PRODUCTION LINES USING TABU SEARCH BASED APPROACHES**

**A Thesis Submitted to the  
Graduate School of Natural and Applied Sciences of Dokuz Eylül University  
In Partial Fulfillment of the Requirements for the Degree of Doctor of  
Philosophy in Industrial Engineering, Industrial Engineering Program**

**by  
Leyla DEMİR**

**April, 2011  
İZMİR**

## Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled “**SOLVING BUFFER ALLOCATION PROBLEM IN PRODUCTION LINES USING TABU SEARCH BASED APPROACHES**” completed by **LEYLA DEMİR** under supervision of **PROF. DR. SEMRA TUNALI** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.



Prof. Dr. Semra TUNALI

Supervisor



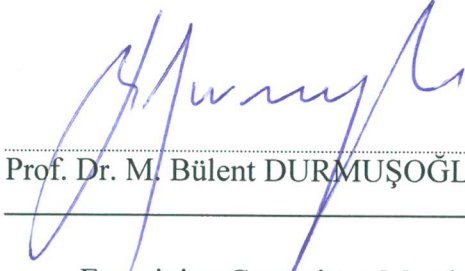
Assoc. Prof. Dr. M. Arslan ÖRNEK

Thesis Committee Member



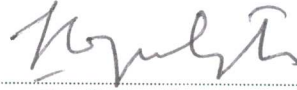
Asst. Prof. Dr. Deniz TÜRSEL ELİİYİ

Thesis Committee Member



Prof. Dr. M. Bülent DURMUŞOĞLU

Examining Committee Member



Assoc. Prof. Dr. Şeyda A. TOPALOĞLU

Examining Committee Member



Prof. Dr. Mustafa SABUNCU

Director

Graduate School of Natural and Applied Sciences

## ACKNOWLEDGMENTS

First and foremost, I would like to thank my supervisor Prof. Dr. Semra Tunalı, for her guidance, insights and encouragement throughout this Ph.D. study. Her inspiration, guidance and counsel throughout the period of my study at Dokuz Eylül University were invaluable. Her advice always gave me the direction especially when I was lost during the research. She always encouraged me when I was desperate. She is not only a Ph.D. supervisor for me but also a great friend who listens all my problems and continuously supports me. It was extremely helpful for my academic career to have a chance to work with her.

I would like to express my appreciation to the other members of my thesis committee. I would like to thank Assoc. Prof. Dr. Arslan M. Örnek for his continuous support during my doctoral education at Dokuz Eylül University. I would also like to express my deepest appreciation to Assist. Prof. Dr. Deniz Türsel Eliiyi, for her valuable suggestions and guidance throughout this Ph.D. study. She spared her precious time for me like a co-advisor and continuously supported me during my Ph.D. study. I really appreciate for her friendship and continuous support.

I would also like to extend my gratitude to Prof. Dr. M. Bülent Durmuşoğlu and Assoc. Prof. Dr. Şeyda A. Topaloğlu for accepting to serve on my dissertation committee in the midst of all their activities.

I would like to thank two important people who continuously supported me when I was in Norway for my Ph.D. researches. First, I would like to express my deepest appreciation to Prof. Arne Løkketangen for his valuable suggestions, insights and guidance during my Ph.D. study. I would also like to thank to Prof. Geir Dahl for his continuous support and encouragement when I was in Norway.

I would also like to thank to my colleagues for their guidance during my studies at Dokuz Eylül University. Great thanks to my friends, Hacer Güner Gören and Simge

Yelkenci Köse, who are special for me, for listening to my complaints and providing continuous support and smile whenever I need the most.

Last, but the most, I would like to emphasize my thankfulness with ultimate respect and gratitude to my parents and siblings. The continuous support, care, and love of my family is the source and encouragement of this work. I would like to thank my mother, Sultan Demir and my father, Hüseyin Önder Demir, from the bottom of my heart. I feel extremely lucky to have such wonderful parents who have made many sacrifices over the years to ensure that their children receive high quality education. My sister, Haval Demir, and my brother, Abidin Demiray Demir, have also had a tremendous positive influence on my life. They were always with me whenever I needed. I would like to emphasize my thankfulness to both because of their love, confidence, encouragement and endless support in my whole life.

Leyla Demir  
Izmir, April, 2011

# **SOLVING BUFFER ALLOCATION PROBLEM IN PRODUCTION LINES USING TABU SEARCH BASED APPROACHES**

## **ABSTRACT**

The buffer allocation problem, which involves the distribution of buffer space among the intermediate buffers of a production line, arises in a wide range of manufacturing systems, and it is one of the most important optimization problems faced by manufacturing systems designers. The primary aim of this Ph.D. study is to introduce novel tabu search based solution approaches for solving buffer allocation problem in production lines. In this thesis, the buffer allocation problem is solved in three stages. In the first stage, a novel TS algorithm including new move definitions is proposed to solve the buffer allocation problem under the objective of throughput maximization for homogeneous production lines involving unreliable machines with deterministic processing times. Following a pilot experiment to identify the best TS parameters, the new move definitions for buffer allocation problem are introduced. In the second stage, the problem is extended to non-homogeneous production lines, and an adaptive TS algorithm is proposed to solve the revised problem under the objective of throughput maximization. Besides proposing a new strategy to tune the parameters of TS adaptively during the search, an experimental study is carried out to select an intelligent initial solution scheme among three alternatives so as to decrease the search effort to obtain the best solutions. Finally, in the last stage, three approaches are proposed to solve the buffer allocation problem for non-homogeneous production lines involving unreliable machines with deterministic processing times. These three approaches which integrate binary search, tabu search, and simulated annealing with an adaptive tabu search mechanism aim at minimizing the total buffer size to achieve a desired throughput level. To improve the searching efficiency of TS and SA algorithms alternative neighborhood generation mechanisms are suggested and their performance are tested.

**Keywords:** Buffer allocation problem, Production lines, Tabu search, Simulated annealing

# ÜRETİM HATLARINDA TAMPON STOK DAĞILIMI PROBLEMİ İÇİN TABU ARAMA TABANLI ÇÖZÜM YAKLAŞIMLARI

## ÖZ

Üretim sistemi tasarımcılarının karşılaştığı başlıca eniyileme problemlerinden biri olan tampon stok dağılımı problemi bir üretim sisteminde tampon stokların bu stoklar için ayrılmış alana en iyi şekilde dağıtılmasını içermektedir. Bu doktora tezinin başlıca amacı üretim hatlarında tampon stok dağılımı problemini çözmek üzere özgün tabu arama tabanlı yaklaşımlar sunmaktır. Tampon stok dağılımı problemi bu tezde üç aşamada çözülmüştür. İlk aşamada, deterministik üretim zamanlarına sahip ve bozulmalara maruz kalan makinelerin oluşturduğu homojen üretim hatlarında tampon stok dağılımı problemini çözmek için yeni hareket tanımlarını içeren özgün bir tabu arama algoritması önerilmiştir. En iyi tabu arama parametrelerini belirlemek üzere yapılan bir pilot çalışmadan sonra tampon stok dağılımı problemi için yeni hareket tanımları sunulmuştur. İkinci aşamada, söz konusu problem, homojen olmayan üretim hatlarında tampon stok dağılımı olarak genişletilerek, üretim oranını maksimize etmek amacıyla özgün bir adaptif tabu arama algoritması önerilmiştir. Tabu arama parametrelerini arama süresince adaptif bir şekilde değiştirmek üzere yeni bir stratejinin önerilmesinin yanı sıra, arama için sarf edilen eforun azaltılması amacıyla üç alternatif başlangıç çözümü önerilmiştir. Önerilen bu alternatif başlangıç çözümlerinden birini seçmek üzere de deneysel bir çalışma yürütülmüştür. Bu doktora çalışmasının son aşamasında da bozulmalara maruz kalan ve deterministik üretim zamanlarına sahip makinelerden oluşan üretim hatlarında tampon stok dağılımı problemini çözmek üzere üç ayrı yaklaşım önerilmiştir. İkili arama, tabu arama ve tavlama benzetimi algoritmalarını bir adaptif tabu arama mekanizması ile birleştiren bu üç yaklaşım istenilen üretim oranını sağlamak üzere hattaki toplam tampon miktarını minimize etmeyi amaçlamaktadır. Tabu arama ve tavlama benzetimi algoritmalarının arama etkinliğini artırmak üzere alternatif komşuluk yaratma mekanizmaları önerilmiş ve bunların performansları test edilmiştir.

**Anahtar sözcükler:** Tampon stok dağılımı problemi, Üretim hatları, Tabu arama, Tavlama benzetimi

## CONTENTS

	<b>Page</b>
Ph.D. THESIS EXAMINATION RESULT FORM .....	ii
ACKNOWLEDGMENTS .....	iii
ABSTRACT.....	v
ÖZ .....	vi
<b>CHAPTER ONE – INTRODUCTION .....</b>	<b>1</b>
1.1. Objectives and Motivations .....	1
1.2. Research Methodology.....	2
1.3 Outline of the Thesis .....	4
<b>CHAPTER TWO - BACKGROUND INFORMATION.....</b>	<b>7</b>
2.1 Introduction .....	7
2.2 The Buffer Allocation Problem.....	7
2.2.1 Characteristics of the Buffer Allocation Problem .....	8
2.2.2 Classification of the Problems .....	11
2.3 General Procedure to Solve the Buffer Allocation Problem .....	13
2.3.1 Evaluative Methods .....	15
2.3.2 Generative Methods.....	16
2.4 Background Information on Solution Approaches Employed .....	18
2.4.1 Decomposition Method .....	19
2.4.2 Tabu Search .....	20
2.4.2.1 Search Space and Neighborhood Structure.....	21
2.4.2.2 Tabus.....	21
2.4.2.3 Aspiration Criteria .....	22
2.4.2.4 Termination Criteria.....	22
2.4.2.5 Intensification .....	23
2.4.2.6 Diversification.....	24



2.4.3 Simulated Annealing .....	25
2.4.3.1 Neighborhood Generation Mechanism .....	27
2.4.3.2 Initial Temperature.....	27
2.4.3.3 Cooling Schedule .....	27
2.4.3.4 Final Temperature .....	28
2.4.3.5 Number of Iterations .....	28
2.5 Chapter Summary .....	29
<b>CHAPTER THREE - LITERATURE SURVEY .....</b>	<b>30</b>
3.1 Introduction .....	30
3.2 Proposed Classification Scheme and Discussion of Current Literature .....	31
3.2.1 Reliable Lines .....	32
3.2.2 Unreliable Lines.....	42
3.3 Motivation .....	55
3.4 Chapter Summary .....	57
<b>CHAPTER FOUR - A TABU SEARCH APPROACH FOR THROUGHPUT MAXIMIZATION IN UNRELIABLE HOMOGENEOUS PRODUCTION LINES.....</b>	<b>58</b>
4.1. Introduction .....	58
4.2 Problem Specifications.....	58
4.3 Proposed TS Algorithm.....	60
4.3.1 Move Representation and Tabu Moves .....	60
4.3.2 Search Space and Neighborhood Structure .....	61
4.3.3 Diversification Strategy .....	62
4.3.4 Aspiration Criterion .....	62
4.3.5 Stopping Condition.....	63
4.4 Computational Experiments .....	65
4.4.1 Identification of the Best Tabu Search Parameters.....	65
4.4.2 Experiments on Benchmark Problems .....	67

4.4.2.1 Five Machine Line .....	68
4.4.2.2 Nine Machine Lines .....	68
4.4.2.3 Ten Machine Line .....	69
4.4.2.4 Long Production Lines .....	71
4.5 Chapter Summary .....	73

**CHAPTER FIVE - AN ADAPTIVE TABU SEARCH APPROACH FOR THROUGHPUT MAXIMIZATION IN UNRELIABLE NON-HOMOGENEOUS PRODUCTION LINES.....75**

5.1 Introduction .....	75
5.2 Problem Specifications .....	75
5.3 Proposed ATS Algorithm .....	77
5.3.1 Move Representation and Tabu Moves .....	77
5.3.2 Search Space and Neighborhood Structure .....	78
5.3.3 Initialization Scheme .....	78
5.3.4 Tabu Tenure.....	80
5.3.5 Intensification Strategy .....	80
5.3.6 Diversification Strategy .....	81
5.3.7 Stopping Condition .....	81
5.4 Computational Experiments .....	83
5.4.1 Results of Small-Sized Problems .....	84
5.4.2 Results of Medium-Sized Problems .....	86
5.4.3 Results for Large-Sized Problems .....	89
5.4.4 Summary of the findings .....	93
5.5 Chapter Summary .....	95

**CHAPTER SIX - AN INTEGRATED APPROACH FOR THROUGHPUT MAXIMIZATION WITH MINIMUM TOTAL BUFFER SIZE.....97**

6.1 Introduction .....	97
6.2 Problem Specifications .....	97

6.3 Proposed Integrated Approach .....	99
6.3.1 Binary Search Algorithm .....	99
6.3.2 Tabu Search Algorithm .....	101
6.3.2.1 Search Space .....	101
6.3.2.2 Move Representation and Tabu Moves .....	101
6.3.2.3 Neighborhood Generation Mechanism .....	101
6.3.2.4 Neighborhood Size and Tabu Tenure .....	102
6.3.2.5 Aspiration Criterion .....	102
6.3.2.6 Stopping Condition .....	102
6.3.3 Simulated Annealing Algorithm.....	104
6.4 Computational Experiments .....	106
6.4.1 Determination of Neighborhood Generation Mechanism .....	107
6.4.2 Experiments on Test Problems .....	114
6.4.2.1 Results of Small-Sized Problems.....	115
6.4.2.2 Results of Medium-Sized Problems.....	118
6.4.2.3 Results of Large-Sized Problems.....	121
6.4.3 Summary of the Findings .....	123
6.5 Chapter Summary .....	124
<b>CHAPTER SEVEN – CONCLUSION.....</b>	<b>125</b>
7.1 Summary of the Thesis.....	125
7.2 Contributions .....	127
7.3 Future Research Directions .....	128
<b>REFERENCES.....</b>	<b>130</b>
<b>APPENDICES .....</b>	<b>143</b>

# CHAPTER ONE

## INTRODUCTION

### 1.1 Objectives and Motivations

Production systems are often organized with machines connected in series and separated by buffers. This arrangement is often called a production line. A five-machine line is presented by Figure 1.1 where the squares represent machines and the circles represent buffers. Each part goes through all the machines exactly in same order in the direction of the arrows, from upstream inventory to the first machine for an operation, to the first buffer where it waits for the second machine, to the second machine, etc.

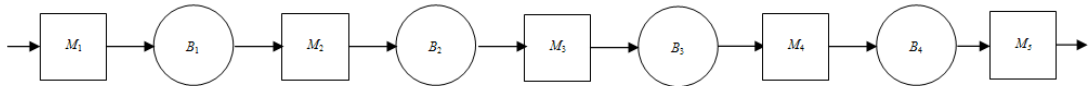


Figure 1.1 Five-machine production line

The performance of such a production line is affected by both variations in the processing times and the reliability parameters of the machines. The effects of such variations can be reduced by using buffers between the machines. Allocating buffers between the machines is to allow machines to operate more independently of each other. This reduces the idle time due to starving (no input available) and blocking (no space to dispose of output). Less idle time increases the average production rate of the line. However, allocating buffers into a production line can be expensive, and there is generally a physical limit to the floor space in the system. The buffer allocation problem (BAP), which is concerned with the allocation of a certain amount of buffers among the intermediate buffer locations of a production line to achieve a specific objective function, is the subject of this Ph.D. thesis.

Due to its importance and complexity, a considerable amount of work has been done in this area. The previous studies in this area mainly focus on characterizing

and describing optimal buffer distributions. In last ten years, the main focus of many research studies has been on developing methods to optimize buffer sizes in production lines.

*The purpose of this Ph.D. thesis is also to construct and describe efficient algorithms for production line design. It is hoped that these algorithms will help manufacturing system designers to determine how buffers should be allocated.*

Generally, the buffer allocation problem is classified into two categories according to the objective function employed to solve this problem. The first one aims at maximizing the throughput rate of the line and the second one focuses on total buffer size minimization. The throughput maximization problem has been studied more extensively in the literature. Moreover, employing meta-heuristic methods to solve buffer allocation problem is a new trend in this area. To better search the solution space, the recent trend is to hybridize the meta-heuristics with other methods. However, a few studies attempt to solve buffer allocation problem by hybrid methods.

*In the light of current relevant literature, this Ph.D. study aims at developing new hybrid approaches to solve buffer allocation problem under the objective of total buffer size minimization.*

## **1.2 Research Methodology**

The general problem involves how to allocate buffers so as to improve the performance of the production line. Solution to this problem depends on the characteristics of the production line studied. In this Ph.D. thesis, the scope of the problem is limited to production lines involving unreliable machines. So, the machines in the line are subject random breakdowns with random repair times.

In general, solution approaches to solve the buffer allocation problem involve a setting where generative methods and evaluative methods are combined in a closed

loop configuration. In such a configuration, an evaluative method is used to obtain the value of the objective function for a set of inputs. The value of the objective function is then communicated to the generative method. Simulation, traditional Markov state models, aggregation methods, generalized expansion method, and decomposition methods are examples of evaluative methods. In this study, the decomposition method is used as an evaluative method due to its ability to obtain the throughput of a production line quite accurately and quickly for unreliable serial lines with deterministic processing times.

There are various optimization techniques used as a generative method. Complete enumeration is the simplest method but it is only applicable for small-sized problems. Since the total number of feasible solutions grows exponentially when the total number of machines and the total buffer capacity increases, it is impossible to employ complete enumeration for large-sized problems. Therefore the researchers employed several traditional optimization and search methods, such as dynamic programming, gradient search methods, Hooke-Jeeves method and knowledge-based methods. However, traditional search methods have disadvantages. The main disadvantage of these methods is that they cannot escape local optima in search of the global optimum. To overcome this difficulty, in recent years, heuristic and meta-heuristic methods, such as simulated annealing (SA), tabu search (TS), genetic algorithms (GA), and ant colony optimization (ACO) are widely used to solve the buffer allocation problem.

Among these meta-heuristics, the application of TS received a considerable attention from the researchers, since it provides an alternative to traditional optimization techniques by using memory-based strategies to escape the local optima and it is also successfully employed on many combinatorial optimization problems. However, as problems get larger and more complex as in real life, basic TS may lack the capability of exploring the search space effectively. As a remedy, over the last years, while some of the studies attempt to employ TS in an adaptive way the others attempt to hybridize TS with other optimization methods.

Within this framework, in this Ph.D. study, the buffer allocation problem is solved in three stages. In the first stage, a TS algorithm is proposed to solve buffer allocation problem under the objective of throughput maximization for unreliable and also homogeneous production lines where all the machines in the line have the same deterministic processing times. Following a pilot experiment to identify the best TS parameters, the new move definitions for buffer allocation problem are introduced.

In the second stage, the problem is extended to non-homogeneous production lines where the processing times of the machines are different, and an adaptive TS algorithm is proposed to solve the revised problem under the objective of throughput maximization. To our knowledge, ours is the first extensive study dealing with buffer allocation problem for unreliable and also non-homogeneous lines. Imposing buffer space constraints for each buffer location makes the problem at hand even harder. Besides proposing a new strategy to tune the parameters of TS adaptively during the search, an experimental study is carried out to select an intelligent initial solution scheme among three alternatives so as to decrease the search effort to obtain the best solutions.

Finally, in the last stage, three approaches are proposed to solve the buffer allocation problem for non-homogeneous production lines involving unreliable machines with deterministic processing times. These three approaches which integrate binary search, tabu search, and simulated annealing with an adaptive tabu search mechanism aim at minimizing the total buffer size to achieve a desired throughput level. To improve searching efficiency of TS and SA algorithms alternative neighborhood generation mechanisms are suggested and their performance are tested.

### **1.3 Outline of the Thesis**

This Ph.D. thesis is organized as follows:

In Chapter 2, to gain a more comprehensive understanding of the problem studied in this Ph.D. thesis, various concepts related to the buffer allocation problem, i.e., characteristics, formulations and the solution methods employed in literature to solve this problem are described. Additionally, the basic concepts of TS and SA which are employed as a solution method in this study are presented.

In Chapter 3, a structural framework is proposed to review the current relevant research on buffer allocation problem in production lines. Using this structural framework, the current research issues are identified and the motivation for this Ph.D. study is presented.

In Chapter 4, a TS approach is proposed to solve the buffer allocation problem in unreliable and homogeneous production lines under the objective of throughput maximization. Prior to using the proposed TS approach, a pilot experiment is carried out to identify the best TS parameters. Next, using these best TS parameters, comparative experiments are carried out on a set of benchmark problems published in the literature.

In Chapter 5, an adaptive TS approach is proposed for solving the buffer allocation problem to maximize the throughput in unreliable and also non-homogeneous production lines. Moreover, a pilot experiment is carried out to identify the best initialization scheme. To test the performance of proposed adaptive TS approach an experimental study is carried out on randomly generated problem sets involving both small and large-sized problems.

In Chapter 6, three solution approaches are proposed to solve buffer allocation problem for unreliable and also non-homogeneous lines under the objective of total buffer size minimization. The proposed solution approaches for solving the problem involve three algorithms: binary search, tabu search and simulated annealing. All of these algorithms involve an adaptive tabu search algorithm to minimize the total buffer size to achieve a desired throughput level. Additionally, to improve the search



performance of TS and SA algorithms, alternative neighborhood generation mechanisms are suggested and they are tested.

Finally in Chapter 7, the summary and the contributions of this Ph.D. study are discussed. Moreover, the possible future research directions are presented.

## **CHAPTER TWO**

### **BACKGROUND INFORMATION**

#### **2.1 Introduction**

The aim of this Ph.D. study is to develop efficient algorithms to solve buffer allocation problem for unreliable serial production lines. In order to gain an understanding of important issues related to the buffer allocation problem and also the solution methodologies proposed to deal with this problem, this chapter gives a general background information.

The chapter is organized as follows. In Section 2.2, the problem characteristics are given and the buffer allocation problem is classified based on these characteristics. In section 2.3, the general process of solution of buffer allocation problems is presented. In section 2.4, the basic principles of decomposition method, tabu search and simulated annealing are explained so that an understanding can be gained to the background of the methods employed in this Ph.D. study. Finally, in section 2.5, the context of this chapter is summarized.

#### **2.2 The Buffer Allocation Problem**

The buffer allocation problem, BAP, is concerned with the allocation of a certain amount of buffers,  $N$ , among the  $K-1$  intermediate buffer locations of a production line to achieve some specific objective and it is one of the major optimization problems faced by manufacturing systems designers. It should be noted that while this problem is being handled, it is assumed that other manufacturing design problems, the workload and server allocation problems, have already been solved.

The primary reason for having storage buffers is to allow sequential workstations to operate more independently of each other. This reduces the idle time due to starving (no input available) and blocking (no space to dispose of output). Less idle

time increases the average production rate of the line. On the other hand, inclusion of buffers requires additional capital investment and floor space, which may be expensive. Buffering also increases in-process inventory. If the buffers are too large then the capital cost incurred may outweigh the benefit of the increased productivity. If the buffers are too small, the machines will be underutilized or demand will not be met. Because of the importance of finding good or optimal buffer configurations, the buffer allocation problem is still an important optimization problem.

The buffer allocation problem arises in a wide range of manufacturing systems, such as transfer lines, flexible manufacturing systems or robotic assembly lines. In this Ph.D. thesis we mainly concerned with the buffer allocation problem in serial production lines. The characteristics of the buffer allocation problem in serial production lines are given in the following section.

### ***2.2.1 Characteristics of the Buffer Allocation Problem***

A production line consists of machines connected in series and separated by buffers. A  $K$ -machine production line is represented in Figure 2.1, in which the squares represent machines and the circles represent buffers. Material moves in the direction of the arrows, from upstream machine to the downstream machine. Material flow may be disrupted by machine failures or variable processing times. Buffers are inserted between machines, so that the propagation of disruptions can be limited and hence, the average production rate of the line can be increased.

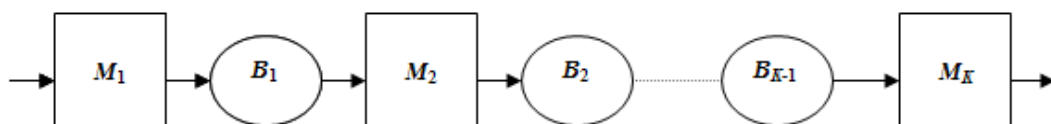


Figure 2.1  $K$ -machine production line

There are several unique characteristics inherent to the buffer allocation problem which complicates the application of existing ordinary search techniques. The

following is the summary of the discussion on these difficulties as it is stated by Park (1993):

- (A) The system performance of throughput rate over buffer size is monotonically increasing. Okamura and Yamashina (1977) show that the throughput rate of the production line, which is composed of more than two stages, steeply increases in the range of small buffer sizes and thereafter this increment continues with gradually smaller improvement until it reaches an upper bound.
  
- (B) There may be one or more stagnant areas in the function of a throughput rate over buffer sizes. Since the throughput rate is not likely to increase strictly as the buffer size increases, no increase in throughput rate may occur through a certain range of buffer sizes, as shown in Figure 2.2. Increasing the size of any buffer in the line may generate a local gain in the throughput rate of the line. However, the local throughput gain may or may not subsequently be propagated through the upstream and/or downstream machines due to complex interactions of processing, failure and repair rates of the machines and buffer sizes. Only if it can be propagated through both upstream and downstream machines, the local gain can be realized as a production gain for overall system. Otherwise, there will be no increase in production rate of the line. In most cases, the traditional optimization techniques get stuck at the stagnant area. This phenomenon is expanded to  $K$ -stage problems with  $K-1$  buffers, in that reduction of a buffer size may be compensated by the increasing sizes of other buffers to obtain the same production rate.

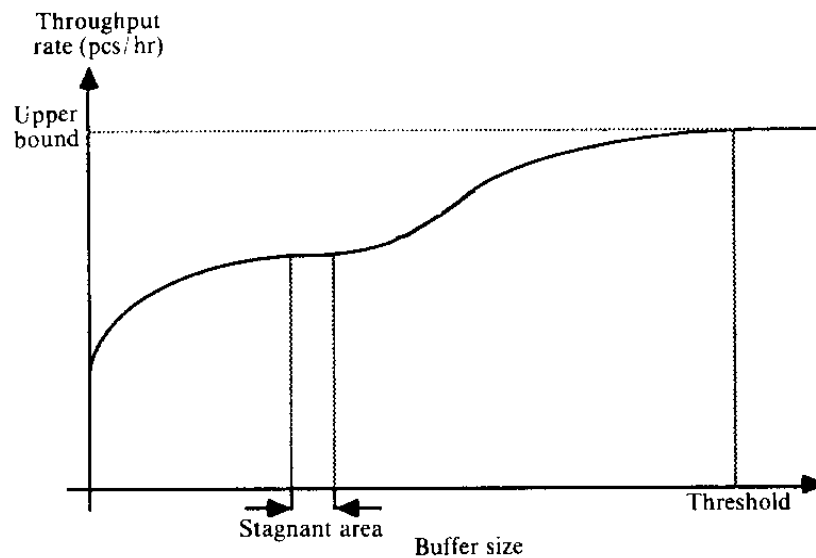


Figure 2.2 A function of throughput rate over buffer size (Park, 1993)

- (C) There is a limit on the degree of the system performance gained by increments in buffer sizes. The “threshold” in Figure 2.2 indicates the upper bound on the throughput rate. Also, in the vicinity of the threshold, considerable buffer storage is usually required to achieve even a small improvement in system performance. Since there is no change in system throughput rate beyond the threshold, one may face difficulties in finding a global optimal solution if the objective of the buffer allocation problem is to maximize the throughput rate of the line.
- (D) The buffer sizing problem is discrete in nature. In general, due to their combinatorial complexity, optimization problems with discrete control variables are more difficult to solve than the problems with continuous decision variables. Moreover, since there is no algebraic relation between the throughput of the line and buffer sizes, it is much harder to solve the buffer allocation problem.
- (E) The throughput rate function over buffer sizes is not usually unimodal in case of multiple buffers. Since many traditional optimization methods

require the unimodality condition to obtain a global optimal solution, they are often likely to fail in finding global optimal buffer sizes.

### 2.2.2 Classification of the Problems

The buffer allocation problem can be expressed mainly in three forms depending on the objective function. These objective functions may be concerned with maximizing throughput rate of the production line, minimizing total buffer size in the line and minimizing average work-in-process inventory. These forms can be given as follows:

**Problem 1 (BAP1)**: This formulation of the problem expresses the maximization of the throughput rate, for a given fixed amount of buffers, as follows:

Find  $B = (B_1, B_2, \dots, B_{K-1})$  so as to

$$\max f(B) \quad (1)$$

subject to

$$\sum_{i=1}^{K-1} B_i = N \quad (2)$$

$$B_i \text{ nonnegative integers } (i = 1, 2, \dots, K-1) \quad (3)$$

where  $N$  is a fixed nonnegative integer denoting the total buffer space available in the system which has to be allocated among the  $K-1$  buffer locations so as to maximize the throughput rate of the  $K$ -machine production line. In this formulation  $B$  represents a buffer size vector,  $B_i$  is the buffer size for each location, and  $f(B)$  represents the throughput rate of the production line as a function of the buffers' size vector.

**Problem 2 (BAP2)**: The solution approaches to this problem aims achieving the desired throughput rate with the minimum total buffer size, as follows:

Find  $B = (B_1, B_2, \dots, B_{K-1})$  so as to

$$\min \sum_{i=1}^{K-1} B_i \quad (4)$$

subject to

$$f(B) \geq f^* \quad (5)$$

$$B_i \text{ nonnegative integers } (i = 1, 2, \dots, K - 1) \quad (6)$$

where  $K$  is the number of machines in the line,  $B$  is a buffer size vector,  $B_i$  is the buffer size for each location,  $f(B)$  is the throughput rate of the production line and  $f^*$  is the desired throughput rate.

**Problem 3 (BAP3):** This last formulation expresses the minimization of the average work-in-process inventory subject to the total buffer size constraint as well as the desired throughput rate constraint, as follows:

Find  $B = (B_1, B_2, \dots, B_{K-1})$  so as to

$$\min Q(B) \quad (7)$$

subject to

$$\sum_{i=1}^{K-1} B_i \leq N \quad (8)$$

$$f(B) \geq f^* \quad (9)$$

$$B_i \text{ nonnegative integers } (i = 1, 2, \dots, K - 1) \quad (10)$$

where  $K$  is the number of machines in the line,  $B$  is a buffer size vector,  $Q(B)$  denotes the average work-in-process inventory as a function of buffer size vector,  $B_i$  is the buffer size for each location,  $N$  is a fixed nonnegative integer denoting the total buffer size,  $f(B)$  is the throughput rate of the production line and  $f^*$  is the desired throughput rate.

As it is stated by Park (1993), allocating buffer storage based on monetary criteria, such as maximizing profit or minimizing total cost, is a management concern in real production systems. Objective functions involving monetary criteria are expressed in a form of profits or costs.

Meester and Shanthikumar (1990) show that the throughput rate of the tandem queuing systems is an increasing, concave function of the buffer sizes. Based on their proof Papadopoulos et al. (2009) stated that the problem BAP1 is an increasing

function of the total buffer size  $N$ . Hence, the results obtained for problem BAP1 can be used to solve the problem BAP2. Thus, the above three problems can be reduced to two problems as it is stated by Papadopoulos et al. (2009).

The buffer allocation problem is difficult for two reasons, as indicated by Chow (1987): (1) the lack of an algebraic relation between the throughput of the line and buffer sizes; and (2) the nature of combinatorial optimization inherent in the buffer design problem. For a production line with  $K$  machines and the total buffer capacity  $N$ , the total number of possible buffer configurations for the problem BAP1 can be calculated as follows:

$$\binom{N+K-2}{K-2} = \frac{(N+1)(N+2)\dots(N+K-2)}{(K-2)!} \quad (11)$$

As it can be observed above, the total number of feasible solutions increases exponentially when  $N$  and  $K$  are large. For instance if the production line involves only ten machines and the number of total buffers to be allocated is 50, then the total number of feasible buffer allocations becomes 1.916.797.311 indicating the computational difficulty to search through the whole solution space by complete enumeration even for small sized problems. So, numerical approaches to the solution of the problems are inevitable even in situations with relatively small problems. Hence, to overcome this difficulty various solution techniques are employed to solve buffer allocation problem. Next section summarizes these solution techniques.

### 2.3 General Procedure to Solve the Buffer Allocation Problem

Solution approaches to solve buffer allocation problem involve applying a generative method and an evaluative method in an iterative manner. In other words generative methods and evaluative methods are combined in a closed loop configuration as depicted in Figure 2.3. In this configuration an evaluative method is used to obtain the value of the objective function for a set of inputs. To search for an



optimal solution, the value of the objective function is then communicated to the generative method.

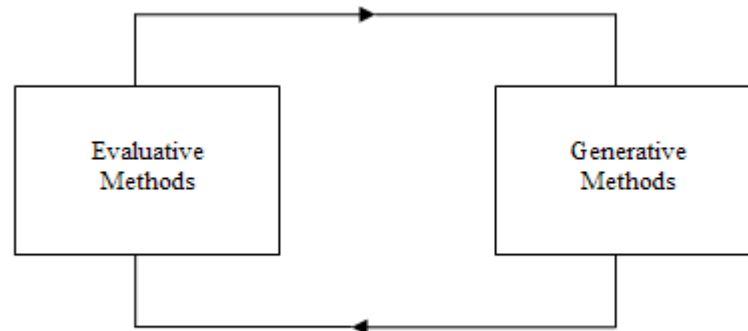


Figure 2.3 General process of solution of buffer allocation problems (Papadopoulos et al., 2009)

Evaluative methods, which provide the prediction of various performance measures such as the throughput rate and the mean queue lengths, are based on analytical methods and simulation. Analytical methods can be classified as exact and approximate methods. Since the analytical methods can be applicable only for small-sized problems, approximate methods are usually employed as evaluative method for solving buffer allocation problem. There are also various optimization techniques used as generative method. Complete enumeration is the simplest method but it is applicable for small-sized problems. Since the total number of feasible solutions grows exponentially when the total number of machines and the total buffer capacity increases, it is impossible to employ complete enumeration for large-sized problems. Therefore, the researchers widely adopted various search methods and meta-heuristics to effectively deal with the buffer allocation problem.

In the following subsections, alternative evaluative and generative methods used for solving buffer allocation problem in the literature are discussed.

### ***2.3.1 Evaluative Methods***

As it is stated before, basically two methods are used for evaluation: analytical methods and simulation. Exact analytical results based on the queuing models are difficult to obtain, and are only available for short production lines. For long production lines, generally approximate evaluative methods are employed. Most frequently used approximate evaluative methods to solve the buffer allocation problem are decomposition method, aggregation method, and generalized expansion method.

Among these methods, the decomposition method is the most widely used evaluation method for solving buffer allocation problem (Gershwin and Schor 2000, Helber, 2001, Shi and Men, 2003, Noureifath et al., 2005, Nahas et al., 2006, Demir and Tunali, 2008, Shi and Gershwin 2009, Massim et al., 2010, and Demir et al., 2011). The common idea in this method is to decompose the analysis of the original model into the analysis of a set of smaller subsystems which are easier to deal with. The main advantage of the decomposition method is its computational efficiency and its accuracy to reach the solution. However, the disadvantage of decomposition method is that it can be applicable only under the assumptions that processing rates are either deterministic or exponentially distributed and failure and repair rates are either geometric or exponentially distributed random variables. In this Ph.D. study, the decomposition method is used as an evaluative method due to its ability to obtain the throughput of a production line quite accurately and quickly. The details of the method are given in section 2.4.1.

Another approximation method based on the queuing models is the generalized expansion method. Contrast to the decomposition method the generalized expansion method can be used for generally distributed service times and reliable machines and it can be applicable to split and merge configurations as well as serial configurations. Applications of generalized expansion method for buffer allocation problem can be found in the studies of Spinellis et al. (2000), Daskalaki and MacGregor Smith

(2004), MacGregor Smith and Cruz (2005), Cruz et al. (2008), Aksoy and Gupta (2010) and Cruz et al. (2010).

Another evaluative method which can be used to solve buffer allocation problem is the aggregation method. Dolgui et al. (2002, 2007) successfully employ the aggregation method to evaluate the performance of buffer allocation decisions in unreliable production lines. The basic idea of aggregation is to first place a two-station one-buffer sub-line by a single equivalent station. Then this equivalent station is combined with a buffer and station of the original line to form a new two-station one-buffer sub-line, which is then aggregated into a single equivalent station. This process is repeated until the last or first station is reached, depending on the direction of the aggregation is performed.

If the objective is to realistically model a large and complex system, simulation provides many advantages in comparison to analytical methods. But the chief disadvantage of simulation modeling is that it is very time consuming. Simulation modeling is best suited to addressing design and operational problems at the detailed level, where other mathematical techniques are not sufficiently accurate to be applied. The studies of Jeong and Kim (2000), Gurkan (2000), Sabuncuoglu et al. (2002, 2006), Bulgak (2006), Altiparmak et al. (2007), Battini et al. (2008), Can and Heavey (2009) and Kose (2010) can be given as applications of simulation for solving buffer allocation problem.

### ***2.3.2 Generative Methods***

Generative methods focus on finding optimal buffer sizes to improve the system performance. The simplest generative method is complete enumeration. However, this method is applicable only for small systems since the total number of feasible solutions grows exponentially when the total number of machines and the total buffer size to be allocated in the system increase. Therefore for large systems it is impossible to search through the whole solution space by complete enumeration. In recent years, the researchers widely adopted various search methods and meta-

heuristics to effectively deal with the combinatorial nature of the buffer allocation problem.

Search methods including both traditional and also heuristic search algorithms tend to resolve the exponential explosion in the number of alternative buffer vectors by quickly shifting through many alternative buffer vectors to discover those which yield close to optimal results. Ho et al. (1979), Gershwin and Schor (2000), Seong et al. (1995, 2000) and Helber (2001) apply gradient search algorithm. Vouros and Papadopoulos (1998) employ knowledge based methods. Altiok and Stidham (1983) use the pattern search technique of Hooke and Jeeves. Nahas et al. (2006) employ degraded ceiling local search heuristic. Fuxman (1998), Harris and Powell (1999), Jeong and Kim (2000), Papadopoulos and Vidalis (2001), Hemachandra and Eedupuganti (2003), Tempelmeier (2003), Sabuncuoglu et al. (2006), Zequeira et al. (2008), and Aksoy and Gupta (2010) develop problem specific search algorithms for solving buffer allocation problem. There are mainly two disadvantages of traditional search methods. One of these disadvantages is that traditional search methods sometimes cannot jump over local optimal solutions in search of the global optimal ones. The other disadvantage is that with these approximate methods it is difficult to observe how small changes in buffer sizes affect the system performance.

Meta-heuristics are search methods which use strategies that guide the search process and explore the search space in order to find optimal/near-optimal solutions. Meta-heuristic algorithms are approximate and usually non-deterministic. Typical solution methods in this area include Tabu Search (Lutz et al., 1998, Demir et al., 2011), Simulated Annealing (Spinellis and Papadopoulos, 2000a, 2000b, Spinellis et al. 2000), Genetic Algorithms (Spinellis and Papadopoulos, 2000b, Dolgui et al., 2002, Qudeiri et al., 2007, 2008, Yamamoto et al., 2008, Cruz et al. 2010, Kose, 2010), and Ant Colony Optimization (Nourelfath et al., 2005, Nahas et al., 2009). To better search the solution space, the recent trend is to hybridize the meta-heuristics with other methods such as Nested Partitions (Shi and Men, 2003) and Branch and Bound methods (Dolgui et al., 2007). The chief advantage of meta-heuristics over traditional search methods is that they can jump over local optimal solutions in

search of the global optimal ones. Their main disadvantage is that they are not problem specific and thus, they have to tune-up to produce solutions to a specific problem type.

Moreover, dynamic programming, a well known optimization method (Chow, 1987, Jafari and Shanthikumar, 1989, Yamashita and Altiok, 1998, Diamantidis and Papadopoulos 2004), artificial neural networks (Bulgak, 2006, Altiparmak et al., 2007) and also immune system algorithm (Massim *et al.*, 2010) are successfully employed for solving buffer allocation problem in production lines.

Lastly, a number of studies including Sabuncuoglu et al. (2002) and Raman and Jamaludin (2008) employed various experimental designs for evaluating the solutions to the buffer allocation problem.

Due to its ability to evaluate the throughput of a production line quite accurately and quickly, in this Ph.D. study, the decomposition method is used as an evaluative method. Unlike population-based search algorithms such as genetic algorithms, which require long time to converge, single point search algorithms such as tabu search and simulated annealing focus on exploitation and they are faster. Hence, to reduce the computational difficulty especially for evaluating the throughput of the line for medium and large-sized problems, tabu search and simulated annealing are employed as a generative method for solving the buffer allocation problem.

The following sections present the details of evaluation (decomposition method) and generative methods (tabu search and simulated annealing) employed in this study.

## **2.4 Background Information on Solution Approaches Employed**

This section gives background information on solution approaches used in this Ph.D. study. Next section presents the basic idea of decomposition method. Basic

information on tabu search and simulated annealing are given in sections 2.4.2 and 2.4.3, respectively.

### ***2.4.1 Decomposition Method***

The decomposition method, proposed by Gershwin (1987), is an efficient method to estimate the performance measures of serial production lines. The method works as follows. An original line  $L$  is broken into  $K-1$  two-machine lines as illustrated in Figure 2.4. Line  $L(i)$  is composed of an upstream machine  $M_u(i)$ , a downstream machine  $M_d(i)$ , and buffer  $B(i)$ . The capacity of  $B(i)$ ,  $N_i$ , is the same as the capacity of buffer  $B_i$  in line  $L$ . In order to determine the average throughput rate of this production line, the system is modeled as a Markov process for which the steady-state behavior is determined. Since the performance characteristics of two-machine lines can be obtained exactly, the decomposition method requires the derivation of a set of equations that link the decomposed two-machine lines together. These nonlinear equations are solved to determine the unknown parameters of each line  $L(i)$ , i.e. the processing rates ( $\mu_i$ ), failure rates ( $p_i$ ) and repair rates ( $r_i$ ) of upstream and downstream machines, so that the behavior of the material flow in buffer  $B(i)$  in line  $L(i)$  closely matches that of the flow in buffer  $B_i$  of original line  $L$ .

Dallery *et al.* (1988) develop the decomposition equations and an algorithm called DDX to solve these equations for homogeneous lines, i.e. the machines in the line have the same processing times amounting to one unit of time. Later, Burman (1995) extends this study to the non-homogeneous lines, i.e. the machines in the line have different processing times, and develops the algorithm called as accelerated-DDX (ADDX). Due to its ability to evaluate the throughput of a production line quite accurately and quickly, in this Ph.D. study, while the DDX algorithm is employed for homogenous lines and the ADDX algorithm is employed for non-homogeneous lines. The details of both algorithms are given in Appendix A.

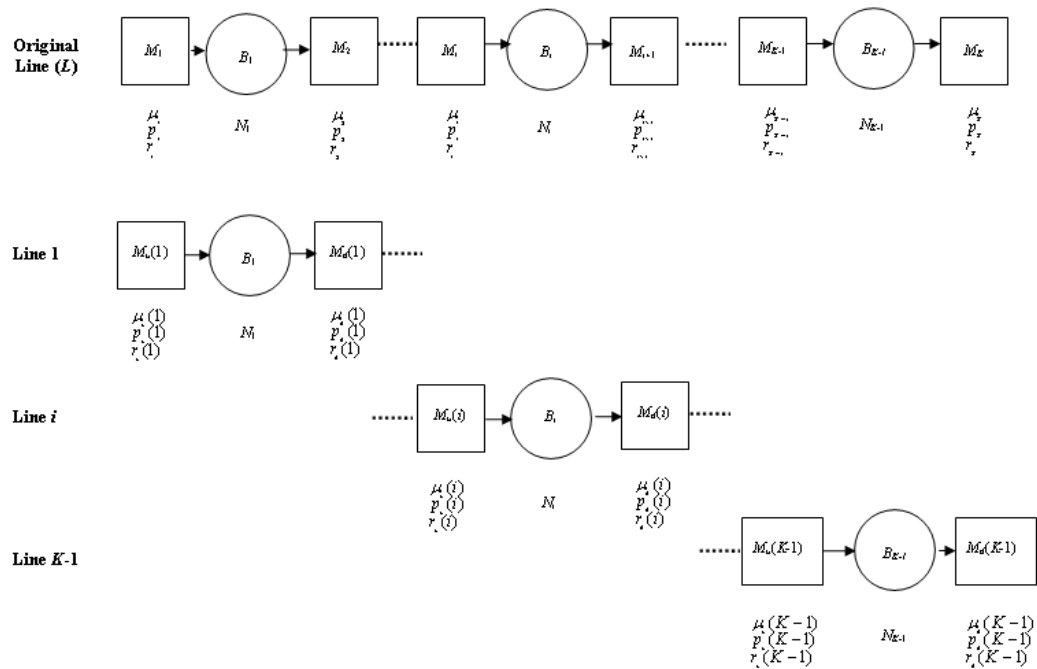


Figure 2.4 The decomposition method (Burman, 1995)

### 2.4.2 Tabu Search

Tabu Search (TS) is a meta-heuristic for solving combinatorial optimization problems. Originating from the work by Glover (1977), TS basic ideas were first introduced in Glover (1986). TS explicitly uses the history of the search, both to escape from local optima and to implement an explorative search. For more details about TS the reader can refer to Glover and Laguna (1997).

Suppose that TS is employed to deal with the following combinatorial optimization problem:

$$(P) \quad \text{Minimize } f(x) : x \in X \text{ in } R_n.$$

The objective function  $f(x)$  may be linear or nonlinear, and the condition  $x \in X$  is assumed to constrain specified components of  $x$  to discrete values. In some settings (P) may represent a modified form of some original problem, as where  $X$  is a superset of the vectors that normally qualify as feasible, and  $f(x)$  is a penalty function, designed to assure that optimal solutions to (P) likewise are optimal for the problem from which it is derived (Glover, 1989).

A fundamental element of TS is the use flexible memory functions, called *tabu lists*, to forbid the transitions, called *moves*, from the current solution to other candidate solutions that are previously visited. A move  $m$  is defined as follows:

$$m: X(m) \rightarrow X .$$

Associated with  $x \in X$  is the set  $NB(x)$  which consists of those moves  $m \in NB$  that can be applied to  $x$ ; i.e.,  $NB(x) = \{m \in NB : x \in X(m)\}$ . The set  $NB(x)$  is called the neighborhood of  $x$ . Within this framework the basic elements of tabu search can be described as follows.

#### 2.4.2.1 Search Space and Neighborhood Structure

Choosing a search space along with a neighborhood structure is the most critical step of any TS implementation. The search space of TS is simply the space of all feasible solutions that can be visited during the search. It should be noted that it is not always a good idea to restrict the search space to feasible solutions. In many cases, allowing the search to move infeasible solutions is desirable (see Gendreau and Potvin, 2005, for further details). To define the neighborhood structures of the current solution, there are several choices depending on the specific problem at hand. For instance in the buffer allocation problem context, one choice could be to consider the full neighborhood of the current buffer configuration while the other could be to consider only a subset of the neighborhood of the current solution.

#### 2.4.2.2 Tabus

Tabus are one of the basic elements of TS. Tabus are used to prevent cycling while escaping from local optima via non-improving moves. Tabus are also useful to help the search move away from previously visited areas of the search space and thus perform more extensive exploration. Tabus are stored in a tabu list, which is the *short term memory* of the search, and in general only a fixed and fairly limited quantity of information is recorded in this list. The most commonly used tabus involve recording the last few moves performed on the current solution and



forbidding reverse moves; others are based on key characteristics of the solutions themselves.

The length of the tabu list, called *tabu tenure* ( $TT$ ), is an important search parameter of TS. Tabu tenure is the number of iterations that tabus stay in the tabu list. As indicated by Glover *et al.* (1993) the size of tabu list providing good results often grows with the size of the problem. However, no single rule has been found to yield an effective tenure for all classes of problems. This is partly because an appropriate list size depends on the strength of the tabu restrictions employed (where stronger restrictions are generally coupled with smaller sizes) (Glover and Laguna, 1997). If the size of the tabu tenure is too small, preventing the cycling might not be achieved; conversely a too long length creates too many restrictions. As indicated by Reeves (1996) a value of 7 for  $TT$  has often found to be sufficient to prevent cycling; other commonly used values are  $TT = \sqrt{n}$  where  $n$  is some natural measure of the problem size. Dynamic rules may be useful too, usually this means choosing lower and upper bounds  $TT_{\min}$  and  $TT_{\max}$  on the tabu tenure, and allowing  $TT$  to vary in some way between them.

#### 2.4.2.3 *Aspiration Criteria*

It is not difficult to realize that tabus may forbid moving to attractive unvisited solutions. It is therefore necessary to overrule the tabu status of moves in certain situations. This is performed by means of *aspiration criteria*. The simplest and most commonly used aspiration criterion consists of allowing a move, even if it is tabu, if it results in a solution with an objective value better than the current best-known solution.

#### 2.4.2.4 *Termination Criteria*

The most commonly used termination criteria in TS are:

- after a fixed number of iterations or a fixed amount of CPU time,

- after some number of iterations without an improvement in the objective function value,
- when the objective function value reaches a pre-specified threshold value.

Table 2.1 represents the basic Tabu Search algorithm and the flowchart of the standard tabu search algorithm is presented in Figure 2.5.

Table 2.1 Basic tabu search algorithm

Select an initial  $x \in X$  and let  $x^* := x$ . Set the iteration counter  $k = 0$  and begin with  $T$  empty.

**while** termination conditions not met **do**

Set  $k := k + 1$  and select the best  $m_k \in NB(x) - T$  where the elements of  $NB(X) - T$  are not tabu or they satisfy at least one aspiration criterion.

Set  $x^* := x$  where  $x^*$  denotes the best solution currently found

Update the tabu list and aspiration criteria

Basic TS as described above can sometimes successfully solve difficult problems, but in most cases, the following additional elements have to be included in the search to make it fully effective.

#### 2.4.2.5 Intensification

The key idea behind the concept of intensification is to implement some strategies so that the areas of the search space that seem promising can be explored more thoroughly. In general, intensification is based on *intermediate-term memory*, such as a *recency memory*, in which one records the number of consecutive iterations that various solution components have been present in the current solution without interruption. Intensification is used in many TS implementations, but it is not always necessary. This is because there are many situations where the search performed by the normal searching process is thorough enough (Gendreau and Potvin, 2005). Thus there is no need to spend time exploring in depth the portions of the search space that have already been visited, and this time can be used more effectively.

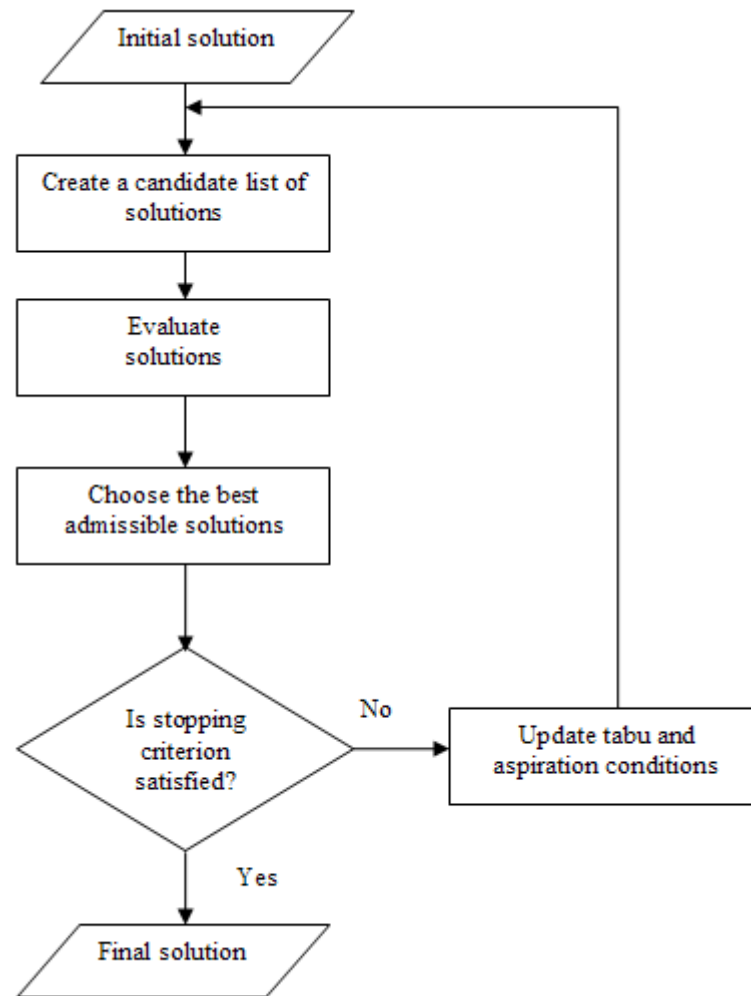


Figure 2.5 The flowchart of a standard tabu search algorithm

#### 2.4.2.6 Diversification

Unlike intensification which helps more intensively searching the regions which contain good solutions diversification guides the search to unexplored regions. Diversification is usually based on *long-term memory*, such as a *frequency memory*, where the total number of iterations of the performed moves or visited solutions is recorded. There are two major diversification techniques known as *restart diversification* and *continuous diversification* while the first one is performed by several random restarts, the second one is integrated into the regular searching process to penalize frequently performed moves or solutions.

### 2.4.3 Simulated Annealing

Simulated Annealing (SA) is another meta-heuristic method used for solving combinatorial optimization problems. The ideas that form the basis of simulated annealing were first published by Metropolis *et al.* (1953). Annealing is the physical process of heating up a solid and then cooling it down slowly until it crystallizes. The atoms in the material have high energies at high temperatures and have more freedom to arrange themselves. As the temperature is reduced, the atomic energies decrease. A crystal with regular structure is obtained at the state where the system has minimum energy. If the cooling is carried out very quickly, which is known as rapid quenching, widespread irregularities and defects are seen in the crystal structure. The system does not reach the minimum energy state and ends in a polycrystalline state which has a higher energy (Pham and Karaboga, 2000).

Essentially, Metropolis's algorithm simulates the change in the energy of the system when subject to cooling process, until it converges to a steady frozen state. In 1983, Kirkpatrick *et al.* (1983) suggested that this type of simulation could be used for solving combinatorial optimization problems.

Simulated annealing algorithm consists of a sequence of iterations. At each iteration, the neighborhoods of the current solution are generated randomly or in a systematic way by using a neighborhood generation mechanism. Once a new solution is created the corresponding change in the acceptance function is computed to decide whether the newly produced solution can be accepted as the current solution. If the change in the acceptance function is negative the newly produced solution is directly taken as the current solution. Otherwise, it is accepted according to Metropolis's criterion based on Boltzman's probability.

According to Metropolis's criterion, if the difference between the acceptance function values of the current and the newly produced solutions is equal to or larger than zero, a random number  $R \in [0,1]$  is generated from a uniform distribution. If

$$R \leq \exp(-\Delta E / T)$$

then the newly produce solution is accepted as the current solution. Otherwise, the current solution is unchanged. Here  $\Delta E$  is the difference between the acceptance function values of the current solution and newly produced solution and  $T$  is the value of temperature. The flowchart of a standard SA algorithm is shown in Figure 2.6.

The important issues need to be considered in implementation the SA algorithm are summarized in the following sections.

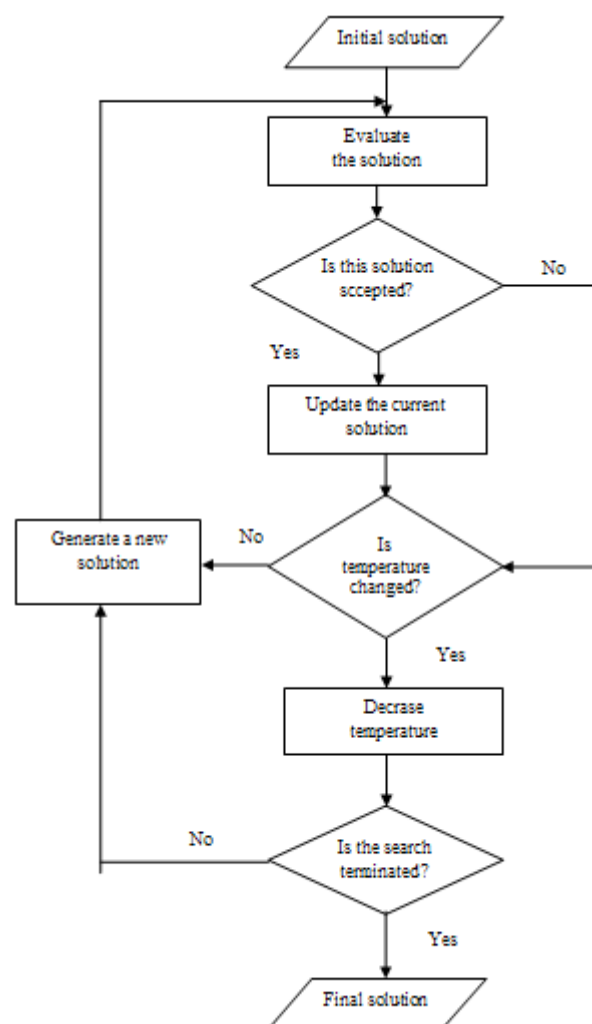


Figure 2.6 Flowchart of a standard simulated annealing algorithm

#### *2.4.3.1 Neighborhood Generation Mechanism*

In general, the neighborhoods are sampled randomly in implementing SA algorithm. However, sampling in a systematic or adaptive has a higher chance to produce better results. To define the neighborhood structures of the current solution, there are several choices depending on the specific problem at hand. In this Ph.D. study, the neighborhood generation to solve the buffer allocation problem is carried out in a systematic way. The details regarding this issue are given in chapter 6.

#### *2.4.3.2 Initial Temperature*

Generally the SA algorithm starts with “high” initial temperature allowing many inferior moves to be accepted. In practice this may require some knowledge of magnitude of neighboring solutions; in the absence of such knowledge, one may choose what appears to be a large value, and run the algorithm for a short time and observe the acceptance rate. As it is stated by Reeves (1996), if this acceptance rate is “suitably high” this value of temperature may be used to start the algorithm. Even if “suitably high” acceptance rate varies from one situation to another, Reeves (1996) states that an acceptance rate of between 40% and 60% seems to give good results in many cases.

#### *2.4.3.3 Cooling Schedule*

There are basically two types of schedule, having analogies to homogeneous and inhomogeneous Markov chains, respectively. In the homogeneous case, annealing is carried out at a fixed temperature until equilibrium is reached. Once this state is judged to have been reached, the temperature is reduced, and the procedure is repeated. The number of attempted moves at each temperature may be quite large, although the temperature steps can be relatively large also. In the inhomogeneous case, the temperature is reduced (but by a very small amount) after every move. This is less complicated than the homogeneous case, and is the one more commonly used in practice (Reeves, 1996).

In either case, one has to decide on the shape of the cooling curve. The simplest and most common one is the geometric schedule. In the geometric schedule, the temperature is updated by the following formula:

$$T_{i+1} = cT_i \quad i = 0, 1, \dots$$

where  $c$  is a temperature factor which is a constant close to 1 (typically in the range 0.90 to 0.99).

The other method is proposed by Lundy and Mees (1986). This method updates the temperature by the following formula:

$$T_{i+1} = \frac{T_i}{1 + \beta T_i} \quad i = 0, 1, \dots$$

where  $\beta$  is the constant near to zero.

#### 2.4.3.4 Final Temperature

In theory, the algorithm continues until the final temperature is zero, but in practice it is sufficient to stop the algorithm when the chance of accepting inferior solutions becomes negligible. This is a problem dependent issue and as in the case of selecting an initial temperature, selecting final temperature may involve some monitoring of the ratio of acceptances. For this purpose Lundy and Mees (1986) proposed stopping when

$$T \leq \frac{\varepsilon}{\ln[(|S|-1)/\Theta]}$$

where  $S$  is the solution space. This is designed to produce a solution which is within  $\varepsilon$  of the optimum with probability  $\Theta$ .

#### 2.4.3.5 Number of Iterations

The number of iterations can be determined by the following formulas (Reeves, 1996):

$$k = \frac{\log T_f - \log T_0}{\log c}$$

and

$$k = \frac{T_0 - T_f}{\beta T_0 T_f}$$

for homogenous and non-homogeneous cases respectively, where  $T_f$  is the final temperature and  $T_0$  is the initial temperature.

Since both methods are point based search methods and it is known that point based methods needs less solution time as compared to the population based search methods such as genetic algorithms, these two methods are employed as generative methods. It is known that simulated annealing is successfully employed for solving buffer allocation problem. Moreover, since tabu search provides an alternative to traditional optimization techniques by using memory-based strategies to escape the local optima and it is also successfully employed on many combinatorial optimization problems.

## 2.5 Chapter Summary

In this chapter, the characteristics, formulations and solutions methods of buffer allocation problem are given in detail. The solution of the buffer allocation problem involves using an evaluative method and a generative method in an iterative manner. In this Ph.D. study, the decomposition method is used as an evaluative method and two meta-heuristic methods - tabu search and simulated annealing- are employed as generative method. Hence, in this chapter, background information on all these methods is given.

Next chapter is devoted to the review of the related literature on buffer allocation problem. To our knowledge since the study of Gershwin and Schor (2000) there is no any comprehensive survey on buffer allocation problem. Thus, in the next chapter we aim at filling the perceived gap in this area and also state our contributions to this area.



## **CHAPTER THREE**

### **LITERATURE SURVEY**

#### **3.1 Introduction**

Due to its importance and complexity, the buffer allocation problem has been studied for over 50 years and numerous publications are available in the literature. The first study in this area is presented by Koenigsberg (1959), which gives an analysis and review of the basic problems associated with the efficient operation of production systems. A detailed analysis of mathematical models describing the effect of the buffer storage can be found in the following references (Buzacott and Shanthikumar, 1993, Papadopoulos et al., 1993, Papadopoulos et al., 2009) and in some comprehensive survey studies (Dallery and Gershwin, 1992, Papodopoulos and Heavey, 1996).

As mentioned in chapter 2, the buffer allocation problem can be formulated in three forms, BAP1, BAP2 and BAP3. Among these, the BAP1 and BAP2 have been studied more extensively in the literature. As stated by Enginarlar (2003), BAP1 and BAP2 can be solved using algorithmic and also rule-based approaches (see Figure 3.1). While algorithmic approaches involve an optimization algorithm to solve the problem, rule-based approaches employ simple rules to obtain a good solution. In comparison to BAP1 and BAP2, the problem BAP3 which involves minimizing average work in process (WIP) in the system is a relatively less studied problem. This could be due to the fact that BAP3 involves more challenging constraints than the other two. The studies addressing these three problems are discussed in following sections.

This chapter presents a comprehensive survey on buffer allocation problem in production lines. Next section introduces our classification scheme to review the studies published after 1998. For other studies published before 1998, the reader can refer to Park (1993) and Gershwin and Schor (2000).

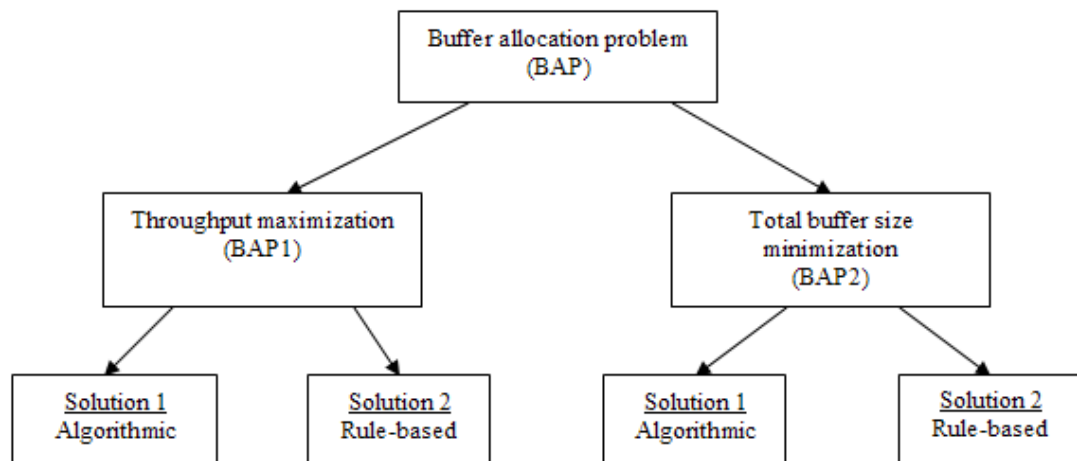


Figure 3.1 Classification of the literature on buffer allocation problem (Enginarlar, 2003)

The insight gained as a result of surveying the current literature and the motivation of this Ph.D. study are given in section 3.3. Finally, the context of this chapter is summarized in section 3.4.

### 3.2 Proposed Classification Scheme and Discussion of Current Literature

In this section the studies dealing with buffer allocation problem in production lines, published since 1998, are reviewed based on the following classification scheme:

**Topology of the line:** The relevant studies are classified according to the topology of line as follows:

- S : Serial
- S-P : Serial-Parallel
- GN : General Network
- A : Assembly
- FMS : Flexible Manufacturing Systems
- CMS : Cellular Manufacturing Systems

**Objective function:** The following objective functions are noted in the current relevant literature:

Objective 1: Throughput maximization

Objective 2: Total buffer size / Work in Process (WIP) minimization

Objective 3: Cost minimization

Objective 4: Profit maximization

Objective 5: Other objective functions, such as maximizing customer service level, minimizing the mean waiting time of a job, reducing idle time, minimizing cycle time, and minimizing average flow time of the product.

**Solution methodology:** The existing literature is classified according to the type of the evaluative and generative solution method employed to solve the buffer allocation problem.

To review the current relevant literature, first the studies are divided into two categories: 1. Reliable lines which are not subject to failure, 2. Unreliable lines which are subject to failure. Based on three criteria defined above, the following two sections present the review of studies done for reliable and unreliable production lines, respectively.

### **3.2.1 Reliable Lines**

Using the classification scheme explained in previous section, Table 3.1 chronologically lists the studies for reliable lines published since 1998. It should be noted that the notation used in this table is given in previous section.

Table 3.1 Overview on buffer allocation literature: Reliable lines

Authors	Topology of the Line						Objective					Solution Methodology	
	S	S-P	GN	A	FMS	CMS	1	2	3	4	5	Evaluative	Generative
Fuxman (1998)				x			x					Levner's Graph	Heuristic
Lutz et al. (1998)	x						x	x				Simulation	Tabu Search
Papadopoulos & Vidalis (1998)	x						x					Markovian State Model	The Modified Hooke-Jeeves Method
Powell and Pyke (1998)				x			x					Simulation	Heuristic
Yamashita & Altiok (1998)	x							x				Simulation	Dynamic Programming
Harris & Powell (1999)	x						x					Simulation	Spendley-Hext & Nelder-Mead Simplex Search Algorithms
Hillier (2000)	x									x		Markov Chain Model	Heuristic
Spinellis & Papadopoulos (2000a)	x						x					Decomposition Method	Simulated Annealing
Spinellis & Papadopoulos (2000b)	x											Decomposition Method	Genetic Algorithms & Simulated Annealing
Spinellis et al. (2000)	x						x					Expansion Method	Simulated Annealing
Huang et al. (2002)	x						x	x				Approximate Analytic Algorithm	Dynamic Programming
Sabuncuoglu et al. (2002)				x			x					Simulation	Design of Experiments
Chaharsooghi & Nahavandi (2003)	x						x					Markov State Model	Heuristic
Hemachandra and Eedupuganti (2003)				x			x	x				Markov State Model	Heuristic
Yamada and Matsui (2003)				x						x		Simulation	Complex Method
Daskalaki, & MacGregor Smith (2004)		x					x	x				Expansion Method	Powell's Algorithm
Diamantidis & Papadopoulos (2004)	x						x					Aggregation Method	Dynamic Programming

Table 3.1 Overview on buffer allocation literature: Reliable lines (cont.)

Authors	Topology of the Line						Objective					Solution Methodology	
	S	S-P	GN	A	FMS	CMS	1	2	3	4	5	Evaluative	Generative
MacGregor Smith & Cruz (2005)			x					x				Expansion Method	Powell's Algorithm
Hillier & Hillier (2006)	x									x		Markov Chain Model	Heuristic
Kwon (2006)					x							Decomposition Method	Heuristic
Nieuwenhuysen et al. (2007)	x										x	Queuing Model	Mathematical Model
Um et al. (2007)					x		x					Simulation	Evolution Strategy
Can et al. (2008)	x						x					Simulation	Genetic Algorithms
Cruz et al. (2008)			x					x				Expansion Method	Lagrange Relaxation & Derivative-free Search
Raman and Jamaludin (2008)	x							x				Simulation	Design of Experiments
Yuzukirmizi & MacGregor Smith (2008)			x				x				x	Expanded Mean Value Analysis	Powell's Algorithm
Can & Heavey (2009)	x						x					Simulation	Genetic Programming & Particle Swarm Optimization
Aksoy & Gupta (2010)						x				x		Expansion Method	Heuristic
Cruz et al. (2010)			x				x	x		x		Expansion Method	Genetic Algorithms
Van Woensel et al. (2010)			x					x				Two-moment Approximation & Expansion Method	Powell's Algorithm
Total	16	1	5	5	2	1	18	10	2	3	2		

The first study employing a meta-heuristic method for solving buffer allocation problem was presented by Lutz et al. (1998). In this study, a tabu search algorithm combined with simulation was suggested to solve the buffer allocation problem in a six-machine reliable production line. The objective was to maximize the throughput of the line while minimizing the total buffer size in the system. The results obtained from this study were consistent with the other studies which considered the same type of production lines. The authors indicated that the advantage of their simulation-search procedure was its ability to model any line configuration regardless of its scheduling policy or any other characteristics. However, it should be noted that this advantage was gained at the expense of a long computation time to model large or extremely complex manufacturing systems.

Another study employing simulation was presented by Yamashita and Altioik (1998). The objective of this study was to minimize the total buffer size for achieving desired throughput rate in a reliable line with phase-type processing times. For this purpose, the authors developed a dynamic programming algorithm. Moreover, they provided numerical examples to show the buffer allocation and compared the corresponding simulated throughput and its bounds with the desired throughput.

Papadopoulos and Vidalis (1998) considered the buffer allocation problem in balanced reliable production lines. The authors developed a search technique, which was a modified Hooke-Jeeves algorithm and they presented two basic design rules using this method. As an evaluative method the authors employed the Markovian state model developed by Heavey et al. (1993).

Fuxman (1998) presented an analytical model which allows computation of the minimum required number of buffers and an optimal allocation thereof in an asynchronous reliable mixed-model assembly line to maintain the highest possible throughput rate. The author developed a simple and efficient algorithm called BELA (i.e., buffer elimination algorithm) to identify buffer configurations and used the Levner's graph to obtain the throughput of the system. Powell and Pyke (1998) also studied simple asynchronous reliable assembly systems with variable processing

times and they developed simple heuristic rules that can be used to improve system performance. The authors employed simulation to evaluate the sensitivity of location of the first buffer against the parameters of the processing time distributions.

Harris and Powell (1999) developed a simple search algorithm for optimal buffer allocation to maximize throughput of the line under total buffer size constraint. The algorithm, which is an adaptation of the Spendley-Hext and Nelder-Mead simplex search algorithms, uses simulation to estimate throughput for every allocation considered.

The characterization of optimal buffer allocation for reliable production lines with variable processing times was presented by Hillier (2000). The production line was modeled by using Markov chains and the optimal buffer sizes were obtained by a heuristic approach so as to maximize the profit. The aim of this study was to suggest the rules of thumbs rather than develop algorithms for optimal buffer allocation in short reliable lines. This study was later extended by Hillier and Hillier (2006) to simultaneous optimization of workload and buffer allocation using Markov chains.

In recent years, meta-heuristic approaches are extensively used for solving buffer allocation problem. Simulated annealing is the most widely used method in this area. One of these studies employing simulated annealing was presented by Spinellis and Papadopoulos (2000a). In this study the decomposition method was employed as an evaluative method and simulated annealing was employed as a generative method for maximizing the production rate of the line. In another study, the authors (Spinellis and Papadopoulos, 2000b) compared the performance of simulated annealing algorithm with the performance of genetic algorithms, which is another meta-heuristic method frequently used for solving buffer allocation problem. The authors aimed at maximizing the throughput of the line for large-sized problems. Like their previous study, the decomposition method was employed to obtain the throughput rate of the line. The numerical experiments showed that simulated annealing produced larger number of optimal configurations than genetic algorithms but the performance of the genetic algorithm was superior to the performance of simulated

annealing. Hence, the authors suggested using these two methods in a complementary way. Genetic algorithms can be employed in real time applications for the swift recalculation of optimal configurations and simulated annealing can be utilized in batch-oriented calculations for obtaining an optimal configuration.

In another study, Spinellis et al. (2000) focused on a series-parallel production line to maximize the throughput under the constraints of total buffer size, total number of available servers and the summation of expected service time. A simulated annealing approach was presented to solve this problem for long production lines including 60 stations and 120 buffers. The expansion method was used to evaluate the system performance. Regarding the allocation of buffers, the number of servers and their service rate, the results of comparative experimental study exhibited some interesting similarities, but also striking differences from the earlier relevant research. The authors stated that these patterns of allocation were one of the most important insights which were emerged in solving very long production lines.

Huang et al. (2002) employed dynamic programming to solve the buffer allocation problem in a flow shop type production system under the objectives of minimizing work-in-process, cycle time and blocking probability, maximizing throughput, or their combinations. They evaluated the performance of this system by using an approximate analytic procedure.

Sabuncuoglu et al. (2002) studied the effect of the number of component stations (parallelism), work transfer, processing time distributions, and buffer allocation schemes on throughput rate and also on inter-departure time variability of assembly systems. The authors conducted simulation experiments so as to give some rule of thumbs that could guide practitioners to design more effective systems. Another study on assembly lines was presented by Hemachandra and Eedupuganti (2003). The authors considered a finite capacity fork-join queuing model for open assembly systems with arrival and departure synchronizations and they proposed a heuristic approach for enumerating the state space. The authors considered three objectives,



such as maximizing the throughput of the system, minimizing the mean waiting time of a typical job and minimizing the WIP of the system, while finding the optimal buffer configurations.

Yamada and Matsui (2003) developed a management design approach for assembly lines by considering both the cost and lead-time under demand fluctuations. In this study, the optimal buffer size which minimizes the buffer and overflow cost at each station was determined by iterative simulation. Here, the complex method was employed to optimize the buffer sizes.

Chararsoghi and Nahavandi (2003) proposed a heuristic approach for optimal allocation of buffers under the objective of throughput maximization. The throughput rate of the studied system was evaluated using simulation. Computational tests showed the efficiency of the proposed approach.

A dynamic programming implementation for solving buffer allocation problem was presented by Diamantidis and Papadopoulos (2004). The objective of this study was to maximize the throughput of the line subject to total buffer space constraint. To obtain the throughput of the system an aggregation method was used. The numerical results showed that the proposed dynamic algorithm was very fast and always converged.

Daskalaki and MacGregor Smith (2004) combined routing and buffer allocation problems in serial-parallel queuing networks. An iterative two-step methodology was proposed to solve the optimal routing and buffer allocation problems with the objectives of maximizing throughput and minimizing total buffer size. This methodology involved the expansion method as an evaluative method and Powell's algorithm as a generative method. The effectiveness of the proposed methodology was demonstrated through several experiments.

MacGregor Smith and Cruz (2005) solved buffer allocation problem for general queuing networks so as to minimize the total buffer size in the system. The

generalized expansion method was employed to evaluate the performance of the system and Powell's unconstrained search algorithm was used to optimize the buffer sizes. The efficiency of the solution approach was demonstrated by extensive computational experiments.

A flexible manufacturing system with parallel workstations was considered by Kwon (2006). The author employed the decomposition method to evaluate the performance of the system and proposed a heuristic algorithm to optimize the buffer sizes so as to maximize the throughput rate of the system. While the numerical tests showed the efficiency of the proposed algorithm for small-sized problems, the experimental results were not encouraging for large-sized problems.

Um et al. (2007) employed simulation methodology for the buffer size determination in a flexible manufacturing system cell line. Buffer allocation was categorized into cell buffer and machine buffer. The authors used the evolution strategy in order to find the optimal buffer sizes so as to maximize the throughput rate of the system and also minimizing AGV congestion and maximizing AGV utilization. Another simulation based study was carried out by Othman et al. (2007) for optimal buffer allocation in short production lines including eight machines. The authors aimed at giving a guideline to manufacturing system designers for the application of simulation methodology to the buffer allocation problem.

Optimization of finished good inventories was considered by Nieuwenhuyse et al. (2007). In this study, the authors developed a queuing model to evaluate customer service levels and buffer size requirements in semi-process industry. The objective of this study was to reduce finished good buffer requirements without compromising customer service level by determining optimal campaign sizes.

Raman and Jamaludin (2008) suggested three strategies to reduce the WIP inventory in a three-machine line and they tested the performance of these strategies via simulation modeling.

Cruz et al. (2008) considered the buffer allocation problem in an arbitrary queuing network. They aimed at finding the minimum total buffer size which achieves the desired throughput rate. Like their previous study (MacGregor Smith and Cruz, 2005) the generalized expansion method was employed to obtain the throughput rate of the system. For optimizing the buffer sizes, an algorithm based on a Lagrangian relaxation was proposed. In comparison with the exact simulation results, their proposed algorithm seemed to produce very fast and accurate solutions.

Yuzukirmizi and MacGregor Smith (2008) presented an optimal buffer allocation procedure for closed queuing networks. The performance measures were evaluated using the expanded mean value analysis and the buffer sizes were optimized by utilizing Powell method which was a non-linear optimization method. The objective function involved maximizing the throughput rate of the system, and also minimizing the total average waiting delay of a customer from entry to completion. The efficiency of the proposed solution approach was demonstrated through several numerical experiments.

In same year, Can et al. (2008) presented a comparative study of different stochastic components of genetic algorithms, such as operators, fitness assignment strategies and elitism, for simulation-based optimization of the buffer allocation problem. They also incorporated problem specific knowledge to further enhance the practicality of genetic algorithms in decision making for buffer allocation problem. Later, Can and Heavey (2009) presented a simulation-based evolutionary framework for constructing analytical meta-models and applied it to solve the buffer allocation problem in manufacturing lines. In this framework, a particle swarm algorithm was integrated with genetic programming to perform symbolic regression of the problem. The sampling data was sequentially generated by the particle swarm algorithm, while genetic programming evolved symbolic functions of the domain. The experimental results were promising in terms of efficiency in design of experiments and accuracy in global meta-modeling.

An efficient algorithm for finding the near optimal buffer allocation of a given number of total buffer sizes in a cellular remanufacturing system where the servers follow the  $N$ -policy with finite buffers was presented by Aksoy and Gupta (2010). In this study, the expected total cost of the remanufacturing system which was a function of the each station's throughput was considered. The system was modeled by using expansion method. The performance of the proposed algorithm was tested for both balanced and unbalanced lines covering a large experimental region and it has been observed that their proposed algorithm produced excellent results in a variety of experimental conditions.

Cruz et al. (2010) developed a multi-objective approach for the buffer allocation and throughput trade-off problem for single server queuing networks. It has been stated that combining generalized expansion method with a multi-objective genetic algorithm gave insightful Pareto curves. These curves explicitly showed the trade-off between buffer spaces and throughput. Their experimental results showed consistency with the optimal solutions found in earlier studies and supported the merits of the proposed solution approach.

Finally, Van Woensel et al. (2010) considered the joint optimization of the number of buffers and servers. The performance of the line was evaluated by using a combination of two-moment approximation and the generalized expansion method. The model presented in this study minimized the number of buffers and servers such that the resulting throughput is greater than a predefined threshold throughput rate. Similar to previous studies considering the same type of production systems (Daskalaki and MacGregor Smith, 2004, MacGregor Smith and Cruz, 2005, Yuzukirmizi and MacGregor Smith, 2008), the authors employed Powell's algorithm for optimization.

### ***3.2.2 Unreliable Lines***

In this section, the studies done since 1998 for buffer allocation problem in unreliable lines published are reviewed. A chronological list of these studies with respect to three classification criteria is presented in Table 3.2.

Vouros and Papadopoulos (1998) employed a knowledge based system to obtain the optimal buffer allocation plan for unreliable lines so as to maximize the throughput rate of the line. Simulation was used an evaluative method. The results of proposed solution approach were compared with the results of exact method. The authors stated that using specific types of knowledge the proposed system was computationally efficient and the results obtained were very close to the optimal.

A study by Gershwin and Schor (2000) which was based on M.S. thesis of Schor (1995) focused on both BAP1 and BAP2 problems which were previously defined in Chapter 2. In this study, the authors named BAP1 as a dual problem and BAP2 as a primal problem and they also considered profit maximization as an objective. Gradient based search algorithms were proposed to solve these problems. They first solved the dual problem and then used this solution to solve the primal problem. The performance of the proposed algorithms was tested on existing benchmark problems and better results were obtained. The throughput rate of the line was evaluated by using decomposition method. Moreover, a survey of related literature was provided in this study.

Seong et al. (2000) also proposed a gradient based search algorithm to solve the buffer allocation problem in a continuous flow unreliable production line so as to maximize the net profit of the system. Unlike the study of Gershwin and Schor (2000), the authors solved the buffer allocation problem under the linear constraints on buffer sizes and total buffer size capacity. Their computational experiments showed that the proposed gradient based algorithm was produced good solutions. The authors also showed the robustness of the proposed algorithm to the initial solution through numerical experiments.

Table 3.2 Overview on buffer allocation literature: Unreliable lines

Authors	Topology of the Line					Objective					Solution Methodology		
	S	S-P	GN	A	FMS	CMS	1	2	3	4	5	Evaluative	Generative
Vouros & Papadopoulos (1998)	x						x					Simulation	Knowledge Based Method
Gershwin & Schor (2000)	x						x	x		x		Decomposition Method	Gradient Search
Gurkan (2000)	x							x				Generalized Semi-Markov Process	Sample Path Optimization & Stochastic Approximation
Jeong and Kim (2000)				x						x		Simulation	Heuristic
Lee (2000)						x				x		Simulation	Heuristic
Seong et al. (2000)	x									x		Decomposition Method	Gradient Search
Helber (2001)	x			x								Decomposition Method	Gradient Search
Kim & Lee (2001)	x							x				Approximate Analytic Algorithm	Heuristic
Sörensen & Janssens (2001)	x									x		Approximate Analytic Algorithm	Heuristic
Papadopoulos & Vidalis (2001)	x						x					Markovian State Model	Heuristic
Dolgui et al. (2002)		x									x	Aggregation Method	Genetic Algorithms
Enginarlar et al. (2002)	x									x		Analytic Model & Simulation	Rules-of-Thumbs
Han & Park (2002)	x									x		Approximate Analytic Algorithm	Penalty Function & Steepest Descent Methods
Lee & Ho (2002)			x								x	Simulation	Response Surface Methodology
Roser et al. (2003)		x										Simulation	Shifting Bottleneck Detecting Approach
Shi & Men (2003)	x						x					Decomposition Method	Hybrid Method (Tabu Search & Nested Partitions)
Tempelmeier (2003)	x			x				x				Decomposition Method	Heuristic

Table 3.2 Overview on buffer allocation literature: Unreliable lines (cont.)

Authors	Topology of the Line					Objective					Solution Methodology		
	S	S-P	GN	A	FMS	CMS	1	2	3	4	5	Evaluative	Generative
Louw & Page (2004)			x								x	Open Queuing Network Model & Simulation	Analytic Method
Zequeira et al. (2004)	x								x			Mathematical Model	Grid Search
Aksoy & Gupta (2005)						x			x			Expansion Method	Heuristic
Allon et al. (2005)	x						x					Simulation	Cross-Entropy Method
Enginarlar et al. (2005)	x							x				Analytic Model	Rules-of-Thumbs
Nourelfath et al. (2005)	x						x					Decomposition Method	Ant Colony Optimization
Matta et al. (2005)	x						x					Simulation	Design of Experiments
Bulgak (2006)				x			x					Simulation	Genetic Algorithms & Artificial Neural Networks
Nahas et al. (2006)	x						x					Decomposition Method	Degraded Ceiling Algorithm & Simulated Annealing
Sabuncuoglu et al. (2006)	x						x					Simulation	Heuristic
Altiparmak et al. (2007)				x			x					Simulation	Artificial Neural Networks
Dolgui et al. (2007)		x									x	Aggregation Method	Hybrid Method (Genetic Algorithms & Branch-and-Bound)
Qudeiri et al. (2007)			x				x					Simulation	Genetic Algorithms
Ribeiro et al. (2007)	x										x	Mixed Integer Linear Programming Model	Mixed Integer Linear Programming Model
Battini et al. (2008)	x							x				Simulation	Experimental Cross Matrix
Demir & Tunali (2008)	x						x					Decomposition Method	Hybrid Method (Genetic Algorithms & Subgradient Method)
Qudeiri et al. (2008)		x					x					Aggregation Method & Simulation	Genetic Algorithms

Table 3.2 Overview on buffer allocation literature: Unreliable lines (cont.)

Authors	Topology of the Line						Objective					Solution Methodology	
	S	S-P	GN	A	FMS	CMS	1	2	3	4	5	Evaluative	Generative
Yamamoto et al. (2008)	x						x					Simulation	Genetic Algorithms
Zequeira et al. (2008)	x								x			Mathematical Model	Heuristic
Lee et al. (2009)	x						x	x				Simulation	Genetic Algorithms & Artificial Neural Networks
Nahas et al. (2009)		x					x					Decomposition Method	Ant Colony Optimization & Simulated Annealing
Shi & Gershwin (2009)	x									x		Decomposition Method	Nonlinear Programming
Vitanov et al. (2009)				x			x					Simulation	Ant Colony Optimization
Cehade et al. (2010)				x			x	x				Simulation	Ant Colony Optimization
Colledani et al. (2010)	x						x					Approximate Analytic Algorithm	Colledani et al. (2002)
Demir et al. (2010)	x						x					Decomposition Method	Adaptive Tabu Search
Kose (2010)			x				x					Simulation	Genetic Algorithms
Massim et al. (2010)	x						x			x		Decomposition Method	Artificial Immune Algorithm
Demir et al. (2011)	x						x	x				Decomposition Method	Tabu Search
Total	30	5	3	7	0	1	24	11	7	8	3		



Gurkan (2000) employed sample path optimization and stochastic approximation in unreliable serial production lines where the product type was fluid. The author derived recursive expressions to compute one-sided directional derivatives of throughput by utilizing a generalized semi-Markov process representation of the system. The objective was to minimize the function involving total buffer size and scaled throughput under some linear constraints. The author employed sample path optimization and stochastic approximation to optimize the buffer sizes. The experimental studies showed that both methods worked equally for small-sized problems while sample path optimization was superior to stochastic approximation in terms of computational effort for large-sized problems.

Lee (2000) addressed the buffer sizing problem under the objective of total cost minimization in a cellular manufacturing system. The cost function to be minimized includes the set-up cost to install the buffer storage and the cost of production loss due to limited buffer storage. This function was minimized by using a heuristic approach based on non-linear search strategy. Later, Lee and Ho (2002) solved to same problem for serial and job shop type production systems with multiple part types. In this study, simulation was used for modeling and buffer sizes were optimized via response surface methodology.

Jeong and Kim (2000) studied a different type of buffer allocation problem involving the selection of machines for each station and determination of buffer capacities in a tree-structured unreliable assembly system. The authors suggested three heuristics to find the minimum cost configuration which achieves a desired throughput rate. Starting from a lower (which consists of less efficient machines and large size buffers) or upper configuration (which consists of more efficient machines and small size buffers) these heuristics first generate promising machine configurations and then find the best buffer configurations for each machine configuration.

Considering the buffer allocation problem as an investment problem Helber (2001) employed gradient algorithm to determine the buffer allocation and the

performance of the algorithm was evaluated by using decomposition technique. To assess the economic consequences of the buffer allocation, the author linked the production rates and inventory levels to the projected cash flow and maximized the expected net present value of the investment including machines, buffers and inventory.

Papadopoulos and Vidalis (2001) investigated the buffer allocation problem in short unbalanced unreliable production lines consisting of up to six machines. They developed a heuristic algorithm based on sectioning approach to maximize the throughput of the line and like in their previous study (Papadopoulos and Vidalis, 1998), for performance evaluation they used Markovian state model method developed by Heavey et al. (1993).

Sörensen and Janssens (2001) developed an approximation model to determine the total availability for an unreliable production system. The total cost of the system was determined as a function of the required or desired availability of the system and of the total or average usage of the available buffer space. This cost was minimized by using a search algorithm.

Kim and Lee (2001) proposed a heuristic algorithm which was the modified version of the algorithm proposed by Seong et al. (1995) to solve the buffer allocation problem in a production line under given total buffer capacity and the minimum required throughput.

Han and Park (2002) presented an approximation method for the analysis of average steady state throughput of serial production lines with unreliable machines, finite buffers and quality inspection machines. Employing this approximation method, the authors proposed an analytic method for the optimal buffer allocation to achieve a desired throughput rate. The method involved penalty function and steepest descent methods and it was validated by computer simulations.

Enginarlar et al. (2002) investigated the smallest level of buffering to ensure the desired throughput rate in serial lines with unreliable machines. In this study, the reliability of machines was assumed to obey either exponential, Erlang or Rayleigh models. The dependence of level of buffering on the reliability model, the number of machines, the average uptime, and the efficiency was analyzed and as a result some rules-of-thumbs were provided to select buffer capacity so as to achieve desired throughput rate. Later, Enginarlar et al. (2005) considered the same problem in unreliable serial lines with identical exponential machines and they provided qualitative insights into the nature of lean buffering in serial production lines.

In recent years, it has been observed that meta-heuristic methods are successfully employed for solving buffer allocation problem. Genetic algorithm is one of these well-known methods. Dolgui et al. (2002) proposed genetic algorithm to solve buffer allocation problem for unreliable serial-parallel production lines so as to maximize the profit. It was assumed that the failure and repair rates of the machines were exponentially distributed and the machines have deterministic processing times. The performance measures of this line were evaluated using Markov-model aggregation method. The performance of the proposed genetic algorithm was compared with the pure genetic algorithm, complete enumeration and the Monte Carlo methods and better quality of solutions were obtained with the proposed GA. Later, Dolgui et al. (2007) proposed a hybrid approach involving genetic algorithms and branch-and-bound method for the same problem. The authors used genetic algorithm to obtain the initial solution for branch-and-bound method and stated that to do so shortens the total running time of the algorithm comparing with pure branch-and-bound algorithm.

Another study hybridizing a meta-heuristic method with any other search techniques was presented by Shi and Men (2003). In this study the nested partitions method was hybridized with tabu search method. The basic tabu search method was incorporated into the nested partitions framework so as to maximize the throughput rate of the line. The decomposition method was used as an evaluative method. The results obtained by the proposed hybrid approach was compared to the results

obtained by the basic tabu search and local search algorithms and it was found that their proposed hybrid approach was the best one among the others in terms of both solution quality and solution time.

Roser et al. (2003) described a prediction model to estimate the effect of increased buffer capacity onto the system performance based on only a single simulation. Although the proposed method was meant to be for large balanced and unbalanced systems and serial/parallel manufacturing systems, the authors stated that it can be adapted to non-manufacturing discrete event systems.

The real life problems considering the optimal buffer allocation problem in unreliable production lines were presented by Tempelmeier (2003). Both deterministic and variable processing times were considered. The author employed decomposition method for evaluating system performance and a heuristic approach for finding the optimal buffer and workload allocation simultaneously.

To estimate the size of the time buffers in theory of constraints controlled flow lines an open queuing network analysis approach was presented by Louw and Page (2004). Simulation experiments performed suggested that this approach could be efficiently applied in practice to estimate the length of the required time buffers.

Zequeira et al. (2004) presented a mathematical model to determine optimal buffer inventories and also optimal operational times to satisfy the demand during the maintenance action on a manufacturing facility. Later, Zequeira et al. (2008) added the issue of imperfect production possibility into the model and assumed that the opportunities to produce the buffer inventory and opportunities to carry out a maintenance action were random. The authors proposed a heuristic algorithm to determine the optimal buffer sizes and they used the mathematical model presented in Zequeira et al. (2004) for performance evaluation.

Nourelfath et al. (2005) employed ant colony optimization for solving buffer allocation problem in unreliable serial lines. The objective of this study was to

maximize the efficiency of the line subject to a total cost constraint. The performance of the line was estimated by using decomposition method. Moreover, to improve the performance of the ant colony optimization algorithm the authors developed two improvement algorithms. It has been observed that by combining these algorithms with ant colony optimization algorithm, the optimal/near-optimal solutions could be obtained more quickly.

The buffer allocation problem in remanufacturing systems was considered by Aksoy and Gupta (2005). In this study, Aksoy and Gupta (2005) developed a near optimal buffer allocation algorithm for cellular remanufacturing system with finite buffer capacities and unreliable servers. Authors considered server unavailability in the remanufacturing system by means of exponential breakdown and repair rate. To analyze this system expansion method was employed and a heuristic algorithm was suggested to optimize the buffer size so as to minimize the total cost. The performance of the algorithm was tested for both balanced and unbalanced cells covering a large experimental region.

A new stochastic algorithm named as cross-entropy method was employed by Allon et al. (2005). The objective was to maximize the throughput of the line and the performance of the line was evaluated using simulation. Matta et al. (2005) also employed simulation to evaluate the performance of flow lines including common buffer area. An experimental design was conducted to determine significant factors on production rate of the system. The authors presented some practical considerations for manufacturing system designers.

Another study which employed simulation as an evaluative method was presented by Sabuncuoglu et al. (2006). The authors studied the cases with single and multiple bottleneck stations under various experimental conditions and presented a heuristic algorithm to maximize the throughput rate of the line. The results of computational experiments showed that the proposed heuristic algorithm was very efficient in terms of both solution quality and also solution time. Moreover, some-rules-of thumbs for characterizing the optimal buffer allocation were given in this study.

Nahas et al. (2006) proposed a degraded ceiling algorithm for solving buffer allocation problem under the objective of throughput maximization for unreliable production lines. To estimate the throughput rate of the line the decomposition method was employed. The results obtained by degraded ceiling algorithm were compared with simulated annealing algorithm using existing benchmark problems which involved both homogeneous and non-homogeneous production lines. It was reported that better results were obtained by degraded ceiling algorithm since degraded ceiling algorithm converged to the optimum solutions quickly than simulated annealing algorithm. In 2009, Nahas et al. (2009) considered both buffer allocation and machine selection problem in a series-parallel production line so as to maximize the production rate of the line. To estimate the performance of this production line an analytical decomposition-type approximation was employed. The decision variables, i.e. buffer sizes and the number of parallel machines were optimized by using ant colony optimization. The results obtained by ant colony optimization were compared with the results obtained by simulated annealing algorithm. It has been observed that ant colony optimization produced better results than simulated annealing algorithm.

Bulgak (2006) studied optimal interstage buffer allocation problem of split-and-merge unpaced open assembly systems. The system studied in this paper was a modified version of the assembly system described by Hemachandra and Eedupuganti (2003). The author developed a simulation model in conjunction with genetic algorithms to find optimal inter-stage buffer configurations yielding a maximum production rate. To make the proposed method computationally more efficient, the author also proposed simulation meta-modeling based on artificial neural networks. Altiparmak et al. (2007) also investigated meta-modeling opportunities in buffer allocation and performance modeling in asynchronous unreliable assembly systems. In this study, a meta-model based on the artificial neural network was developed and simulation was employed as an evaluative method. The authors stated that the artificial neural network could successfully be used for modeling assembly systems.

Another study employing genetic algorithm was presented by Qudeiri et al. (2007). In this study, a simulator which includes a genetic algorithm to optimize the buffer sizes for complex production systems was developed. The authors introduced a new encoding method called multi-vector encoding method for genetic algorithms and stated that using this new encoding method the optimal buffer sizes can be obtained after a few number of generations. Following this study, Yamamoto et al. (2008) employed the same solution methodology to determine the buffer sizes for a flexible transfer line with bypass lines and Qudeiri et al. (2008) proposed genetic algorithms for optimal buffer allocation in an unreliable serial-parallel production line. Qudeiri et al. (2008) used the gene family arrangement method to express the chromosome and proposed new crossover and mutation methods. The authors evaluated the performance of the line using aggregation method and they optimized buffer sizes between each pair of workstations, number of machines in each workstation and the types of machines. It was observed that the production efficiency was improved by using their proposed solution method.

Ribeiro et al. (2007) aimed at jointly optimizing the maintenance of a capacity constrained resource, its feed machine/operation and inlet buffer size. A mixed integer linear programming model was developed to maximize total profit in the planning horizon. A two-machine line example was used to illustrate the implementation of the model.

Demir and Tunali (2008) proposed a hybrid solution approach combining subgradient algorithm proposed by Gasimov and Ustun (2007) and genetic algorithms within the same framework to maximize the throughput of the production line. To evaluate the throughput of the line an analytical decomposition approximation method was used. The performance of the proposed approach was demonstrated by a case study and promising results were obtained.

In same year, Battini et al. (2008) presented a new paradigm: the buffer design for availability. The main idea was to construct the buffer configuration based on availability performance of production lines. In this study, only micro downtimes

were considered. Simulation was used as an evaluation method and a new experimental cross matrix based on the guide index was provided as a generative method to determine the optimal buffer sizes. Moreover, the authors evaluated the effects of workstation reliability parameters on buffer capacities using simulation and also presented simple guidelines to support and help manufacturing system designers for rapid and robust buffer design.

In a recent study, Shi and Gershwin (2009) presented a nonlinear programming approach for maximizing profits through buffer size optimization in production lines. In this study, both buffer space cost and average inventory cost with distinct cost coefficients for different buffers and also a nonlinear production rate constraint have been considered. Moreover, numerical results were provided to show the efficiency and accuracy of the proposed algorithm for both short and long production lines.

Another ant colony optimization algorithm for solving buffer allocation problem in assembly lines was proposed by Vitanov et al. (2009). The algorithm was designed to work in conjunction with a simulation model and adapted to have both combinatorial and stochastic problem solving capacity.

Lee et al. (2009) proposed an artificial intelligence based method to investigate the buffer allocation of unbalanced-unreliable flow type production lines. Genetic algorithm combined with simulation method was used in attempting to quickly figure out the best solutions. In turn, these optimal solutions were fed into an artificial neural network for predicting buffer allocations. The performance of the proposed method was evaluated by using benchmark problems previously published in literature.

Colledani et al. (2010) proposed a new methodology based on analytical methods to support Scania, the manufacturer of heavy trucks, buses, and industrial and marine diesel engines. Besides improving the performance of the system through re-configuration, the authors also considered repair crew optimization and buffer size



optimization problems. The buffer allocation problem was solved using the method proposed by Colledani et al. (2004) under the objective of throughput maximization.

Demir et al. (2010) presented an adaptive tabu search approach to maximize the throughput rate of the line for non-homogeneous lines. In this study, tabu search parameters were tuned adaptively during the search so as to improve the search process. The performance of the proposed solution approach was tested on randomly generated test problems. In another study, Demir et al. (2011) proposed a tabu search approach for solving buffer allocation problem under the objective of both throughput maximization and also total buffer size minimization in homogeneous lines. The performance of the proposed approach was tested on previously published benchmark problems and promising results were obtained.

Another implementation of genetic algorithms for solving buffer allocation problem was presented by Kose (2010). In this study a real production system was modeled using simulation and buffer sizes were optimized using genetic algorithms.

Cehade et al. (2010) proposed a new multi-objective solution approach for solving buffer allocation problem in assembly lines. For each buffer size a range was considered including a lower and upper bound. Two objectives were taken in consideration: throughput maximization and total buffer size minimization. The proposed solution method was based on a multi-objective ant colony optimization algorithm using the Lorenz dominance instead of the well-known Pareto dominance relationship. The author stated that the Lorenz dominance relationship provided a better domination area by rejecting the solutions founded on the extreme sides of the Pareto front. The results obtained were compared with those of a classical multi-objective ant colony optimization algorithm. The numerical results showed the advantages and the efficiency of the Lorenz dominance.

Finally, in the study of Massim et al. (2010), an artificial immune algorithm was proposed to solve buffer allocation problem for unreliable production lines so as to maximize the throughput of the line and also profit under the total buffer capacity

constraint. The algorithm was tested on previously published benchmark problems, and equivalent or improved results were obtained.

### 3.3 Motivation

Based on the survey of the current studies on buffer allocation problem (see Tables 3.1 and 3.2), we could list our findings as follows:

- Most of the studies consider the issue of machine breakdowns (46 out of 76 studies).
- The serial line configuration is the most studied line configuration (46 out of 76 studies).
- The throughput maximization problem (i.e. BAP1) is the most studied problem in the current literature (41 out of 76 studies).
- Only 8 out of 76 studies solve the problem under the objective of both throughput maximization and also total buffer size minimization.
- A great majority of studies (49 out of 76 studies) employ analytic methods as an evaluative method for solving the problem. It has been observed that when the line studied becomes more complex as in the case of assembly lines, simulation is used as an evaluative method. It has been noted that 8 out of 11 studies involving assembly lines employ simulation as an evaluative method.
- In recent years, the meta-heuristics are widely used for solving the buffer allocation problem because of their capability in handling combinatorial optimization problems. More than half of these studies (13 out of 22 studies) employ genetic algorithms as a generative method.
- Only 3 out of 76 studies employ a hybrid approach to optimize the buffer sizes (Shi and Men, 2003, Dolgui et al., 2007, and Demir and Tunali, 2008) and all of these studies solve the buffer allocation problem under the objective of throughput maximization.

In the light of the current literature, we could state that a great majority of researchers studied the buffer allocation problem either under the objective of

throughput maximization or total buffer size minimization. Since the buffering allows the machines to operate nearly independently of each other and in this way it helps to increase the throughput rate of the system, there are usually floor space and budget constraints in reality. Generally, the ultimate aim is to improve the system performance with the minimum cost as in the case of all manufacturing system problems. So, one should consider both objectives while solving buffer allocation problem so as to obtain the best solution.

Moreover, there were a few studies using hybrid meta-heuristic approaches to solve this complex problem. As it is stated before, the buffer allocation problem is an NP-hard combinatorial optimization problem where the decision variables, i.e. buffer sizes, are integer. As in all cases of combinatorial problems, to find optimum solutions by exact methods in a reasonable amount of time is impossible when the problem size increases. To overcome to this difficulty and to find (near-) optimal solutions, in recent years, meta-heuristic methods are widely employed for solving combinatorial optimization problems. Moreover, to improve the search process, meta-heuristic methods are hybridized with any other meta-heuristic or optimization methods in a complementary fashion. Over the last years, it has been reported that the best results for many combinatorial problems are obtained by hybrid algorithms. Combinations of algorithms such as simulated annealing, tabu search and evolutionary algorithms have provided very powerful algorithms as it can be seen in the buffer allocation problem context (see Shi and Men, 2003, Dolgiu et al. 2007, and Demir and Tunali, 2008). However, the buffer allocation problem is solved only under the objective of throughput maximization in all mentioned studies. Hence, solving buffer allocation problem by using a hybrid meta-heuristic approach and testing its efficiency on total buffer size minimization problem seems an open area in buffer allocation literature.

Considering these opportunities and unfulfilled research potential in this area, this Ph.D. study aims at suggesting hybrid algorithms to solve the buffer allocation problem in unreliable production lines under the objective of both throughput maximization and also total buffer size minimization.

### 3.4 Chapter Summary

In this chapter, the current studies on buffer allocation problem were extensively reviewed to identify the current research gaps and also state the motivation for this Ph.D. thesis.

Our review includes the studies published after 1998. To review the current relevant literature, first the studies are divided into two categories: reliable lines which are not subject to failure, and unreliable lines which are subject to failure. The studies are classified according to the topology of the line, considered objective function and the type of the evaluative and generative solution method employed to solve the buffer allocation problem.

Based on the findings of this survey, this Ph.D. study aims at suggesting tabu search algorithms to solve the buffer allocation problem in unreliable production lines under the objective of both throughput maximization and also total buffer size minimization. For this purpose, the buffer allocation problem is first solved under the objective of throughput maximization. Next chapter presents the details of our proposed TS algorithm for throughput maximization problem in unreliable homogeneous production lines.

## **CHAPTER FOUR**

### **A TABU SEARCH APPROACH FOR THROUGHPUT MAXIMIZATION IN UNRELIABLE HOMOGENEOUS PRODUCTION LINES**

#### **4.1 Introduction**

The main objective of this Ph.D. study is to develop efficient algorithms for solving buffer allocation problem in unreliable production lines. For this purpose, we solved the problem in three stages as it is stated in the first chapter. This chapter presents what we have done at the first stage of this Ph.D. study.

In this chapter, a TS algorithm is proposed to solve buffer allocation problem under the objective of throughput maximization for homogeneous production lines involving unreliable machines with deterministic processing times. The new move definitions for buffer allocation problem are introduced and a pilot experiment is carried out to identify the best TS parameters. The performance of the proposed TS algorithm is tested on benchmark problems previously published in literature.

The rest of this chapter is organized as follows. In section 4.2, the specifications of the problem are given. The details of the proposed TS algorithm are presented in section 4.3. The results of experimental studies carried out to test the performance of the proposed TS algorithm are discussed in Section 4.4. Finally, in Section 4.5 the context of this chapter is summarized.

#### **4.2 Problem Specifications**

This chapter focuses on serial production lines involving unreliable machines with the same processing rates. The objective is to find the optimal buffer allocations so that the throughput rate of the line can be maximized for a given fixed amount of buffers. As given earlier in chapter 2, this problem is called as BAP1. To deal with

this problem, a tabu search algorithm which is adapted to the intrinsic features of this problem is proposed in this chapter.

The following gives the summary of the features of the production line studied:

- All the machines along the line have the same processing rate of one unit of time.
- There is an intermediate location for storage between each pair of machines.
- Each part goes through all the machines in exactly the same order.
- The machines are subject to breakdown and the repair and failure rates of the machines are geometrically distributed.
- The first machine is never starved, i.e. input is always available, and the last machine is never blocked, i.e. there is always space to dispose of the output.

Assuming that there are  $K$  machines and  $K-1$  buffers in the production line, our objective is to maximize the throughput rate of the production line, subject to a given total buffer capacity. The problem can be formulated as follows:

Find  $B = (B_1, B_2, \dots, B_{K-1})$  so as to

$$\max f(B) \tag{1}$$

subject to

$$\sum_{i=1}^{K-1} B_i = N \tag{2}$$

$$B_i \text{ nonnegative integers } (i=1, 2, \dots, K-1) \tag{3}$$

where  $N$  is a fixed nonnegative integer denoting the total buffer space available in the system which has to be allocated among the  $K-1$  buffer locations so as to maximize the throughput rate of the production line. In this formulation  $B$  represents a buffer vector,  $B_i$  is the buffer size for each location and  $f(B)$  represents the throughput rate of the production line with respect to the buffer configuration  $B$ .

To solve this buffer allocation problem, we propose a tabu search based meta-heuristic approach specifically adapted to the intrinsic features of this problem.

Unlike the studies of Lutz et al. (1998) and Shi and Men (2003), who also employ tabu search for the buffer allocation problem, a different tabu criterion, which is explained in next section, is employed in our TS algorithm and the problem is solved for both short and long production lines. In addition, considering the large size of the search space, a diversification strategy is incorporated into the algorithm to improve the search process for long production lines. The performance of the proposed algorithm (i.e., the throughput rate of the production line) is evaluated by using the decomposition method (Gershwin, 1987) which is explained in chapter 2.

### 4.3 Proposed TS Algorithm

The design of a meta-heuristic method to solve a combinatorial optimization problem first requires the definition of the basic components of the method. The following sections present the specifics of the proposed TS algorithm for solving buffer allocation problem.

#### 4.3.1 Move Representation and Tabu Moves

In this study, the moves are depicted by the notation  $[i, j]$ , meaning that *one* buffer is added at location  $i$ , and *one* buffer is subtracted at the location  $j$  ( $i$  and  $j$  can be any locations). As in all TS applications, constructing the tabu criterion is the most important element of the tabu search process. There are three natural ways to create the tabu criterion for the buffer allocation problem. The first one is to choose the *full move* as tabu which means if the move  $[i, j]$  produces the best objective function, then the reverse move is not permitted. Therefore the move  $[j, i]$  becomes a tabu for  $TT$  number of iterations. The second one is to choose the location as a tabu where one buffer is added. More precisely if the move  $[i, j]$  produces the best objective function, then the move  $[\text{all } j (i \neq j), i]$  is not permitted for  $TT$  number of iterations. In this way more than one move is blocked. This move is called as *move to target location*. The third way to create the tabu criterion is to choose the location as a tabu where one buffer is subtracted. More precisely if the move  $[i, j]$  produces the

best objective function, then the move  $[j, \text{all } i (i \neq j)]$  is not permitted for  $TT$  number of iterations. This move is called as *move from source location*.

In all previous TS applications on buffer allocation problem and also many other combinatorial problems, employing *full move* as a tabu criterion is the common sense. It should be noted that the performance of TS can be affected by the type of move chosen. Hence, we carried out an experimental study to test the effects of move types on the search performance and we showed that the performance of the search process is affected by the move type (see section 4.4.1).

### ***4.3.2 Search Space and Neighborhood Structure***

Identifying a search space along with a neighborhood structure is the most critical step of any TS implementation. The search space of the TS is simply the space of all feasible solutions that can be visited during the search. As it is stated by Gendreau and Potvin (2005) in some cases, allowing the search to move infeasible solutions is desirable. In buffer allocation problem context, since there is an integrality constraint on buffer sizes and also the total buffer size constraint, it is not reasonable to allow the infeasible solutions. So, in this study, the search space involves only the feasible solutions.

To define the neighborhood structures of the current solution, there are several choices. For instance, one choice could be to consider the full neighborhood of the current buffer configuration while the other could be to consider only a subset of the neighborhood of the current solution as in the study of Lutz. et al. (1998). In their study, since simulation is employed as an evaluative method and it takes more time to evaluate all neighborhoods by simulation, they consider only a small subset of the neighborhood of the current solution. Unlike their study, since we obtain the throughput rate of the line by employing decomposition method quickly, we evaluate all neighbors of the current solution. It also allows us not to disregard good solutions during the search.



### ***4.3.3 Diversification Strategy***

Diversification guides the search to unexplored regions to avoid local optimality. If the solution space is too large diversification should be used to explore the search space effectively. There are two major diversification techniques known as restart diversification and continuous diversification. While the first is performed by several random restarts, the latter is integrated into the regular search process.

Considering the large size of the search space, using the frequency based memory structures of TS we employ the continuous diversification strategy in a systematic way to improve the search process for large sized problems. Since we have a knowledge of the performed moves during the search it is more reasonable to employ continuous diversification than restart diversification which involves randomness and has a chance to perform same moves previously visited.

This strategy is implemented as follows. A counter is added to the algorithm to count the moves at each position. When the counter becomes a pre-specified value for any position, it is penalized to make it less attractive. In doing so, the search is moved to other unexplored areas of the search space.

### ***4.3.4 Aspiration Criterion***

Tabus may prohibit attractive moves, even when there is no danger of cycling, or they may lead to an overall stagnation of the searching process. Thus it is necessary to use aspiration criterion to allow the attractive moves. The aspiration criterion used in this study is the most common criterion used in the literature. According to aspiration criterion a tabu move is allowed to be made if and only if the resulting buffer configuration is better than the best configuration found so far. It should be noted that the new solution has not been previously visited.

### 4.3.5 Stopping Condition

The algorithm is terminated if within a certain number of iterations, no better solution has been found.

The proposed TS algorithm is outlined in Table 4.1. The notation used in this table is explained as follows:

$B^0$	initial buffer configuration
$B^k$	buffer configuration at iteration $k$
$NB(B^k)$	the full neighborhood of the current buffer configuration
$B^{best}$	the best buffer configuration during the whole TS algorithm
$B^{best-tabu}$	the best buffer configuration in $NB(B^k)$ that can be reached from $B^k$ by a tabu move in the current neighborhood
$B^{best-nontabu}$	the best buffer configuration $NB(B^k)$ that can be reached from $B^k$ by a non-tabu move in the current neighborhood
$f(B)$	the throughput rate of the line for the given buffer configuration $B$
$f^{best}$	the best throughput rate of the line during the whole TS algorithm
$TT$	tabu tenure

Table 4.1 Proposed TS algorithm for throughput maximization

<p><i>Initialization</i></p> <p>Generate an initial buffer configuration randomly. Initially tabu list (<math>TL</math>) is empty and iteration number <math>k = 0</math>. Set <math>f^{best} = f(B^0)</math> and <math>B^{best} = B^0</math>.</p> <p><i>Step 1</i></p> <p>Create all neighborhoods of the current solution, <math>NB(B^k)</math>, and invoke the evaluation function for each neighbor of the current solution to determine the potential new objective function value, i.e. the throughput rate of the line.</p> <p><i>Step 2</i></p> <p>Select the buffer configuration in <math>NB(B^k)</math> such that one of the following will hold:</p> <ol style="list-style-type: none"> <li>1. If <math>f(B^{best}) \geq \max(f(B^{best-nontabu}), f(B^{best-tabu}))</math>, set <math>B^{k+1} = B^{best-nontabu}</math>, put the moves <math>[all\ j\ (i \neq j), i]</math> in <math>TL</math> for the <math>TT</math> number of iterations where the move <math>[i, j]</math> produces the buffer configuration <math>B^{best-nontabu}</math>,</li> <li>2. If <math>f(B^{best-nontabu}) \geq \max(f(B^{best}), f(B^{best-tabu}))</math>, set <math>B^{k+1} = B^{best-nontabu}</math>, <math>B^{best} = B^{k+1}</math>, <math>f^{best} = f(B^{k+1})</math>, put the moves <math>[all\ j\ (i \neq j), i]</math> in <math>TL</math> for the <math>TT</math> number of iterations where the move <math>[i, j]</math> produces the buffer configuration <math>B^{best-nontabu}</math>,</li> <li>3. <i>Aspiration criterion:</i> If <math>f(B^{best-tabu}) \geq \max(f(B^{best}), f(B^{best-nontabu}))</math>, then set <math>B^{k+1} = B^{best-tabu}</math>, <math>B^{best} = B^{k+1}</math>, <math>f^{best} = f(B^{k+1})</math>, put the moves <math>[all\ j\ (i \neq j), i]</math> in <math>TL</math> for the <math>TT</math> number of iterations where the move <math>[i, j]</math> produces the buffer configuration <math>B^{best-tabu}</math>.</li> </ol> <p><i>Step 3</i></p> <p>Set <math>k = k + 1</math> and go to Step 1 until the termination criterion is satisfied.</p> <p><i>Termination criterion</i></p> <p>The algorithm is terminated if within a certain number of iterations, no better solution has been found.</p>
---

As it is seen in Table 4.1, the step 2 is the phase where the best solution is updated. At each iteration, the best buffer configuration which can be reached by a non-tabu move in the current neighborhood is selected for the next step. The only exception is defined by the aspiration criterion which allows a tabu move to be made if and only if the resulting buffer configuration is better than the best configuration found so far. The algorithm is terminated if within a certain number of iterations, no better solution has been found.

## 4.4 Computational Experiments

In this section, the computational experiments carried out to test the performance of the proposed TS algorithm by using existing benchmark problems are presented. Before giving the details of our computational study, we first present how tabu search parameters used throughout the search process are determined.

### 4.4.1 Identification of the Best Tabu Search Parameters

Since the search performance is heavily affected by the search parameters chosen, identifying best search parameters is an important step of any meta-heuristic application. So this section is devoted to identification of efficient tabu search parameters for solving buffer allocation problem so as to increase the search performance. The search parameters investigated are the type of tabu move and the size of the tabu list, i.e. tabu tenure. The types of moves are explained in section 4.3.1.

The other search parameter investigated is tabu tenure. The tabu tenure can be constant or it may change dynamically during the search as it is stated in section 2.4.2.2. In this study, the performance of these two types of tabu tenure is evaluated in three levels (see Table 4.2). Regarding three levels of constant  $TT$ , the tabu tenure is set to  $\sqrt{n}$  where  $n$  is the total number of all neighborhoods of the current buffer configuration. Likewise the region of experimentation for dynamic  $TT$  has been identified as a result of some number of pilot experiments.

Table 4.2 The levels for tabu tenure

Tabu Parameter	Levels		
	1	2	3
Constant $TT$	3	9	19
Dynamic $TT$	Uniform(3,7)	Uniform(7,15)	Uniform(9,27)

The performance of two types of  $TT$  and three types of tabu moves are evaluated on six sets of problems each involving three instances. It should be noted that these three instances are generated by changing the failure/repair rates of machines and

also the total buffer size to be allocated to the machines. Using a 3 GHz Pentium (R) 4 CPU processor 10 runs are carried out at each design point leading to 180 runs in total.

The results of experiments are summarized in Table 4.3 with respect to the number of iterations required for convergence and also the CPU time used by the TS algorithm with different search parameters. As seen in Table 4.3, regarding the performance of three tabu move criteria investigated, except for the fifth problem set, both in small-sized problems (i.e., involving five machines) and also large-sized problems (i.e., involving 20 machines), move to target location has slightly better rate of convergence. As summarized in Table 4.3, employing the tabu search algorithm with constant  $TT$  and employing full move as a tabu criterion solves the fifth set of problems with higher convergence rates and in less CPU time. For small-sized problems, involving five machines, a significant difference between the performances of three moves has not been observed.

Lastly, regarding the three levels of constant and dynamic  $TT$ , it has been noted that using constant  $TT$  results in higher rate of convergence and also less CPU time in solving great majority of the problems studied. As a result of experimental study, in testing the performance of our TS algorithm, we decided to employ constant  $TT$  in solving all types of problems, the full move in solving only large-sized problems with same machine failure/repair rates and the move to target location in solving all other problems.

Table 4.3 The results of experimental study for tabu search parameters

Problem Set	# of Machines	Failure/Repair rates*	Tabu Tenure*	Tabu Move Criterion***	Instance 1		Instance 2		Instance 3	
					CPU (sec)	# of Iteration for Convergence	CPU (sec)	# of Iteration for Convergence	CPU (sec)	# of Iteration for Convergence
1	5	S	Con/1	1	0.01	7	0.01	7	0.02	7
			Con/1	2	0.01	7	0.01	7	0.02	7
			Con/1	3	0.01	7	0.01	7	0.02	7
			Dyn/1	1	0.01	7	0.01	7	0.02	7
			Dyn/1	2	0.01	7	0.01	7	0.02	7
			Dyn/1	3	0.01	7	0.01	7	0.02	7
2	5	D	Con/1	1	0.11	7	0.17	9	0.36	35
			Con/1	2	0.12	5	0.17	7	0.34	33
			Con/1	3	0.12	6	0.17	8	0.35	45
			Dyn/1	1	0.13	7	0.17	9	0.34	63
			Dyn/1	2	0.14	7	0.18	9	0.34	66
			Dyn/1	3	0.14	8	0.19	9	0.37	60
3	10	S	Con/2	1	2.84	14	4.59	13	5.16	14
			Con/2	2	2.84	15	4.70	14	5.28	16
			Con/2	3	2.95	14	4.70	13	5.26	16
			Dyn/2	1	2.85	15	4.61	14	5.17	14
			Dyn/2	2	2.85	13	4.71	14	5.31	14
			Dyn/2	3	2.98	17	4.71	14	5.37	14
4	10	D	Con/2	1	4.45	23	6.77	63	7.24	102
			Con/2	2	4.42	24	6.68	68	7.18	107
			Con/2	3	4.48	24	6.74	69	7.21	102
			Dyn/2	1	4.45	27	6.87	67	7.24	108
			Dyn/2	2	4.51	26	6.73	75	7.32	101
			Dyn/2	3	4.52	26	6.72	83	7.32	97
5	20	S	Con/3	1	31.44	30	79.38	35	80.36	32
			Con/3	2	31.95	34	79.73	71	81.79	73
			Con/3	3	32.08	55	79.70	85	81.11	56
			Dyn/3	1	32.10	30	79.45	38	82.55	40
			Dyn/3	2	31.87	34	80.94	88	83.84	36
			Dyn/3	3	31.98	42	79.85	72	83.43	59
6	20	D	Con/3	1	63.58	79	198.80	151	109.27	250
			Con/3	2	62.34	64	194.94	155	104.59	236
			Con/3	3	62.39	70	197.56	297	108.58	243
			Dyn/3	1	63.34	69	285.17	116	219.19	269
			Dyn/3	2	62.54	75	278.91	165	236.80	248
			Dyn/3	3	62.75	83	280.62	269	239.86	290

\* S: Failure/repair rates are same for all machines in the line, D: Failure/repair rates are different for all machines in the line  
\*\* Con/*i*: Con:Constant, *i*=1,2,3: Levels of tabu tenure ( See Table 4.2)  
\*\* Dyn/*i*: Dyn:Dynamic, *i*=1,2,3: Levels of tabu tenure ( See Table 4.2)  
\*\*\* 1: Full move, 2: Move to target location, 3: Move from source location

#### 4.4.2 Experiments on Benchmark Problems

The experimental studies consisted of five sets of problems (i.e., five, nine, ten, twenty and forty-machine production lines). The machine parameters, i.e. the mean time to repair and the mean time between failures are denoted by MTTR and MTBF, respectively, in the following tables. Both the proposed TS algorithm and DDX algorithm to evaluate the throughput of the line are implemented in the C language.

Tabu search parameters are determined as follows:

- Initial buffer sizes are generated randomly from a uniform distribution,  $U(0, (2 \times \frac{N}{K-1}))$ . Since the buffer sizes must be integer, the upper bound value is rounded up to the next integer value. This procedure is repeated until the constraint related to the total number of buffers to be allocated is satisfied.
- The search continues until the best throughput value does not change during the certain number of iterations.

Using a 3 GHz Pentium (R) 4 CPU processor, 10 runs are carried out for each example under different initial buffer allocations.

#### 4.4.2.1 Five Machine Line

This example is initially proposed by Ho et al. (1979) and used in Gershwin and Schor (2000). The parameters of the line are given in Table 4.4. The total buffer size is set to 31.

Table 4.4 Five machine line parameters

Machine	1	2	3	4	5
MTTR= $1/r_i$	11	19	12	7	7
MTBF= $1/p_i$	20	167	22	22	26

As a result of experimental studies, it has been noted that the proposed TS algorithm managed to obtain the same throughput value, 0.4943 as in Gershwin and Schor (2000) with optimal buffer allocation,  $B = \{7,10,10,4\}$  after six iterations on average. It should be noted that reaching the optimal solution quickly, as in this case, could be a great advantage in solving large sized problems.

#### 4.4.2.2 Nine Machine Lines

These test cases are proposed in Shi and Men (2003) where the authors employ the DDX algorithm for evaluation and the Nested Partitions/Tabu Search (NP/TS) method for optimization. The authors implement the TS in its simplest form to speed

up the NP method. The results of comparative experimental studies are given in Table 4.5. All the machines in the line are subject to the same probability of failure ( $p_i$ ) and probability of repair ( $r_i$ ) (see the first and second columns). Moreover, the total buffer size is set to 160.

As given in Table 4.5, the proposed TS algorithm results in nearly the same throughput in comparison to the two approaches used in Shi and Men (2003). The observed subtle differences among three approaches can be attributed to applying different convergence rates. While in this study the convergence rate of the DDX algorithm is set  $10^{-6}$ , in Shi and Men (2003) this rate has not been reported.

Table 4.5 Results of comparative experimental studies

Parameters		Throughput		
$p_i$	$r_i$	TS (Shi&Men, 2003)	NP/TS (Shi&Men, 2003)	Proposed TS
0.3	0.05	0.108143	0.108143	0.108159
0.3	0.1	0.197546	0.200250	0.200360
0.3	0.2	0.345491	0.345491	0.345576
0.3	0.3	0.452074	0.452074	0.451660
0.3	0.4	0.528466	0.532002	0.532184
0.4	0.05	0.088777	0.088777	0.088771
0.4	0.1	0.166232	0.166232	0.166226
0.4	0.2	0.293041	0.293041	0.293060
0.4	0.3	0.388517	0.390814	0.390962

#### 4.4.2.3 Ten Machine Line

The failure and repair rates for this ten-machine production line are given in Table 4.6 (Nahas *et al.* 2006). The total buffer size is set to 270.

Table 4.6 Ten machine line parameters

Machine	1	2	3	4	5	6	7	8	9	10
MTTR= $1/r_i$	7	7	5	10	9	14	5	8	10	10
MTBF= $1/p_i$	20	30	22	22	25	40	23	30	45	20

In Nahas *et al.* (2006) the authors employ the DDX algorithm suggested in Dallery *et al.* (1989) to obtain the throughput of the line and they use the degraded



ceiling method for optimization. In this example the same evaluation method, i.e. DDX algorithm, is employed and the convergence rate of the DDX algorithm is set to  $10^{-4}$  as in the study of Nahas et al. (2006). The best throughput obtained by the proposed TS method is 0.641348 with buffer allocation  $B = \{14, 19, 30, 54, 45, 26, 23, 25, 34\}$ .

The two algorithms are also compared with respect to the number of iterations required for convergence. As it is seen in Figure 4.1 and 4.2, the proposed TS algorithm converges faster (78 iterations) to the optimal solution than the degraded ceiling algorithm (14 000 iterations). Having set the same convergence rate for two algorithms, we can state that the proposed TS has great advantage to reach the best solution as it requires much less number of iterations.

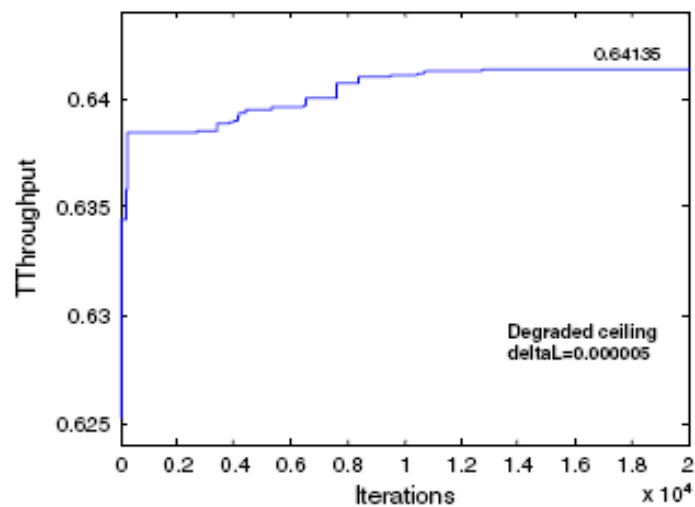


Figure 4.1 Convergence of degraded ceiling algorithm (Nahas et al., 2006)

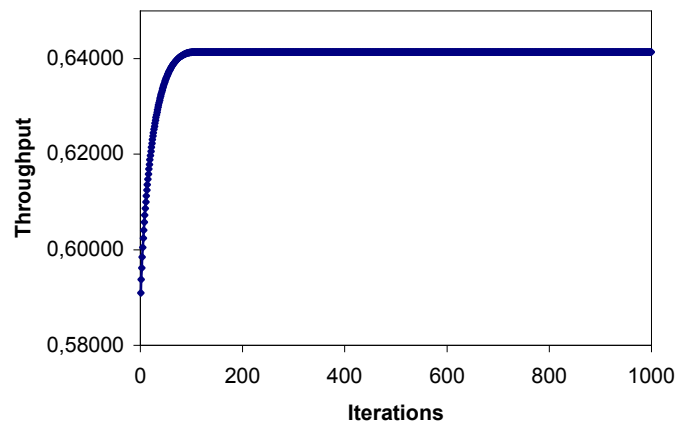


Figure 4.2 Convergence of the proposed TS algorithm

#### 4.4.2.4 Long Production Lines

The motivation for this example is to test the efficiency of our TS algorithm in solving the buffer allocation problem on long production lines. Throughout the experiments it has been assumed that the machines have the same probability and failure rates, *i.e.*  $p_i = r_i = 0.1$  up to  $p_i = r_i = 0.9$  in increments of 0.1. Considering the large size of the search space, the continuous diversification strategy is employed to improve the search process so that the most interesting parts of the search space can be explored more thoroughly. This strategy is implemented as follows:

- A counter is added to the algorithm to count the moves at each position.
- When the counter becomes 100 for any position, it is penalized to make it less attractive.
- As a result of pilot experimental studies, the weight to penalize is determined as  $10^{-4}$  for all cases. In doing so, the search is moved to other unexplored areas of the search space.

The total buffer size to be allocated is set to 100 for 20-machine line and 200 for 40-machine line, respectively. The convergence rate of the DDX algorithm is set to  $10^{-4}$ . The results of experimental studies comparing the proposed TS to the basic TS with respect to both throughput and the CPU time are given in Table 4.7. As opposed to the earlier experiments involving 10 runs for each example, due to the large size of

problem only five runs with different starting solutions have been conducted in these sets of experiments.

Table 4.7 Results of experimental studies for 20-machine lines

Parameters		Throughput		CPU (sec.)	
$p_i$	$r_i$	Basic TS	Proposed TS	Basic TS	Proposed TS
0.1	0.1	0.233963	0.234191	361.90	1082.65
0.2	0.2	0.296989	0.297025	436.86	1162.94
0.3	0.3	0.334450	0.334450	475.63	1139.95
0.4	0.4	0.359637	0.359637	506.37	1148.21
0.5	0.5	0.377842	0.377895	512.10	1213.13
0.6	0.6	0.391712	0.391846	518.04	1291.44
0.7	0.7	0.402760	0.402760	541.05	1011.59
0.8	0.8	0.411629	0.411638	531.87	1033.17
0.9	0.9	0.419017	0.419018	516.60	1009.83

As seen in Table 4.7, in all of the cases, the proposed TS algorithm results in equal or slightly better throughput compared to the basic TS algorithm. This can be attributed to adopting the continuous diversification method into the regular tabu search in this study. Furthermore, by using the frequency memory the search is extended to the unexplored areas of the search space, thus it has been escaped from local optima. However, as seen in Table 4.7, the good performance of the proposed TS with respect to throughput has been achieved at the expense of significantly increasing the search time. This shows the trade-off between search time and solution quality. As it is presented in Table 4.8, the results of comparative experimental studies involving 40-machine line are similar to those of 20-machine line.

In summary, the solution quality of the proposed TS algorithm in solving buffer allocation problem in long production lines is good enough. However, if the proposed TS algorithm is to be implemented in a real world system involving much larger number of machines, further research needs to be done to reduce the solution time.

Table 4.8 Results of experimental studies for 40-machine lines

Parameters		Throughput		CPU (sec.)	
$p_i$	$r_i$	Basic TS	Proposed TS	Basic TS	Proposed TS
0.1	0.1	0.219060	0.221273	5573.56	8399.04
0.2	0.2	0.286347	0.282169	7615.08	9780.96
0.3	0.3	0.325256	0.325256	8319.73	12796.32
0.4	0.4	0.351391	0.351494	8681.03	11272.68
0.5	0.5	0.370653	0.370666	7584.05	12581.64
0.6	0.6	0.385290	0.385328	7613.19	11897.04
0.7	0.7	0.396903	0.397255	7478.19	9773.04
0.8	0.8	0.406232	0.406559	7929.55	9473.40
0.9	0.9	0.413541	0.413990	7274.45	10815.12

#### 4.5 Chapter Summary

In this chapter a tabu search algorithm is proposed to solve the buffer allocation problem for unreliable homogeneous production lines, and its performance is compared to other algorithms using previously published benchmark problems. It should be noted that these benchmark problems include both short and long production lines so that the efficiency of the proposed TS can be tested thoroughly on a wide range of problem instances. The results of experimental studies are found to be quite encouraging. Namely, it has been observed that the proposed TS algorithm finds the best configuration much faster than other algorithms for short production lines. Likewise, the solution quality of the proposed TS algorithm in solving buffer allocation problem in long production lines is also good, but this is achieved at the expense of increased search time.

During the experimental studies, it has been noted that the search time greatly depends on the quality of the initial solution and the value of the convergence rate used in the DDX algorithm. If the initial buffers are equally distributed among the machines where all the machines in the line have the same reliability parameters it has been observed that search time is significantly reduced and likewise, if the convergence criterion used in the DDX algorithm is set to a large value then the search time is reduced too. Also, in some problem sets where the same repair and

failure rates are assumed for all machines in the line it has been noted that, the evaluation function quickly converges.

Due to its restrictive assumptions, the buffer allocation problem in homogeneous production lines does not realistically represent most of the real-world production lines. Consequently, it is absolutely necessary to consider a more realistic version of the production lines, i.e., non-homogeneous production lines. In the following chapter, the problem will be extended to the buffer allocation problem in unreliable non-homogenous production lines and an adaptive tabu search algorithm will be proposed to deal with this problem.

**CHAPTER FIVE**  
**AN ADAPTIVE TABU SEARCH APPROACH FOR THROUGHPUT**  
**MAXIMIZATION IN UNRELIABLE NON-HOMOGENEOUS**  
**PRODUCTION LINES**

**5.1 Introduction**

In this chapter, an adaptive tabu search is proposed to solve the buffer allocation problem in unreliable non-homogeneous production lines. To our knowledge, ours is the first extensive study dealing with buffer allocation problem for unreliable and also non-homogeneous lines. Imposing buffer space constraints for each buffer location makes the problem at hand even harder. An adaptive search strategy of intensification and diversification is proposed to solve the buffer allocation problem in both short and long production lines. An experimental study is carried out to select an intelligent initial solution scheme among three alternatives so as to decrease the search effort to obtain the best solutions.

This chapter is organized as follows. Section 5.2 presents the problem specifications. The proposed adaptive tabu search is described in detail in Section 5.3. The results of computational experiments to test the performance of the proposed adaptive tabu search algorithm are discussed in Section 5.4. Finally, Section 5.5 summarizes the context of this chapter.

**5.2 Problem Specifications**

The features of the buffer allocation problem in a non-homogeneous unreliable production line can be listed as follows:

- Each part goes through all machines in exactly the same order.
- There is an intermediate location for storage (buffer) between each pair of machines.
- Machines in the line have unique deterministic processing times.

- Machines are subject to breakdown, and the repair and failure rates are geometrically distributed.
- The first machine is never starved, i.e. input is always available, and the last machine is never blocked, i.e. there is always space to dispose of the output.

Assuming there are  $K$  machines and  $K-1$  buffers in a production line, the objective is to maximize the throughput rate of the production line, subject to buffer size constraints for each buffer location. The mathematical model for the problem is given as follows:

Find  $B = (B_1, B_2, \dots, B_{K-1})$  so as to

$$\max f(B) \quad (1)$$

subject to

$$\sum_{i=1}^{K-1} B_i = N \quad (2)$$

$$0 \leq B_i \leq u_i \quad (i=1, 2, \dots, K-1) \quad (3)$$

$$B_i \text{ nonnegative integers } (i=1, 2, \dots, K-1) \quad (4)$$

where  $N$  is a fixed nonnegative integer denoting the total buffer space available in the system,  $B$  represents a buffer vector, and  $f(B)$  represents the throughput rate of the production line. Constraint (3) shows upper ( $u_i$ ) bounds for each buffer location. It should be noted that upper ( $u_i$ ) bounds for each buffer location are chosen such that their summation will be larger than  $N$ .

To solve the buffer allocation problem under the above assumptions, an adaptive tabu search is proposed as a generative method. To evaluate the throughput of the line Accelerated-DDX (ADDX) algorithm proposed by Burman (1995) is employed. In next section, the proposed adaptive tabu search algorithm (ATS) is described in detail.

### 5.3 Proposed ATS Algorithm

To employ a tabu search on any combinatorial optimization problem, first the basic components of tabu search should be defined. The following sections present the specifics of the proposed ATS algorithm for solving buffer allocation problem.

#### 5.3.1 Move Representation and Tabu Moves

The moves are represented by  $[i, j]$ , where  $i$  shows the location that a given amount of buffer is added and  $j$  shows the location that the same amount of buffer is subtracted. These locations can correspond to any two buffer storages. For instance, for a five-machine line there will be four locations for buffer storages. If the initial buffer configuration is assumed to be  $[5, 5, 5, 5]$  the first neighborhood of this initial solution is created by subtracting one buffer from the first location and adding one buffer to the second location, while the size of all other buffers remains the same and the total buffer size is constant. One neighboring solution then becomes  $[4, 6, 5, 5]$ . The process is repeated until all neighboring solutions of the current solution are created. Since there are four locations for allocating buffers in a five-machine-line, there will be twelve solutions in the neighborhood for this example.

The increment (decrement) of a buffer size is problem-dependent. We tune the increment (decrement) of the buffer sizes based on problem size. While it is set to one for small and medium-sized problems, i.e. five and ten machine lines, for large-sized problems involving twenty and forty machine lines, the increment (decrement) is set to %1 of the total buffer size.

Once a move is realized, the reverse of this move is recorded as a tabu. Namely, if the move  $[i, j]$  produces the best solution for the current step, then the reverse move  $[j, i]$  is considered as a tabu for a certain number of iterations.



### 5.3.2 Search Space and Neighborhood Structure

Identifying a search space along with a neighborhood structure is the most critical step of any TS implementation. The search space of the TS is simply the space of all feasible solutions that can be visited during the search. In this study, all feasible solutions are considered as search space.

To define the neighborhood structures of the current solution, there are several choices depending on the specific problem at hand. In the buffer allocation problem context, one choice could be to consider the full neighborhood of the current buffer configuration while the other could be to consider only a subset of the neighborhood of the current solution. In the proposed ATS approach, the complete neighborhood of the current solution is created and evaluated.

### 5.3.3 Initialization Scheme

It is known that for some problems the performance of the meta-heuristics is affected by the choice of the initial solution/solutions. If the initial solution is good enough, the probability of finding better solutions generally increases and the convergence to the near-optimal or optimal solution can be faster. In this study, a pilot experiment was carried out to assess the performance of the following initialization methods:

- *The ratio of failure to repair rate*: Buffer sizes are allocated according to the ratio of failure to repair rate of each machine. More buffers are allocated to the machines having high ratios of failure to repair rate.
- *Processing time*: More buffers are allocated to the machines having long processing times.
- *Random initialization*: Buffer sizes are randomly allocated to each machine.

The experiments involve four problem sets, each containing 10 instances. Table 5.1 shows the results of the pilot experiment. In the table,  $K$  and  $N$  stand for the number of machines and the total buffer capacity, respectively.

Table 5.1 Results of pilot experiments on the performance of initial algorithms

Problem Set	K	N	Instance	Random Initialization		Processing Rate Initialization		F/R Ratio Initialization	
				# of Iteration for Convergence	Best throughput rate	# of Iteration for Convergence	Best throughput rate	# of Iteration for Convergence	Best throughput rate
1	10	100	1	172	0.0439	120	0.0424	<b>52</b>	<b>0.0441</b>
			2	37	0.0497	59	0.0497	<b>25</b>	<b>0.0497</b>
			3	1389	0.0526	25	0.0521	<b>74</b>	<b>0.0526</b>
			4	40	0.0531	55	0.0531	<b>35</b>	<b>0.0531</b>
			5	132	0.0564	101	0.0564	<b>96</b>	<b>0.0564</b>
			6	147	0.0121	21	0.0121	179	0.0121
			7	81	0.0146	40	0.0146	<b>34</b>	<b>0.0146</b>
			8	54	0.0159	41	0.0160	178	0.0160
			9	43	0.0165	34	0.0165	<b>31</b>	<b>0.0165</b>
			10	44	0.0201	31	0.0200	<b>23</b>	<b>0.0265</b>
2	10	200	1	277	0.0433	3	0.0425	<b>68</b>	<b>0.0444</b>
			2	181	0.0508	71	0.0522	<b>49</b>	<b>0.0522</b>
			3	67	0.0526	122	0.0526	623	0.0526
			4	1169	0.0542	72	0.0542	4822	0.0542
			5	939	0.0580	4077	0.0580	1680	0.0574
			6	895	0.0121	357	0.0121	<b>167</b>	<b>0.0121</b>
			7	3166	0.0148	2787	0.0148	<b>2725</b>	<b>0.0148</b>
			8	3491	0.0160	3172	0.0160	4423	0.0160
			9	677	0.0165	1459	0.0165	<b>53</b>	<b>0.0165</b>
			10	1398	0.0201	3671	0.0200	4459	0.0200
3	20	200	1	196	0.0437	89	0.0409	<b>34</b>	<b>0.0441</b>
			2	3346	0.0493	55	0.0483	4998	<b>0.0497</b>
			3	1723	0.0520	67	0.0526	133	0.0526
			4	27	0.0545	17	0.0553	24	0.0553
			5	6	0.0491	66	0.0514	<b>51</b>	<b>0.0519</b>
			6	1674	0.0121	4998	0.0121	4998	0.0121
			7	4998	0.0146	4998	0.0146	4998	0.0146
			8	4998	0.0160	2612	0.0160	4998	0.0160
			9	4998	0.0162	4497	0.0157	<b>4994</b>	<b>0.0162</b>
			10	1908	0.0164	4938	0.0165	<b>4833</b>	<b>0.0165</b>
4	20	400	1	4997	0.0434	743	0.0423	<b>444</b>	<b>0.0440</b>
			2	2673	0.0520	1034	0.0522	<b>3253</b>	<b>0.0523</b>
			3	2714	0.0526	4999	0.0526	<b>18</b>	<b>0.0526</b>
			4	1141	0.0575	4998	0.0575	<b>119</b>	<b>0.0575</b>
			5	1703	0.0493	28	0.0525	2380	0.0491
			6	4998	0.0121	3200	0.0121	4994	0.0121
			7	4995	0.0148	4999	0.0148	<b>2595</b>	<b>0.0148</b>
			8	4998	0.0160	4680	0.0160	4992	0.0160
			9	4996	0.0162	4999	0.0162	4999	0.0162
			10	4578	0.0165	4998	0.0165	4998	0.0165

As shown in Table 5.1 above, both the solution quality and the convergence rate are observed. The initialization methods are first compared with respect to solution quality, and in case the same solution quality is observed, the convergence rate is used as a tie-breaking rule. As can be seen from Table 5.1, the performance of ratio of failure to repair rate method is better in 23 problem instances out of 40. Especially

for the first problem set, its performance is quite remarkable. Hence, this method is used as an initialization scheme in the proposed adaptive tabu search approach.

#### 5.3.4 Tabu Tenure

The length of the tabu list, called the *tabu tenure* ( $TT$ ), is another important search parameter of TS. Tabu tenure is the number of iterations that tabu moves stay in the tabu list. In this study, the tabu tenure is tuned adaptively based on the quality of the current solution and the frequency of the moves. Initially, the tabu tenure is set to a predefined minimum value. Then, the value of the tabu tenure for each move is determined using the following formula (Lü and Hao, 2010):

$$TT(m_i) = T + T_i * freq(T_i) \quad (5)$$

$$TT_{\min} \leq TT(m_i) \leq TT_{\max} \quad (6)$$

The first part of the formula given by Eq. (5) represents the effect of solution quality on the tabu tenure. Namely, if the selected move does not improve the objective function (i.e., throughput rate of the production line), the value of  $TT$  is increased by 1. Likewise, if the selected move improves the objective function, the value of  $TT$  is decreased by 1. While making these changes, the tenure is not allowed to surpass the pre-specified minimum or maximum values, as indicated by Eq. (6). The basic idea behind the second part of the formula is to penalize a move which is repeated too often.

#### 5.3.5 Intensification Strategy

The key idea behind the concept of intensification is to implement some strategies so that the areas of the search space that seem promising can be explored more thoroughly. In general, intensification is based on a recency memory, in which one records the number of consecutive iterations that various solution components have been presented in the current solution without interruption.

In this study, the intensification strategy is implemented only for large-sized problems involving 20 and 40 machine lines, since the increment (decrement) values of the moves is set to greater than one for these problems. During the iterative search, if a solution found to be best so far remains to be the best one for a certain number of iterations, the increment (decrement) of moves is set to one. For example, if there are 400 total buffers to be allocated to the twenty-machine line, the increment (decrement) value of moves is set to 4 in the regular phase of the ATS. If incumbent solution remains to be same for 100 iterations, the increment (decrement) of moves is set to 1. In this way, the areas of the search space that contain promising solutions are investigated more thoroughly.

### ***5.3.6 Diversification Strategy***

Unlike intensification that is used to search regions containing good solutions more intensively, diversification guides the search to unexplored regions to avoid local optimality. Diversification is usually based on a frequency memory where the total number of iterations of the performed moves or visited solutions is recorded. There are two major diversification techniques known as restart diversification and continuous diversification. While the first is performed by several random restarts, the latter is integrated into the regular search process.

We employ both of the diversification methods explained above to diversify the search. In the continuous diversification, frequently performed moves are penalized in the regular search process. If the objective function does not change for a certain number of iterations, a random restart is applied, directing the search toward unvisited areas of the search space.

### ***5.3.7 Stopping Condition***

The algorithm is terminated if one of the following criteria is satisfied: (a) within a certain number of iterations, no better solution is found, or (b) the number of iterations reaches the maximum allowable number. The values of these two

parameters are problem-dependent. While the maximum allowable number is set to 50 times of total buffer size for each problem set, the iteration number for the first condition is set to half of the maximum allowable number.

The framework of the proposed ATS procedure is given in Figure 5.1.

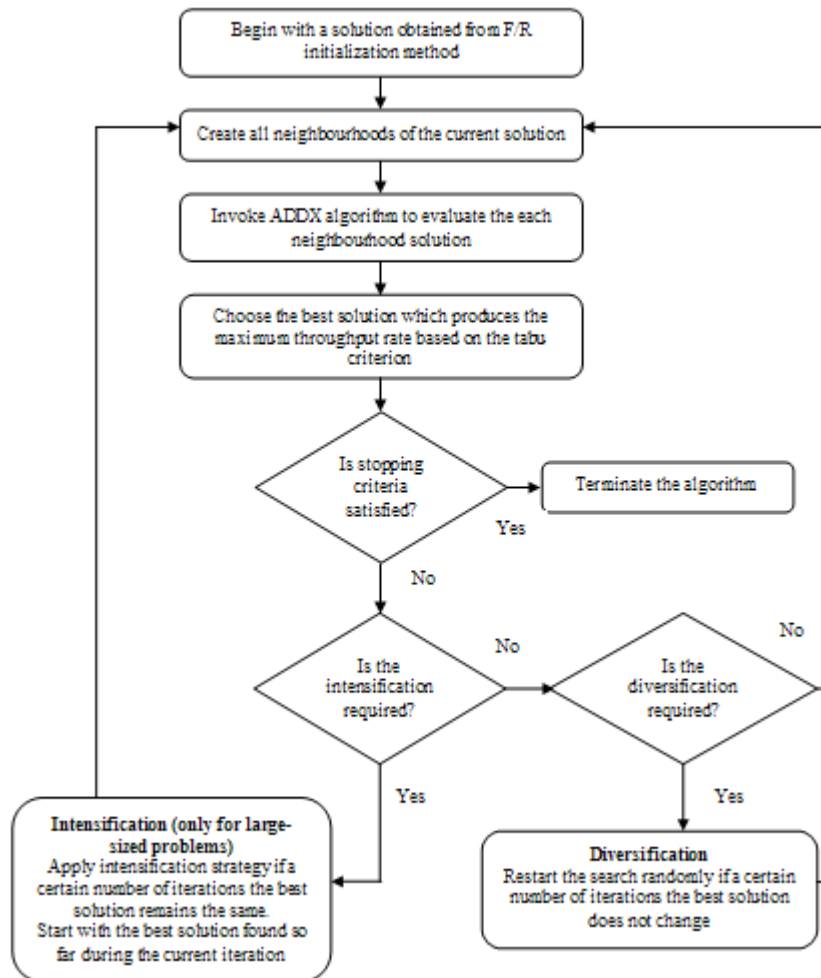


Figure 5.1 The framework of the proposed ATS procedure

The next section presents the results of our experimental studies for evaluating the performance of the proposed ATS algorithm.

## 5.4 Computational Experiments

We design a computational experiment to observe the effects of parameter changes on the performance of the developed algorithm. In our computational experiments, five, ten, twenty and forty-machine lines ( $K = 5, 10, 20, 40$ ) are considered. For each of these levels, the total buffer size is set to 5, 10 and 20 times of the number of machines in the line. Hence, a total of 12 problem sets are considered. As seen in Table 5.2, by identifying different levels for processing rates and reliability parameters, eight different settings are generated for each problem set. The processing rates of the machines are generated from a uniform distribution and the failure and repair rates are generated from a geometric distribution.

Table 5.2 Properties of problem instances

Setting	Processing Rate	Failure Rate	Repair Rate
1	(5,15)	(1,200)	(1,10)
2	(5,15)	(1,200)	(1,40)
3	(5,15)	(1,2000)	(1,100)
4	(5,15)	(1,2000)	(1,400)
5	(5,45)	(1,200)	(1,10)
6	(5,45)	(1,200)	(1,40)
7	(5,45)	(1,2000)	(1,100)
8	(5,45)	(1,2000)	(1,400)

The performance of the proposed ATS algorithm is compared to the basic TS algorithm over the generated settings. For each setting, 10 runs are made totaling to 960 runs. All algorithms are implemented in C language. The execution is done on a computer having 2.26 GHz Intel Core i5 430M CPU processor and 4 GB of RAM.

In our experimental study, the problem sets are denoted by  $K.N$ , where  $K$  is the number of machines in the line and  $N$  is the total buffer size to be allocated. It should be noted that for small-sized problems involving 5 machines and medium-sized problems involving the problem set 10.50, the results were compared to those found by complete enumeration (CE) method. However, for all other problem sets the performance of proposed ATS algorithm is compared to the basic TS (BTS)

algorithm since large problem sets can be solved to optimality within a reasonable amount of time.

The following sections are devoted to the discussion of our experimental results. We give the average results for all problem sets in the following tables. The detailed results for all problems are given in the tables in Appendix B.

#### ***5.4.1 Results of Small-Sized Problems***

Table 5.3 presents the results of our experimental studies for small-sized problems. The second column of this table presents the problem settings which are explained in Table 5.2. For each setting 10 instances are considered. The third and fourth columns represent the number of instances optimally solved for each algorithm. For both BTS and ATS, the average deviation from the optimal solution is measured using the following formulas and they are presented in fifth and sixth columns:

$$\text{Deviation for BTS:} \quad 100 \times \left( \frac{f(CE) - f(BTS)}{f(CE)} \right)$$

$$\text{Deviation for ATS:} \quad 100 \times \left( \frac{f(CE) - f(ATS)}{f(CE)} \right)$$

Finally, the last three columns show the solution times of each method in CPU seconds. For these problems, the convergence rate of the ADDX algorithm is set to  $10^{-6}$ .

As seen in Table 5.3, while the proposed ATS algorithm finds the optimum solution for all problem settings in problem set 5.25, BTS algorithm gets stuck at the local optimum for problem settings 2, 3, and 4 for this problem set. Moreover, no discernible pattern has been observed for the solution times of the algorithms. While the solution time of ATS algorithm is noted to be nearly two times of the solution time of BTS for the problem settings 1, 4, 6, and 7 (see Table 5.3), for other problem settings, no significant difference has been observed at all.

Table 5.3 Results of experimental studies for small-sized problems

Problem Set	Setting	# of Instances Optimally Solved		Avg. Deviation From Optimum (%)		CPU (sec.)		
		BTS	ATS	BTS	ATS	CE	BTS	ATS
5.25	1	10	10	0.00	0.00	0.01	0.12	0.21
	2	7	10	0.71	0.00	0.03	0.50	0.65
	3	7	10	1.42	0.00	0.03	0.42	0.57
	4	8	10	2.19	0.00	0.03	0.32	0.50
	5	10	10	0.00	0.00	0.01	0.14	0.18
	6	10	10	0.00	0.00	0.03	0.34	0.79
	7	10	10	0.00	0.00	0.02	0.14	0.30
	8	10	10	0.00	0.00	0.03	0.36	0.41
5.50	1	10	10	0.00	0.00	0.07	0.41	0.55
	2	7	10	0.53	0.00	0.19	0.53	1.34
	3	7	10	0.18	0.00	0.19	1.02	1.26
	4	4	10	3.84	0.00	0.19	0.68	1.07
	5	10	10	0.00	0.00	0.08	0.30	0.60
	6	9	10	0.15	0.00	0.16	1.00	1.63
	7	9	10	0.78	0.00	0.09	0.42	0.69
	8	8	10	0.33	0.00	0.16	0.65	1.12
5.100	1	10	10	0.00	0.00	0.48	0.81	1.50
	2	6	9	4.33	0.12	1.79	1.82	2.26
	3	7	9	0.17	0.04	1.57	5.01	6.64
	4	6	7	1.36	1.16	1.38	1.66	2.77
	5	8	10	5.56	0.00	0.73	0.77	0.99
	6	10	10	0.00	0.00	0.94	0.99	2.36
	7	9	10	0.40	0.00	0.56	0.73	1.84
	8	8	10	0.89	0.00	1.06	1.25	2.62

For the problem set 5.50, the proposed ATS algorithm finds the optimum solution for all problem settings while BTS algorithm reaches the optimum solution for only problem settings 1 and 5 which involve machines with short repair times. The superiority of ATS algorithm over BTS algorithm becomes much more apparent when the repair times of the machines get much longer (see problem settings 2, 4 and 6 in Table 5.3). The results with respect to solution times of the two algorithms are similar to the ones obtained for problem set 5.25. As noted earlier, the solution time heavily depends on the convergence time required by the ADDX algorithm. Changing the parameters, i.e. machine processing times, failure and repair rates in each problem setting either increases or decreases the complexity of the problem which affects the convergence time of ADDX algorithm and hence solution times of algorithms change across the problem types. As expected, the solution time of both



algorithms increase as problem size increases. Moreover, on average, the solution time of ATS algorithm is higher than the solution time of BTS algorithm for all problem settings.

For the problem set 5.100, the proposed ATS algorithm achieves the optimum solution in 5 out of 8 settings, and for remaining three problem settings (2, 3, and 4) small deviations from the optimum values are observed. As seen in Table 5.3, BTS algorithm can reach the optimal solution only for two problems settings (1 and 6). Moreover, the superiority of proposed ATS over BTS becomes more apparent as variability in machine processing times increases (see the results for the last four problem settings in Table 5.3). Unlike apparent success of ATS in dealing with variability in processing times, for BTS, the highest deviation from the optimum solution has been observed for the problem setting 5 involving more variability in processing times and frequently failed machines with short repair times. As for the solution time, similar trends have been observed like in problem sets 5.25 and 5.50.

In summary, for small-sized problem sets involving five machines the proposed ATS algorithm reaches the optimum solution in 235 out of 240 in very small computation times. Hence it can be said that the proposed ATS algorithm is very efficient to solve buffer allocation problem for small-sized instances.

#### ***5.4.2 Results of Medium-Sized Problems***

Table 5.4 presents the results of our experimental studies for medium-sized problems. As it is stated before, for the problem set 10.50 the results obtained from both TS algorithms are compared to CE results. For all other problem sets including 20 and 40 machine lines, since complete enumeration cannot provide optimal results within a reasonable amount of time, the comparison of BTS and ATS algorithms is done with respect to solution quality as follows:

$$\text{Improvement over BTS} = 100 \times \left( \frac{f(ATS) - f(BTS)}{f(BTS)} \right)$$

For the problem set 10.50, BTS algorithm cannot find the optimal solution for any problem type, while the proposed ATS algorithm reaches the optimal solution for 4 out of 8 problem settings (1, 5, 6 and 7). It should be noted that the proposed ATS finds the optimal solution for 63 out of 80 problems (see Appendix B4). As seen in Table 5.4, when the variability of processing time increases the efficiency of the proposed ATS also increases except for the problem setting 8 where the machines fail less frequently. Moreover, it has been observed that the proposed ATS algorithm is less efficient for the problem setting 3 with the average deviation from optimum, 2.08%. It should be noted that, this problem setting involves rarely failed machines having long repair times. Overall, it can be said that the proposed ATS algorithm is more successful for the problem settings involving more variability in processing times.

It has been observed that for ATS the average deviations from the optimal are very small for this problem set. It should be noted that the convergence rate of the ADDX algorithm is set to  $10^{-5}$  for the cases involving 10 machines and the comparison is based on 6 decimal digits, hence the difference between the optimal and the best solution found by the proposed ATS algorithm is not large for this problem set. So it can be concluded that the proposed ATS algorithm obtains very good solutions for this problem set. Moreover, the CPU times for both algorithms are comparable, and all CPU times are under one minute as seen in Table 5.4.

For the problem set 10.100, the proposed ATS algorithm finds better solutions than BTS algorithm in 31 out of 80 problems (see Appendix B5). Unlike the previous experiments, in these experiments the proposed ATS algorithm is found to be more efficient for problem settings involving less variability in processing times. Moreover, it has been observed that the superiority of ATS is more observable for the problem setting 4 with the average improvement over BTS, 6.45%. It should be noted that this problem set involves rarely failed machines having long repair times and also less variability processing times.

Table 5.4 Results of experimental studies for medium-sized problems

Problem Set	Setting	# of Instances Optimally Solved		Avg. Deviation From Optimum (%)		CPU (sec.)		
		BTS	ATS	BTS	ATS	CE	BTS	ATS
10.50	1	8	10	0.46	0.00	10118.53	16.66	20.25
	2	3	7	2.45	0.10	13656.45	53.11	33.06
	3	4	6	2.84	2.08	15230.31	38.25	42.30
	4	1	5	9.47	0.55	9142.00	16.68	18.05
	5	7	10	0.76	0.00	8804.06	14.59	13.23
	6	5	10	1.09	0.00	12268.71	21.12	29.26
	7	6	10	0.21	0.00	10201.89	16.97	25.59
	8	1	5	6.18	0.68	13841.16	31.31	28.60
Problem Set	Setting	# of Instances Obtained Better Solutions than BTS	Avg. Improvement Over BTS (%)	CPU (sec.)				
				BTS	ATS			
10.100	1	2	0.81	31.05	30.52			
	2	4	1.18	47.37	54.29			
	3	4	1.35	41.84	62.62			
	4	8	6.45	71.35	45.91			
	5	2	0.30	29.19	41.14			
	6	2	0.63	37.62	50.64			
	7	2	0.28	25.63	40.52			
	8	7	3.87	45.03	64.27			
10.200	1	2	0.57	86.53	115.6			
	2	6	1.17	66.69	136.2			
	3	4	1.57	116.21	124.4			
	4	5	3.87	70.42	132.4			
	5	1	0.81	42.45	68.94			
	6	3	4.92	85.06	82.08			
	7	3	2.69	47.76	80.93			
	8	5	5.11	64.23	133.9			

As stated before, the solution time of the algorithms heavily depends on the convergence of the ADDX algorithm and hence, the solution times of BTS and ATS change across the problem types. As expected, the solution times of both algorithms increase as problem size increases.

As in the case of problem set 10.100 the solution quality of the proposed ATS algorithm is better than BTS algorithm for the problem set 10.200. In comparison to BTS algorithm, the proposed ATS algorithm finds better solutions in 29 out of 80 problems (see Appendix B6). Like in the previous problem sets involving ten

machines, the superiority of the proposed ATS algorithm is more observable for problem settings 4 and 8 involving less frequently failed machines with long repair times. However, the effect of processing time variability on solution quality is not significant unlike the previous problem sets involving ten machine lines. Moreover, the solution times of both algorithms show similar trend like in the case of problem set 10.100.

In summary, for medium-sized problems involving ten machines, the solution quality of proposed ATS algorithm is better than BTS algorithm. This is achieved at the expense of a small increase in computation time for some settings. Based on our experimental results, it can be concluded that the proposed ATS algorithm is more efficient to solve medium-sized problems for the lines having rarely failed machines with long repair times.

#### ***5.4.3 Results for Large-Sized Problems***

In this section we discuss the experimental results for large sized problems involving 20 and 40 machines. It should be noted that the convergence rate of the ADDX algorithm is set to  $10^{-4}$  for these problems.

The experimental results for large-sized problems involving 20 machines are given in Table 5.5. For the problem set 20.100, the proposed ATS algorithm outperforms BTS algorithm with respect to solution quality. The superiority of the proposed ATS is more observable as compared to the medium-sized problems involving ten machine lines. For this problem set, the proposed ATS find better solutions in 51 out of 80 problems (see Appendix B7). The superiority of proposed ATS algorithm is more observable for the problem settings 2, 4 and 8. While the problem settings 4 and 8 involve rarely failed machines with long repair times, the problem setting 2 involve frequently failed machines with long repair times. Moreover, the proposed ATS algorithm is more efficient for solving the problem settings having less variability in processing times. Based on these results, it can be said that the proposed ATS algorithm is more efficient for the problem settings

having less variable processing times and long repair times. As for the solution time, the proposed ATS algorithm requires slightly longer CPU times than BTS, except for problem setting 7.

Table 5.5 Results of experimental studies for large-sized problems: 20 machines

Problem Set	Setting	# of Instances Obtained Better Solutions than BTS	Avg. Improvement Over BTS (%)	CPU (sec.)	
				BTS	ATS
20.100	1	3	7.82	472.59	610.15
	2	7	16.82	535.80	582.61
	3	9	3.66	743.63	881.92
	4	10	31.27	631.30	659.17
	5	6	6.72	525.12	457.88
	6	5	3.93	326.89	436.40
	7	4	0.41	1305.61	773.77
	8	7	16.55	461.63	664.97
20.200	1	3	0.58	721.13	838.76
	2	5	16.76	744.34	861.97
	3	7	3.03	1495.78	1512.25
	4	9	15.22	946.05	1448.12
	5	4	1.54	900.11	741.51
	6	6	10.97	518.78	848.52
	7	2	1.19	922.39	1501.18
	8	8	20.95	780.61	1175.98
20.400	1	4	0.62	1646.59	2221.27
	2	4	12.29	1156.94	2323.79
	3	5	2.23	2431.42	4060.22
	4	8	5.76	1066.12	2885.77
	5	3	2.75	1072.34	1535.51
	6	4	9.29	1066.70	2076.95
	7	4	1.70	2634.22	3099.10
	8	3	6.93	1265.69	1988.09

The results obtained for the problem set 20.200 are similar to the ones obtained for problem set 20.100. As it can be seen from Table 5.5, as problem size increases the superiority of ATS algorithm over BTS algorithm becomes more apparent for all problems having long repair times (see problem settings 2, 4, 6, and 8 in Table 5.5). So, overall ATS algorithm is more efficient for problem settings having long repair times. However, unlike the previous problem set where the solution quality of the proposed ATS algorithm is affected by the variability in machine processing times,

in this problem set, no such behavior has been observed. As for the solution time, the same trend is observed as in the case of problem set 20.100.

For the problem set 20.400, the proposed ATS algorithm finds better results in 35 out of 80 problems (see Appendix B9). The superiority of the proposed ATS over BTS is more observable for the problem settings 2, 4, 6 and 8 like in the previous problem set (see Table 5.5.) As in the case of the problem set 20.200, the solution quality of proposed ATS is much better for the problems involving machines with long repair times. However, this is achieved at the expense of increasing the computation time. For this problem set, the solution time of ATS algorithm is nearly two times of BTS algorithm. Similar to the earlier problem, in this problem set it is also noted that the solution quality of the proposed ATS is not affected by the variability in processing times.

In summary, the proposed ATS algorithm is very efficient to solve the buffer allocation problem for large-sized problems involving 20 machines (130 out of 240 problems). When the problem size increases, especially for the lines having long repair times, the superiority of proposed ATS becomes more apparent. Moreover, as problem size increases the solution time of the proposed ATS algorithm increases more than the solution time of BTS algorithm.

Table 5.6 shows the results of experimental study for large-sized problems involving 40 machines. As in the case of 20-machine lines, the solution quality of the proposed ATS algorithm is better than BTS for the problem set 40.200. The proposed ATS finds better solutions in 55 out of 80 problems for this problem set (see Appendix B10). Similar to the previous problem set involving 20 machines, the superiority of ATS algorithm over BTS algorithm is observed in the same problem settings, i.e., 2, 4, 6 and 8 (see Table 5.6). Besides these four settings, the proposed ATS is found to be more successful than BTS in dealing with the problems involving more frequently failing machines with short repair times (see the results for the first problem setting in Table 5.6). Overall, it can be said that when the problem size increases the efficiency of the proposed ATS also increases. As for

solution time, the proposed ATS algorithm takes much longer time than BTS algorithm as it is seen in Table 5.6.

Table 5.6 Results of experimental studies for large-sized problems: 40 machines

Problem Set	Setting	# of Instances Obtained Better Solutions than BTS	Avg. Improvement Over BTS (%)	CPU (sec.)	
				BTS	ATS
40.200	1	5	5.03	3055.47	14182.22
	2	7	6.35	5926.02	13734.16
	3	9	2.73	6846.08	19377.56
	4	9	11.68	6249.59	15585.69
	5	2	2.50	2806.04	11062.05
	6	5	5.08	2949.22	11130.49
	7	8	2.03	5511.91	12972.34
	8	10	15.04	7501.24	20265.23
40.400	1	5	2.98	6493.01	29218.54
	2	5	2.02	8994.71	30270.03
	3	9	0.77	10989.18	35234.01
	4	6	6.22	11215.27	32214.51
	5	2	0.76	14228.44	31228.77
	6	6	5.35	9449.98	24737.21
	7	6	1.42	9197.03	34408.30
	8	10	15.34	9419.63	30980.34
40.800	1	4	3.98	11433.08	38311.94
	2	6	5.38	17578.23	44480.83
	3	8	1.52	18410.88	40041.72
	4	10	6.30	22582.94	52410.80
	5	3	1.06	16594.29	41678.42
	6	7	9.62	20066.05	39523.48
	7	3	1.31	20189.94	42415.71
	8	8	18.03	18777.31	46475.92

For the problem set 40.400, the proposed ATS algorithm finds better solution than BTS algorithm in 49 out of 80 problems (see Appendix B11). The superiority for ATS algorithm over BTS algorithm is more apparent especially for the problem settings 4, 6 and 8 involving machines with long repair times. However, the improvement of ATS algorithm over BTS algorithm is not too much in comparison to the improvement in problem set 40.200. Also, it has been observed that when the variability of processing times increases, the efficiency of the proposed ATS algorithm also increases (see problem settings 2, 4, 6 and 8 in Table 5.6). As in the

case of problem set 40.200, the solution time of the proposed ATS increases when the problem size increases.

The proposed ATS algorithm finds better solutions in 49 out of 80 problems (see Appendix B12) for the problem set 40.800. Similar to the problem sets 40.200 and 40.400, the efficiency of the proposed ATS algorithm is apparent for the problem settings 2, 4, 6 and 8 involving machines with long repair times. As in the previous problem set, it has been observed that when the variability of processing times increases the efficiency of the proposed ATS algorithm also increases. As for the solution time, the proposed ATS algorithm takes much longer time than BTS algorithm.

In summary, the proposed ATS algorithm is quite efficient to solve buffer allocation problem for long production lines involving 40 machines. The proposed ATS algorithm finds better solutions in 153 out of 240 problems. The efficiency of ATS algorithm is much better especially for the problem settings involving machines with long repair times and having more variability in machine processing times. However, the solution time of proposed ATS greatly increases when the problem size increases.

#### ***5.4.4 Summary of the Findings***

In this section, the performance of the proposed ATS algorithm is compared to that of BTS algorithm over wide range of problems with varying difficulty. The overall results are summarized in Table 5.7. It can be concluded that the proposed ATS algorithm works very well to obtain good quality solutions across all problem types studied. Moreover, it has been observed that the solution quality of ATS algorithm is much better especially for the problem settings involving machines with long repair times. Also, the solution quality of ATS algorithm gets much better while the problem size increases where the average improvement over BTS becomes 6.91% (see Table 5.7). Embedding intensification and diversification strategies into the BTS algorithm and also tuning the tabu tenure adaptively clearly improves the



solution quality of the proposed ATS algorithm. However, as expected, the execution time of the proposed ATS algorithm gets longer for large-sized problems since the convergence of the ADDX algorithm and evaluation of the complete neighborhood of a solution during the search requires longer time (see Figure 5.2). Nevertheless, it should be noted that the ATS approach proposed in this study is not designed to solve buffer allocation problem in “real-time”. In fact, since the buffer allocation problem is a manufacturing design problem, the emphasis should be placed on finding good quality solutions in reasonable times rather than obtaining quick solutions.

Table 5.7 Summary of experimental studies

Problem Size	Problem Set	Avg. Improvement Over BTS (%) <sup>*</sup>	CPU Time (sec.)	
			BTS	ATS
Small	5.25	0.54	0.29	0.45
	5.50	0.73	0.63	1.03
	5.100	1.81	1.63	2.62
	<b>Average</b>	<b>1.02</b>	<b>0.85</b>	<b>1.37</b>
Medium	10.50	2.96	26.09	26.29
	10.100	1.86	41.14	48.74
	10.200	2.59	72.42	109.34
	<b>Average</b>	<b>2.47</b>	<b>46.55</b>	<b>61.46</b>
Large	20.100	10.90	625.32	633.36
	20.200	8.78	878.65	1116.04
	20.400	5.20	1542.50	2523.84
	40.200	6.31	5105.69	14788.72
	40.400	4.36	9454.77	29231.16
	40.800	5.90	17231.94	41618.89
	<b>Average</b>	<b>6.91</b>	<b>5806.48</b>	<b>14985.33</b>

<sup>\*</sup> Avg. Improvement Over BTS (%) =  $100 * [f(\text{ATS}) - f(\text{BTS})] / f(\text{BTS})$

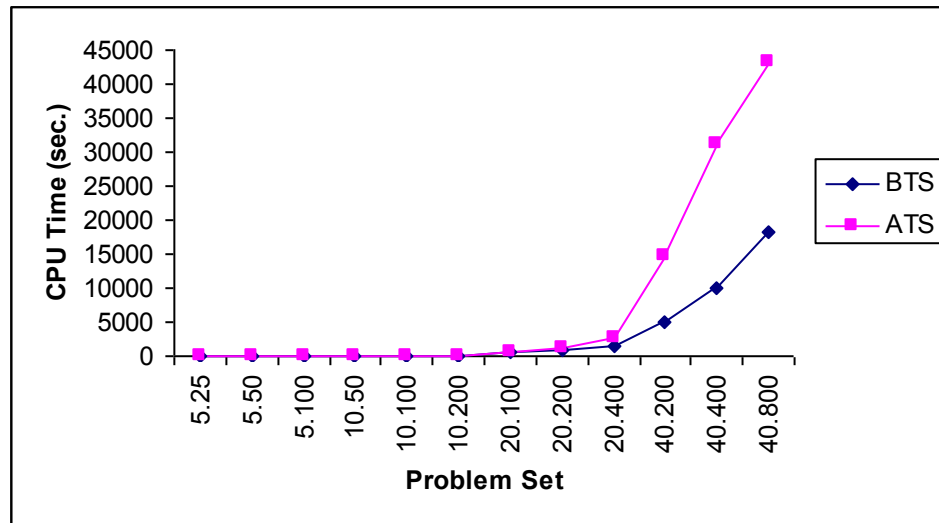


Figure 5.2 Solution times of both algorithms with respect to the problem sets

As a result, it can be said that the proposed ATS algorithm is quite effective in solving the buffer allocation problem in unreliable non-homogeneous production lines for all problem instances studied.

## 5.5 Chapter Summary

In this chapter, we proposed an adaptive tabu search algorithm to obtain the optimal buffer allocations for maximizing the throughput rate of the line in unreliable non-homogeneous production lines. To our knowledge, this is the first extensive study to deal with buffer allocation problem in unreliable and also non-homogeneous production lines. Imposing buffer space constraints for each buffer location makes the problem harder. In the proposed solution approach, intensification and diversification strategies in tabu search are utilized to solve the buffer allocation problem in both short and long lines. Moreover, an experimental study is carried out to select an initial solution generation scheme.

The performance of the proposed ATS algorithm is tested on randomly generated test problems and its performance is compared to the basic TS algorithm. The results of experimental study show that the proposed ATS algorithm is quite efficient to reach optimal/near optimal solutions in very small computation times. The superiority of the proposed algorithm is observed especially for large-sized

problems, i.e. 20 and 40 machine lines, since the solution quality is much better than BTS algorithm.

We consider the problem from a throughput maximization perspective throughout this chapter. Since our ultimate objective is to propose a solution approach for total buffer size minimization problem, in the next chapter, this study will be extended to minimize the total buffer size by using proposed ATS algorithm.

## **CHAPTER SIX**

### **AN INTEGRATED APPROACH FOR THROUGHPUT MAXIMIZATION WITH MINIMUM TOTAL BUFFER SIZE**

#### **6.1 Introduction**

In this chapter, an integrated approach is proposed to solve the buffer allocation to maximize the throughput rate of the line with minimum total buffer size. The proposed integrated approach has two control loops, i.e., the inner loop and outer loop. While the inner loop control includes the adaptive tabu search algorithm proposed in the previous chapter, the outer loop control includes three different algorithms, i.e., binary search, tabu search and simulated annealing. These nested loops aim at minimizing the total buffer size to achieve a desired throughput level. To improve searching efficiency of the proposed tabu search and simulated annealing algorithms alternative neighborhood generation mechanisms are suggested and their performance are tested.

This chapter is organized as follows. Next section presents the specifications of the problem. In section 6.3, the proposed solution approaches are described in detail. The results of computational experiments to test the performance of the proposed algorithms are discussed in Section 6.4. Finally, Section 6.5 summarizes the context of this chapter.

#### **6.2 Problem Specifications**

Like in the previous chapter, we address the buffer allocation problem in an unreliable serial production line with deterministic processing times. The features of this line can be listed as follows:

- Each part goes through all machines in exactly the same order.
- There is an intermediate location for storage (buffer) between each pair of machines.

- Machines in the line have unique deterministic processing times.
- Machines are subject to breakdown, and the repair and failure rates are geometrically distributed.
- The first machine is never starved, i.e. input is always available, and the last machine is never blocked, i.e. there is always space to dispose of the output.

Assuming that there are  $K$  machines and  $K-1$  buffers in a production line, our main objective is to minimize the total buffer size so as to achieve desired throughput rate. This problem can be formulated as follows:

Find  $B = (B_1, B_2, \dots, B_{K-1})$  so as to

$$\min N = \sum_{i=1}^{K-1} B_i \quad (1)$$

subject to

$$f(B^N) \geq f^* \quad (2)$$

$$0 \leq B_i \leq u_i \quad (i = 1, 2, \dots, K-1) \quad (3)$$

$$B_i \text{ nonnegative integers } (i = 1, 2, \dots, K-1) \quad (4)$$

where  $K$  is the number of machines in the line,  $B$  is a buffer vector,  $N$  is the total buffer size,  $u_i$  is the upper bound for each location,  $f(B^N)$  is the throughput rate of the production line obtained by total buffer size  $N$  and  $f^*$  is the desired throughput rate. It should be noted that upper ( $u_i$ ) bounds for each buffer location are chosen such that their summation will be larger than the total buffer size ( $N$ ) in the system.

To solve this buffer allocation problem, an integrated approach involving two control loops is proposed. While the inner loop control includes an adaptive tabu search algorithm to obtain the maximum throughput rate of the line for a given total buffer size, the outer loop control includes three different algorithms, i.e., binary search, tabu search and simulated annealing to minimize the total buffer size in the system so that the desired throughput rate can be achieved.

The details of this integrated approach are presented in next section.

### 6.3 Proposed Integrated Approach

As stated before, this integrated approach involves an inner and an outer loop algorithm. The inner loop algorithm is the proposed adaptive tabu search algorithm presented in Chapter 5. The outer loop algorithm is implemented by either using binary search, or tabu search or simulated annealing algorithms.

Figure 6.1 shows the execution mechanism of the proposed integrated approach. As it is seen from Figure 6.1, each outer algorithm is started with a pre-specified  $N$  value, the maximum throughput rate which can be obtained with this  $N$  value is calculated and it is compared to the desired throughput rate. In an iterative way, the outer loop algorithm is run again to obtain new  $N$  values, and this procedure continues until the desired throughput rate is achieved with the minimum total buffer size.

The following sections present the details of the proposed outer loop algorithms.

#### 6.3.1 Binary Search Algorithm

The first algorithm is a binary search algorithm. Table 6.1 shows the steps of the algorithm. This algorithm starts with a number  $M$  which is big enough and continues to search between  $H$  and  $L$  until the desired throughput rate is obtained. Hereafter, when  $H$  is equal to  $L$ , the algorithm continues to search between 0 and  $N^{min}$  which is the best total buffer size value found so far. In this way, the algorithm explores the whole search space.

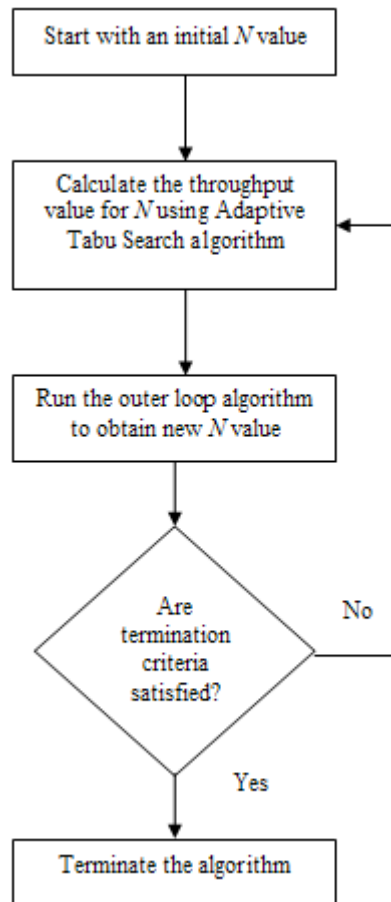


Figure 6.1 The framework of algorithms for total buffer size minimization

Table 6.1 Binary search algorithm

<p><i>Initialization</i></p> <p>Set <math>H=M</math> and <math>L=0</math></p> <p><i>Step 1.</i> Set <math>N = (H + L) / 2</math></p> <p><i>Step 2.</i> Run ATS algorithm for <math>N</math>.</p> <p><i>Step 3.</i> If <math>f(B^N) &lt; f^*</math>, set <math>L = N</math> and <math>N = (H + L) / 2</math>          If <math>f(B^N) \geq f^*</math>, set <math>H = N</math> and <math>N = (H + L) / 2</math>,          If <math>H=L</math>, set <math>H=N^{min}</math>, <math>L=0</math>, <math>N = (H + L) / 2</math>.</p> <p><i>Step 4.</i> until <math>N=1</math> go to Step 2, otherwise terminate the algorithm.</p>
--

However, exploring search space in this way needs more computational effort. Therefore, to improve search efficiency so that computational cost can be reduced it is proposed to employ two meta-heuristic methods as outer loop control algorithms,

i.e., tabu search and simulated annealing. The details about these approaches are given in the following sections.

### ***6.3.2 Tabu Search Algorithm***

The tabu search algorithm employs memory-based strategies to reach the global optimum. The following sections explain the specifics of the proposed tabu search algorithm for solving buffer allocation problem under the objective of total buffer size minimization.

#### *6.3.2.1 Search Space*

Identifying a search space along with a neighborhood structure is the most critical step of any TS implementation. The search space of the TS is simply the space of all feasible solutions that can be visited during the search. In this study, all feasible solutions are considered as the search space.

#### *6.3.2.2 Move Representation and Tabu Moves*

In the proposed TS algorithm, the moves are defined depending on the values of the total buffer size,  $N$ . Throughout the algorithm the value of total buffer size is decreased by some specified value determined by the neighborhood generation mechanism which is explained in next section. At each iteration, the neighborhood of the current solution becomes tabu for a certain number of iterations. For instance if the current solution is 100 and the neighbors of this solutions are 90, 80, 70 and 60, then all of those values become tabu and they are not evaluated for the next  $TT$  (tabu tenure) number of iterations.

#### *6.3.2.3 Neighborhood Generation Mechanism*

Since the aim of this study is to minimize the total buffer size it is important to decide how the current value of  $N$  will be decremented during the search so that the



search space can be explored effectively to find the global minima. To deal with this issue, a pilot experiment is carried out and the decrement value is determined based on the results of this study. The details of neighborhood generation mechanism and the results of these pilot experiments are discussed in section 6.4.1.

#### *6.3.2.4 Neighborhood Size and Tabu Tenure*

The neighborhood size, i.e. the number of solutions evaluated at each iteration is set to 4 and it is kept constant during the search. Likewise, the value of tabu tenure, i.e. the number of iterations that tabus stay in the tabu list is kept constant depending on problem size during the search.

#### *6.3.2.5 Aspiration Criterion*

At each iteration, the minimum  $N$  value which can be reached by a non-tabu move in the current solution is selected for the next step. The only exception is defined by the aspiration criterion which allows a tabu move to be made if and only if the resulting total buffer size is better than the best total buffer size found so far.

#### *6.3.2.6 Stopping Condition*

The algorithm is terminated if one of the following criteria is satisfied: (a) within a certain number of iterations, no better solution has been found, or (b) the number of iterations reaches the maximum allowable number. The values of these two parameters are also problem-dependent. Depending on the size of the problem, these parameters are set to small values with small size of  $N$  and they are kept big enough for large size of  $N$ .

The proposed TS algorithm is outlined in Table 6.2. The notation used in this table is explained as follows:

$B^{best}$                       the best buffer configuration during the whole TS algorithm

$B^{N^k}$	the best buffer configuration obtained by the total buffer size $N$ at $k$ th iteration
$f(N^k)$	the best throughput rate value which is equal to or greater than the desired value obtained by total buffer size $N$ at $k$ th iteration.
$f^{best}$	the best throughput rate value which is equal to or greater than the desired value during the whole TS algorithm
$TT$	tabu tenure

Table 6.2 Tabu search algorithm

**Initialization**

Set  $N$  to its pre-specified value,  $N^0 = N$ . Initially tabu list is empty and iteration count is set to  $k = 0$ . Run the ATS algorithm to obtain the best throughput value for  $N$ .

Set  $f^{best} = f(N^0)$ , and  $B^{best} = B^{N^0}$ . Put  $N^0$  into *tabu list* for  $TT$  number of iterations.

**Step 1**

Create all neighbors of the current solution according to the neighborhood generation mechanism and run the ATS algorithm for each neighbor to obtain the throughput rate values.

**Step 2**

If all neighbors produce throughput values equal or greater than the desired value:

1. Select the minimum  $N$  that is non-tabu and less than the current best value.
2. Select the minimum tabu move that has a lower value than the current best value if all moves are tabu, and update the neighborhood generation mechanism (*Aspiration criterion*).
3. Set  $f^{best} = f(N^k)$  and  $B^{best} = B^{N^k}$  and update the tabu list.

Else

1. Set  $N^{k+1} = N^k$  and update the neighborhood generation mechanism.
2. Update the tabu list.

**Step 3**

Set  $k = k + 1$ , and go to Step 1 until one of the termination criteria is satisfied.

**Termination criteria**

The algorithm is terminated if one of the following criteria is satisfied:

1. within a certain number of iterations, no better solution is found, or
2. the number of iterations reaches the maximum allowable number.

As seen from Table 6.2, the algorithm is started with a pre-specified  $N$  value and the neighborhoods of this value are generated by using neighborhood generation mechanism which is explained in section 6.3.2.3. The ATS algorithm is run for each neighborhood solution to obtain the throughput value. It should be noted that if one of these neighborhood solutions is in tabu list, the ATS algorithm is run only for non-tabu values of  $N$ . Using these memory properties of tabu search reduces the running time of the algorithm. At each iteration, the minimum  $N$  value which can be reached by a non-tabu move in the current solution is selected for the next step. The only exception is defined by the aspiration criterion which allows a tabu move to be made if and only if the resulting  $N$  is lower than the minimum total size value found so far. This procedure continues until one of the stopping conditions is satisfied.

### ***6.3.3 Simulated Annealing Algorithm***

Simulated annealing is a meta-heuristic method derived from statistical mechanics. Kirkpatrick et al. (1983) proposed an algorithm which is based on the analogy between the annealing of solids and the problem of solving combinatorial optimization problems.

The analogy between the buffer allocation problem and the annealing process can be stated as follows: The states of the solid represent the feasible solutions of the problem, the energies of the states correspond to the values of the throughput rates computed at those solutions, the minimum energy state corresponds to the optimal solution to the problem and rapid quenching can be viewed as local optimization.

The proposed simulated annealing algorithm is outlined in Table 6.3. The notation used in this table is the same as in Table 6.2. The proposed algorithm consists of a sequence of iterations. Each iteration consists of changing the current solution to create a new solution in the neighborhood of the current solution. The neighborhoods of the current solution are generated by using neighborhood generation mechanism as defined in section 6.3.2.3 (Step 3). To select the best neighborhood generation mechanism a pilot experiment is carried out as in the case of proposed TS algorithm

and the results of these experiments are discussed in section 6.4.1. Once a new solution is created the corresponding change in the throughput rate is computed as given in the step 4 of the algorithm to decide whether the newly produced solution can be accepted as the current solution.

Table 6.3 Simulated annealing algorithm

---

**Step 0 (Initialization)**  
Set  $N$  to its pre-specified value. Run the ATS algorithm to obtain the best throughput value for  $N$ . Set  $f^{best} = f(N)$ , and  $B^{best} = B^0$ .

---

**Step 1 (Initialization of temperature)**  
Set the initial temperature  $T = -10$ .

---

**Step 2 (Initialization of step and success count)**  
Set the iteration count to  $k = 0$  and success count to  $U = 0$

---

**Step 3 (Neighborhood generation)**  
Create all neighbors of the current solution according to the neighborhood generation mechanism and run the ATS algorithm for each neighbor to obtain the throughput rate values.

---

**Step 4 (Calculation of energy differential)**  
Set  $\Delta E = f^* - f(N^k)$ .

---

**Step 5 (Acceptance criteria)**

- If  $\Delta E \leq 0$ , select the minimum  $N$  value that satisfies this condition for the next iteration. Set  $f^{best} = f(N^k)$ ,  $B^{best} = B^{N^k}$ , and  $U = U + 1$ .
- If  $\Delta E > 0$  and (buffer size reduction) $>1$ , keep the current best  $N$  value for the next iteration, and update the neighborhood generation mechanism.
- If  $\Delta E > 0$  and (buffer size reduction) $=1$ , select the  $N$  value that satisfies the following condition:  $e^{(-\Delta E/T)} > \text{rand}(0...1)$ .

---

**Step 6 (Repeat for current temperature)**  
Set  $k = k + 1$ , if  $k < (\text{maximum number of steps})$  go to Step 3, otherwise go to Step 5.

---

**Step 7 (Lower the annealing temperature)**  
Set  $T = cT$  ( $0 < c < 1$ ).

---

**Step 8 (Termination criterion)**  
If  $U > 0$  go to step 2, otherwise terminate the algorithm.

---

As given in step 5 of the proposed algorithm (see Table 6.3), if the desired throughput rate is obtained by less total buffer size then this value is directly taken as

the current solution. Otherwise it is accepted according to Metropolis's criterion (Metropolis et al., 1953).

According to Metropolis's criterion, if the difference between the throughput rate values of the current and newly produced solutions is greater than zero, a random number in  $(0,1)$  is generated from a uniform distribution and if  $e^{(-\Delta E/T)} > \text{rand}(0...1)$  then the newly produced solution is accepted as the current solution. Since the addressed problem is a minimization problem and our algorithm is designed in the form of decreasing  $N$  value we consider the Metropolis's criterion if and only if the value of buffer size reduction is 1.

In designing cooling schedule for the proposed SA algorithm, the initial temperature is set to -10. Since the problem addressed in this study is a minimization problem, the cooling schedule becomes like a heating schedule. In the proposed algorithm we employ the geometric cooling rule which is a common and very simple cooling strategy. This rule updates the temperature by the formula given in the seventh step of the algorithm. In this formula  $c$  is a temperature factor which is a constant smaller than 1 but close to 1. In our computational study, we tried different values for  $c$  and decided to use  $c = 0.98$  throughout the experiments.

The experimental studies carried out to test the performance of the proposed integrated approach are given in the next section.

#### **6.4 Computational Experiments**

In this section, randomly generated test problems are used to test the performance of the proposed algorithms. Before giving the details of our computational study, first the results of pilot experiments which are carried out to determine the neighborhood generation mechanism of the proposed TS and SA algorithms are presented.

### 6.4.1 Determination of Neighborhood Generation Mechanism

In this study, the neighborhood generation mechanism is defined as follows:

- At the initialization step of both TS and SA algorithms, the total buffer size,  $N$ , is set to a pre-specified value and the neighbors of this solution are generated by setting the value of buffer reduction to  $N/x$ . Here  $x$  is a divisor of  $N$ . During the pilot experiments, different values of  $x$ , i.e., 2, 3, 4 and 5 are used to generate the neighborhoods of the current solution. For instance, if the total buffer size, the neighborhood size and  $x$  are initially set to 200, 3 and 4, respectively, then the neighborhoods of this solution become 150, 100, and 50.
- In the following steps, the value of buffer reduction is set to  $\frac{1}{2}(N^{min} / x)$ . Here  $N^{min}$  is the current best solution found so far. Continuing with the previous example, if a total buffer size of 100 achieves the best throughput rate during the second step of the algorithm, then the neighborhoods of this solution become 87, 74, and 61 by setting the value of buffer reduction to 13, i.e.,  $100/8=12,5\sim 13$  (since buffer sizes must be integer,  $N^{min}/8$  is rounded to integer value). This process continues until the value of buffer reduction becomes 1, i.e.,  $N^{min}/16$  during the third step,  $N^{min} /32$  during the fourth step, etc. In this way, it is focused on promising solution areas of the search space, i.e., intensification.

As mentioned above, a set of pilot experiments is carried out to determine the value of divisor,  $x$ . The values of  $x$  tested are 2, 3, 4 and 5. These experiments involve a ten-machine line with initial total buffer size of 100. By changing the failure/repair rates of machines (see Table 5.2 in chapter 5) eight sets of problems are generated. To determine the desired throughput rates using ATS algorithm, all considered problems are solved by setting the total buffer size value to 50 which is the half of the initial total buffer size value. In doing so, we aim at searching for whether the proposed algorithms can achieve the desired throughput rate value with

the pre-specified total buffer size. Using a 2 GHz Intel ® Core™ 2 Duo CPU processor 5 runs are carried out at each design point leading to 320 runs in total.

Tables 6.4 and 6.5 present the results obtained by the proposed TS and SA algorithms, respectively. In these tables, the first two columns show the problem setting and the problem instances for each setting. While the third column presents the desired throughput rate, the fourth column presents the minimum total buffer size found by the binary search algorithm to achieve this desired throughput rate. The following columns show the performance of the proposed algorithm for each  $x$  value.

Table 6.4 shows the results obtained by the proposed TS algorithm for each  $x$  value. As seen in Table 6.4, for the problem settings 1, 3, 5 and 7, all  $x$  values produce the same results. For the third problem instances in problem setting 2 and 8, and for the first problem instance in problem setting 4, none of  $x$  values reaches the optimum solution. Considering the results of pilot experiments, setting the divisor,  $x$  to 3 seems to be a good decision since it can reach the optimum solution in 37 out of 40 problems and for other problems it produces better results than the other  $x$  values.

Table 6.4 Results of pilot experiments for neighborhood generation mechanism by using the proposed TS algorithm

Problem Setting	Instance No	Desired Rate	N	x=2				x=3				x=4				x=5							
				Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)	Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)	Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)	Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)
1	1	0.08694	13	0.08694	13	14	4	172.02	0.08694	13	14	4	208.84	0.08694	13	13	3	198.67	0.08694	13	21	11	263.16
	2	0.07338	38	0.07338	38	13	3	923.55	0.07338	38	17	7	1091.97	0.07338	38	14	4	1578.45	0.07338	38	14	4	1158.71
	3	0.06442	13	0.06442	13	14	4	276.28	0.06442	13	14	4	309.77	0.06442	13	13	3	311.56	0.06442	13	15	5	344.33
	4	0.07554	23	0.07554	23	15	5	389.50	0.07554	23	15	5	440.11	0.07554	23	14	4	431.30	0.07554	23	17	7	619.52
	5	0.07907	28	0.07907	28	17	7	180.80	0.07907	28	17	7	234.56	0.07907	28	20	10	171.13	0.07907	28	15	5	166.24
	Avg	0.07587	23	0.07587	23	15	5	388.43	0.07587	23	15	5	457.05	0.07587	23	15	5	538.22	0.07587	23	16	6	510.39
2	1	0.05596	48	0.05603	48	15	5	571.31	0.05598	48	21	11	773.94	0.05602	48	16	6	720.92	0.05603	49	16	6	714.81
	2	0.06253	48	0.06266	48	15	5	455.88	0.06266	48	15	5	507.97	0.06266	48	16	6	590.11	0.06266	48	14	4	472.67
	3	0.06730	38	0.07281	45	19	9	1182.66	0.07281	39	23	13	1398.20	0.06739	48	22	12	1575.69	0.07024	48	17	7	1143.97
	4	0.05925	48	0.05925	48	15	5	799.69	0.05925	48	15	5	920.55	0.05925	48	16	6	1032.42	0.05925	48	14	4	837.84
	5	0.06936	48	0.06936	48	15	5	999.42	0.06936	48	15	5	1017.11	0.06936	48	16	6	1195.30	0.06936	48	17	7	1235.75
	Avg	0.06288	46	0.06402	47	16	6	801.79	0.06401	46	18	8	923.55	0.06294	48	17	7	1022.89	0.06351	48	16	6	881.01
3	1	0.07214	48	0.07214	48	15	5	377.16	0.07214	48	16	6	478.39	0.07214	48	15	5	420.81	0.07214	48	14	4	385.39
	2	0.06911	43	0.06911	43	16	6	430.67	0.06911	43	15	5	457.83	0.06911	43	16	6	463.09	0.06911	43	16	6	539.53
	3	0.08565	53	0.08571	53	20	10	2453.53	0.08618	53	19	9	1755.80	0.08571	53	16	6	1609.52	0.08618	53	16	6	1334.78
	4	0.06365	48	0.06365	48	15	5	901.91	0.06365	48	16	6	1091.73	0.06365	48	15	5	1000.59	0.06365	48	14	4	975.45
	5	0.06659	48	0.06660	48	15	5	250.73	0.06660	48	16	6	306.84	0.06660	48	15	5	281.47	0.06660	48	14	4	273.81
	Avg	0.07143	48	0.07144	48	16	6	882.80	0.07154	48	16	6	818.12	0.07144	48	15	5	755.10	0.07154	48	15	5	701.79
4	1	0.05021	13	0.17492	16	30	28	877.47	0.13110	14	13	3	248.97	0.22953	20	30	23	1833.86	0.36402	20	30	30	1994.13
	2	0.04972	78	0.05134	78	18	8	2205.36	0.05134	78	16	6	1756.86	0.05134	78	20	10	2520.56	0.05134	78	15	5	1534.56
	3	0.05610	38	0.05610	38	13	3	376.03	0.05610	38	17	7	720.39	0.05610	38	14	4	531.36	0.05610	38	14	4	524.45
	4	0.04156	48	0.04156	48	15	5	1241.27	0.04156	48	24	14	2341.20	0.04156	50	12	2	1009.63	0.04156	50	12	2	1245.76
	5	0.05589	9	0.05631	9	4	4	244.55	0.05631	9	4	4	273.13	0.05631	9	9	4	250.98	0.05631	9	9	4	257.88
	Avg	0.05070	33	0.07604	38	16	10	988.94	0.06728	37	15	7	1068.11	0.08697	39	17	9	1229.28	0.11386	39	16	9	1111.36



Table 6.4 Results of pilot experiments for neighborhood generation mechanism by using the proposed TS algorithm (cont.)

Problem Setting	Instance No	Desired Rate	N	x=2				x=3				x=4				x=5							
				Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)	Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)	Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)					
5	1	0.03089	6	0.03089	6	5	5	127.16	0.03089	6	5	5	153.34	0.03089	6	5	5	169.72	0.03089	6	5	5	186.97
	2	0.02444	12	0.02444	12	14	4	68.19	0.02444	12	14	4	76.69	0.02444	12	5	5	54.88	0.02444	12	13	3	87.02
	3	0.02340	8	0.02340	8	5	5	128.78	0.02340	8	5	5	161.88	0.02340	8	5	5	166.42	0.02340	8	5	4	196.76
	4	0.02347	8	0.02347	8	5	5	103.95	0.02347	8	5	5	124.91	0.02347	8	5	5	132.03	0.02347	8	5	5	143.05
	5	0.03700	28	0.03700	28	17	7	223.06	0.03700	28	15	5	181.67	0.03700	28	17	7	282.73	0.03700	28	15	5	239.06
	Avg	0.02784	12	0.02784	12	9	5	130.23	0.02784	12	9	5	139.70	0.02784	12	7	5	161.16	0.02784	12	9	4	170.57
6	1	0.01849	23	0.01849	23	15	5	475.19	0.01849	23	15	5	501.75	0.01849	23	14	4	500.84	0.01849	23	17	7	763.14
	2	0.02342	23	0.02343	23	30	27	1610.44	0.02343	23	21	11	596.19	0.02343	23	30	29	1780.81	0.02342	94	28	18	2959.75
	3	0.02322	48	0.02322	48	15	5	405.23	0.02322	48	16	6	471.76	0.02322	48	15	5	411.03	0.02322	48	14	4	399.17
	4	0.02892	48	0.02892	48	15	5	1153.42	0.02892	48	16	6	1461.27	0.02892	48	15	5	1333.83	0.02892	48	14	4	1156.58
	5	0.02505	48	0.02505	48	15	5	463.33	0.02505	48	16	6	587.45	0.02505	48	15	5	517.25	0.02505	48	14	4	496.86
	Avg	0.02382	38	0.02382	38	18	9	821.52	0.02382	38	17	7	723.68	0.02382	38	18	10	908.75	0.02382	52	17	7	1155.10
7	1	0.02165	43	0.02165	43	16	6	183.25	0.02165	43	15	5	196.28	0.02165	43	16	6	202.41	0.02165	43	16	6	234.14
	2	0.02169	48	0.02169	48	28	18	844.33	0.02169	48	20	10	508.72	0.02169	48	17	7	510.98	0.02169	48	20	10	461.84
	3	0.02167	18	0.02167	18	16	6	102.52	0.02167	18	15	5	89.67	0.02167	18	15	5	93.72	0.02167	18	13	3	88.55
	4	0.02558	43	0.02558	43	16	6	1193.31	0.02558	43	15	5	1220.34	0.02558	43	16	6	1289.27	0.02558	43	16	6	1428.91
	5	0.01396	23	0.01396	23	15	5	71.44	0.01396	23	15	5	76.63	0.01396	23	14	4	77.92	0.01396	23	17	7	116.38
	Avg	0.02091	35	0.02091	35	18	8	478.97	0.02091	35	16	6	418.33	0.02091	35	16	6	434.86	0.02091	35	16	6	465.96
8	1	0.01742	43	0.01744	43	16	6	238.61	0.01744	43	24	14	639.58	0.01744	43	21	11	470.92	0.01744	43	20	10	498.61
	2	0.01280	48	0.01280	49	27	17	1068.11	0.01280	48	24	14	1068.11	0.01293	53	16	6	712.23	0.01280	48	26	16	1003.89
	3	0.03790	48	0.03790	48	15	5	731.67	0.03790	48	16	6	913.09	0.03790	48	17	7	1091.75	0.03790	48	18	8	1111.13
	4	0.01337	40	0.01337	48	15	5	306.70	0.01337	48	16	6	365.40	0.01337	48	17	7	425.28	0.01337	48	14	4	414.72
	5	0.01610	49	0.01610	49	15	5	462.63	0.01610	49	17	7	705.51	0.01610	49	15	5	537.75	0.01610	49	16	6	603.36
	Avg	0.01952	46	0.01952	47	18	8	602.52	0.01952	47	19	9	738.34	0.01955	48	17	7	647.59	0.01952	47	19	9	726.34

The results of pilot experiments for neighborhood generation mechanism by using the proposed SA algorithm are shown in Table 6.5. As seen in Table 6.5, for the problem settings 1, 3, 5 and 6 all  $x$  values produce the same results. For the third problem instance in problem setting 2, none of  $x$  values reaches the optimum solution. Considering the results of pilot experiments, setting the divisor,  $x$  to 3 seems to be a good decision since it can reach to the optimum solution in 38 out of 40 problems.

Table 6.5 Results of pilot experiments for neighborhood generation mechanism by using the proposed SA algorithm

Problem Setting	Instance No	Desired Rate	N	$x=2$					$x=3$					$x=4$					$x=5$				
				Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)	Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)	Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)	Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)
1	1	0.08694	13	0.08694	13	8	4	161.94	0.08694	13	9	5	199.76	0.08694	13	8	3	199.88	0.08694	13	9	5	242.28
	2	0.07338	38	0.07338	38	10	3	1088.23	0.07338	38	10	5	1338.22	0.07338	38	10	3	1141.78	0.07338	38	10	4	1335.70
	3	0.06442	13	0.06442	13	5	4	185.92	0.06442	13	5	4	225.23	0.06442	13	5	3	226.45	0.06442	13	6	5	313.23
	4	0.07554	23	0.07554	23	9	4	366.24	0.07554	23	10	5	470.67	0.07554	23	9	4	376.08	0.07554	23	10	4	523.53
	5	0.07907	28	0.07907	28	15	6	204.97	0.07907	28	10	5	154.08	0.07907	29	10	5	174.46	0.07907	28	10	3	166.06
	Avg	0.07587	23	0.07587	23	9	4	401.46	0.07587	23	9	5	477.59	0.07587	23	8	4	423.73	0.07587	23	9	4	516.16
2	1	0.05596	48	0.05602	49	10	5	660.36	0.05602	48	10	4	707.34	0.05597	49	10	5	736.20	0.05597	48	10	5	743.19
	2	0.06253	48	0.06266	48	10	5	491.00	0.06266	48	10	4	535.79	0.06266	48	10	5	747.46	0.06266	48	10	3	564.14
	3	0.06730	38	0.06923	39	15	10	1349.50	0.06730	48	15	8	1673.11	0.07281	44	10	4	952.02	0.07281	42	10	4	970.75
	4	0.05925	48	0.05925	48	10	5	1062.94	0.05925	48	10	4	1266.22	0.05925	48	10	5	1160.25	0.05925	48	10	3	1256.39
	5	0.06936	48	0.06936	48	10	5	953.95	0.06936	48	10	5	1242.23	0.06936	48	10	5	1146.46	0.06936	48	10	3	1164.31
	Avg	0.06288	46	0.06331	46	11	6	903.55	0.06292	48	11	5	1084.94	0.06401	47	10	5	948.48	0.06401	47	10	4	939.76
3	1	0.07214	48	0.07214	48	10	5	423.31	0.07214	48	10	4	458.23	0.07214	48	10	5	441.14	0.07214	48	10	3	489.16
	2	0.06911	43	0.06911	43	10	5	415.73	0.06911	43	10	5	476.19	0.06911	43	10	5	629.23	0.06911	43	10	5	524.91
	3	0.08565	50	0.08589	50	15	7	2100.92	0.08616	50	15	6	1880.45	0.08589	50	10	5	1896.14	0.08640	50	10	5	1433.55
	4	0.06365	48	0.06365	48	10	5	1057.22	0.06365	48	10	4	1113.55	0.06365	48	10	5	1215.94	0.06365	48	10	3	1150.17
	5	0.06659	48	0.06660	48	10	5	269.17	0.06660	48	10	4	295.13	0.06660	48	10	5	311.43	0.06660	48	10	3	313.36
	Avg	0.07143	47	0.07148	47	11	5	853.27	0.07153	47	11	5	844.71	0.07148	47	10	5	898.78	0.07158	47	10	4	782.23
4	1	0.05021	13	0.22174	23	13	9	597.44	0.13110	13	13	12	910.34	0.13110	14	19	17	1342.31	0.22174	23	11	7	707.17
	2	0.04972	78	0.05134	78	15	7	2564.76	0.05134	78	15	6	2499.44	0.05134	78	10	4	1798.88	0.05134	78	10	5	1788.16
	3	0.05610	38	0.05610	38	10	3	431.33	0.05610	38	10	4	492.34	0.05610	38	10	3	535.64	0.05610	38	10	4	541.67
	4	0.04156	48	0.04156	50	10	2	1293.22	0.04156	48	10	3	1476.85	0.04156	50	10	3	1347.89	0.04156	48	10	4	1420.56
	5	0.05589	9	0.05631	9	5	5	187.68	0.05631	9	5	5	200.50	0.05631	9	5	5	229.83	0.05631	9	5	5	215.67
	Avg	0.05070	37	0.08541	40	11	5	1014.89	0.06728	37	11	6	1115.89	0.06728	38	11	6	1050.91	0.08541	39	9	5	934.65

Table 6.5 Results of pilot experiments for neighborhood generation mechanism by using the proposed SA algorithm (cont.)

Problem Setting	Instance No	Desired Rate	N	x=2				x=3				x=4				x=5							
				Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)	Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)	Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)	Achieved Rate	N	Tot. # of Iterations	# of Iter. for Conv.	CPU (sec.)
5	1	0.03089	6	0.03089	6	6	6	128.25	0.03089	6	6	6	167.80	0.03089	6	6	6	192.20	0.03089	6	6	6	204.31
	2	0.02444	12	0.02444	12	6	4	55.05	0.02444	12	6	4	63.59	0.02444	12	7	5	79.66	0.02444	12	6	3	70.83
	3	0.02340	8	0.02340	8	6	6	130.78	0.02340	8	6	6	63.59	0.02340	8	6	6	195.51	0.02340	8	6	6	219.70
	4	0.02347	8	0.02347	8	6	6	102.30	0.02347	8	6	6	124.52	0.02347	8	6	6	153.22	0.02347	8	6	6	152.44
	5	0.03700	28	0.03700	28	10	5	167.25	0.03700	28	10	5	181.42	0.03700	28	10	4	233.23	0.03700	28	10	4	252.47
	Avg	0.02784	12	0.02784	12	7	5	116.73	0.02784	12	7	5	120.18	0.02784	12	7	5	170.76	0.02784	12	7	5	179.95
6	1	0.01849	23	0.01849	23	10	4	430.69	0.01849	23	10	5	508.09	0.01849	23	10	4	461.91	0.01849	23	10	4	578.83
	2	0.02342	23	0.02343	23	10	4	698.75	0.02343	23	10	5	726.24	0.02343	23	16	11	817.46	0.02342	23	15	10	865.12
	3	0.02322	48	0.02322	48	10	5	411.31	0.02322	48	10	4	440.95	0.02322	48	10	5	491.15	0.02322	48	10	3	466.19
	4	0.02892	48	0.02892	48	10	5	1270.39	0.02892	48	10	4	1455.66	0.02892	48	10	5	1419.43	0.02892	48	10	3	1516.69
	5	0.02505	48	0.02505	48	10	5	529.22	0.02505	48	10	4	573.45	0.02505	48	10	5	803.23	0.02505	48	10	3	614.11
	Avg	0.02382	38	0.02382	38	10	5	668.07	0.02382	38	10	4	740.88	0.02382	38	11	6	798.64	0.02382	38	11	5	808.19
7	1	0.02165	43	0.02165	43	10	5	182.26	0.02165	43	10	5	204.66	0.02165	43	10	5	208.25	0.02165	43	10	5	227.98
	2	0.02169	48	0.02169	53	10	5	426.24	0.02169	48	15	7	581.33	0.02169	50	10	2	432.48	0.02169	48	10	3	383.55
	3	0.02167	18	0.02167	18	10	5	81.81	0.02167	18	9	4	79.72	0.02167	18	9	4	82.42	0.02167	18	9	3	93.45
	4	0.02558	43	0.02558	43	10	5	1235.56	0.02558	43	10	5	1312.13	0.02558	43	10	5	1220.02	0.02558	43	10	5	1475.77
	5	0.01396	23	0.01396	23	9	4	60.78	0.01396	23	10	5	75.05	0.01396	23	9	4	73.88	0.01396	23	9	4	85.05
	Avg	0.02091	35	0.02091	36	10	5	397.33	0.02091	35	11	5	450.58	0.02091	35	10	4	403.41	0.02091	35	10	4	453.16
8	1	0.01742	43	0.01744	43	15	5	372.93	0.01744	43	15	6	383.33	0.01744	43	10	5	268.83	0.01744	43	10	5	338.31
	2	0.01280	43	0.01280	49	10	5	605.75	0.01280	48	10	4	620.94	0.01280	43	10	3	600.98	0.01280	50	10	4	658.34
	3	0.03790	48	0.03790	50	10	2	671.70	0.03790	48	10	3	737.89	0.03790	48	10	3	820.37	0.03790	48	10	3	795.67
	4	0.01818	48	0.01818	48	10	5	550.92	0.01818	48	10	4	594.27	0.01818	48	10	5	614.58	0.01818	48	10	3	587.65
	5	0.01610	49	0.01610	49	15	5	723.77	0.01610	49	15	6	967.06	0.01610	49	10	5	893.46	0.01610	49	10	5	659.08
	Avg	0.02048	46	0.02048	48	12	4	585.01	0.02048	47	12	5	660.70	0.02048	46	10	4	639.64	0.02048	48	10	4	607.81

### 6.4.2 Experiments on Test Problems

We design a computational experiment to test the performance of the proposed algorithms. In our computational experiments, five, ten, and twenty-machine lines ( $K = 5, 10, 20$ ) are considered. For small and medium-sized problems, i.e. 5 and 10 machine-lines, the initial total buffer size is set to 10, 20 and 40 times of the number of machines in the line. However, to reduce the computational cost, for large-sized problems involving 20 machines the initial total buffer size is set to 200. To determine the desired throughput rates using ATS algorithm, all considered problems are solved by setting the total buffer size value to the half of the initial total buffer size value. Hence, a total of 7 problem sets are considered. As seen in Table 6.6, by identifying different levels for processing rates and reliability parameters, eight different settings are generated for each problem set. The processing rates of the machines are generated from a uniform distribution and the failure and repair rates are generated from a geometric distribution. For each setting, 5 runs are made totaling to 800 runs. All algorithms are implemented in C language. The execution is done on a computer having 2.26 GHz Intel Core i5 430M CPU processor and 4 GB of RAM.

Table 6.6 Properties of problem instances

Setting	Processing rate	Failure Rate	Repair Rate
1	(5,15)	(1,200)	(1,10)
2	(5,15)	(1,200)	(1,40)
3	(5,15)	(1,2000)	(1,100)
4	(5,15)	(1,2000)	(1,400)
5	(5,45)	(1,200)	(1,10)
6	(5,45)	(1,200)	(1,40)
7	(5,45)	(1,2000)	(1,100)
8	(5,45)	(1,2000)	(1,400)

Performance comparison of the three algorithms, i.e., binary search, tabu search and simulated annealing is done with respect to solution quality and solution time. It should be noted that since the optimal solution for large sized problems using binary search algorithm can not be reached in reasonable computation time, the comparative

experiments for large sized problems involved only the comparison of the tabu search algorithm to the simulated annealing algorithm.

In our experimental study, the problem sets are denoted by  $K.N$ , where  $K$  is the number of machines in the line and  $N$  is the initial total buffer size to be allocated. In the following tables, we give the average results obtained for all problem sets. The detailed results are given in Appendix C.

#### *6.4.2.1 Results of Small-Sized Problems*

Table 6.7 presents the results of our experimental studies for small-sized problems. In this table, the first two columns represent the problem set and problem setting. Other columns represent the performance of three algorithms with respect to the number of instances achieved the desired throughput rate, minimum value of total buffer size, total number of iterations, the number of iterations for convergence and the CPU time in seconds.

As seen in Table 6.7, for the first problem set, all algorithms converge the optimum solution for all problems. However, the proposed TS and SA algorithms reach the optimum solution with less number of iterations than the binary search algorithm and hence, the solution times of proposed TS and SA algorithms are much less than the solution time of binary search algorithm. Comparing the solution time of TS algorithm to SA, except for the problem setting 5 where the two algorithms have the same solution time, on average, the solution time of proposed TS algorithm is less than the proposed SA algorithm. Overall, it can be said that for solving the problem set 5.50, the solution quality of three algorithms is the same, and the proposed TS algorithm dominates the other two with respect to the solution time.

For the problem set 5.100, both two algorithms get stuck at the local optima for only one problem (see the fourth problem in problem setting 3 for TS and the fourth problem in problem setting 1 for SA in Appendix C2). The convergence rate of TS and SA algorithms is almost the same. As for the solution time, TS algorithm is

slightly better than SA algorithm and in overall, as expected the binary search algorithm has worse performance than the other two.

For problem set 5.200, both TS and SA algorithms reach the optimum solution. Moreover, the convergence rate and the solution times of both algorithms are very close. Like in the case of problem set 5.100, the solution times of both algorithms are better than the binary search algorithm.

In summary, for small-sized problems, the proposed TS and SA algorithms are efficient to solve buffer allocation problem under the objective of total buffer size minimization. The solution times and also the convergence rates of both algorithms are very close and the proposed TS algorithm gets stuck at the local optimum for only one problem in all considered small-sized problems.

Table 6.7 Results of experimental studies for small-sized problems

Problem Set	Problem Setting	Binary Search				Tabu Search				Simulated Annealing					
		# of Instances Achieved Desired Rate	N	Total # of Iterations	CPU (sec.)	# of Instances Achieved Desired Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)	# of Instances Achieved Desired Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)
5.50	1	5	15	23	2.10	5	15	11	6	1.77	5	15	10	5	2.12
	2	5	22	29	7.60	5	22	8	3	3.12	5	22	10	3	4.72
	3	5	24	31	8.04	5	24	8	3	3.83	5	24	10	4	5.37
	4	5	25	32	6.17	5	25	7	2	2.73	5	25	10	2	3.71
	5	5	10	16	1.72	5	10	8	3	1.54	5	10	6	4	1.54
	6	5	17	24	4.90	5	17	9	4	3.31	5	17	8	4	3.46
	7	5	24	32	5.98	5	24	8	3	2.59	5	24	10	3	3.67
	8	5	25	32	9.24	5	25	8	3	4.06	5	25	10	3	5.67
	Avg.	5	20	27	5.72	5	20	8	3	2.87	5	20	9	3	3.78
5.100	1	5	20	29	4.58	5	20	16	6	5.25	5	21	10	5	5.63
	2	5	49	57	20.89	5	49	14	4	9.64	5	49	10	4	11.64
	3	5	45	55	22.68	5	48	14	4	14.10	5	45	11	4	16.64
	4	5	50	58	27.85	5	50	13	3	12.37	5	50	10	3	14.96
	5	5	15	23	4.27	5	15	13	5	4.46	5	15	8	5	3.90
	6	5	39	47	16.62	5	39	16	6	11.42	5	39	11	6	11.36
	7	5	46	55	20.17	5	46	15	5	10.52	5	46	12	5	12.77
	8	5	50	58	34.85	5	50	13	3	14.91	5	50	10	3	18.78
	Avg.	5	39	48	18.99	5	39	14	4	10.33	5	39	10	4	11.96
5.200	1	5	20	33	9.67	5	20	22	7	10.32	5	20	11	5	10.25
	2	5	75	103	80.69	5	75	21	6	37.94	5	75	12	5	34.81
	3	5	96	106	146.13	5	96	20	5	64.66	5	96	14	5	75.16
	4	5	100	109	129.46	5	100	19	4	50.96	5	100	12	4	53.90
	5	5	15	24	7.92	5	15	21	6	9.78	5	15	9	6	8.68
	6	5	58	68	40.48	5	58	21	6	27.54	5	58	13	6	28.63
	7	5	60	125	148.72	5	60	24	9	39.49	5	60	14	6	36.88
	8	5	89	98	103.48	5	89	21	6	51.60	5	89	15	6	60.35
	Avg.	5	64	83	83.32	5	64	21	6	36.54	5	64	12	5	38.58



#### 6.4.2.2 Results of Medium-Sized Problems

Table 6.8 presents the results of our experimental studies for medium-sized problems.

For the problem set 10.100, both the TS and SA algorithms get stuck at the local optimum for the same problems (see the third problem in problem setting 2 and the first problem in problem setting 1, in Appendix C4). For the problem setting 2, SA algorithm finds better solution than TS algorithm while TS finds better solution than SA for the problem setting 4. Moreover, except for the problem settings 6 and 8 (see Table 6.8), on average, the convergence rate of both TS and SA algorithms are very close. Also, the proposed TS and SA algorithms reach the optimum solution with less number of iterations than binary search algorithm and hence, on average, the solution times of proposed TS and SA algorithms are much less than the solution time of binary search algorithm.

Comparing the solution time of TS algorithm to SA, on average, SA algorithm has better performance than the TS algorithm in 6 out of 8 problem settings. Overall, it can be said that for solving the problem set 10.100, the solution quality of both TS and SA algorithms is nearly the same, and the proposed SA algorithm dominates the other two with respect to the solution time.

For the problem set 10.200 the proposed TS algorithm reaches the optimum solution in 36 out of 40 problems while the proposed SA algorithm finds the optimum solution in 35 out of 40 problems (see Appendix C5). Unlike the problem set 10.100, the proposed TS finds better solution than the proposed SA for the problem setting 2 and SA algorithm reaches the desired throughput rate with less total buffer size than TS algorithm for the problem setting 4. Moreover, on average, the convergence rate of both TS and SA algorithms are very close. Also, the proposed TS and SA algorithms reach the optimum solution with less number of iterations than binary search algorithm and hence, on average, the solution times of

proposed TS and SA algorithms are much less than the solution time of binary search algorithm as in the case of the problem set 10.100.

Comparing the solution time of TS algorithm to SA, on average, the solution time of proposed TS algorithm is less than the proposed SA algorithm in 5 out of 8 problem settings. Overall, it can be said that for solving the problem set 10.200, the solution quality of TS is slightly better than SA algorithm, and the proposed TS algorithm dominates the other two with respect to solution time.

For the problem set 10.400, the proposed TS algorithm reaches the optimum solution in 36 out of 40 problems while the proposed SA algorithm finds the optimum solution in 34 out of 40 problems (see Appendix C6). Moreover, on average, the convergence rate of both TS and SA algorithms are very close. Also, the proposed TS and SA algorithms reach the optimum solution with less number of iterations than binary search algorithm and hence, on average, the solution times of proposed TS and SA algorithms are much less than the solution time of binary search algorithm as it is expected.

Comparing the solution time of TS algorithm to SA, on average, the solution time of proposed TS algorithm is less than the proposed SA algorithm in 6 out of 8 problem settings. Overall, it can be said that for solving the problem set 10.400, the solution quality of TS is slightly better than SA algorithm, and the proposed TS algorithm dominates the other two with respect to solution time.

Table 6.8 Results of experimental studies for medium-sized problems

Problem Set	Problem Setting	Binary Search				Tabu Search				Simulated Annealing					
		# of Instances Achieved Desired Rate	N	Total # of Iterations	CPU (sec.)	# of Instances Achieved Desired Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)	# of Instances Achieved Desired Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)
10.100	1	5	23	39	321.01	5	23	16	6	324.84	5	23	8	4	275.42
	2	5	46	64	964.01	5	48	16	6	863.86	5	47	10	5	616.51
	3	5	48	67	1091.25	5	48	15	5	672.14	5	48	10	5	584.20
	4	5	32	47	676.24	5	33	13	5	417.36	5	34	10	5	485.92
	5	5	12	21	80.92	5	12	15	5	117.56	5	12	7	5	117.90
	6	5	38	50	735.12	5	38	18	10	573.28	5	38	11	6	519.11
	7	5	35	53	426.19	5	35	15	5	287.37	5	35	10	4	262.21
	8	5	46	74	727.73	5	46	19	9	539.71	5	46	10	4	418.54
	Avg.	5	35	52	627.81	5	35	16	6	474.51	5	36	10	5	409.98
10.200	1	5	23	32	366.42	5	23	23	8	432.78	5	23	11	5	426.31
	2	5	81	129	2910.72	5	93	22	7	1685.59	5	94	14	7	2015.40
	3	5	80	92	1929.69	5	82	20	5	872.67	5	82	13	5	1131.55
	4	5	64	79	2221.59	5	70	27	12	1996.59	5	64	17	9	1856.38
	5	5	12	27	145.16	5	12	19	7	206.65	5	12	9	6	202.85
	6	5	61	72	1529.89	5	61	21	6	913.16	5	61	15	7	1195.00
	7	5	48	58	683.12	5	48	21	6	501.58	5	48	12	5	511.10
	8	5	81	96	1707.80	5	81	22	7	713.23	5	81	12	5	891.38
	Avg.	5	56	73	1436.80	5	59	22	7	915.28	5	58	13	6	1028.75
10.400	1	5	23	38	577.60	5	23	25	10	916.92	5	23	13	8	828.24
	2	5	133	249	10545.51	5	140	29	16	4622.00	5	136	18	12	4695.92
	3	5	140	153	4728.79	5	142	20	5	1565.74	5	142	14	5	2100.76
	4	5	146	158	4362.45	5	182	23	8	2554.89	5	175	17	9	3091.36
	5	5	12	23	318.96	5	12	25	10	552.72	5	12	13	10	548.64
	6	5	99	151	3898.07	5	99	25	10	2531.18	5	101	16	7	2580.87
	7	5	79	90	1989.83	5	79	23	8	1150.16	5	79	14	7	1178.61
	8	5	179	233	8391.94	5	194	24	9	2440.90	5	195	14	6	2494.23
	Avg.	5	102	137	4351.65	5	109	24	9	2041.81	5	108	15	8	2189.83

In summary, both TS and SA algorithms are efficient to solve the buffer allocation under the objective of total buffer size minimization for medium-sized problems. It is noted that in comparison to the SA algorithm, the TS algorithm needs less number of iterations for terminating the algorithm leading to a better solution time performance for TS. This could be attributed to the fact that during the search some values become tabu, hence tabu algorithm evaluates less number of  $N$  values throughout the search. Moreover, TS algorithm reaches the optimum solution for larger number of problems than SA algorithm. So it can be said that the proposed TS algorithm is more efficient than SA algorithm in solving medium-sized problems in terms of both solution quality and also solution time.

#### *6.4.2.3 Results of Large-Sized Problems*

Table 6.9 represents the results of large-sized problems. As it is stated before, since the optimal solution for large sized problems using binary search algorithm can not be reached in a reasonable computation time, the comparative experiments for large sized problems involved only the comparison of the tabu search algorithm to the simulated annealing algorithm.

As in the previous problem sets, the solution quality of SA algorithm is better than TS algorithm for problem setting 4 which involves rarely failed machines with long repair times and for problem setting 7, TS algorithm achieves the desired throughput rate with less buffer sizes than SA algorithm. Although, on average, the performance of TS and SA algorithms is very close (see Table 6.9), it is not possible to draw a general conclusion. It seems that the solution quality of the proposed algorithms is somehow affected by the features of the problem at hand. As for the solution time, TS algorithm is much better than SA algorithm even if, the total number of iterations for stopping the proposed TS algorithm is higher than the proposed SA algorithm. In summary, TS algorithm is clearly better than SA algorithm in terms of solution time for large-sized problems involving 20 machines.

Table 6.9 Results of experimental studies for large-sized problems

Problem Set	Problem Setting	Tabu Search					Simulated Annealing				
		# of Instances Achieved Desired Rate	<i>N</i>	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)	# of Instances Achieved Desired Rate	<i>N</i>	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)
20.200	1	5	48	21	6	5666.20	5	48	11	5	12228.99
	2	5	46	16	7	5256.70	5	46	22	19	26546.11
	3	5	80	22	10	14238.91	5	80	16	12	36599.90
	4	5	64	18	6	9754.57	5	62	13	8	16372.69
	5	5	30	18	6	7759.27	5	30	10	6	7593.38
	6	5	73	22	7	9955.45	5	73	10	5	6147.75
	7	5	69	26	11	24429.62	5	70	13	7	28809.44
	8	5	91	22	7	15826.44	5	91	11	6	17340.51
	<b>Avg.</b>	<b>5</b>	<b>63</b>	<b>21</b>	<b>8</b>	<b>11610.90</b>	<b>5</b>	<b>63</b>	<b>13</b>	<b>9</b>	<b>18954.85</b>

### 6.4.3 Summary of the Findings

In this section, the performance of three heuristic algorithms is tested on wide range of problems with varying difficulty. The results are summarized in Table 6.10. As seen in Table 6.10, regarding solution quality, the proposed TS and SA algorithms have the same performance for small-sized problems and as for solution time, TS algorithm has slightly better performance than SA algorithm. For medium-sized and large-sized problems TS algorithm finds better solution for larger number of problems in less time than the proposed SA algorithm (see Table 6.10 and also Figure 6.2). Hence, it can be concluded that the proposed TS is better than the proposed SA in terms of both solution quality and also solution time when the problem size increases. It should be noted that the superiority of SA algorithm is more observable for the problem setting 4 which involves rarely failed machines with long repair times for all considered problem sets. As a result, it can be concluded that the proposed TS algorithm is the most efficient algorithms among the others for solving buffer allocation problem in unreliable and also non-homogeneous lines under the objective of total buffer size minimization.

Table 6.10 Summary of experimental studies

Problem Size	Problem Set	# Of Instances Find Optimum		CPU Time (sec.)	
		TS	SA	TS	SA
Small	5.50	40	40	2.87	3.78
	5.100	39	39	10.33	11.96
	5.200	40	40	36.54	38.58
	<b>Average</b>	<b>40</b>	<b>40</b>	<b>16.58</b>	<b>18.11</b>
Medium	10.100	38	37	474.51	409.98
	10.200	36	35	915.28	1028.75
	10.400	36	34	2041.81	2189.83
	<b>Average</b>	<b>37</b>	<b>35</b>	<b>1143.87</b>	<b>1209.52</b>
Problem Size	Problem Set	# Of Instances Find Better Solution than SA		CPU Time (sec.)	
Large	20.200	6		11610.90	18954.85

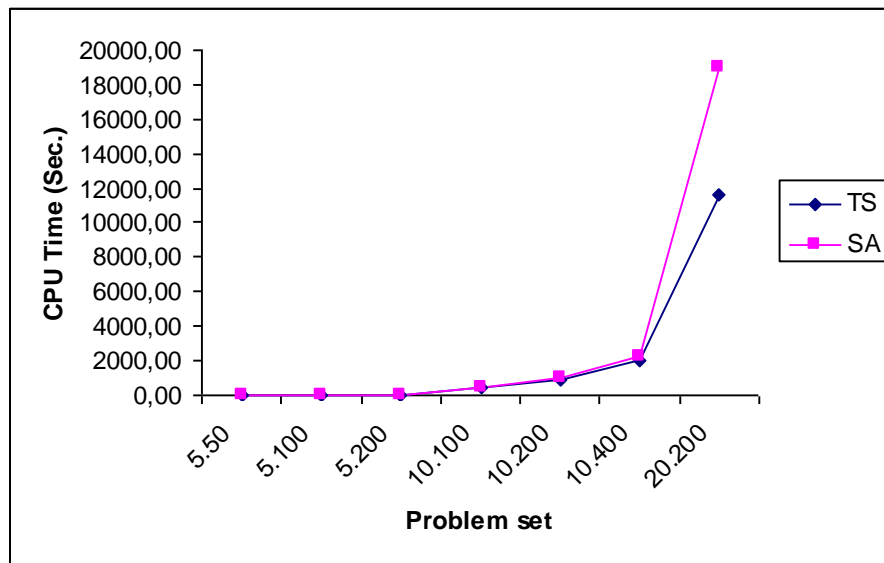


Figure 6.2 Solution times of both algorithms with respect to the problem sets

## 6.5 Chapter Summary

In this chapter, we proposed an integrated approach to solve buffer allocation problem for unreliable and also non-homogeneous production lines to maximize the throughput rate of the line with minimum total buffer size. The proposed integrated approach has two control loops, i.e., the inner loop and outer loop. While the inner loop control includes the adaptive tabu search algorithm proposed in the previous chapter, the outer loop control includes three different algorithms, i.e., binary search, tabu search and simulated annealing. These nested loops aim at minimizing the total buffer size to achieve a desired throughput level. To improve searching efficiency of the proposed TS and SA algorithms alternative neighborhood generation mechanisms are suggested and their performance are tested.

The performance of proposed algorithms is tested on randomly generated test problems. In general, the results of experimental study show that the proposed TS algorithm is much better than the proposed SA algorithm in terms of both the solution quality and solution time especially when problem size increases. It is also observed that the solution quality of the proposed algorithms can be somehow affected by the features of the problem at hand.

## **CHAPTER SEVEN**

### **CONCLUSION**

#### **7.1 Summary of the Thesis**

The buffer allocation problem, which involves the distribution of buffer space among the intermediate buffers of a production line, arises in a wide range of manufacturing systems, such as transfer lines, flexible manufacturing systems or robotic assembly lines and it is one of the major optimization problems faced by manufacturing systems designers.

Due to its importance and complexity, a considerable amount of work has been done in this area. The previous studies in this area mainly focus on characterizing and describing optimal buffer distributions. In last ten years, the main focus of many research studies has been on developing methods to optimize buffer sizes in production lines. The purpose of this Ph.D. thesis is also to construct and describe efficient algorithms to be used in the design of production lines.

Generally, the buffer allocation problem is classified into two categories according to the objective function employed to solve this problem. The first one aims at maximizing the throughput rate of the line and the second one focuses on total buffer size minimization. The throughput maximization problem has been studied more extensively in the literature. Moreover, employing meta-heuristic methods to solve buffer allocation problem is a new trend in this area. To better search the solution space, the recent trend is to hybridize the meta-heuristics with other methods. However, a few studies attempt to solve buffer allocation problem by hybrid methods.

Hence, to fill the perceived gaps in current relevant literature, this Ph.D. study:

- focuses on the buffer allocation problem in unreliable non-homogeneous production lines



- proposes a new adaptive tabu search algorithm to solve this problem under the objective of throughput maximization,
- hybridizes this adaptive tabu search algorithm with other meta-heuristics to integrate the issue of total buffer size minimization into the solution of this problem,
- minimizes the total buffer size subject to achieve the desired level of throughput.

In summary, to solve the buffer allocation problem, this Ph.D. study not only proposes stand-alone search methods it also proposes new hybrid approaches which consider the minimization of total buffer subject to the desired throughput level. The objectives of this Ph.D. study listed above are fulfilled in three stages. In the first stage, a TS algorithm is proposed to solve the buffer allocation problem under the objective of throughput maximization for homogeneous production lines involving unreliable machines with deterministic processing times. Following a pilot experiment to identify the best TS parameters, the new move definitions for buffer allocation problem are introduced.

In the second stage, the problem is extended to non-homogeneous production lines, and an adaptive TS algorithm is proposed to solve the revised problem under the objective of throughput maximization. Besides proposing a new strategy to tune the parameters of TS adaptively during the search, an experimental study is carried out to select an intelligent initial solution scheme among three alternatives so as to decrease the search effort to obtain the best solutions.

Finally, in the last stage, three hybrid approaches are proposed to solve the buffer allocation problem for non-homogeneous production lines involving unreliable machines with deterministic processing times. These three approaches which integrate binary search, tabu search, and simulated annealing with an adaptive tabu search mechanism aim at minimizing the total buffer size to achieve a desired throughput level. To improve the searching efficiency of the proposed TS and SA

algorithms alternative neighborhood generation mechanisms are suggested and their performance are tested.

## 7.2 Contributions

The original contributions of this thesis can be summarized as follows:

- This is the first extensive study employing tabu search to solve the buffer allocation problem for unreliable and also non-homogeneous production lines. Only two studies employing tabu search for buffer allocation problem are noted in the literature. While Shi and Men (2003) employ tabu search in a simple form and combines it with the nested partitions method to solve the problem in unreliable *homogenous lines* with nine machines, Lutz et al. (1998) employ tabu search for solving the problem in *reliable lines* with six machines. Unlike these studies, we focus on the buffer allocation problem in *unreliable non-homogeneous lines* and we test the performance of proposed search methods not only *in small-sized production lines* but also *large-sized production lines*.
- Proposed tabu search methods are not implemented in a simple form, *new move definitions* are introduced and the efficiency of these moves is tested in a large range of problems.
- The buffer allocation problem is generalized by adding buffer space constraints for each buffer location and an *adaptive search strategy of intensification and diversification* is proposed to solve the buffer allocation problem. The performance of the proposed adaptive search algorithms is tested both in short and long production lines.
- Unlike the current relevant research employing random initialization, to improve the search efficiency, alternative initialization schemes utilizing problem specific features such as machine failure/repair rate and machine

processing time are suggested and their performance are tested on various size of problems.

- Unlike the current relevant studies which deal with buffer allocation problem only under the objective of throughput maximization, this study also integrates the issue of total buffer size minimization into the solution approaches. Specifically, we hybridize the proposed adaptive tabu search algorithm with other search methods to solve the buffer allocation problem under the objective of total buffer size minimization subject to desired throughput rate.

### **7.3 Future Research Directions**

Future research directions can be summarized with respect to the scope the problem studied and the scope of the solution approaches proposed to deal with this problem.

Regarding the scope of the problem, some of the future research directions can be stated as follows:

- In this study we only consider the serial lines. The proposed algorithms can be tested on serial-parallel lines, assembly lines or other systems having general network topology.
- The assumption of deterministic machine processing times can be extended to stochastic processing times.
- Besides geometric distribution for machine reliability parameters other distributions such as phase-type distribution can be considered.
- The problem studied in this thesis can be extended to BAP3 which considers work in process minimization and the performance of the

proposed algorithms can be tested on this problem. As stated in Chapter 3, this problem is the least studied one in buffer allocation literature.

- Lastly, objective functions involving monetary criteria which are expressed in a form of profits or costs, can be considered and solved by the proposed algorithms.

Regarding the proposed solution methods, some of the future research directions can be stated as follows:

- The proposed algorithms can be combined with other evaluation methods, such as aggregation method or simulation.
- The proposed adaptive tabu search algorithm can be hybridized with other meta-heuristic algorithms such as genetic algorithms or ant colony optimization for total buffer size minimization.
- The problem can be solved in a multi-objective manner by using pareto optimization methods.

Furthermore, as it is stated in the beginning of this study, the aim of the study is construct efficient algorithms for buffer allocation in production lines. When implemented in a real world environment, the proposed algorithms can provide the practitioners with valuable information about how to allocate limited amount of buffers. However, it is not realistic to expect that the practitioners will be able to easily use these approaches in designing production lines. Hence, to improve the practicality of the proposed approaches, in a future study, these buffer allocation algorithms can be integrated into a decision support system with a user-friendly interface.

**REFERENCES**

- Aksoy, K. H. & Gupta, S. M. (2005). Buffer allocation plan for a remanufacturing cell. *Computers&Industrial Engineering*, 48, 657-677.
- Aksoy, K. H. & Gupta, S. M. (2010). Near optimal buffer allocation in remanufacturing systems with  $N$ -policy. *Computers&Industrial Engineering*, 59, 496-508
- Allon, G., Kroese, D.P., Raviv, T. & Rubinstein. R.Y. (2005). Application of the Cross-Entropy Method to the Buffer Allocation Problem in a Simulation-Based Environment. *Annals of Operations Research*, 134 (1), 137-151.
- Altioek, T. & Stidham, S. (1983). The allocation of interstage buffer capacities in production lines. *IIE Transactions*, 18, 251-261.
- Altiparmak, F., Dengiz, B. & Bulgak, A. A. (2007). Buffer allocation and performance modeling in asynchronous assembly system operations: An artificial neural network metamodeling approach. *Applied Soft Computing*, 7, 946-956.
- Battini, D., Persona, A. & Regattieri, A. (2009). Buffer size design linked to reliability performance: A simulative study. *Computers&Industrial Engineering*, 56(4), 1633-1641.
- Bulgak, A. A. (2006). Analysis and design of split and merge unpaced assembly systems by metamodeling and stochastic search. *International Journal of Production Research*, 44 (18-19), 4067-4080.
- Burman. M. (1995). *New results in flow line analysis*. Ph.D. Thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge MA.
- Buzacott, J. A. (1967). Automatic Transfer Lines with Buffer Stocks. *International Journal of Production Research*, 5 (3), 183-200.

- Buzacott, J.A. & Shanthikumar, J.G. (1993). *Stochastic Models of Manufacturing Systems*. New Jersey:Prentice-Hall.
- Can, B., Beham, A. & Heavey, C. (2008). A comparative study of genetic algorithm components in simulation-based optimisation. *Proceedings of the 2008 Winter Simulation Conference*, 1829-1837.
- Can, B. & Heavey, C. (2009). Sequential metamodelling with genetic programming and particle swarms. *Proceedings of the 2009 Winter Simulation Conference*, 3150-3157.
- Chaharsooghi, S. K. & Nahavandi, N. (2003). Buffer Allocation Problem, A Heuristic Approach, *Scientia Iranica*, 10 (4), 401-409.
- Chehade, H., Yalaoui, F., Amodeo, L. & Dugardin, F. (2010). Buffers sizing in assembly lines using a Lorenz multiobjective ant colony optimization algorithm. *IEEE International Conference on Machine and Web Intelligence*, 283-287.
- Chow, W-M. (1987). Buffer capacity analysis for sequential production lines with variable process times. *International Journal of Production Research*, 25 (8), 1183-1196.
- Colledani, M., et al. (2004). A new analytical method for buffer space allocation in production lines. *37<sup>th</sup> CIRP International Seminar on Manufacturing Systems*, 231-237.
- Colledani, M., Ekvall, M., Lundholm, T, Moriggi, P., Polato A. & Tolio, T. (2010). Analytical methods to support continuous improvements at Scania. *International Journal of Production Research*, 48 (7), 1913-1945.
- Cruz, F. R. B., Duarte, A. R. & Van Woensel, T. (2008). Buffer allocation in general single-server queuing networks. *Computers and Operations Research*. 35(11), 3581-3598.

- Cruz, F. R. B., Van Woensel, T. & MacGregor Smith, J. (2010). Buffer and throughput trade-offs in M/G/1/K queuing networks: A bi-criteria approach. *International Journal of Production Economics*, 125, 224-234.
- Dallery, Y., David, R. & Xie, X-L. (1988). An efficient algorithm for analysis of transfer lines with unreliable machines and finite buffers. *IIE Transactions*, 23 (3), 280-283.
- Dallery, Y., David, R. and Xie, X-L. (1989). Approximate analysis of transfer lines with unreliable machines and finite buffers. *IEEE Transactions on Automatic Control*, 34 (9), 943-953.
- Dallery, Y. & Gershwin, S.B. (1992). Manufacturing flow line systems: a review of models and analytical results. *Queuing Systems*, 12 (1-2), 3-94.
- Daskalaki, S. & MacGregor Smith, J. (2004). Combining routing and buffer allocation problems in serial-parallel queuing networks. *Annals of Operations Research*, 125, 47-68.
- Demir, L. & Tunali, S. (2008). A new approach for optimal buffer allocation in unreliable production lines. *Proceedings of 38th International Conference on Computers and Industrial Engineering*, 1962-1970.
- Demir, L., Tunali, S. & Eliiyi, D. T. (2010). An adaptive tabu search approach for buffer allocation problem in unreliable production lines. *24th Mini EURO Conference on Continuous Optimization and Information-based Technologies in the Financial Sector*, Selected Papers, 207-212.
- Demir, L., Tunali, S. & Løkketangen A. (2011). A tabu search approach for buffer allocation in production lines with unreliable machines. *Engineering Optimization*, 43 (2), 213-231.
- Diamantidis, A. C. & Papadopoulos, C. T. (2004). A dynamic programming algorithm for the buffer allocation problem in homogeneous asymptotically

- reliable serial production lines. *Mathematical Problems in Engineering*, 3, 209-223.
- Dolgui, A., Ereemeev, A., Kolokolov, A. & Sigaev, V. (2002). A genetic algorithm for the allocation of buffer storage capacities in a production line with unreliable machines. *Journal of Mathematical Modeling and Algorithms*, 1, 89-104.
- Dolgui, A., Ereemeev, A. & Sigaev, V. (2007). HBBA: hybrid algorithm for buffer allocation in tandem production lines. *Journal of Intelligent Manufacturing*, 18, 411-420.
- Enginarlar, E., Li, J., Meerkov., S.M. & Zhang, R.Q. (2002). Buffer capacity for accommodating machine downtime in serial production lines. *International Journal of Production Research*, 40, 601–624.
- Enginarlar, E. (2003). *Lean buffering in production systems: A quantitative approach*. Ph.D. Thesis, The University of Michigan, Electrical Engineering: Systems.
- Enginarlar, E., Li, J. & Meerkov, S. M. (2005). How lean can lean buffers be? *IIE Transactions*, 37, 333-342.
- Fuxman, L. (1998). Optimal buffer allocation in asynchronous cyclic mixed model assembly lines. *Production and Operations Management*, 7 (3), 294-311.
- Gasimov, R. N. & Ustun, O. (2007). Solving the quadratic assignment problem using F-MSG algorithm”, *Journal of Industrial and Management Optimization*, 3 (2), 173-191.
- Gendreau, M. & Potvin. J-Y. (2005). Tabu Search. *In: E. Burke and G. Kendall, eds. Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Newyork: Springer, 165-186.



- Gershwin, S. B., and Schick, I. C. (1983). Modeling and Analysis of Three-Stage Transfer Lines with Unreliable Machines and Finite Buffers, *Operations Research*, Vol. 31, No. 2, 354-380.
- Gershwin, S.B. (1987). An efficient decomposition method for the approximate evaluation of tandem queues with finite storage space and blocking. *Operations Research*, 35 (2), 291-305.
- Gershwin, S.B., & Schor. J.E. (2000). Efficient algorithms for buffer space allocation. *Annals of Operations Research*, 93, 117-144.
- Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8, 156-166.
- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers&Operations Research*, 13, 533-549.
- Glover, F. (1989). Tabu Search-Part I. *ORSA Journal on Computing*, 1 (3), 190-206.
- Glover, F., Taillard, E. & Werra. D. (1993). A user's guide to tabu search. *Annals of Operations Research*, 41, 3-28.
- Glover, F. & Laguna. M. (1997). *Tabu Search*. Dordrecht:Kluwer Academic Publishers.
- Gurkan, G. (2000). Simulation optimization of buffer allocations in production lines with unreliable machines. *Annals of Operations Research*, 93, 177-216.
- Han, M-S. & Park, D-J. (2002). Optimal buffer allocation of serial production lines with quality inspection machines. *Computers&Industrial Engineering*, 42, 75-89.
- Harris, J. H. & Powell, S. G. (1999). An algorithm for optimal buffer placement in reliable serial lines. *IIE Transactions*, Vol. 31, 287-302.

- Heavey, C., Papadopoulos, H. T. & Browne, J. (1993). The throughput rate of multistation unreliable production lines. *European Journal of Operational Research*, 68, 69-89.
- Helber, S. (2001). Cash-flow-oriented buffer allocation in stochastic flow lines. *International Journal of Production Research*, 39 (14), 3061-3083.
- Hemachandra, N. & Eedupuganti, S. K. (2003). Performance analysis and buffer allocations in some open assembly systems. *Computers & Operations Research*, 30, 695-704.
- Hillier, S. M. (2000). Characterizing the optimal allocation of storage space in production line systems with variable processing times. *IIE Transactions*, 32, 1-8.
- Hillier, S. M. & Hillier F. S. (2006). Simultaneous optimization of work and buffer space in unpaced production lines with random processing times. *IIE Transactions*, 38, 39-51.
- Ho, Y.C., Eyster, M.A. & Chien, T.T. (1979). A gradient technique for general buffer storage design in a production line. *International Journal of Production Research*, 17 (2), 557-580.
- Huang, M-G., Chang, P-L. & Chou Y-C. (2002). Buffer allocation in flow-shop-type production systems with general arrival and service patterns. *Computers & Operations Research*, 29,103-121.
- Jafari, M.A. & Shanthikumar, J.G. (1989). Determination of optimal buffer storage capacities and optimal allocation in multistage automatic transfer lines. *IIE Transactions*, 21 (2), 130-135.
- Jeong, K.-C. & Kim, Y.-D. (2000). Heuristics for selecting machines and determining buffer capacities in assembly systems. *Computers & Industrial Engineering*, 38, 341-360.

- Kim, S. & Lee, H-J. (2001). Allocation of buffer capacity to minimize average work-in-process. *Production Planning & Control*, 12 (7), 706-716.
- Kirkpatrick, S., Gelatt, C. D. Jr. & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220 (4598), 671-680.
- Koenigsberg, E. (1959). Production lines and internal storage-a review. *Management Science*, 5, 410-433.
- Kose, S. Y. (2010). *Capacity improvement in a real manufacturing system using a hybrid simulation/genetic algorithm approach*. M.S. Thesis, Dokuz Eylul University, Graduate School of Natural and Applied Sciences.
- Kwon, S-T. (2006). On the optimal buffer allocation of an FMS with finite in-process buffers. *LNCS 3982*, 767-776.
- Lee, S-D. (2000). Buffer sizing in complex cellular manufacturing systems. *International Journal of Systems Science*. 31 (8), 937-948.
- Lee, S-D. & Ho, S-H. (2002). Buffer sizing in manufacturing production systems with complex routings. *Int. J. Computer Integrated Manufacturing*, 15 (5), 440-452.
- Lee, H-T., Chen, S-K. & Shunder Chang S. (2009). A meta-heuristic approach to buffer allocation in production line. *Journal of C.C.I.T.*, 38 (1), 167-178.
- Louw, L. & Page, D. C. (2004). Queuing network analysis approach for estimating the sizes of the time buffers in Theory of Constraints-controlled production systems. *International Journal of Production Research*, 42 (6), 1207-1226.
- Lutz, C.M., Davis, K.R. & Sun, M. (1998). Determining buffer location and size in production lines using tabu search. *European Journal of Operational Research*, 106, 301-316.
- Lü, Z. & Hao, J. K. (2010). Adaptive tabu search for course timetabling. *European Journal of Operational Research*. 200(1), 235-244.

- MacGregor Smith, J. & Cruz, F. R. B. (2005). The buffer allocation problem for general finite buffer queuing networks. *IIE Transactions*, 37(4), 343-365.
- Massim, Y., Yalaoui, F., Amodeo, L., Chatelet, E. & Zeblah, A. (2010). Efficient combined immune-decomposition algorithm for optimal buffer allocation in production lines for throughput and profit maximization. *Computers & Operations Research*, 37(4), 611-620.
- Matta, A., Runchina, M. & Tolio, T. (2005). Automated flow lines with shared buffer. *OR Spectrum*, 27, 265-286.
- Meester, L. E. & Shanthikumar, J. G. (1990). Concavity of the throughput of tandem queuing systems with finite buffer storage space. *Advances in Applied Probability*, 22, 764-767.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. (1953). Equation of state calculations by fast computing machines. *J. of Chem. Phys.*, 21 (6), 1087-1092.
- Nahas, N., Ait-Kadi, D. & Nourelfath, M. (2006). A new approach for buffer allocation in unreliable production lines. *International Journal of Production Economics*, 103, 873-881.
- Nahas, N., Ait-Kadi, D. & Nourelfath, M. (2009). Selecting machines and buffers in unreliable series-parallel production lines. *International Journal of Production Research*. 47 (14), 3741-3774.
- Nieuwenhuysse, I. V., Vandaele, N., Rajaram, K & Karmarkar, U. S. (2007). Buffer sizing in multi-product multi-reactor batch processes: Impact of allocation and campaign sizing policies. *European Journal of Operational Research*, 179, 424-443.
- Nourelfath, M., Nahas, N. & Ait-Kadi, D. (2005). Optimal design of series production lines with unreliable machines and finite buffers. *Journal of Quality in Maintenance Engineering*, 11 (2), 121-138.

- Okamura, K. & Yamashina, H. (1977). Analysis of the effect of buffer storage capacity in transfer line systems. *AIIE Transactions*, 9 (2), 127-135.
- Othman, Z., Kamaruddin, S. & Ismail, M. S. (2007). Optimal Buffer Allocation for Unpaced Balanced and Unbalanced Mean Processing Time, *Jurnal Teknologi*, 46 (A), 31–42.
- Papadopoulos, H.T., Heavey, C. & Browne, J. (1993). *Queuing Theory in Manufacturing Systems Analysis and Design*. London: Chapman and Hall.
- Papadopoulos, H.T. & Heavey, C. (1996). Queuing theory in manufacturing systems analysis and design: A classification of models for production and transfer lines, *European Journal of Operational Research*, 92, 1-27.
- Papadopoulos, H. T. & Vidalis, M. I. (1998). Optimal buffer storage allocation in balanced reliable production lines. *Int. Trans. Opl Res.*, 5 (4), 325-339.
- Papadopoulos, H. T. & Vidalis, M. I. (2001). A heuristic algorithm for the buffer allocation in unreliable unbalanced production lines. *Computers&Industrial Engineering*, 41, 261-277.
- Papadopoulos, C. T., O’Kelly, M. E. J., Vidalis, M. J. & Spinellis, D. (2009). *Analysis and Design of Discrete Part Production Lines*. New York: Springer Science+Business Media.
- Park, T. (1993). A two-phase heuristic algorithm for determining buffer sizes of production lines. *International Journal of Production Research*, 31 (3), 613-631.
- Pham, D. T. & Karaboga, D. (2000). *Intelligent Optimization Techniques*. London: Springer-Verlag.
- Powell, S. G. & Pyke, D. F. (1998). Buffering unbalanced assembly systems. *IIE Transactions*, 30, 55-65.
- Qudeiri, J. A., Yamamoto, H., Ramli, R. & Al-Momani, K.R. (2007). Development of production simulator for buffer size decisions in complex production systems

- using genetic algorithms. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, 1(3), 418-429.
- Qudeiri, J. A., Yamamoto, H., Ramli, R. & Jamali, A. (2008). Genetic algorithm for buffer size and work station capacity in serial–parallel production lines. *Artificial Life and Robotics*, 12, 102-106.
- Raman, N. A. & Jamaludin, E. K. R. (2008). Implementation of Toyota Production System (TPS) in the production line of a local automotive parts manufacturer. *Proceedings of International Conference on Mechanical & Manufacturing Engineering*.
- Reeves, C.R. (1996). Modern Heuristic Techniques. In: Rayward-Smith, V.J., Osman, I.H., Reeves, C.R, and Smith, G.D., eds. *Modern Heuristic Search Methods* (1<sup>st</sup> ed.) (1-25). England: John Wiley&Sons.
- Ribeiro, M. A., Silveira, J. L. & Qassim, R. Y. (2007). Joint optimization of maintenance and buffer size in a manufacturing system. *European Journal of Operational Research*, 176, 405-413.
- Roser, C., Nakano, M. & Tanaka, M. (2003). Buffer allocation model based on a single simulation. *Proceedings of the Winter Simulation Conference*, 1238-1246.
- Sabuncuoglu, I., Erel E., & Kok, A. G. (2002). Analysis of assembly systems for interdeparture time variability and throughput. *IIE Transactions*, 34, 23-40.
- Sabuncuoglu, I., Erel, E., & Gocgun, Y. (2006). Analysis of serial production lines: characterization study and a new heuristic procedure for optimal buffer allocation. *International Journal of Production Research*, 44 (13), 2499-2523.
- Schor, J. E. (1995). *Efficient algorithms for buffer allocation*. M.S. Thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA.

- Seong, D., Chang, Y.S. & Hong, Y. (1995). Heuristic algorithms for buffer allocation in a production line with unreliable machines. *International Journal of Production Research*, 33 (7), 1989-2005.
- Seong, D., Chang, Y.S., and Hong, Y. (2000). An algorithm for buffer allocation with linear resource constraints in a continuous-flow unreliable production line. *Asia-Pacific Journal of Operational Research*, 17, 169-180.
- Shi, L. & Men, S. (2003). Optimal buffer allocation in production lines. *IIE Transactions*, 35, 1-10.
- Shi, C. & Gershwin, S. B. (2009). An efficient buffer design algorithm for production line profit maximization. *International Journal of Production Economics*. 122, 725-740.
- Spinellis, D. D. & Papadopoulos, C.T. (2000a). A simulated annealing approach for buffer allocation in reliable production lines. *Annals of Operations Research*, 93, 373-384.
- Spinellis, D. D. & Papadopoulos, C. T. (2000b). Stochastic Algorithms for Buffer Allocation in Reliable Production Lines. *Mathematical Problems in Engineering*, Vol. 5, 441-458.
- Spinellis, D., Papadopoulos, C. & MacGregor-Smith, J. (2000). Large production line optimization using simulated annealing. *International Journal of Production Research*, 38 (3), 509-541.
- Sörensen K. & Janssens, G. K. (2001). Buffer allocation and required availability in a transfer line with unreliable machines. *International Journal of Production Economics*, 74, 163-173.
- Tempelmeier, H. (2003). Practical considerations in the optimization of flow production systems. *International Journal of Production Research*, 41 (1), 149-170.

- Um, I., Lee, H. & Cheon, H. (2007). Determination of Buffer Sizes in Flexible Manufacturing System by Using the Aspect-oriented Simulation. *International Conference on Control, Automation and Systems*, 1729-1733.
- Van Woensel, T., Andriansyah R., Cruz, F. R. B., MacGregor Smith, J. & Kerbache, L. (2010). Buffer and server allocation in general multi-server queuing Networks. *International Transactions in Operational Research*, 17, 257-286.
- Vitanov, I. V., Vitanov, V. I. & Harrison, D. K. (2009). Buffer capacity allocation using ant colony optimization algorithm. *Proceedings of the 2009 Winter Simulation Conference*, 3158-3168.
- Vouros, G.A. & Papadopoulos, H.T. (1998). Buffer allocation in unreliable production lines using a knowledge-based system, *Computers and Operations Research*, 25 (12), 1005-1067.
- Yamada, T. & Matsui, M. (2003). A management design approach to assembly line systems. *International Journal of Production Economics*, 84, 193-204.
- Yamamoto, H., Qudeiri, J. A. & Marui, E. (2008). Definition of FTL with bypass lines and its simulator for buffer size decision. *International Journal of Production Economics*, 112, 18-25.
- Yamashita, H. & Altiok, T. (1998). Buffer capacity allocation for a desired throughput in production lines. *IIE Transactions*, 30, 883-891.
- Yuzukirmizi, M. & MacGregor Smith, J. (2008). Optimal buffer allocation in finite closed networks with multiple servers. *Computers & Operations Research*, 35, 2579-2598.
- Zequeira, R., Prida, B. & Valdes, J. E. (2004). Optimal buffer inventory and preventive maintenance for an imperfect production process. *International Journal of Production Research*, 42 (5), 959-974.



Zequeira, R., Valdes, J. E. & Berenguer C. (2008). Optimal buffer inventory and opportunistic preventive maintenance under random production capacity availability. *International Journal of Production Economics*, 111, 686-696.

## **APPENDICES**

**APPENDIX A**  
**DDX AND ADDX ALGORITHMS**

## A1. Model and Assumptions

In this section, we consider a homogeneous production line, denoted by  $L$ , composed of  $K$  machines,  $\{M_1, \dots, M_K\}$ , separated by  $K-1$  buffers,  $\{B_1, \dots, B_{K-1}\}$ . Let  $N_i$  be the finite capacity of buffer  $B_i$ . A part enters the first machine from outside the system. Each part is processed by machine 1, after which it moves to buffer  $B_1$ . The part moves in the downstream direction, from machine  $i$  to buffer  $i$  and on to machine  $i+1$ , until it is processed by the last station, machine  $K$  and leaves the system. It is assumed that raw parts are always available at the input of the line and there are always empty spaces at the output of the line. The processing time of a part on a machine is deterministic, i.e., it requires a fixed amount of time. Moreover, it is assumed that this time is the same for all machines and, without loss of generality, is taken as the time unit. The machines are unreliable: during the processing of a part, machine  $M_i$  has a probability  $p_i$  of failing. When machine  $M_i$  is down, it is under repair and it has a probability  $r_i$  of being repaired during a time unit. The mean time to failures (MTTF) and the mean time to repair (MTTR) of machine  $M_i$  are  $1/p_i$ , and  $1/r_i$ , respectively. The isolated efficiency of machine  $M_i$  is given by Buzacott (1967):

$$e_i = \frac{r_i}{r_i + p_i} \quad (1)$$

The behavior of such a production line is fairly complex. This is especially due to the interaction between machines due to the finite buffers. When machine  $M_i$  is down, the number of parts in buffer  $B_{i-1}$  increases while the number of parts in buffer  $B_i$  decreases. If this condition persists,  $B_{i-1}$  may become full and therefore machine  $M_{i-1}$  is blocked, or  $B_i$  may become empty and therefore machine  $M_{i+1}$  is starved. It is assumed that the first machine is never starved and the last machine is never blocked. Also it is assumed that the machines can be failed only when it is working, starved or blocked machine cannot be failed.

In such system the production rate  $P$  when all buffer sizes are infinite is the minimum of the production rates of all the machines in the line. That is,

$$P(\infty, \dots, \infty) = \min_{i=1, \dots, K} e_i \quad (2)$$

The production rate when all buffers have size 0 is:

$$P(0, \dots, 0) = \frac{1}{1 + \sum_{i=1}^K p_i / r_i} \quad (3)$$

When there are more than two machines and all  $N_i$  are neither infinite nor zero, the production rate and average inventory levels cannot be calculated analytically or exactly even numerically. In these situations approximate decomposition methods or simulation can be used. Because of simulation requires long computational time decomposition method is widely used for modeling such production systems. In the following sections first we describe decomposition method then we give DDX and ADDX algorithms to solve decomposition equations.

## A2. Decomposition Method and DDX Algorithm

Decomposition method is proposed by Gershwin (1987). This method is based on a decomposition of the line into a set of  $K-1$  two machine lines  $L(i)$ , for  $i=1, \dots, K-1$ . Line  $L(i)$  is composed of an upstream machine  $M_u(i)$ , and a downstream machine  $M_d(i)$  and buffer  $B(i)$  which has the same capacity as buffer  $B_i$  in line  $L$ ,  $N_i$ . Machines  $M_u(i)$  and  $M_d(i)$  are defined by their failure, repair and processing rates  $p_u(i)$ ,  $r_u(i)$ ,  $\mu_u(i)$  and  $p_d(i)$ ,  $r_d(i)$ ,  $\mu_d(i)$ , respectively. The purpose of the method is to determine these parameters so that the behavior of the material flow in buffer  $B(i)$  in line  $L(i)$  closely matches that of the flow in buffer  $B_i$  of line  $L$ . This decomposition is previously illustrated in Figure 2.4 in Chapter 2.

For each two-machine line  $L(i)$ , for  $i=1, \dots, K-1$ , we define:

$E(i)$ : efficiency (production rate) of line  $L(i)$ ;

$p_s(i)$ : probability of buffer  $B(i)$  being empty in line  $L(i)$ ;

$p_b(i)$ : probability of buffer  $B(i)$  being full in line  $L(i)$ .

These quantities are functions of the four unknown parameters  $p_u(i)$ ,  $r_u(i)$ ,  $p_d(i)$ ,  $r_d(i)$ . Since the DDX algorithm is designed for homogeneous lines and it is assumed that all machines have one unit processing rates, the unknown parameters involve only failure and repair rates. The exact solution of the two-machine line derived in Gershwin and Shick (1983) can be used to calculate these quantities. The required formulas can be found in Gershwin (1987).

In order to determine the unknown parameters of each line  $L(i)$  Gershwin(1987) establish the following set of equations:

$$E(1) = E(2) = \dots = E(K-1) \quad (4)$$

$$\frac{p_d(i-1)}{r_d(i-1)} + \frac{p_u(i)}{r_u(i)} = \frac{1}{E(i)} + \frac{1}{e_i} - 2 \text{ for } i = 2, \dots, K-1 \quad (5)$$

$$r_u(i) = Zr_u(i-1) + (1-Z)r_i \text{ for } i = 2, \dots, K-1 \quad (6)$$

where  $Z = \frac{r_u(i)p_s(i-1)}{p_u(i)E(i)}$

$$r_d(i-1) = Tr_d(i) + (1-T)r_i \text{ for } i = 2, \dots, K-1 \quad (7)$$

where  $T = \frac{r_d(i-1)p_b(i)}{p_d(i-1)E(i-1)}$

$$\begin{cases} p_u(1) = p_1 \\ p_d(K-1) = p_K \\ r_u(1) = r_1 \\ r_d(K-1) = r_K \end{cases} \quad (8)$$

Equation (4) is related to the conservation of flow. Because there is no mechanism for the creation or destruction of material flow is conserved. Equation (5) is called

the flow rate idle time equation which is based on the assumption that the probability of a station being simultaneously blocked and starved is zero. Equation (6) and equation (7) show the relationship between repair probabilities in neighboring two machine lines and in the original line. Equation (6) is obtained by stating that a failure of machine  $M_u(i)$  represents either a failure of machine  $M_i$ , or a starvation of machine  $M_i$  due to a failure of one of the upstream machines ( $M_{i-1}, M_{i-2}, \dots$ ), which is represented by a failure of machine  $M_u(i-1)$ . Equation (7) can similarly be obtained by considering the failure of machine  $M_d(i-1)$ . Finally equation (8) denotes the boundary conditions.

There are a total of  $4(K-1)$  equations among (4), (5), (6), (7), and (8) in  $4(K-1)$  unknowns:  $p_u(i), r_u(i), p_d(i), r_d(i)$ . Gershwin used an iterative procedure to solve this set of equations. The algorithm, which consists of three loops, is fairly complicated. Because of the complexity and some numerical problems of Gershwin's algorithm Dallery *et al.* (1988) propose a new algorithm to solve these equations efficiently. The idea is replace to previous set of equations by an equivalent one. Using equation (4) and replacing  $E(i)$  by  $E(i-1)$  in equation (5) and in expression (6) and similarly, replacing  $E(i-1)$  by  $E(i)$  in expression(7) the followings are obtained:

$$\frac{p_d(i-1)}{r_d(i-1)} + \frac{p_u(i)}{r_u(i)} = \frac{1}{E(i-1)} + \frac{1}{e_i} - 2 \text{ for } i = 2, \dots, K-1 \quad (9)$$

$$Z = \frac{r_u(i)p_s(i-1)}{p_u(i)E(i-1)} \quad (10)$$

$$T = \frac{r_d(i-1)p_b(i)}{p_d(i-1)E(i)} \quad (11)$$

Dallery *et al.* (1988) defined following quantities:

$$\begin{cases} I_u(i) = \frac{p_u(i)}{r_u(i)} \\ I_d(i) = \frac{p_d(i)}{r_d(i)} \end{cases} \text{ for } i = 1, \dots, K-1 \quad (12)$$

Using these quantities and expression (10), equations (9) and (6) may be written as:

$$I_u(i) = \frac{1}{E(i-1)} + \frac{1}{e_i} - I_d(i-1) - 2 \text{ for } i = 2, \dots, K-1 \quad (13)$$

$$r_u(i) = Zr_u(i-1) + (1-Z)r_i \text{ for } i = 2, \dots, K-1 \quad (14)$$

where  $Z = \frac{p_s(i-1)}{I_u(i)E(i-1)}$

Similarly, using expression (11) and by shifting  $i-1$  to  $i$ , equations (5) and (7) can be expressed as:

$$I_d(i) = \frac{1}{E(i+1)} + \frac{1}{e_{i+1}} - I_u(i+1) - 2 \text{ for } i = 1, \dots, K-2 \quad (15)$$

$$r_d(i) = Tr_d(i+1) + (1-T)r_{i+1} \text{ for } i = 1, \dots, K-2 \quad (16)$$

where  $T = \frac{p_b(i+1)}{I_d(i)E(i+1)}$

The parameters of upstream machine of line  $L(i)$ ,  $I_u(i)$ ,  $r_u(i)$ , and therefore  $p_u(i)$  can be obtained from the parameters of line  $L(i-1)$  by means of equations (13) and (14). Similarly, the parameters of the downstream machine of line  $L(i)$ ,  $I_d(i)$ ,  $r_d(i)$ , and therefore  $p_d(i)$ , can be obtained from the parameters of line  $L(i+1)$  by means of equations (15) and (16).  $E(i-1)$  and  $p_s(i-1)$  and  $E(i+1)$  and  $p_b(i+1)$  can be derived from the parameters of lines  $L(i-1)$  and  $L(i+1)$  respectively using the formulas given in Table 1. This new formulation leads to the following algorithm which iteratively calculates the unknown parameters, i.e.  $p_u(i)$ ,  $r_u(i)$ ,  $p_d(i)$ ,  $r_d(i)$ . The algorithm is given in Table A1.



Table A1 DDX Algorithm

**Initialization**

$$\begin{cases} p_d(i) = p_{i+1} & i=1, \dots, K-1 \\ r_d(i) = r_{i+1} & i=1, \dots, K-1 \\ p_u(i) = p_1 \\ r_u(i) = r_1 \end{cases}$$

**Step 1:**

For  $i=2, 3, \dots, K-1$ , calculate  $I_u(i)$ ,  $r_u(i)$ , and  $p_u(i)$  using equations (13), (14), and (12).

**Step 2:**

For  $i=K-2, K-3, \dots, 1$ , calculate  $I_d(i)$ ,  $r_d(i)$ , and  $p_d(i)$  using equations (15), (16), and (12).

Go to step 1 until convergence.

**A3. ADXX Algorithm**

Burman (1995) extends to DDX algorithm for non-homogenous production lines where the machines may have different processing times. Burman (1995) presents closed form solution for unknown parameters, i.e.  $p_u(i)$ ,  $r_u(i)$ ,  $\mu_u(i)$  and  $p_d(i)$ ,  $r_d(i)$ ,  $\mu_d(i)$ .

The equations for these unknown parameters are given as follows:

$$p_u(i) = \frac{p_i K_2 K_3 + r_i p_i + r_i K_1 K_3}{r_i + K_2 K_3 - K_1 K_3} \quad (17)$$

$$r_u(i) = \frac{p_i K_2 K_3 + r_i p_i + r_i K_1 K_3}{p_i + K_1 K_3 - K_2 K_3} \quad (18)$$

$$\mu_u(i) = \frac{K_3(p_i + r_i)}{r_i + K_2 K_3 - K_1 K_3} \quad (19)$$

where

$$K_1 = p_i \left( \frac{P_{i-1}(0,1,1)}{P(i-1)} \left( \frac{\mu_u(i-1)}{\mu_d(i-1)} - 1 \right) \right) + \left( \frac{P_{i-1}(0,0,1)}{P(i-1)} \right) r_u(i-1) \quad (20)$$

$$K_2 = (r_u(i-1) - r_i) \left( \frac{p_{i-1}(0,0,1)}{P(i-1)} \right) \quad (21)$$

$$K_3 = \frac{1}{\frac{1}{P(i-1)} + \frac{1}{e_i \mu_i} - \frac{1}{e_d(i-1) \mu_d(i-1)}} \quad (22)$$

A similar procedure can be used to generate:

$$p_d(i) = \frac{p_{i+1} K_5 K_6 + r_{i+1} p_{i+1} + r_{i+1} K_4 K_6}{r_{i+1} + K_5 K_6 - K_4 K_6} \quad (23)$$

$$r_d(i) = \frac{p_{i+1} K_5 K_6 + r_{i+1} p_{i+1} + r_{i+1} K_4 K_6}{p_{i+1} + K_4 K_6 - K_5 K_6} \quad (24)$$

$$\mu_d(i) = \frac{K_6 (p_{i+1} + r_{i+1})}{r_{i+1} + K_5 K_6 - K_4 K_6} \quad (25)$$

where

$$K_4 = p_{i+1} \left( \frac{p_{i+1}(N_{i+1}, 1, 1)}{P(i+1)} \left( \frac{\mu_d(i+1)}{\mu_u(i+1)} - 1 \right) \right) + \left( \frac{p_{i+1}(N_{i+1}, 1, 0)}{P(i+1)} \right) r_d(i+1) \quad (26)$$

$$K_5 = (r_d(i+1) - r_{i+1}) \left( \frac{p_{i+1}(N_{i+1}, 1, 0)}{P(i+1)} \right) \quad (27)$$

$$K_6 = \frac{1}{\frac{1}{P(i+1)} + \frac{1}{e_{i+1} \mu_{i+1}} - \frac{1}{e_u(i+1) \mu_u(i+1)}} \quad (28)$$

In these formulas  $p(n, \alpha_1, \alpha_2)$  is defined as the probability that there are  $n$  parts in the buffer and that  $M_i$  is in state  $\alpha_i$  where

$$\alpha_i = \begin{cases} 0 & \text{if machine } i \text{ is under repair} \\ 1 & \text{if machine } i \text{ is operational} \end{cases} \quad i = 1, \dots, K$$

Together (17) - (28) form a new set of decomposition equations. The ADDX algorithm for solving these equations is given in Table A2.

Table A2 ADDX Algorithm

<p><b>Initialization</b></p> $p_u(i) = p_i$ $r_u(i) = r_i$ $\mu_u(i) = \mu_i$ $p_d(i) = p_{i+1}$ $r_d(i) = r_{i+1}$ $\mu_d(i) = \mu_{i+1}$ <p style="text-align: center;"><math>i=1, \dots, K-1</math></p> <p><b>Step 1:</b></p> <p>For <math>i=2, 3, \dots, K-1</math>, calculate <math>p_u(i)</math>, <math>r_u(i)</math>, <math>\mu_u(i)</math> using equations (17), (18), and (19).</p> <p><b>Step 2:</b></p> <p>For <math>i=K-2, K-3, \dots, 1</math>, calculate <math>p_d(i)</math>, <math>r_d(i)</math>, <math>\mu_d(i)</math> using equations (23), (24), and (25).</p> <p>Go to step 1 until convergence.</p>
---

As it is stated by Burman (1995), the ADDX algorithm is faster than the original DDX algorithm by as much as ten times and has a reliability of convergence of nearly 100%.

**APPENDIX B**  
**DETAILED RESULTS FOR CHAPTER 5**

Appendix B1 Results of experimental studies for the problem set 5.25

Problem Setting	Instance No	Throughput Rate			Deviation(%)		CPU (sec.)		
		CE	BTS	ATS	BTS	ATS	CE	BTS	ATS
1	1	0.07327	<b>0.07327</b>	<b>0.07327</b>	0.00	0.00	0.01	0.14	0.34
	2	0.07495	<b>0.07495</b>	<b>0.07495</b>	0.00	0.00	0.00	0.06	0.20
	3	0.07952	<b>0.07952</b>	<b>0.07952</b>	0.00	0.00	0.01	0.09	0.27
	4	0.07952	<b>0.07952</b>	<b>0.07952</b>	0.00	0.00	0.00	0.27	0.14
	5	0.07338	<b>0.07338</b>	<b>0.07338</b>	0.00	0.00	0.00	0.13	0.17
	6	0.08682	<b>0.08682</b>	<b>0.08682</b>	0.00	0.00	0.01	0.14	0.31
	7	0.06803	<b>0.06803</b>	<b>0.06803</b>	0.00	0.00	0.01	0.09	0.20
	8	0.09560	<b>0.09560</b>	<b>0.09560</b>	0.00	0.00	0.00	0.11	0.17
	9	0.06901	<b>0.06901</b>	<b>0.06901</b>	0.00	0.00	0.01	0.13	0.28
	10	0.08674	<b>0.08674</b>	<b>0.08674</b>	0.00	0.00	0.00	0.08	0.00
	Avg	<b>0.07868</b>	<b>0.07868</b>	<b>0.07868</b>	<b>0.00</b>	<b>0.00</b>	<b>0.01</b>	<b>0.12</b>	<b>0.21</b>
2	1	0.06404	<b>0.06404</b>	<b>0.06404</b>	0.00	0.00	0.03	0.16	0.20
	2	0.05846	<b>0.05846</b>	<b>0.05846</b>	0.00	0.00	0.01	0.11	0.25
	3	0.07958	0.07883	<b>0.07958</b>	0.95	0.00	0.02	0.50	0.34
	4	0.06952	<b>0.06952</b>	<b>0.06952</b>	0.00	0.00	0.06	0.72	1.17
	5	0.06355	<b>0.06355</b>	<b>0.06355</b>	0.00	0.00	0.02	0.33	0.39
	6	0.07527	0.07470	<b>0.07527</b>	0.76	0.00	0.05	0.72	1.20
	7	0.06923	0.06551	<b>0.06923</b>	5.37	0.00	0.02	0.30	0.50
	8	0.08164	<b>0.08164</b>	<b>0.08164</b>	0.00	0.00	0.02	0.37	0.70
	9	0.07360	<b>0.07360</b>	<b>0.07360</b>	0.00	0.00	0.05	1.61	1.45
	10	0.07570	<b>0.07570</b>	<b>0.07570</b>	0.00	0.00	0.00	0.14	0.31
	Avg	<b>0.07106</b>	<b>0.07055</b>	<b>0.07106</b>	<b>0.71</b>	<b>0.00</b>	<b>0.03</b>	<b>0.50</b>	<b>0.65</b>
3	1	0.07297	<b>0.07297</b>	<b>0.07297</b>	0.00	0.00	0.01	0.16	0.39
	2	0.06543	0.06540	<b>0.06543</b>	0.06	0.00	0.08	1.84	2.53
	3	0.07763	<b>0.07763</b>	<b>0.07763</b>	0.00	0.00	0.01	0.16	0.30
	4	0.06032	0.06024	<b>0.06032</b>	0.15	0.00	0.03	0.44	0.30
	5	0.07235	<b>0.07235</b>	<b>0.07235</b>	0.00	0.00	0.03	0.11	0.31
	6	0.09005	<b>0.09005</b>	<b>0.09005</b>	0.00	0.00	0.03	0.30	0.37
	7	0.07852	0.06750	<b>0.07852</b>	14.03	0.00	0.01	0.12	0.23
	8	0.09482	<b>0.09482</b>	<b>0.09482</b>	0.00	0.00	0.03	0.55	0.37
	9	0.08473	<b>0.08473</b>	<b>0.08473</b>	0.00	0.00	0.03	0.41	0.59
	10	0.07297	<b>0.07297</b>	<b>0.07297</b>	0.00	0.00	0.01	0.11	0.30
	Avg	<b>0.07698</b>	<b>0.07586</b>	<b>0.07698</b>	<b>1.42</b>	<b>0.00</b>	<b>0.03</b>	<b>0.42</b>	<b>0.57</b>
4	1	0.05930	<b>0.05930</b>	<b>0.05930</b>	0.00	0.00	0.03	0.24	0.44
	2	0.04374	<b>0.04374</b>	<b>0.04374</b>	0.00	0.00	0.05	0.36	0.41
	3	0.05317	<b>0.05317</b>	<b>0.05317</b>	0.00	0.00	0.02	0.17	0.22
	4	0.06064	0.05599	<b>0.06064</b>	7.67	0.00	0.05	0.78	1.23
	5	0.05273	<b>0.05273</b>	<b>0.05273</b>	0.00	0.00	0.03	0.20	0.27
	6	0.06283	<b>0.06283</b>	<b>0.06283</b>	0.00	0.00	0.03	0.58	1.03
	7	0.06209	0.05326	<b>0.06209</b>	14.22	0.00	0.03	0.41	0.36
	8	0.06773	<b>0.06773</b>	<b>0.06773</b>	0.00	0.00	0.01	0.14	0.33
	9	0.06215	<b>0.06215</b>	<b>0.06215</b>	0.00	0.00	0.02	0.19	0.39
	10	0.06180	<b>0.06180</b>	<b>0.06180</b>	0.00	0.00	0.03	0.16	0.30
	Avg	<b>0.05862</b>	<b>0.05727</b>	<b>0.05862</b>	<b>2.19</b>	<b>0.00</b>	<b>0.03</b>	<b>0.32</b>	<b>0.50</b>

Appendix B1 Results of experimental studies for the problem set 5.25 (cont.)

Problem Setting	Instance No	Throughput Rate			Deviation(%)		CPU (sec.)		
		CE	BTS	ATS	BTS	ATS	CE	BTS	ATS
5	1	0.02619	<b>0.02619</b>	<b>0.02619</b>	0.00	0.00	0.01	0.11	0.30
	2	0.02252	<b>0.02252</b>	<b>0.02252</b>	0.00	0.00	0.00	0.05	0.08
	3	0.02843	<b>0.02843</b>	<b>0.02843</b>	0.00	0.00	0.01	0.17	0.00
	4	0.02770	<b>0.02770</b>	<b>0.02770</b>	0.00	0.00	0.00	0.08	0.16
	5	0.02454	<b>0.02454</b>	<b>0.02454</b>	0.00	0.00	0.01	0.17	0.23
	6	0.03676	<b>0.03676</b>	<b>0.03676</b>	0.00	0.00	0.03	0.23	0.51
	7	0.02362	<b>0.02362</b>	<b>0.02362</b>	0.00	0.00	0.01	0.20	0.27
	8	0.03713	<b>0.03713</b>	<b>0.03713</b>	0.00	0.00	0.00	0.08	0.11
	9	0.03085	<b>0.03085</b>	<b>0.03085</b>	0.00	0.00	0.03	0.25	0.02
	10	0.03369	<b>0.03369</b>	<b>0.03369</b>	0.00	0.00	0.01	0.05	0.11
		<b>Avg</b>	<b>0.02914</b>	<b>0.02914</b>	<b>0.02914</b>	<b>0.00</b>	<b>0.00</b>	<b>0.01</b>	<b>0.14</b>
6	1	0.02831	<b>0.02831</b>	<b>0.02831</b>	0.00	0.00	0.03	0.39	0.59
	2	0.02256	<b>0.02256</b>	<b>0.02256</b>	0.00	0.00	0.03	0.27	0.53
	3	0.02540	<b>0.02540</b>	<b>0.02540</b>	0.00	0.00	0.09	1.22	2.17
	4	0.02375	<b>0.02375</b>	<b>0.02375</b>	0.00	0.00	0.02	0.27	0.23
	5	0.02508	<b>0.02508</b>	<b>0.02508</b>	0.00	0.00	0.01	0.09	0.20
	6	0.01966	<b>0.01966</b>	<b>0.01966</b>	0.00	0.00	0.06	0.34	2.06
	7	0.02145	<b>0.02145</b>	<b>0.02145</b>	0.00	0.00	0.01	0.09	0.25
	8	0.03383	<b>0.03383</b>	<b>0.03383</b>	0.00	0.00	0.06	0.58	1.44
	9	0.02838	<b>0.02838</b>	<b>0.02838</b>	0.00	0.00	0.02	0.09	0.27
	10	0.01800	<b>0.01800</b>	<b>0.01800</b>	0.00	0.00	0.01	0.08	0.20
		<b>Avg</b>	<b>0.02464</b>	<b>0.02464</b>	<b>0.02464</b>	<b>0.00</b>	<b>0.00</b>	<b>0.03</b>	<b>0.34</b>
7	1	0.02616	<b>0.02616</b>	<b>0.02616</b>	0.00	0.00	0.03	0.11	0.27
	2	0.02221	<b>0.02221</b>	<b>0.02221</b>	0.00	0.00	0.02	0.09	0.30
	3	0.02844	<b>0.02844</b>	<b>0.02844</b>	0.00	0.00	0.01	0.08	0.22
	4	0.02767	<b>0.02767</b>	<b>0.02767</b>	0.00	0.00	0.01	0.11	0.00
	5	0.02593	<b>0.02593</b>	<b>0.02593</b>	0.00	0.00	0.01	0.16	0.33
	6	0.03342	<b>0.03342</b>	<b>0.03342</b>	0.00	0.00	0.05	0.25	0.37
	7	0.02349	<b>0.02349</b>	<b>0.02349</b>	0.00	0.00	0.02	0.11	0.23
	8	0.03719	<b>0.03719</b>	<b>0.03718</b>	0.00	0.01	0.03	0.39	1.01
	9	0.03101	<b>0.03101</b>	<b>0.03101</b>	0.00	0.01	0.00	0.08	0.17
	10	0.03369	<b>0.03369</b>	<b>0.03369</b>	0.00	0.00	0.00	0.05	0.08
		<b>Avg</b>	<b>0.02892</b>	<b>0.02892</b>	<b>0.02892</b>	<b>0.00</b>	<b>0.00</b>	<b>0.02</b>	<b>0.14</b>
8	1	0.02349	<b>0.02349</b>	<b>0.02349</b>	0.00	0.00	0.01	0.23	0.42
	2	0.01835	<b>0.01835</b>	<b>0.01835</b>	0.00	0.00	0.05	0.28	0.59
	3	0.02401	<b>0.02401</b>	<b>0.02401</b>	0.00	0.00	0.02	0.31	0.23
	4	0.02235	<b>0.02235</b>	<b>0.02235</b>	0.00	0.00	0.05	0.94	0.58
	5	0.02461	<b>0.02461</b>	<b>0.02461</b>	0.00	0.00	0.00	0.09	0.25
	6	0.02256	<b>0.02256</b>	<b>0.02256</b>	0.00	0.00	0.01	0.47	0.50
	7	0.01986	<b>0.01986</b>	<b>0.01986</b>	0.00	0.00	0.02	0.28	0.33
	8	0.02757	<b>0.02757</b>	<b>0.02757</b>	0.00	0.00	0.05	0.41	0.39
	9	0.02751	<b>0.02751</b>	<b>0.02751</b>	0.00	0.00	0.03	0.34	0.50
	10	0.01864	<b>0.01864</b>	<b>0.01864</b>	0.00	0.00	0.01	0.27	0.33
		<b>Avg</b>	<b>0.02290</b>	<b>0.02290</b>	<b>0.02290</b>	<b>0.00</b>	<b>0.00</b>	<b>0.03</b>	<b>0.36</b>

Appendix B2 Results of experimental studies for the problem set 5.50

Problem Setting	Instance No	Throughput Rate			Deviation(%)		CPU (sec.)		
		CE	BTS	ATS	BTS	ATS	CE	BTS	ATS
1	1	0.08200	<b>0.08200</b>	<b>0.08200</b>	0.00	0.00	0.09	0.53	0.41
	2	0.07495	<b>0.07495</b>	<b>0.07495</b>	0.00	0.00	0.05	0.22	0.51
	3	0.07952	<b>0.07952</b>	<b>0.07952</b>	0.00	0.00	0.08	0.36	0.66
	4	0.07952	<b>0.07952</b>	<b>0.07952</b>	0.00	0.00	0.05	0.55	0.19
	5	0.07338	<b>0.07338</b>	<b>0.07338</b>	0.00	0.00	0.08	0.33	0.67
	6	0.08682	<b>0.08682</b>	<b>0.08682</b>	0.00	0.00	0.08	0.14	0.73
	7	0.06803	<b>0.06803</b>	<b>0.06803</b>	0.00	0.00	0.09	0.53	0.81
	8	0.09560	<b>0.09560</b>	<b>0.09560</b>	0.00	0.00	0.06	0.80	0.64
	9	0.06901	<b>0.06901</b>	<b>0.06901</b>	0.00	0.00	0.08	0.45	0.76
	10	0.08674	<b>0.08674</b>	<b>0.08674</b>	0.00	0.00	0.02	0.16	0.13
	Avg	<b>0.07956</b>	<b>0.07956</b>	<b>0.07956</b>	<b>0.00</b>	<b>0.00</b>	<b>0.07</b>	<b>0.41</b>	<b>0.55</b>
2	1	0.06418	<b>0.06418</b>	<b>0.06418</b>	0.00	0.00	0.11	0.28	0.53
	2	0.05947	<b>0.05947</b>	<b>0.05947</b>	0.00	0.00	0.09	0.19	0.73
	3	0.08163	0.08084	<b>0.08163</b>	0.97	0.00	0.14	0.72	0.80
	4	0.06958	<b>0.06958</b>	<b>0.06958</b>	0.00	0.00	0.20	0.70	1.09
	5	0.06415	<b>0.06415</b>	<b>0.06415</b>	0.00	0.00	0.11	0.31	0.48
	6	0.07641	0.07572	<b>0.07641</b>	0.91	0.00	0.30	1.33	1.15
	7	0.06947	0.06710	<b>0.06947</b>	3.41	0.00	0.14	0.33	1.08
	8	0.08303	<b>0.08303</b>	<b>0.08303</b>	0.00	0.00	0.14	0.48	1.11
	9	0.07537	<b>0.07537</b>	<b>0.07537</b>	0.00	0.00	0.61	0.58	5.82
	10	0.07575	<b>0.07575</b>	<b>0.07575</b>	0.00	0.00	0.05	0.37	0.61
	Avg	<b>0.07190</b>	<b>0.07152</b>	<b>0.07190</b>	<b>0.53</b>	<b>0.00</b>	<b>0.19</b>	<b>0.53</b>	<b>1.34</b>
3	1	0.07327	<b>0.07327</b>	<b>0.07327</b>	0.00	0.00	0.11	0.33	0.80
	2	0.06719	0.06695	<b>0.06719</b>	0.36	0.00	0.67	5.66	4.26
	3	0.07893	<b>0.07893</b>	<b>0.07893</b>	0.00	0.00	0.09	0.25	0.86
	4	0.06109	0.06107	<b>0.06109</b>	0.04	0.00	0.17	1.08	1.00
	5	0.07309	<b>0.07309</b>	<b>0.07309</b>	0.00	0.00	0.09	0.51	0.76
	6	0.09384	0.09257	<b>0.09384</b>	1.36	0.00	0.16	0.58	1.29
	7	0.06772	<b>0.06772</b>	<b>0.06772</b>	0.00	0.01	0.11	0.23	0.66
	8	0.09529	<b>0.09529</b>	<b>0.09529</b>	0.00	0.00	0.14	0.28	0.92
	9	0.08591	<b>0.08591</b>	<b>0.08591</b>	0.00	0.00	0.25	1.03	1.06
	10	0.08395	<b>0.08395</b>	<b>0.08395</b>	0.00	0.00	0.12	0.25	0.94
	Avg	<b>0.07803</b>	<b>0.07787</b>	<b>0.07803</b>	<b>0.18</b>	<b>0.00</b>	<b>0.19</b>	<b>1.02</b>	<b>1.26</b>
4	1	0.07486	0.07201	<b>0.07486</b>	3.81	0.00	0.16	0.84	0.55
	2	0.04864	0.04708	<b>0.04864</b>	3.20	0.00	0.23	1.00	1.28
	3	0.05754	<b>0.05754</b>	<b>0.05754</b>	0.00	0.00	0.12	0.39	0.61
	4	0.06345	0.06228	<b>0.06345</b>	1.85	0.00	0.33	1.36	1.69
	5	0.05682	<b>0.05682</b>	<b>0.05682</b>	0.00	0.00	0.13	0.28	1.04
	6	0.08785	0.08033	<b>0.08785</b>	8.56	0.00	0.28	0.89	1.33
	7	0.06440	0.05626	<b>0.06440</b>	12.65	0.00	0.16	0.98	1.09
	8	0.07299	<b>0.07299</b>	<b>0.07299</b>	0.00	0.00	0.16	0.34	0.97
	9	0.06396	<b>0.06396</b>	<b>0.06396</b>	0.00	0.00	0.14	0.28	1.12
	10	0.06735	0.06176	<b>0.06735</b>	8.30	0.00	0.14	0.48	1.00
	Avg	<b>0.06579</b>	<b>0.06310</b>	<b>0.06579</b>	<b>3.84</b>	<b>0.00</b>	<b>0.19</b>	<b>0.68</b>	<b>1.07</b>

Appendix B2 Results of experimental studies for the problem set 5.50 (cont.)

Problem Setting	Instance No	Throughput Rate			Deviation(%)		CPU (sec.)		
		CE	BTS	ATS	BTS	ATS	CE	BTS	ATS
5	1	0.02619	<b>0.02619</b>	<b>0.02619</b>	0.00	0.00	0.09	0.38	0.72
	2	0.02252	<b>0.02252</b>	<b>0.02252</b>	0.00	0.00	0.03	0.09	0.14
	3	0.02843	<b>0.02843</b>	<b>0.02843</b>	0.00	0.00	0.06	0.38	0.58
	4	0.02770	<b>0.02770</b>	<b>0.02770</b>	0.00	0.00	0.03	0.05	0.42
	5	0.02454	<b>0.02454</b>	<b>0.02454</b>	0.00	0.00	0.09	0.45	0.62
	6	0.03676	<b>0.03676</b>	<b>0.03676</b>	0.00	0.00	0.25	0.41	1.16
	7	0.02901	<b>0.02901</b>	<b>0.02901</b>	0.00	0.00	0.14	0.55	0.41
	8	0.03713	<b>0.03713</b>	<b>0.03713</b>	0.00	0.00	0.03	0.13	0.67
	9	0.03085	<b>0.03085</b>	<b>0.03085</b>	0.00	0.00	0.09	0.45	0.84
	10	0.03369	<b>0.03369</b>	<b>0.03369</b>	0.00	0.00	0.01	0.11	0.42
		<b>Avg</b>	<b>0.02968</b>	<b>0.02968</b>	<b>0.02968</b>	<b>0.00</b>	<b>0.00</b>	<b>0.08</b>	<b>0.30</b>
6	1	0.02775	0.02732	<b>0.02775</b>	1.55	0.00	0.14	0.47	0.59
	2	0.02256	<b>0.02256</b>	<b>0.02256</b>	0.00	0.00	0.20	0.34	0.92
	3	0.02540	<b>0.02540</b>	<b>0.02540</b>	0.00	0.00	0.37	6.14	8.46
	4	0.02375	<b>0.02375</b>	<b>0.02375</b>	0.00	0.00	0.05	0.06	0.58
	5	0.02510	<b>0.02510</b>	<b>0.02510</b>	0.00	0.00	0.06	0.19	0.53
	6	0.01966	<b>0.01966</b>	<b>0.01966</b>	0.00	0.00	0.30	1.45	1.51
	7	0.02145	<b>0.02145</b>	<b>0.02145</b>	0.00	0.00	0.08	0.27	0.69
	8	0.03383	<b>0.03383</b>	<b>0.03383</b>	0.00	0.00	0.28	0.58	1.79
	9	0.02855	<b>0.02855</b>	<b>0.02855</b>	0.00	0.00	0.08	0.36	0.64
	10	0.01800	<b>0.01800</b>	<b>0.01800</b>	0.00	0.00	0.05	0.13	0.59
		<b>Avg</b>	<b>0.02460</b>	<b>0.02456</b>	<b>0.02460</b>	<b>0.15</b>	<b>0.00</b>	<b>0.16</b>	<b>1.00</b>
7	1	0.02618	<b>0.02618</b>	<b>0.02618</b>	0.00	0.00	0.11	0.45	0.86
	2	0.02225	<b>0.02225</b>	<b>0.02225</b>	0.01	0.00	0.09	0.19	0.45
	3	0.02844	<b>0.02844</b>	<b>0.02844</b>	0.00	0.00	0.08	0.16	0.58
	4	0.02767	<b>0.02767</b>	<b>0.02767</b>	0.00	0.00	0.05	0.34	0.78
	5	0.02593	<b>0.02593</b>	<b>0.02593</b>	0.00	0.00	0.09	0.30	0.76
	6	0.03613	0.03332	<b>0.03613</b>	7.77	0.00	0.20	1.48	1.15
	7	0.02350	<b>0.02350</b>	<b>0.02350</b>	0.00	0.00	0.08	0.19	0.70
	8	0.03723	<b>0.03723</b>	<b>0.03723</b>	0.00	0.00	0.09	0.86	0.55
	9	0.03102	<b>0.03102</b>	<b>0.03102</b>	0.00	0.00	0.05	0.09	0.58
	10	0.03387	<b>0.03387</b>	<b>0.03387</b>	0.00	0.00	0.05	0.11	0.44
		<b>Avg</b>	<b>0.02922</b>	<b>0.02894</b>	<b>0.02922</b>	<b>0.78</b>	<b>0.00</b>	<b>0.09</b>	<b>0.42</b>
8	1	0.02526	0.02445	<b>0.02526</b>	3.18	0.00	0.17	1.03	0.97
	2	0.01892	<b>0.01892</b>	<b>0.01892</b>	0.00	0.00	0.22	0.52	1.01
	3	0.02517	<b>0.02517</b>	<b>0.02517</b>	0.00	0.00	0.11	0.33	1.08
	4	0.02349	<b>0.02349</b>	<b>0.02349</b>	0.00	0.00	0.25	1.40	2.12
	5	0.02500	0.02497	<b>0.02500</b>	0.14	0.00	0.08	0.11	0.72
	6	0.02490	<b>0.02490</b>	<b>0.02490</b>	0.00	0.00	0.19	0.69	1.42
	7	0.02066	<b>0.02066</b>	<b>0.02066</b>	0.00	0.00	0.09	0.34	0.47
	8	0.03164	<b>0.03164</b>	<b>0.03164</b>	0.00	0.00	0.23	1.04	1.58
	9	0.02751	<b>0.02751</b>	<b>0.02751</b>	0.00	0.00	0.14	0.58	1.11
	10	0.02012	<b>0.02012</b>	<b>0.02012</b>	0.00	0.00	0.08	0.47	0.73
		<b>Avg</b>	<b>0.02427</b>	<b>0.02418</b>	<b>0.02427</b>	<b>0.33</b>	<b>0.00</b>	<b>0.16</b>	<b>0.65</b>



Appendix B3 Results of experimental studies for the problem set 5.100

Problem Setting	Instance No	Throughput Rate			Deviation(%)		CPU (sec.)		
		CE	BTS	ATS	BTS	ATS	CE	BTS	ATS
1	1	0.08200	<b>0.08200</b>	<b>0.08200</b>	0.00	0.00	0.67	0.94	2.31
	2	0.07495	<b>0.07495</b>	<b>0.07495</b>	0.00	0.00	0.28	0.33	1.86
	3	0.07952	<b>0.07952</b>	<b>0.07952</b>	0.00	0.00	0.56	0.77	0.78
	4	0.07952	<b>0.07952</b>	<b>0.07952</b>	0.00	0.00	0.20	1.09	0.51
	5	0.07338	<b>0.07338</b>	<b>0.07338</b>	0.00	0.00	0.55	0.77	0.89
	6	0.08682	<b>0.08682</b>	<b>0.08682</b>	0.00	0.00	0.53	0.86	1.87
	7	0.06803	<b>0.06803</b>	<b>0.06803</b>	0.00	0.00	0.66	0.91	2.34
	8	0.09560	<b>0.09560</b>	<b>0.09560</b>	0.00	0.00	0.50	1.25	1.95
	9	0.06901	<b>0.06901</b>	<b>0.06901</b>	0.00	0.00	0.66	0.86	2.03
	10	0.08674	<b>0.08674</b>	<b>0.08674</b>	0.00	0.00	0.16	0.31	0.44
		<b>Avg</b>	<b>0.07956</b>	<b>0.07956</b>	<b>0.07956</b>	<b>0.00</b>	<b>0.00</b>	<b>0.48</b>	<b>0.81</b>
2	1	0.07352	0.06419	<b>0.07352</b>	12.70	0.00	0.76	0.42	1.84
	2	0.05961	<b>0.05961</b>	<b>0.05961</b>	0.00	0.00	0.48	0.28	1.34
	3	0.08246	0.06780	<b>0.08246</b>	17.78	0.00	0.83	0.59	2.57
	4	0.06959	<b>0.06959</b>	<b>0.06959</b>	0.00	0.00	0.78	0.55	2.56
	5	0.06419	<b>0.06419</b>	<b>0.06419</b>	0.00	0.00	0.67	0.36	2.25
	6	0.07681	0.07590	0.07590	1.18	1.18	1.65	1.79	2.00
	7	0.06749	0.05961	<b>0.06749</b>	11.67	0.00	0.89	0.48	2.56
	8	0.08318	<b>0.08318</b>	<b>0.08318</b>	0.00	0.00	0.66	0.52	2.62
	9	0.07591	<b>0.07591</b>	<b>0.07591</b>	0.00	0.00	10.87	12.92	4.27
	10	0.07575	<b>0.07575</b>	<b>0.07575</b>	0.00	0.00	0.28	0.33	0.56
		<b>Avg</b>	<b>0.07285</b>	<b>0.06957</b>	<b>0.07276</b>	<b>4.33</b>	<b>0.12</b>	<b>1.79</b>	<b>1.82</b>
3	1	0.07332	0.07332	<b>0.07332</b>	0.00	0.00	0.80	0.52	1.20
	2	0.06790	0.06783	0.06783	0.10	0.10	6.96	43.24	44.05
	3	0.07923	<b>0.07923</b>	<b>0.07923</b>	0.00	0.00	0.67	0.45	1.95
	4	0.06124	<b>0.06124</b>	<b>0.06124</b>	0.00	0.00	1.03	0.92	2.68
	5	0.07321	<b>0.07321</b>	<b>0.07321</b>	0.01	0.00	0.69	0.45	1.92
	6	0.09485	0.09330	<b>0.09458</b>	1.63	0.28	1.01	1.00	2.62
	7	0.06774	<b>0.06774</b>	<b>0.06774</b>	0.00	0.00	0.80	0.50	2.31
	8	0.09536	<b>0.09536</b>	<b>0.09536</b>	0.00	0.00	0.61	0.38	1.17
	9	0.08653	<b>0.08653</b>	<b>0.08653</b>	0.00	0.00	2.36	1.93	5.88
	10	0.07316	<b>0.07316</b>	<b>0.07316</b>	0.00	0.00	0.81	0.70	2.59
		<b>Avg</b>	<b>0.07725</b>	<b>0.07709</b>	<b>0.07722</b>	<b>0.17</b>	<b>0.04</b>	<b>1.57</b>	<b>5.01</b>
4	1	0.08250	0.07790	0.07808	5.57	5.35	1.08	3.21	2.45
	2	0.05115	<b>0.05115</b>	<b>0.05115</b>	0.00	0.00	1.64	0.95	4.71
	3	0.06230	<b>0.06230</b>	<b>0.06230</b>	0.00	0.00	1.03	0.73	3.06
	4	0.06809	<b>0.06809</b>	<b>0.06809</b>	0.00	0.00	2.67	3.42	2.65
	5	0.06075	<b>0.06075</b>	<b>0.06075</b>	0.00	0.00	1.06	1.33	2.95
	6	0.09046	0.08724	0.08761	3.56	3.15	1.92	4.07	3.13
	7	0.06698	0.06691	<b>0.06698</b>	0.11	0.00	1.11	0.86	1.36
	8	0.07840	<b>0.07840</b>	<b>0.07840</b>	0.00	0.00	1.04	0.76	2.79
	9	0.06648	<b>0.06648</b>	<b>0.06648</b>	0.00	0.00	1.04	0.42	2.85
	10	0.07352	0.07032	0.07125	4.35	3.08	1.17	0.86	1.76
		<b>Avg</b>	<b>0.07006</b>	<b>0.06895</b>	<b>0.06911</b>	<b>1.36</b>	<b>1.16</b>	<b>1.38</b>	<b>1.66</b>

Appendix B3 Results of experimental studies for the problem set 5.100 (cont.)

Problem Setting	Instance No	Throughput Rate			Deviation(%)		CPU (sec.)		
		CE	BTS	ATS	BTS	ATS	CE	BTS	ATS
5	1	0.03457	0.02619	<b>0.03457</b>	24.24	0.00	0.67	0.87	0.87
	2	0.02252	<b>0.02252</b>	<b>0.02252</b>	0.00	0.00	0.72	0.94	0.76
	3	0.03475	<b>0.03475</b>	<b>0.03475</b>	0.00	0.00	0.87	1.26	1.01
	4	0.02770	<b>0.02770</b>	<b>0.02770</b>	0.00	0.00	0.31	0.01	0.61
	5	0.03578	0.02454	<b>0.03578</b>	31.41	0.00	0.75	1.25	0.89
	6	0.03676	<b>0.03676</b>	<b>0.03676</b>	0.00	0.00	1.90	1.56	1.65
	7	0.02901	<b>0.02901</b>	<b>0.02901</b>	0.00	0.00	1.14	0.89	1.19
	8	0.03713	<b>0.03713</b>	<b>0.03713</b>	0.00	0.00	0.19	0.73	1.64
	9	0.03085	<b>0.03085</b>	<b>0.03085</b>	0.00	0.00	0.62	0.00	0.83
	10	0.03369	<b>0.03369</b>	<b>0.03369</b>	0.00	0.00	0.14	0.19	0.44
	<b>Avg</b>	<b>0.03228</b>	<b>0.03032</b>	<b>0.03228</b>	<b>5.56</b>	<b>0.00</b>	<b>0.73</b>	<b>0.77</b>	<b>0.99</b>
6	1	0.02908	<b>0.02908</b>	<b>0.02908</b>	0.00	0.00	0.86	0.97	2.61
	2	0.02256	<b>0.02256</b>	<b>0.02256</b>	0.00	0.00	1.34	1.79	3.01
	3	0.02540	<b>0.02540</b>	<b>0.02540</b>	0.00	0.00	1.39	1.22	3.32
	4	0.02375	<b>0.02375</b>	<b>0.02375</b>	0.00	0.00	0.27	0.62	2.20
	5	0.02510	<b>0.02510</b>	<b>0.02510</b>	0.00	0.00	0.42	0.50	1.54
	6	0.01966	<b>0.01966</b>	<b>0.01966</b>	0.00	0.00	1.93	2.15	3.49
	7	0.02862	<b>0.02862</b>	<b>0.02862</b>	0.00	0.00	1.01	0.73	1.01
	8	0.03383	<b>0.03383</b>	<b>0.03383</b>	0.00	0.00	1.34	0.98	3.84
	9	0.02857	<b>0.02857</b>	<b>0.02857</b>	0.00	0.00	0.51	0.58	1.92
	10	0.01800	<b>0.01800</b>	<b>0.01800</b>	0.00	0.00	0.33	0.31	0.61
	<b>Avg</b>	<b>0.02546</b>	<b>0.02546</b>	<b>0.02546</b>	<b>0.00</b>	<b>0.00</b>	<b>0.94</b>	<b>0.99</b>	<b>2.36</b>
7	1	0.02618	<b>0.02618</b>	<b>0.02618</b>	0.00	0.00	0.73	1.08	2.43
	2	0.02225	<b>0.02225</b>	<b>0.02225</b>	0.00	0.00	0.62	0.31	1.26
	3	0.02844	<b>0.02844</b>	<b>0.02844</b>	0.00	0.00	0.55	0.58	0.92
	4	0.02767	<b>0.02767</b>	<b>0.02767</b>	0.00	0.00	0.30	0.89	2.45
	5	0.02593	<b>0.02593</b>	<b>0.02593</b>	0.00	0.00	0.69	0.69	2.70
	6	0.03368	0.03233	<b>0.03368</b>	4.01	0.00	1.25	1.03	3.09
	7	0.02350	<b>0.02350</b>	<b>0.02350</b>	0.00	0.00	0.53	0.66	0.84
	8	0.03723	<b>0.03723</b>	<b>0.03723</b>	0.00	0.00	0.45	1.59	1.28
	9	0.03102	<b>0.03102</b>	<b>0.03102</b>	0.00	0.00	0.25	0.30	1.68
	10	0.03388	<b>0.03388</b>	<b>0.03388</b>	0.00	0.00	0.25	0.19	1.73
	<b>Avg</b>	<b>0.02898</b>	<b>0.02884</b>	<b>0.02898</b>	<b>0.40</b>	<b>0.00</b>	<b>0.56</b>	<b>0.73</b>	<b>1.84</b>
8	1	0.02767	0.02570	<b>0.02767</b>	7.11	0.00	1.40	1.40	3.43
	2	0.01928	<b>0.01928</b>	<b>0.01928</b>	0.00	0.00	1.36	0.83	3.40
	3	0.02578	<b>0.02578</b>	<b>0.02578</b>	0.00	0.00	0.78	1.52	3.63
	4	0.02352	<b>0.02352</b>	<b>0.02352</b>	0.00	0.00	1.29	0.94	2.53
	5	0.02507	<b>0.02507</b>	<b>0.02507</b>	0.00	0.00	0.51	0.44	2.04
	6	0.02707	<b>0.02707</b>	<b>0.02707</b>	0.00	0.00	1.42	1.50	4.23
	7	0.02107	<b>0.02107</b>	<b>0.02107</b>	0.00	0.00	0.62	0.34	0.92
	8	0.03230	0.03171	<b>0.03230</b>	1.81	0.00	1.76	4.35	2.37
	9	0.02786	<b>0.02786</b>	<b>0.02786</b>	0.00	0.00	0.87	0.39	1.78
	10	0.02096	<b>0.02096</b>	<b>0.02096</b>	0.00	0.00	0.56	0.83	1.84
	<b>Avg</b>	<b>0.02506</b>	<b>0.02480</b>	<b>0.02506</b>	<b>0.89</b>	<b>0.00</b>	<b>1.06</b>	<b>1.25</b>	<b>2.62</b>

Appendix B4 Results of experimental studies for the problem set 10.50

Problem Setting	Instance No	Throughput Rate			Deviation(%)		CPU (sec.)		
		CE	BTS	ATS	BTS	ATS	CE	BTS	ATS
1	1	0.08694	<b>0.08694</b>	<b>0.08694</b>	0.00	0.00	7470.40	12.61	18.30
	2	0.07338	<b>0.07338</b>	<b>0.07338</b>	0.00	0.00	17425.47	33.94	54.38
	3	0.06442	<b>0.06442</b>	<b>0.06442</b>	0.00	0.00	9986.93	44.55	12.45
	4	0.07554	<b>0.07554</b>	<b>0.07554</b>	0.00	0.00	6792.48	8.98	20.67
	5	0.07907	<b>0.07907</b>	<b>0.07907</b>	0.00	0.00	4080.16	4.54	12.20
	6	0.07976	0.07832	<b>0.07976</b>	1.80	0.00	13410.86	31.06	16.22
	7	0.07049	<b>0.07049</b>	<b>0.07049</b>	0.00	0.00	11591.44	7.32	14.77
	8	0.06952	<b>0.06952</b>	<b>0.06952</b>	0.00	0.00	5045.41	7.82	7.78
	9	0.07713	0.07500	<b>0.07713</b>	2.77	0.00	17304.16	7.96	33.25
	10	0.06880	<b>0.06880</b>	<b>0.06880</b>	0.00	0.00	8078.02	7.85	12.45
		<b>Avg</b>	<b>0.07450</b>	<b>0.07415</b>	<b>0.07450</b>	<b>0.46</b>	<b>0.00</b>	<b>10118.53</b>	<b>16.66</b>
2	1	0.05603	0.05027	0.05596	10.28	0.12	9309.26	23.48	15.06
	2	0.06253	<b>0.06253</b>	<b>0.06253</b>	0.00	0.01	13022.07	23.78	18.63
	3	0.05223	<b>0.05223</b>	<b>0.05223</b>	0.00	0.00	13255.60	17.29	13.68
	4	0.07191	0.06950	0.07142	3.35	0.68	21165.35	34.54	28.37
	5	0.06572	0.06542	<b>0.06572</b>	0.46	0.00	12597.77	32.23	36.66
	6	0.05925	<b>0.05925</b>	<b>0.05925</b>	0.00	0.00	11265.77	47.12	22.76
	7	0.06188	0.06038	<b>0.06188</b>	2.42	0.00	12243.06	100.76	17.24
	8	0.07324	0.07157	<b>0.07324</b>	2.28	0.00	15954.19	95.57	56.30
	9	0.06897	0.06502	0.06882	5.72	0.21	9645.34	108.16	65.11
	10	0.06608	0.06606	<b>0.06608</b>	0.03	0.00	18106.07	48.18	56.77
		<b>Avg</b>	<b>0.06378</b>	<b>0.06222</b>	<b>0.06371</b>	<b>2.45</b>	<b>0.10</b>	<b>13656.45</b>	<b>53.11</b>
3	1	0.07214	<b>0.07214</b>	<b>0.07214</b>	0.00	0.00	18184.14	38.59	27.54
	2	0.07361	0.07335	0.07351	0.35	0.13	12089.04	13.91	21.47
	3	0.06815	<b>0.06815</b>	<b>0.06815</b>	0.00	0.00	7591.47	22.68	38.27
	4	0.06909	0.06748	0.06888	2.33	0.31	9052.20	21.48	36.13
	5	0.07086	<b>0.07086</b>	<b>0.07086</b>	0.00	0.00	31618.15	49.17	115.89
	6	0.06865	0.06828	<b>0.06865</b>	0.54	0.00	22935.23	113.27	84.68
	7	0.07799	0.07360	0.07714	5.63	1.10	19787.61	40.16	38.48
	8	0.06779	0.06762	<b>0.06779</b>	0.26	0.01	9781.58	67.32	39.67
	9	0.08079	0.06521	0.06521	19.29	19.28	6033.33	9.59	13.09
	10	0.06660	<b>0.06660</b>	<b>0.06660</b>	0.00	0.01	2065.69	6.35	7.80
		<b>Avg</b>	<b>0.07157</b>	<b>0.06933</b>	<b>0.06989</b>	<b>2.84</b>	<b>2.08</b>	<b>15230.31</b>	<b>38.25</b>
4	1	0.05029	0.05021	0.05021	0.17	0.17	6534.82	8.70	11.19
	2	0.00785	<b>0.00785</b>	<b>0.00785</b>	0.00	0.00	8560.73	6.14	6.26
	3	0.04144	0.04044	0.04091	2.42	1.29	11279.95	16.34	22.39
	4	0.05311	0.04662	0.05151	12.22	3.02	10741.37	20.76	37.50
	5	0.04406	0.04304	0.04380	2.32	0.59	11160.58	11.18	20.22
	6	0.05610	0.03711	<b>0.05610</b>	33.86	0.00	7399.05	8.25	14.48
	7	0.04242	0.03991	<b>0.04242</b>	5.92	0.00	8381.59	21.24	17.39
	8	0.03423	0.03376	<b>0.03423</b>	1.36	0.00	9683.85	25.02	16.66
	9	0.03164	0.02557	0.03149	19.18	0.46	7153.07	15.94	15.76
	10	0.05576	0.04616	<b>0.05576</b>	17.22	0.00	10524.99	33.25	18.60
		<b>Avg</b>	<b>0.04169</b>	<b>0.03707</b>	<b>0.04143</b>	<b>9.47</b>	<b>0.55</b>	<b>9142.00</b>	<b>16.68</b>

Appendix B4 Results of experimental studies for the problem set 10.50 (cont.)

Problem Setting	Instance No	Throughput Rate			Deviation(%)		CPU (sec.)		
		CE	BTS	ATS	BTS	ATS	CE	BTS	ATS
5	1	0.03089	0.03007	<b>0.03089</b>	2.65	0.00	8113.45	31.32	13.49
	2	0.02345	0.02344	<b>0.02345</b>	0.03	0.00	11238.31	12.28	16.33
	3	0.02444	<b>0.02444</b>	<b>0.02444</b>	0.00	0.00	3249.22	3.04	2.31
	4	0.02614	0.02485	<b>0.02614</b>	4.96	0.00	11950.31	10.55	9.87
	5	0.02769	<b>0.02769</b>	<b>0.02769</b>	0.00	0.00	5130.26	11.15	6.05
	6	0.02340	<b>0.02340</b>	<b>0.02340</b>	0.00	0.00	9696.40	10.76	13.23
	7	0.02347	<b>0.02347</b>	<b>0.02347</b>	0.00	0.00	11516.88	4.55	9.50
	8	0.02437	<b>0.02437</b>	<b>0.02437</b>	0.00	0.00	7850.06	7.91	9.38
	9	0.03700	<b>0.03700</b>	<b>0.03700</b>	0.00	0.00	7339.22	13.03	15.07
	10	0.02351	<b>0.02351</b>	<b>0.02351</b>	0.00	0.00	11956.50	41.29	37.06
		<b>Avg</b>	<b>0.02644</b>	<b>0.02622</b>	<b>0.02644</b>	<b>0.76</b>	<b>0.00</b>	<b>8804.06</b>	<b>14.59</b>
6	1	0.01849	<b>0.01849</b>	<b>0.01849</b>	0.00	0.00	20235.51	10.37	21.29
	2	0.02103	<b>0.02103</b>	<b>0.02103</b>	0.00	0.00	8075.22	13.22	25.05
	3	0.02463	0.02368	<b>0.02463</b>	0.04	0.00	6055.65	10.35	14.59
	4	0.02786	0.02785	<b>0.02786</b>	0.00	0.00	5561.70	21.41	25.33
	5	0.02347	0.02333	<b>0.02347</b>	0.01	0.00	5453.36	17.17	24.56
	6	0.01986	<b>0.01986</b>	<b>0.01986</b>	0.00	0.00	17703.50	20.37	27.71
	7	0.02322	0.02174	<b>0.02322</b>	0.06	0.00	9320.15	12.46	14.10
	8	0.02892	<b>0.02892</b>	<b>0.02892</b>	0.00	0.00	12330.67	21.35	37.53
	9	0.02507	0.02506	<b>0.02507</b>	0.00	0.00	22505.36	65.76	78.33
	10	0.02213	<b>0.02213</b>	<b>0.02213</b>	0.00	0.00	15445.97	18.78	24.06
		<b>Avg</b>	<b>0.02347</b>	<b>0.02321</b>	<b>0.02347</b>	<b>1.09</b>	<b>0.00</b>	<b>12268.71</b>	<b>21.12</b>
7	1	0.02488	<b>0.02488</b>	<b>0.02488</b>	0.00	0.00	5306.97	4.71	3.63
	2	0.02199	0.02196	<b>0.02199</b>	0.00	0.00	7656.30	8.52	10.19
	3	0.02167	<b>0.02167</b>	<b>0.02167</b>	0.00	0.00	4888.31	4.98	4.06
	4	0.02558	<b>0.02558</b>	<b>0.02558</b>	0.00	0.00	15848.16	23.34	37.08
	5	0.02365	<b>0.02365</b>	<b>0.02365</b>	0.00	0.00	8961.77	14.48	14.74
	6	0.02250	0.02216	<b>0.02250</b>	0.01	0.00	18571.17	37.11	67.73
	7	0.02796	0.02786	<b>0.02796</b>	0.00	0.00	8609.40	50.59	71.12
	8	0.02224	<b>0.02224</b>	<b>0.02224</b>	0.00	0.00	15506.98	5.04	13.07
	9	0.02909	0.02906	<b>0.02909</b>	0.00	0.00	7033.93	16.24	20.25
	10	0.02397	<b>0.02397</b>	<b>0.02397</b>	0.00	0.00	9635.94	4.65	14.01
		<b>Avg</b>	<b>0.02435</b>	<b>0.02430</b>	<b>0.02435</b>	<b>0.21</b>	<b>0.00</b>	<b>10201.89</b>	<b>16.97</b>
8	1	0.01845	0.01730	<b>0.01845</b>	6.20	0.00	10353.56	4.81	15.99
	2	0.01333	0.01140	<b>0.01333</b>	14.51	0.00	7373.68	6.50	17.72
	3	0.01843	0.01832	0.01839	0.64	0.23	18605.25	85.86	58.77
	4	0.02214	0.01984	0.02181	10.36	1.48	17981.73	35.21	33.57
	5	0.00670	<b>0.00670</b>	<b>0.00670</b>	0.00	0.00	4380.44	5.74	5.05
	6	0.02194	0.02017	<b>0.02194</b>	8.04	0.00	18336.02	50.62	25.85
	7	0.02104	0.01885	<b>0.02104</b>	10.44	0.00	13823.67	27.33	26.02
	8	0.01657	0.01656	0.01656	0.07	0.07	10636.85	14.65	16.21
	9	0.01819	0.01695	0.01760	6.81	3.25	13771.64	36.40	33.73
	10	0.02464	0.02348	0.02422	4.71	1.72	23148.75	45.99	53.07
		<b>Avg</b>	<b>0.01814</b>	<b>0.01696</b>	<b>0.01800</b>	<b>6.18</b>	<b>0.68</b>	<b>13841.16</b>	<b>31.31</b>

Appendix B5 Results of experimental studies for the problem set 10.100

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
1	1	0.08694	0.08694	0.00	14.85	38.00
	2	0.07338	0.07338	0.00	12.14	43.76
	3	0.06442	0.06442	0.00	86.70	13.43
	4	0.07554	0.07554	0.00	7.05	27.58
	5	0.07907	0.07907	0.01	3.88	6.46
	6	0.07398	<b>0.07976</b>	7.81	42.06	25.49
	7	0.07049	0.07049	0.00	15.48	22.99
	8	0.06952	0.06952	0.00	6.79	5.43
	9	0.07501	<b>0.07519</b>	0.24	37.35	58.19
	10	0.07061	0.07061	0.00	84.24	63.87
	<b>Avg</b>	<b>0.07389</b>	<b>0.07449</b>	<b>0.81</b>	<b>31.05</b>	<b>30.52</b>
2	1	0.05128	0.05128	0.00	35.71	50.06
	2	0.06398	0.06398	0.00	17.89	27.91
	3	0.05225	0.05225	0.00	9.66	13.82
	4	0.07505	<b>0.07669</b>	2.19	69.37	70.20
	5	0.06620	<b>0.06651</b>	0.47	18.27	40.34
	6	0.05929	0.05929	0.00	9.05	25.16
	7	0.06119	<b>0.06442</b>	5.27	12.64	20.05
	8	0.07488	0.07488	0.00	76.19	90.64
	9	0.06852	<b>0.07147</b>	4.31	151.49	160.28
	10	0.07019	0.07024	0.08	73.38	44.46
	<b>Avg</b>	<b>0.06428</b>	<b>0.06510</b>	<b>1.18</b>	<b>47.37</b>	<b>54.29</b>
3	1	0.07394	0.07394	0.00	36.46	53.26
	2	0.06911	<b>0.07353</b>	6.39	22.64	34.29
	3	0.06816	0.06816	0.00	18.89	20.45
	4	0.06959	<b>0.07065</b>	1.52	33.06	48.56
	5	0.07067	<b>0.07087</b>	0.29	102.81	89.11
	6	0.06876	0.06876	0.00	44.13	154.25
	7	0.07802	<b>0.08215</b>	5.29	44.93	54.65
	8	0.06808	0.06808	0.00	92.12	92.04
	9	0.06615	0.06616	0.02	7.61	23.41
	10	0.07010	0.07010	0.00	15.79	56.21
	<b>Avg</b>	<b>0.07026</b>	<b>0.07124</b>	<b>1.35</b>	<b>41.84</b>	<b>62.62</b>
4	1	0.05082	<b>0.05371</b>	5.69	31.47	55.49
	2	0.00847	<b>0.00848</b>	0.13	7.89	14.84
	3	0.04700	0.04700	0.00	81.70	39.09
	4	0.05417	<b>0.05896</b>	8.85	81.98	81.20
	5	0.00682	0.00682	0.00	20.08	23.87
	6	0.05147	<b>0.05151</b>	0.08	325.51	58.28
	7	0.03937	<b>0.04751</b>	20.68	30.34	37.83
	8	0.03876	<b>0.03922</b>	1.18	22.31	32.43
	9	0.03036	<b>0.03733</b>	22.98	26.04	62.95
	10	0.05522	<b>0.05793</b>	4.89	86.21	53.09
	<b>Avg</b>	<b>0.03825</b>	<b>0.04085</b>	<b>6.45</b>	<b>71.35</b>	<b>45.91</b>

Appendix B5 Results of experimental studies for the problem set 10.100 (cont.)

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
5	1	0.02999	<b>0.03089</b>	2.99	51.78	36.77
	2	0.02345	0.02345	0.00	23.57	53.45
	3	0.02444	0.02444	0.00	6.12	9.83
	4	0.02708	<b>0.02709</b>	0.02	30.47	65.86
	5	0.02769	0.02769	0.01	29.06	36.22
	6	0.02359	0.02359	0.00	32.07	27.75
	7	0.02487	0.02487	0.01	15.77	17.43
	8	0.02437	0.02437	0.00	11.17	23.31
	9	0.03700	0.03700	0.01	28.81	70.14
	10	0.02351	0.02351	0.00	63.04	70.64
	<b>Avg</b>	<b>0.02660</b>	<b>0.02669</b>	<b>0.30</b>	<b>29.19</b>	<b>41.14</b>
6	1	0.03012	0.03012	0.00	14.07	47.27
	2	0.02104	0.02104	0.00	22.37	24.09
	3	0.02467	0.02467	0.00	55.99	46.64
	4	0.02788	0.02788	0.00	20.11	33.46
	5	0.02091	<b>0.02212</b>	5.76	17.61	47.00
	6	0.02004	0.02004	0.00	33.32	88.62
	7	0.02364	0.02364	0.00	17.57	16.13
	8	0.02876	<b>0.02893</b>	0.58	53.41	42.18
	9	0.02507	0.02507	0.00	125.38	131.79
	10	0.02213	0.02213	0.00	16.38	29.25
	<b>Avg</b>	<b>0.02443</b>	<b>0.02456</b>	<b>0.63</b>	<b>37.62</b>	<b>50.64</b>
7	1	0.02493	0.02493	0.00	10.02	15.54
	2	0.02196	0.02199	0.13	13.56	40.06
	3	0.02167	0.02167	0.01	6.18	14.45
	4	0.02558	0.02558	0.00	35.94	56.02
	5	0.02365	0.02365	0.00	31.03	48.48
	6	0.02187	<b>0.02217</b>	1.38	44.77	61.59
	7	0.02794	0.02828	1.24	42.98	59.61
	8	0.02224	0.02224	0.00	18.61	37.63
	9	0.02910	<b>0.02911</b>	0.05	44.44	48.67
	10	0.02397	0.02397	0.00	8.74	23.13
	<b>Avg</b>	<b>0.02429</b>	<b>0.02436</b>	<b>0.28</b>	<b>25.63</b>	<b>40.52</b>
8	1	0.01042	<b>0.01049</b>	0.68	32.71	40.58
	2	0.01367	<b>0.01635</b>	19.59	12.61	35.69
	3	0.01948	0.01948	0.02	99.76	193.96
	4	0.02151	<b>0.02303</b>	7.02	32.14	80.62
	5	0.00673	0.00673	0.00	11.19	26.72
	6	0.02278	<b>0.02288</b>	0.44	47.64	58.05
	7	0.02151	<b>0.02197</b>	2.18	46.02	37.03
	8	0.01825	0.01825	0.00	15.55	56.21
	9	0.01953	<b>0.02052</b>	5.09	66.64	55.88
	10	0.02539	<b>0.02632</b>	3.67	85.99	57.98
	<b>Avg</b>	<b>0.01793</b>	<b>0.01860</b>	<b>3.87</b>	<b>45.03</b>	<b>64.27</b>

Appendix B6 Results of experimental studies for the problem set 10.200

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
1	1	0.08694	0.08694	0.00	58.14	134.77
	2	0.07338	0.07338	0.00	165.55	231.74
	3	0.06442	0.06442	0.00	80.78	22.79
	4	0.07554	0.07554	0.00	75.24	105.41
	5	0.07906	<b>0.07907</b>	0.01	101.06	60.17
	6	0.07976	0.07976	0.00	17.77	53.60
	7	0.07011	<b>0.07406</b>	5.64	42.29	75.19
	8	0.06952	0.06952	0.00	29.61	24.43
	9	0.07499	0.07499	0.00	168.14	163.93
	10	0.07061	0.07061	0.00	126.75	284.31
		<b>Avg</b>	<b>0.07443</b>	<b>0.07483</b>	<b>0.57</b>	<b>86.53</b>
2	1	0.05128	0.05128	0.00	48.78	137.72
	2	0.06422	<b>0.06462</b>	0.62	90.67	120.08
	3	0.05224	<b>0.05225</b>	0.02	67.20	40.42
	4	0.07726	<b>0.07809</b>	1.08	139.81	158.94
	5	0.06516	<b>0.06629</b>	1.74	34.40	128.04
	6	0.05929	0.05929	0.00	17.85	42.45
	7	0.06247	<b>0.06760</b>	8.22	122.69	70.82
	8	0.07513	0.07513	0.00	62.31	248.27
	9	0.07267	<b>0.07269</b>	0.03	63.27	359.10
	10	0.06612	0.06612	0.00	19.95	56.48
		<b>Avg</b>	<b>0.06458</b>	<b>0.06534</b>	<b>1.17</b>	<b>66.69</b>
3	1	0.07476	0.07476	0.00	265.39	52.23
	2	0.06911	<b>0.07379</b>	6.76	47.59	158.44
	3	0.06816	0.06816	0.00	34.82	34.96
	4	0.07080	<b>0.07154</b>	1.04	38.97	108.11
	5	0.07087	0.07087	0.00	25.07	54.65
	6	0.06877	<b>0.07039</b>	2.36	335.18	475.80
	7	0.07923	<b>0.08363</b>	5.56	141.94	175.67
	8	0.06815	0.06815	0.00	237.25	90.12
	9	0.06645	0.06645	0.00	19.03	35.21
	10	0.07014	0.07014	0.00	16.85	59.53
		<b>Avg</b>	<b>0.07064</b>	<b>0.07179</b>	<b>1.57</b>	<b>116.21</b>
4	1	0.01059	0.01059	0.00	63.02	72.68
	2	0.00954	0.00954	0.00	54.60	60.33
	3	0.05259	0.05259	0.00	110.24	136.98
	4	0.06323	<b>0.06447</b>	1.96	89.61	219.79
	5	0.00682	0.00682	0.00	40.23	57.45
	6	0.05915	<b>0.05916</b>	0.02	99.82	280.64
	7	0.04559	<b>0.05441</b>	19.34	50.28	69.87
	8	0.04450	0.04450	0.00	39.31	62.60
	9	0.03560	<b>0.04048</b>	13.69	80.71	150.51
	10	0.06202	<b>0.06428</b>	3.65	76.36	213.50
		<b>Avg</b>	<b>0.03896</b>	<b>0.04068</b>	<b>3.87</b>	<b>70.42</b>

Appendix B6 Results of experimental studies for the problem set 10.200 (cont.)

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
5	1	0.03089	0.03089	0.00	19.34	88.51
	2	0.02345	0.02345	0.01	60.75	100.89
	3	0.02444	0.02444	0.00	12.37	23.13
	4	0.02709	0.02709	0.00	58.58	109.20
	5	0.02769	0.02769	0.00	19.44	40.31
	6	0.02363	0.02363	0.00	19.31	57.35
	7	0.02487	0.02487	0.00	26.94	46.80
	8	0.03099	<b>0.03352</b>	8.13	32.07	69.11
	9	0.03700	0.03700	0.00	65.46	74.79
	10	0.02351	0.02351	0.00	110.28	79.33
	<b>Avg</b>	<b>0.02736</b>	<b>0.02806</b>	<b>0.81</b>	<b>42.45</b>	<b>68.94</b>
6	1	0.02122	<b>0.03012</b>	41.95	53.98	71.81
	2	0.02104	0.02104	0.00	48.67	41.59
	3	0.02467	0.02467	0.00	57.70	88.55
	4	0.02788	0.02788	0.00	75.43	73.40
	5	0.02212	<b>0.02356</b>	6.51	56.03	62.99
	6	0.02013	0.02013	0.00	40.00	67.84
	7	0.02371	<b>0.02388</b>	0.70	31.30	116.84
	8	0.02893	0.02893	0.00	148.51	64.61
	9	0.02507	0.02507	0.00	312.73	171.37
	10	0.02213	0.02213	0.00	26.27	61.78
	<b>Avg</b>	<b>0.02369</b>	<b>0.02474</b>	<b>4.92</b>	<b>85.06</b>	<b>82.08</b>
7	1	0.02493	0.02493	0.00	17.72	26.22
	2	0.02196	<b>0.02199</b>	0.12	38.76	45.71
	3	0.02167	0.02167	0.00	13.72	23.93
	4	0.02558	0.02558	0.00	54.27	148.07
	5	0.02365	0.02365	0.00	19.34	56.39
	6	0.02187	<b>0.02220</b>	1.52	91.32	178.62
	7	0.02830	0.02830	0.00	77.52	103.08
	8	0.02224	0.02224	0.00	19.31	63.30
	9	0.02324	<b>0.02912</b>	25.28	33.68	109.32
	10	0.02397	0.02397	0.00	111.98	54.65
	<b>Avg</b>	<b>0.02374</b>	<b>0.02436</b>	<b>2.69</b>	<b>47.76</b>	<b>80.93</b>
8	1	0.01565	0.01565	0.00	54.09	71.45
	2	0.01449	<b>0.01842</b>	27.14	58.55	63.62
	3	0.01981	0.01981	0.00	50.93	208.88
	4	0.01898	<b>0.02335</b>	23.02	46.65	144.97
	5	0.00673	0.00673	0.00	16.74	27.38
	6	0.02293	0.02293	0.00	130.67	222.21
	7	0.02220	<b>0.02221</b>	0.02	51.29	159.95
	8	0.01929	0.01929	0.01	18.55	55.61
	9	0.02147	<b>0.02156</b>	0.38	62.98	87.39
	10	0.02749	<b>0.02763</b>	0.51	151.81	298.35
	<b>Avg</b>	<b>0.01890</b>	<b>0.01976</b>	<b>5.11</b>	<b>64.23</b>	<b>133.98</b>



Appendix B7 Results of experimental studies for the problem set 20.100

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
1	1	0.07983	0.07983	0.00	514.09	632.35
	2	0.07093	<b>0.07349</b>	3.62	528.01	479.06
	3	0.06884	0.06884	0.00	526.12	1919.35
	4	0.08056	0.08056	0.00	441.91	523.46
	5	0.07924	0.07924	0.00	407.51	618.62
	6	0.10068	0.10068	0.00	451.13	267.48
	7	0.07070	<b>0.07167</b>	1.38	515.25	243.64
	8	0.09473	0.09473	0.00	510.36	525.47
	9	0.06566	<b>0.11369</b>	73.15	399.82	445.11
	10	0.08877	0.08877	0.00	431.66	446.97
		<b>Avg</b>	<b>0.07999</b>	<b>0.08515</b>	<b>7.82</b>	<b>472.59</b>
2	1	0.04606	<b>0.06961</b>	51.15	455.09	348.55
	2	0.02104	<b>0.02106</b>	0.10	473.20	316.42
	3	0.00641	<b>0.00642</b>	0.09	376.86	258.35
	4	0.06209	<b>0.07121</b>	14.69	714.02	872.43
	5	0.03307	0.03307	0.00	418.32	215.00
	6	0.06506	0.06506	0.00	663.18	995.41
	7	0.04643	<b>0.06737</b>	45.09	415.34	894.88
	8	0.06498	0.06498	0.00	489.57	717.46
	9	0.06303	<b>0.09318</b>	47.83	809.98	873.65
	10	0.07589	<b>0.08292</b>	9.26	542.39	333.96
		<b>Avg</b>	<b>0.04841</b>	<b>0.05749</b>	<b>16.82</b>	<b>535.80</b>
3	1	0.06623	<b>0.07356</b>	11.07	1088.44	1827.51
	2	0.06990	<b>0.07080</b>	1.29	521.13	577.29
	3	0.06912	0.06912	0.00	617.04	655.50
	4	0.06561	<b>0.06652</b>	1.39	530.23	489.88
	5	0.06438	<b>0.06933</b>	7.70	858.22	1383.10
	6	0.06796	<b>0.06890</b>	1.38	858.22	620.27
	7	0.07172	<b>0.07570</b>	5.54	635.22	856.18
	8	0.06947	<b>0.06950</b>	0.04	630.24	947.95
	9	0.07755	<b>0.07900</b>	1.87	821.25	766.60
	10	0.06466	<b>0.06877</b>	6.35	876.28	694.95
		<b>Avg</b>	<b>0.06866</b>	<b>0.07112</b>	<b>3.66</b>	<b>743.63</b>
4	1	0.08216	<b>0.08816</b>	7.30	444.59	616.22
	2	0.04566	<b>0.04593</b>	0.60	500.48	276.38
	3	0.04511	<b>0.07416</b>	64.39	1026.45	1160.45
	4	0.06005	<b>0.06137</b>	2.20	557.64	628.68
	5	0.03277	<b>0.04197</b>	28.06	638.92	865.91
	6	0.08740	<b>0.08857</b>	1.33	525.45	412.09
	7	0.04388	<b>0.12184</b>	177.69	771.14	956.75
	8	0.04205	<b>0.04786</b>	13.80	631.43	430.28
	9	0.10754	<b>0.11973</b>	11.33	466.62	477.55
	10	0.04757	<b>0.05041</b>	5.97	750.24	767.40
		<b>Avg</b>	<b>0.05942</b>	<b>0.07400</b>	<b>31.27</b>	<b>631.30</b>

Appendix B7 Results of experimental studies for the problem set 20.100 (cont.)

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
5	1	0.02566	<b>0.02800</b>	9.10	347.16	370.13
	2	0.02663	0.02663	0.00	663.86	767.41
	3	0.02516	<b>0.02575</b>	2.35	527.71	598.45
	4	0.02278	<b>0.02280</b>	0.06	506.05	479.25
	5	0.02625	0.02625	0.00	509.67	340.38
	6	0.02251	<b>0.02666</b>	18.45	523.91	305.07
	7	0.02414	<b>0.02610</b>	8.12	551.04	241.35
	8	0.02225	<b>0.02872</b>	29.09	502.95	821.64
	9	0.02593	0.02593	0.00	486.74	240.22
	10	0.03369	0.03369	0.00	632.13	414.90
		<b>Avg</b>	<b>0.02550</b>	<b>0.02705</b>	<b>6.72</b>	<b>525.12</b>
6	1	0.00441	0.00441	0.00	247.80	130.49
	2	0.01701	<b>0.02104</b>	23.73	467.62	273.95
	3	0.01724	<b>0.01732</b>	0.45	193.11	429.13
	4	0.02466	<b>0.02475</b>	0.34	532.64	492.56
	5	0.02483	<b>0.02799</b>	12.71	314.07	394.15
	6	0.02751	0.02751	0.00	475.62	542.87
	7	0.02337	0.02337	0.00	306.00	1062.50
	8	0.02131	0.02131	0.00	282.93	368.43
	9	0.02073	0.02073	0.00	222.00	188.76
	10	0.03375	<b>0.03444</b>	2.06	227.15	481.17
		<b>Avg</b>	<b>0.02148</b>	<b>0.02229</b>	<b>3.93</b>	<b>326.89</b>
7	1	0.02477	0.02477	0.00	1182.51	771.95
	2	0.02173	<b>0.02176</b>	0.12	1082.48	215.39
	3	0.02226	0.02226	0.01	2194.73	1052.91
	4	0.02520	0.02520	0.00	1216.48	875.19
	5	0.02210	<b>0.02284</b>	3.35	1112.16	739.00
	6	0.02212	<b>0.02219</b>	0.33	1250.37	762.00
	7	0.03132	0.03132	0.00	1239.29	920.61
	8	0.02279	0.02279	0.00	1178.15	904.21
	9	0.02796	<b>0.02804</b>	0.31	1150.21	916.72
	10	0.02352	0.02352	0.00	1449.70	579.76
		<b>Avg</b>	<b>0.02438</b>	<b>0.02447</b>	<b>0.41</b>	<b>1305.61</b>
8	1	0.00820	<b>0.00821</b>	0.05	289.53	311.68
	2	0.01666	0.01666	0.00	425.08	395.50
	3	0.02017	<b>0.02127</b>	5.48	572.84	1240.66
	4	0.02218	0.02380	7.34	462.05	879.67
	5	0.01748	<b>0.02072</b>	18.54	582.67	556.57
	6	0.02037	<b>0.02974</b>	46.01	535.14	667.84
	7	0.01609	<b>0.02544</b>	58.07	662.19	792.75
	8	0.01911	<b>0.02180</b>	14.07	351.73	617.72
	9	0.02226	<b>0.02582</b>	15.99	263.36	537.19
	10	0.02377	0.02377	0.00	471.72	650.15
		<b>Avg</b>	<b>0.01863</b>	<b>0.02172</b>	<b>16.55</b>	<b>461.63</b>

Appendix B8 Results of experimental studies for the problem set 20.200

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
1	1	0.07998	0.07998	0.00	1263.56	1034.76
	2	0.07044	<b>0.07349</b>	4.34	748.64	1023.74
	3	0.06884	0.06884	0.00	919.56	1208.49
	4	0.08056	0.08056	0.00	486.73	963.68
	5	0.07924	0.07924	0.00	625.46	1206.60
	6	0.10068	0.10068	0.00	548.12	769.74
	7	0.07070	<b>0.07167</b>	1.38	406.66	336.57
	8	0.09477	0.09477	0.00	679.37	517.56
	9	0.06570	<b>0.06573</b>	0.04	487.08	575.83
	10	0.08877	0.08877	0.00	1046.09	750.66
	<b>Avg</b>	<b>0.07997</b>	<b>0.08037</b>	<b>0.58</b>	<b>721.13</b>	<b>838.76</b>
2	1	0.04612	<b>0.06961</b>	50.93	808.25	1282.70
	2	0.05492	0.05492	0.00	535.20	676.34
	3	0.00641	0.00641	0.00	432.05	299.01
	4	0.07217	<b>0.07227</b>	0.13	657.38	1140.36
	5	0.03307	0.03307	0.00	465.61	502.93
	6	0.06508	0.06508	0.00	508.49	1491.16
	7	0.04003	<b>0.06737</b>	68.32	595.60	336.13
	8	0.06529	0.06529	0.00	805.05	973.21
	9	0.06341	<b>0.09134</b>	44.03	1513.90	1242.85
	10	0.08220	<b>0.08567</b>	4.22	1121.84	675.00
	<b>Avg</b>	<b>0.05287</b>	<b>0.06110</b>	<b>16.76</b>	<b>744.34</b>	<b>861.97</b>
3	1	0.07441	0.07441	0.00	581.01	1689.59
	2	0.06663	<b>0.06695</b>	0.48	195.76	785.74
	3	0.06986	<b>0.06987</b>	0.02	2187.92	1642.96
	4	0.06576	<b>0.06769</b>	2.92	230.74	814.10
	5	0.06668	<b>0.07042</b>	5.61	308.60	1441.47
	6	0.06952	<b>0.06953</b>	0.01	638.04	1572.50
	7	0.07793	<b>0.08391</b>	7.67	347.99	1489.51
	8	0.06977	0.06977	0.00	506.27	1535.59
	9	0.06990	<b>0.07942</b>	13.62	2171.74	1953.90
	10	0.06883	0.06883	0.00	1809.32	2197.18
	<b>Avg</b>	<b>0.06993</b>	<b>0.07208</b>	<b>3.03</b>	<b>1495.78</b>	<b>1512.25</b>
4	1	0.08227	<b>0.10730</b>	30.43	796.73	1417.61
	2	0.04860	<b>0.04902</b>	0.87	221.86	594.99
	3	0.07585	<b>0.07792</b>	2.73	2700.41	2752.77
	4	0.06352	<b>0.06575</b>	3.50	510.25	1268.11
	5	0.04296	<b>0.04925</b>	14.63	2414.79	1115.89
	6	0.08915	<b>0.09207</b>	3.28	227.59	986.78
	7	0.04940	<b>0.06661</b>	34.83	539.90	2403.47
	8	0.04675	<b>0.05133</b>	9.81	632.72	1334.41
	9	0.09477	<b>0.14420</b>	52.16	464.40	833.93
	10	0.05634	0.05634	0.00	951.85	1773.26
	<b>Avg</b>	<b>0.06496</b>	<b>0.07598</b>	<b>15.22</b>	<b>946.05</b>	<b>1448.12</b>

Appendix B8 Results of experimental studies for the problem set 20.200 (cont.)

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
5	1	0.02656	<b>0.02800</b>	5.40	1407.30	654.51
	2	0.02663	0.02663	0.00	597.55	877.89
	3	0.02516	<b>0.02554</b>	1.50	895.55	968.64
	4	0.02313	0.02313	0.00	632.33	901.37
	5	0.02625	0.02625	0.00	1405.96	721.86
	6	0.02666	0.02666	0.00	460.53	669.51
	7	0.02414	<b>0.02610</b>	8.12	559.75	490.53
	8	0.02872	0.02872	0.00	523.12	705.68
	9	0.02583	<b>0.02593</b>	0.38	1486.00	555.08
	10	0.03369	0.03369	0.00	1032.97	870.03
	<b>Avg</b>	<b>0.02668</b>	<b>0.02706</b>	<b>1.54</b>	<b>900.11</b>	<b>741.51</b>
6	1	0.00441	0.00441	0.00	451.34	388.86
	2	0.01678	<b>0.02318</b>	38.18	472.02	837.78
	3	0.01724	0.01724	0.00	1658.10	1597.24
	4	0.01987	<b>0.02478</b>	24.70	292.89	928.53
	5	0.02972	<b>0.02991</b>	0.66	409.22	682.77
	6	0.02261	<b>0.02751</b>	21.67	546.31	1111.25
	7	0.02337	0.02337	0.00	750.69	723.31
	8	0.02131	0.02131	0.00	263.50	944.19
	9	0.02069	<b>0.02073</b>	0.20	191.65	408.19
	10	0.02805	<b>0.03486</b>	24.27	152.09	863.12
	<b>Avg</b>	<b>0.02040</b>	<b>0.02902</b>	<b>10.97</b>	<b>518.78</b>	<b>848.52</b>
7	1	0.02477	0.02477	0.00	3228.14	1441.97
	2	0.02153	0.02153	0.00	240.29	486.41
	3	0.02227	0.02227	0.00	257.87	3825.08
	4	0.02520	0.02520	0.00	1110.18	1511.89
	5	0.02305	0.02305	0.00	238.15	851.12
	6	0.02212	<b>0.02221</b>	0.41	717.54	607.14
	7	0.02830	<b>0.03155</b>	11.47	398.89	984.61
	8	0.02279	0.02279	0.00	440.15	2645.83
	9	0.02805	0.02805	0.00	1481.54	1355.20
	10	0.02352	0.02352	0.00	1111.17	1302.56
	<b>Avg</b>	<b>0.02416</b>	<b>0.02449</b>	<b>1.19</b>	<b>922.39</b>	<b>1501.18</b>
8	1	0.00823	0.00823	0.01	235.31	213.28
	2	0.01783	<b>0.01785</b>	0.15	749.56	554.14
	3	0.02273	<b>0.02274</b>	0.05	791.49	1137.40
	4	0.02384	0.02384	0.00	564.15	756.12
	5	0.02187	<b>0.02366</b>	8.15	1732.98	1391.76
	6	0.02806	<b>0.03096</b>	10.32	333.41	1340.03
	7	0.02067	<b>0.02697</b>	30.44	561.75	2476.94
	8	0.02137	<b>0.02465</b>	15.34	855.17	2380.58
	9	0.01040	<b>0.02547</b>	144.98	565.29	486.63
	10	0.02530	<b>0.02531</b>	0.00	1416.98	1022.91
	<b>Avg</b>	<b>0.02003</b>	<b>0.02297</b>	<b>20.95</b>	<b>780.61</b>	<b>1175.98</b>

Appendix B9 Results of experimental studies for the problem set 20.400

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
1	1	0.08000	0.08000	0.02	1402.42	2274.33
	2	0.07096	<b>0.07420</b>	4.57	1668.24	2974.49
	3	0.06884	0.06884	0.00	1789.24	3947.73
	4	0.08056	0.08056	0.00	1354.84	2014.96
	5	0.07924	0.07924	0.00	4395.31	2748.54
	6	0.10045	<b>0.10068</b>	0.22	797.75	1371.94
	7	0.07070	<b>0.07167</b>	1.38	979.87	1545.36
	8	0.09477	0.09477	0.00	834.75	1073.44
	9	0.06570	<b>0.06573</b>	0.04	1276.22	2228.18
	10	0.08877	0.08877	0.00	1967.24	2033.73
		<b>Avg</b>	<b>0.08000</b>	<b>0.08045</b>	<b>0.62</b>	<b>1646.59</b>
2	1	0.04612	<b>0.07070</b>	53.29	1047.53	4041.17
	2	0.05652	0.05652	0.00	503.60	1567.10
	3	0.00641	0.00641	0.02	148.29	802.76
	4	0.07206	<b>0.07303</b>	1.35	1495.71	4569.22
	5	0.03307	0.03307	0.00	742.33	1241.17
	6	0.06508	0.06508	0.00	1359.79	2875.22
	7	0.04008	<b>0.06737</b>	68.09	1291.20	757.94
	8	0.06529	0.06529	0.00	2842.96	1994.40
	9	0.06335	<b>0.06345</b>	0.16	1815.47	3900.43
	10	0.08707	0.08707	0.00	322.47	1488.52
		<b>Avg</b>	<b>0.05351</b>	<b>0.05880</b>	<b>12.29</b>	<b>1156.94</b>
3	1	0.07441	<b>0.08440</b>	13.43	3683.31	3767.83
	2	0.07205	<b>0.07324</b>	1.65	1662.74	4083.45
	3	0.07003	0.07003	0.00	1475.67	10726.03
	4	0.06804	<b>0.06806</b>	0.04	1203.65	1371.73
	5	0.07068	0.07068	0.00	2669.55	3456.61
	6	0.06956	<b>0.06966</b>	0.14	2242.61	3780.87
	7	0.07896	<b>0.08455</b>	7.08	1366.78	4399.89
	8	0.06983	0.06983	0.00	3310.92	3742.49
	9	0.07946	0.07946	0.00	3865.24	2656.06
	10	0.06884	0.06884	0.00	2833.76	2617.19
		<b>Avg</b>	<b>0.07218</b>	<b>0.07387</b>	<b>2.23</b>	<b>2431.42</b>
4	1	0.00823	0.00823	0.00	123.46	4108.05
	2	0.05263	0.05263	0.00	727.57	1174.79
	3	0.06011	<b>0.06221</b>	3.50	1946.38	3329.65
	4	0.06260	<b>0.06753</b>	7.88	1115.54	3060.88
	5	0.05430	<b>0.06133</b>	12.94	1358.84	4003.40
	6	0.09098	<b>0.09544</b>	4.90	436.61	1921.41
	7	0.06000	<b>0.07227</b>	20.45	806.43	4560.43
	8	0.05564	<b>0.05881</b>	5.69	1772.24	3390.74
	9	0.06349	<b>0.06439</b>	1.43	959.54	856.33
	10	0.06089	<b>0.06141</b>	0.86	1414.56	2452.06
		<b>Avg</b>	<b>0.05689</b>	<b>0.06043</b>	<b>5.76</b>	<b>1066.12</b>

Appendix B9 Results of experimental studies for the problem set 20.400 (cont.)

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
5	1	0.03207	<b>0.03540</b>	10.38	1318.12	1238.61
	2	0.02663	0.02663	0.00	1569.85	2033.28
	3	0.02595	<b>0.02598</b>	0.12	684.51	1424.64
	4	0.02326	0.02326	0.00	982.66	1638.13
	5	0.03638	0.03638	0.00	378.71	967.74
	6	0.02666	0.02666	0.00	1671.68	1123.48
	7	0.02414	<b>0.02825</b>	17.04	1720.37	1527.65
	8	0.02872	0.02872	0.00	535.32	1782.82
	9	0.02593	0.02593	0.00	900.06	1867.39
	10	0.03369	0.03369	0.00	962.16	1751.40
	<b>Avg</b>	<b>0.02834</b>	<b>0.02909</b>	<b>2.75</b>	<b>1072.34</b>	<b>1535.51</b>
6	1	0.00440	0.00440	0.11	753.37	1232.76
	2	0.01678	<b>0.02341</b>	39.49	1639.39	3150.58
	3	0.01724	0.01724	0.00	676.85	1396.25
	4	0.01987	<b>0.02478</b>	24.67	537.33	2419.84
	5	0.02380	<b>0.03060</b>	28.58	2091.39	2213.61
	6	0.02751	0.02751	0.00	2146.80	2188.06
	7	0.02337	0.02337	0.00	383.79	1118.32
	8	0.02131	0.02131	0.00	299.47	1953.43
	9	0.02073	0.02073	0.00	730.24	3365.64
	10	0.03485	<b>0.03488</b>	0.09	1408.37	1731.01
	<b>Avg</b>	<b>0.02099</b>	<b>0.02282</b>	<b>9.29</b>	<b>1066.70</b>	<b>2076.95</b>
7	1	0.02477	<b>0.02480</b>	0.12	6452.50	4941.10
	2	0.02167	<b>0.02170</b>	0.11	1812.66	4880.82
	3	0.02227	0.02227	0.00	3034.03	2380.59
	4	0.02520	0.02520	0.00	2729.32	2665.11
	5	0.02306	0.02306	0.00	1954.15	3741.20
	6	0.02215	<b>0.02221</b>	0.24	3630.24	1063.50
	7	0.02707	<b>0.03155</b>	16.57	2567.84	2147.27
	8	0.02279	0.02279	0.00	1486.08	4062.45
	9	0.02805	0.02805	0.00	1574.10	3677.55
	10	0.02352	0.02352	0.00	1101.30	1431.41
	<b>Avg</b>	<b>0.02405</b>	<b>0.02451</b>	<b>1.70</b>	<b>2634.22</b>	<b>3099.10</b>
8	1	0.00823	0.00823	0.00	1009.38	1647.06
	2	0.01867	0.01867	0.00	1050.45	1296.64
	3	0.02380	0.02380	0.00	1052.00	1777.92
	4	0.02384	0.02384	0.00	739.72	1637.61
	5	0.02372	<b>0.02493</b>	5.11	4558.11	3550.95
	6	0.02371	<b>0.03149</b>	32.84	996.92	2242.15
	7	0.02122	<b>0.02787</b>	31.32	922.18	1974.11
	8	0.02701	0.02701	0.00	599.85	2429.41
	9	0.02578	0.02578	0.00	610.03	1146.93
	10	0.02637	0.02637	0.00	1118.27	2178.11
	<b>Avg</b>	<b>0.02223</b>	<b>0.02379</b>	<b>6.93</b>	<b>1265.69</b>	<b>1988.09</b>

Appendix B10 Results of experimental studies for the problem set 40.200

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
1	1	0.07327	<b>0.08200</b>	11.92	2815.94	13552.11
	2	0.06027	0.06027	0.00	1174.70	11863.92
	3	0.07957	0.07957	0.00	3056.91	12311.99
	4	0.07952	0.07952	0.00	1973.86	10925.34
	5	0.07338	<b>0.07346</b>	0.11	2646.05	9988.17
	6	0.06721	<b>0.08597</b>	27.90	4216.20	16540.13
	7	0.06803	<b>0.06812</b>	0.13	4100.84	14278.52
	8	0.09560	0.09560	0.00	1421.41	16651.79
	9	0.03156	<b>0.03480</b>	10.26	4995.30	23352.77
	10	0.08674	0.08674	0.00	4153.45	12357.47
		<b>Avg</b>	<b>0.07151</b>	<b>0.07460</b>	<b>5.03</b>	<b>3055.47</b>
2	1	0.06547	<b>0.09746</b>	48.87	6543.52	15792.22
	2	0.09626	<b>0.09663</b>	0.38	1115.05	6271.30
	3	0.07888	<b>0.08022</b>	1.71	2458.77	11470.12
	4	0.06936	<b>0.06954</b>	0.25	6546.20	11141.63
	5	0.06366	<b>0.06500</b>	2.12	7864.83	33952.58
	6	0.05883	0.05883	0.00	16168.44	25836.86
	7	0.06923	0.06923	0.00	1518.34	5728.60
	8	0.08170	<b>0.08210</b>	0.50	7365.01	13978.12
	9	0.07312	0.07312	0.00	3835.86	7802.66
	10	0.06904	<b>0.07571</b>	9.66	5844.13	5367.52
		<b>Avg</b>	<b>0.07255</b>	<b>0.07679</b>	<b>6.35</b>	<b>5926.02</b>
3	1	0.07292	<b>0.07305</b>	0.19	6023.73	15447.66
	2	0.06707	0.06707	0.00	1963.70	5027.43
	3	0.07483	<b>0.07830</b>	4.65	15815.53	26778.01
	4	0.06085	<b>0.06109</b>	0.39	1756.45	4391.06
	5	0.07268	<b>0.07274</b>	0.08	8936.36	25387.56
	6	0.07966	<b>0.08312</b>	4.34	8919.91	24489.68
	7	0.06754	<b>0.07852</b>	16.26	5560.72	21465.61
	8	0.09433	<b>0.09451</b>	0.19	6255.59	20900.17
	9	0.08458	<b>0.08527</b>	0.82	8022.93	28562.42
	10	0.08606	<b>0.08643</b>	0.43	5205.87	21326.04
		<b>Avg</b>	<b>0.07605</b>	<b>0.07801</b>	<b>2.73</b>	<b>6846.08</b>
4	1	0.05897	<b>0.06984</b>	18.43	5080.15	21412.47
	2	0.04542	<b>0.04666</b>	2.71	2189.63	6539.40
	3	0.04307	<b>0.05292</b>	22.88	16311.28	26558.22
	4	0.05751	<b>0.06156</b>	7.05	4219.17	14515.56
	5	0.04565	<b>0.05165</b>	13.14	13663.22	21208.65
	6	0.07553	<b>0.07789</b>	3.13	2075.05	6922.93
	7	0.05281	<b>0.06209</b>	17.56	5153.79	12044.58
	8	0.05788	<b>0.06896</b>	19.14	7650.00	12440.17
	9	0.06642	0.06642	0.00	1207.70	22490.63
	10	0.05748	<b>0.06483</b>	12.79	4945.88	17551.04
		<b>Avg</b>	<b>0.05607</b>	<b>0.06228</b>	<b>11.68</b>	<b>6249.59</b>

Appendix B10 Results of experimental studies for the problem set 40.200 (cont.)

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
5	1	0.02572	0.02572	0.00	4532.03	14535.20
	2	0.02513	0.02513	0.00	2003.23	7599.48
	3	0.02495	<b>0.02496</b>	0.04	2488.64	13870.25
	4	0.02650	0.02650	0.00	2351.25	16177.85
	5	0.02487	0.02487	0.00	2700.03	10753.72
	6	0.02293	0.02293	0.00	3307.66	7461.38
	7	0.02334	0.02334	0.00	1987.13	6461.80
	8	0.02169	<b>0.02710</b>	24.92	1681.25	13156.19
	9	0.02168	0.02168	0.00	5759.06	12850.29
	10	0.03292	0.03292	0.00	1250.16	7754.29
	<b>Avg</b>	<b>0.02497</b>	<b>0.02552</b>	<b>2.50</b>	<b>2806.04</b>	<b>11062.05</b>
6	1	0.02875	<b>0.02948</b>	2.56	2176.67	16729.51
	2	0.01930	0.01930	0.00	3371.52	9610.69
	3	0.02337	<b>0.03011</b>	28.84	2558.16	9431.27
	4	0.02456	0.02456	0.00	2446.48	8042.12
	5	0.02203	0.02203	0.00	2745.19	11029.80
	6	0.02232	<b>0.02242</b>	0.44	3028.26	7631.35
	7	0.02047	<b>0.02307</b>	12.73	2801.36	10328.14
	8	0.03202	0.03202	0.00	2400.70	6771.24
	9	0.02614	<b>0.02778</b>	6.25	3497.80	20571.80
	10	0.02875	0.02875	0.00	4466.03	11159.02
	<b>Avg</b>	<b>0.02477</b>	<b>0.02595</b>	<b>5.08</b>	<b>2949.22</b>	<b>11130.49</b>
7	1	0.02574	<b>0.02582</b>	0.30	6034.09	14136.17
	2	0.02215	<b>0.02229</b>	0.66	4438.19	10203.99
	3	0.02781	<b>0.02803</b>	0.76	5089.61	10985.18
	4	0.02796	<b>0.02797</b>	0.05	7293.75	13342.20
	5	0.02595	<b>0.02609</b>	0.51	6702.91	18100.61
	6	0.03269	<b>0.03841</b>	17.51	5132.66	10328.95
	7	0.02317	<b>0.02325</b>	0.35	3998.66	8833.30
	8	0.03670	0.03670	0.00	4081.36	8575.73
	9	0.02980	<b>0.02986</b>	0.21	8100.55	24915.04
	10	0.03287	0.03287	0.00	4247.28	10302.23
	<b>Avg</b>	<b>0.02848</b>	<b>0.02913</b>	<b>2.03</b>	<b>5511.91</b>	<b>12972.34</b>
8	1	0.01856	<b>0.02773</b>	49.37	12618.81	42253.54
	2	0.01703	<b>0.01705</b>	0.12	5054.92	13085.27
	3	0.01912	<b>0.02241</b>	17.22	10970.84	28875.46
	4	0.02224	<b>0.02396</b>	7.73	4596.08	14181.67
	5	0.02054	<b>0.02064</b>	0.46	12404.01	13233.24
	6	0.02079	<b>0.02953</b>	42.05	6831.69	16844.55
	7	0.01892	<b>0.02332</b>	23.26	7025.14	19843.68
	8	0.02729	<b>0.02904</b>	6.42	2963.74	16434.03
	9	0.02463	<b>0.02485</b>	0.91	8298.10	22987.44
	10	0.02723	<b>0.02801</b>	2.89	4249.05	14913.41
	<b>Avg</b>	<b>0.02163</b>	<b>0.02465</b>	<b>15.04</b>	<b>7501.24</b>	<b>20265.23</b>



Appendix B11 Results of experimental studies for the problem set 40.400

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
1	1	0.08200	0.08200	0.00	6078.41	29163.88
	2	0.06027	0.06027	0.00	3539.51	16926.48
	3	0.07952	<b>0.07957</b>	0.06	10658.80	37448.29
	4	0.07952	0.07952	0.00	4636.64	28288.55
	5	0.07338	<b>0.07346</b>	0.11	4988.56	32450.96
	6	0.06721	<b>0.08597</b>	27.90	7568.82	34732.79
	7	0.06803	<b>0.06808</b>	0.08	5863.07	29255.94
	8	0.09560	0.09560	0.00	1591.14	18620.74
	9	0.06788	<b>0.06902</b>	1.68	10247.36	29178.62
	10	0.08674	0.08674	0.00	9757.81	36119.17
		<b>Avg</b>	<b>0.07601</b>	<b>0.07802</b>	<b>2.98</b>	<b>6493.01</b>
2	1	0.06428	<b>0.07352</b>	14.38	10298.55	27928.16
	2	0.09669	0.09669	0.00	3528.71	15018.90
	3	0.07908	<b>0.08178</b>	3.41	5356.44	22185.66
	4	0.06947	<b>0.06958</b>	0.16	15060.64	26849.99
	5	0.06424	<b>0.06547</b>	1.92	7459.03	62964.94
	6	0.05886	0.05886	0.00	16739.39	40051.36
	7	0.06923	<b>0.06948</b>	0.36	6350.44	30914.72
	8	0.08304	0.08304	0.00	9700.77	28932.40
	9	0.07496	0.07496	0.00	12795.39	36496.87
	10	0.07575	0.07575	0.00	2657.76	11357.30
		<b>Avg</b>	<b>0.07356</b>	<b>0.07492</b>	<b>2.02</b>	<b>8994.71</b>
3	1	0.07332	<b>0.07337</b>	0.06	16280.78	35756.84
	2	0.06799	<b>0.06831</b>	0.48	5754.05	11151.21
	3	0.07619	<b>0.07908</b>	3.79	5392.91	13368.90
	4	0.06124	0.06124	0.00	13642.39	29584.57
	5	0.07313	<b>0.07315</b>	0.03	4379.03	11426.90
	6	0.08569	<b>0.08614</b>	0.53	24482.97	72147.65
	7	0.07852	<b>0.07858</b>	0.08	12603.60	77072.13
	8	0.09519	<b>0.09521</b>	0.02	9418.82	49508.60
	9	0.08381	<b>0.08605</b>	2.68	7524.56	21249.27
	10	0.08658	<b>0.08659</b>	0.01	10412.73	31074.05
		<b>Avg</b>	<b>0.07816</b>	<b>0.07877</b>	<b>0.77</b>	<b>10989.18</b>
4	1	0.05897	<b>0.06227</b>	5.59	9299.57	41080.21
	2	0.04991	<b>0.05172</b>	3.63	3531.21	15363.42
	3	0.05309	<b>0.05871</b>	10.59	24695.76	47090.40
	4	0.05751	<b>0.06378</b>	10.91	5809.01	23556.20
	5	0.05332	<b>0.06087</b>	14.16	23637.81	44445.38
	6	0.07989	0.07989	0.00	4164.93	20755.54
	7	0.05281	0.05281	0.00	5068.12	22251.41
	8	0.05242	0.05242	0.00	9717.13	24551.60
	9	0.06642	0.06642	0.00	16158.59	41511.09
	10	0.05213	<b>0.06115</b>	17.30	10070.61	41539.81
		<b>Avg</b>	<b>0.05765</b>	<b>0.06100</b>	<b>6.22</b>	<b>11215.27</b>

Appendix B11 Results of experimental studies for the problem set 40.400 (cont.)

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
5	1	0.02572	0.02572	0.00	11414.59	32423.50
	2	0.02513	<b>0.02590</b>	3.04	21482.39	44447.34
	3	0.02574	0.02574	0.00	19847.79	39543.70
	4	0.02650	0.02650	0.00	12657.03	35981.93
	5	0.02487	<b>0.02600</b>	4.56	18826.44	36479.01
	6	0.02293	0.02293	0.00	12353.07	35059.82
	7	0.02334	0.02334	0.00	13364.70	36094.15
	8	0.02710	0.02710	0.00	6862.80	19588.40
	9	0.02168	0.02168	0.00	17086.35	22161.86
	10	0.03292	0.03292	0.00	8389.19	10508.01
		<b>Avg</b>	<b>0.02559</b>	<b>0.02578</b>	<b>0.76</b>	<b>14228.44</b>
6	1	0.02875	<b>0.02948</b>	2.56	4497.03	13401.10
	2	0.01946	<b>0.02085</b>	7.10	11029.17	20771.88
	3	0.02453	<b>0.03011</b>	22.76	10700.12	32010.98
	4	0.02456	0.02456	0.00	3950.45	10852.74
	5	0.02203	0.02203	0.00	7749.09	21745.00
	6	0.02232	<b>0.02275</b>	1.92	10290.66	30781.95
	7	0.02043	<b>0.02307</b>	12.91	4154.28	12642.09
	8	0.03203	0.03203	0.02	11771.58	35413.34
	9	0.02614	<b>0.02778</b>	6.25	14988.45	34162.73
	10	0.02875	0.02875	0.00	15369.00	35590.27
		<b>Avg</b>	<b>0.02490</b>	<b>0.02614</b>	<b>5.35</b>	<b>9449.98</b>
7	1	0.02558	<b>0.02582</b>	0.93	7548.97	23869.70
	2	0.02230	<b>0.02246</b>	0.69	6418.31	25349.30
	3	0.02797	<b>0.02804</b>	0.24	8159.45	32424.01
	4	0.02797	0.02797	0.00	16931.75	46688.22
	5	0.02355	<b>0.02609</b>	10.80	14365.40	47299.64
	6	0.03798	<b>0.03842</b>	1.15	4553.02	28686.44
	7	0.02315	<b>0.02324</b>	0.38	6451.81	31213.94
	8	0.03671	0.03671	0.00	7339.54	38413.25
	9	0.02986	0.02986	0.00	13421.89	33886.45
	10	0.03287	0.03287	0.00	6780.16	36252.01
		<b>Avg</b>	<b>0.02879</b>	<b>0.02915</b>	<b>1.42</b>	<b>9197.03</b>
8	1	0.02009	<b>0.04073</b>	102.74	8779.82	46878.15
	2	0.01815	<b>0.01897</b>	4.52	7666.96	25017.55
	3	0.02364	<b>0.02426</b>	2.61	8105.85	26452.88
	4	0.02138	<b>0.02435</b>	13.89	18219.98	44569.98
	5	0.02035	<b>0.02186</b>	7.43	5508.15	16425.51
	6	0.02578	<b>0.02653</b>	2.91	6817.52	21542.36
	7	0.01932	<b>0.02018</b>	4.46	7193.17	25976.72
	8	0.02728	<b>0.03106</b>	13.89	11964.66	40895.15
	9	0.02548	<b>0.02555</b>	0.27	9091.24	27732.27
	10	0.02848	<b>0.02866</b>	0.65	10848.98	34312.86
		<b>Avg</b>	<b>0.02299</b>	<b>0.02621</b>	<b>15.34</b>	<b>9419.63</b>

Appendix B12 Results of experimental studies for the problem set 40.800

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
1	1	0.07331	<b>0.08200</b>	11.86	13709.00	38312.57
	2	0.06027	0.06027	0.00	4179.39	21848.03
	3	0.07952	<b>0.07961</b>	0.11	10845.56	44140.25
	4	0.07952	0.07952	0.00	8472.16	45321.93
	5	0.07338	<b>0.07346</b>	0.11	10549.55	40101.03
	6	0.06729	<b>0.08597</b>	27.76	16988.08	48964.56
	7	0.06808	0.06808	0.00	6670.13	33550.01
	8	0.09560	0.09560	0.00	3537.11	18955.65
	9	0.06910	0.06910	0.00	20775.13	47392.53
	10	0.08674	0.08674	0.00	18604.66	44532.86
		<b>Avg</b>	<b>0.07528</b>	<b>0.07803</b>	<b>3.98</b>	<b>11433.08</b>
2	1	0.06428	<b>0.07351</b>	14.36	21650.11	54905.34
	2	0.09626	<b>0.09669</b>	0.45	4266.89	12806.70
	3	0.06788	<b>0.08180</b>	20.51	13645.66	40939.68
	4	0.06959	0.06959	0.00	25250.64	55751.93
	5	0.06419	<b>0.06547</b>	2.00	17222.05	49666.51
	6	0.05886	0.05886	0.00	23665.34	55995.01
	7	0.05970	<b>0.06948</b>	16.40	15697.89	40390.76
	8	0.08318	0.08318	0.00	23800.69	53204.70
	9	0.07573	<b>0.07579</b>	0.07	17050.95	40452.18
	10	0.07575	0.07575	0.00	13532.06	40695.47
		<b>Avg</b>	<b>0.07154</b>	<b>0.07500</b>	<b>5.38</b>	<b>17578.23</b>
3	1	0.07332	<b>0.07337</b>	0.06	17838.03	39576.32
	2	0.06831	<b>0.06895</b>	0.94	10390.50	25371.06
	3	0.07908	<b>0.07923</b>	0.20	28219.16	59343.12
	4	0.06120	<b>0.06124</b>	0.07	6987.19	15979.81
	5	0.07321	0.07321	0.00	32224.67	67949.00
	6	0.08281	<b>0.08614</b>	4.02	9926.56	22858.26
	7	0.07238	<b>0.07858</b>	8.55	11081.59	26861.02
	8	0.09535	0.09535	0.00	35735.00	70970.38
	9	0.08563	<b>0.08665</b>	1.19	13974.13	32332.18
	10	0.08660	<b>0.08675</b>	0.18	17732.01	39176.06
		<b>Avg</b>	<b>0.07779</b>	<b>0.07893</b>	<b>1.52</b>	<b>18410.88</b>
4	1	0.06227	<b>0.06305</b>	1.27	18864.14	43542.39
	2	0.05160	<b>0.05256</b>	1.85	19836.88	45565.83
	3	0.06135	<b>0.06817</b>	11.11	25559.53	58457.08
	4	0.06541	<b>0.06849</b>	4.71	14812.42	36393.63
	5	0.06087	<b>0.06763</b>	11.11	40075.88	91099.92
	6	0.07989	<b>0.08070</b>	1.01	5025.30	14444.38
	7	0.06600	<b>0.06805</b>	3.09	7276.46	21935.54
	8	0.07767	<b>0.08175</b>	5.26	16417.22	40569.39
	9	0.06326	<b>0.07415</b>	17.21	67775.68	148747.75
	10	0.07323	<b>0.07790</b>	6.38	10185.89	23352.08
		<b>Avg</b>	<b>0.06615</b>	<b>0.07024</b>	<b>6.30</b>	<b>22582.94</b>

Appendix B12 Results of experimental studies for the problem set 40.800 (cont.)

Problem Setting	Instance No	Throughput Rate		Deviation(%)	CPU (sec.)	
		BTS	ATS		BTS	ATS
5	1	0.02989	0.02989	0.00	22134.55	52758.95
	2	0.02590	0.02590	0.00	2653.14	13796.13
	3	0.02574	<b>0.02614</b>	1.52	32486.17	73462.19
	4	0.02650	0.02650	0.00	17656.86	43803.57
	5	0.02600	<b>0.02728</b>	4.93	25197.63	58885.11
	6	0.02293	<b>0.02389</b>	4.17	5006.55	18502.95
	7	0.02468	0.02468	0.00	25757.34	60004.53
	8	0.02710	0.02710	0.00	5044.58	18579.01
	9	0.02168	0.02168	0.00	12466.87	33423.59
	10	0.03292	0.03292	0.00	17539.17	43568.19
	<b>Avg</b>	<b>0.02633</b>	<b>0.02660</b>	<b>1.06</b>	<b>16594.29</b>	<b>41678.42</b>
6	1	0.02261	<b>0.02948</b>	30.38	35920.93	71233.23
	2	0.01919	<b>0.02085</b>	8.63	16449.11	32289.59
	3	0.02362	<b>0.03011</b>	27.48	40649.29	80689.95
	4	0.02456	0.02456	0.00	9674.54	18740.45
	5	0.02203	0.02203	0.00	29029.44	57450.25
	6	0.02232	<b>0.02275</b>	1.92	21382.10	42155.57
	7	0.02275	<b>0.02307</b>	1.43	10325.74	20042.85
	8	0.03203	<b>0.03481</b>	8.70	5927.10	11245.57
	9	0.02614	<b>0.03076</b>	17.65	15790.94	30973.25
	10	0.02875	0.02875	0.00	15511.35	30414.07
	<b>Avg</b>	<b>0.02440</b>	<b>0.02672</b>	<b>9.62</b>	<b>20066.05</b>	<b>39523.48</b>
7	1	0.02574	0.02574	0.00	30085.50	62206.83
	2	0.02234	<b>0.02351</b>	5.26	17673.63	37383.09
	3	0.02804	<b>0.02951</b>	5.26	7142.50	16320.83
	4	0.02797	0.02797	0.00	23306.31	48648.45
	5	0.02355	<b>0.02414</b>	2.51	29987.11	62010.05
	6	0.03366	0.03368	0.05	10170.83	22377.49
	7	0.02315	0.02315	0.00	13053.20	28142.23
	8	0.03671	0.03671	0.00	21161.33	44358.49
	9	0.02986	0.02986	0.00	30271.90	62579.63
	10	0.03287	0.03287	0.00	19047.09	40130.01
	<b>Avg</b>	<b>0.02839</b>	<b>0.02871</b>	<b>1.31</b>	<b>20189.94</b>	<b>42415.71</b>
8	1	0.02255	<b>0.05540</b>	145.73	21592.05	59194.26
	2	0.01897	<b>0.02108</b>	11.11	18210.42	44402.02
	3	0.02426	<b>0.02451</b>	1.01	12935.56	30687.38
	4	0.02435	<b>0.02705</b>	11.11	24371.60	60421.09
	5	0.02186	0.02186	0.00	16576.69	40154.32
	6	0.02653	<b>0.02764</b>	4.17	15678.35	37818.64
	7	0.02018	<b>0.02102</b>	4.17	15804.35	38146.24
	8	0.03106	<b>0.03170</b>	2.04	16665.03	40384.01
	9	0.02613	0.02613	0.00	27711.02	69103.58
	10	0.02848	<b>0.02875</b>	0.95	18227.98	44447.68
	<b>Avg</b>	<b>0.02444</b>	<b>0.02851</b>	<b>18.03</b>	<b>18777.31</b>	<b>46475.92</b>

**APPENDIX C**  
**DETAILED RESULTS FOR CHAPTER 6**

Appendix C1 Results of experimental studies for the problem set 5.50

Problem Setting	Instance	Desired Rate (N=25)	Binary Search				Tabu Search				Simulated Annealing					
			Achieved Rate	N	Total # of Iterations	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)
1	1	0.07327	0.07327	14	22	2.39	0.07327	14	11	6	2.15	0.07327	14	10	5	2.34
	2	0.07495	0.07495	20	28	2.56	0.07495	20	10	5	1.78	0.07495	20	10	5	2.53
	3	0.07952	0.07952	17	25	2.43	0.07952	17	10	5	1.73	0.07952	17	10	4	2.17
	4	0.07952	0.07952	9	17	1.12	0.07952	9	11	6	1.34	0.07952	9	10	5	1.42
	5	0.07338	0.07338	14	22	2.01	0.07338	14	11	6	1.86	0.07338	14	10	5	2.12
	Avg	0.07613	0.07613	15	23	2.10	0.07613	15	11	6	1.77	0.07613	15	10	5	2.12
2	1	0.06404	0.06930	9	17	1.79	0.06930	9	11	6	2.09	0.06930	9	9	7	2.78
	2	0.05846	0.05846	25	32	4.49	0.05846	25	7	2	1.79	0.05846	25	10	2	2.67
	3	0.06376	0.06376	25	32	6.05	0.06376	25	7	2	2.48	0.06376	25	10	2	3.50
	4	0.06952	0.06952	25	32	19.59	0.06952	25	7	2	6.85	0.06952	25	10	2	11.00
	5	0.06355	0.06355	25	32	6.08	0.06355	25	7	2	2.39	0.06355	25	10	2	3.63
	Avg	0.06386	0.06492	22	29	7.60	0.06492	22	8	3	3.12	0.06492	22	10	3	4.72
3	1	0.09005	0.09018	20	28	9.47	0.09018	20	10	5	5.35	0.09018	20	12	6	7.47
	2	0.06728	0.06731	24	32	3.52	0.06731	24	9	4	1.87	0.06731	24	10	4	2.33
	3	0.09482	0.09482	25	32	11.30	0.09482	25	7	2	4.15	0.09482	25	10	2	6.75
	4	0.08473	0.08473	24	32	10.28	0.08473	24	9	4	5.48	0.08473	24	10	4	7.00
	5	0.07306	0.07306	25	32	5.65	0.07306	25	7	2	2.29	0.07306	25	10	2	3.32
	Avg	0.08198	0.08202	24	31	8.04	0.08202	24	8	3	3.83	0.08202	24	10	4	5.37
4	1	0.06093	0.06093	25	32	6.68	0.06093	25	7	2	2.82	0.06093	25	10	2	3.85
	2	0.04998	0.04998	25	32	5.02	0.04998	25	7	2	2.04	0.04998	25	10	2	2.93
	3	0.06773	0.06773	25	32	6.07	0.06773	25	7	2	2.50	0.06773	25	10	2	3.57
	4	0.06215	0.06230	24	32	6.21	0.06230	24	9	4	3.39	0.06230	24	10	4	4.23
	5	0.05550	0.05550	25	32	6.86	0.05550	25	7	2	2.90	0.05550	25	10	2	3.98
	Avg	0.05926	0.05929	25	32	6.17	0.05929	25	7	2	2.73	0.05929	25	10	2	3.71

Appendix C1 Results of experimental studies for the problem set 5.50 (cont.)

Problem Setting	Instance	Desired Rate (N=25)	Binary Search				Tabu Search				Simulated Annealing					
			Achieved Rate	N	Total # of Iterations	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for	CPU (sec.)
5	1	0.03676	0.03676	14	22	3.73	0.03676	14	11	6	3.54	0.03676	14	9	5	3.49
	2	0.02362	0.02362	5	9	0.89	0.02362	5	5	0	0.92	0.02362	5	5	4	0.97
	3	0.03713	0.03713	5	9	0.55	0.03713	5	5	0	0.61	0.03713	5	5	4	0.62
	4	0.03085	0.03085	17	25	3.03	0.03085	17	10	5	2.21	0.03085	17	9	4	2.23
	5	0.03369	0.03369	7	14	0.42	0.03369	7	8	3	0.44	0.03369	7	4	3	0.41
	Avg	0.03241	0.03241	10	16	1.72	0.03241	10	8	3	1.54	0.03241	10	6	4	1.54
6	1	0.01485	0.01485	17	25	1.97	0.01485	17	10	5	1.42	0.01485	17	9	4	1.45
	2	0.02256	0.02256	22	29	7.19	0.02256	22	9	4	3.84	0.02256	22	10	4	5.23
	3	0.02540	0.02540	13	20	10.27	0.02540	13	8	3	7.41	0.02540	13	6	3	6.60
	4	0.02375	0.02375	8	16	1.62	0.02375	8	11	6	2.01	0.02375	8	7	5	1.70
	5	0.02508	0.02508	24	32	3.45	0.02508	24	9	4	1.86	0.02508	24	10	4	2.34
	Avg	0.02233	0.02233	17	24	4.90	0.02233	17	9	4	3.31	0.02233	17	8	4	3.46
7	1	0.03088	0.03088	25	32	10.12	0.03088	25	7	2	4.07	0.03088	25	10	2	5.72
	2	0.02349	0.02349	24	32	3.82	0.02349	24	9	4	2.12	0.02349	24	10	4	2.62
	3	0.03719	0.03719	25	32	10.08	0.03719	25	7	2	3.51	0.03719	25	10	2	6.07
	4	0.03101	0.03101	24	32	3.07	0.03101	24	9	4	1.70	0.03101	24	10	4	2.04
	5	0.03369	0.03369	24	32	2.79	0.03369	24	9	4	1.56	0.03369	24	10	4	1.92
	Avg	0.03125	0.03125	24	32	5.98	0.03125	24	8	3	2.59	0.03125	24	10	3	3.67
8	1	0.01460	0.01460	24	32	10.87	0.01460	24	9	4	5.60	0.01460	24	10	4	7.35
	2	0.01835	0.01835	25	32	10.17	0.01835	25	7	2	4.09	0.01835	25	10	2	5.88
	3	0.02401	0.02401	25	32	5.68	0.02401	25	7	2	2.37	0.02401	25	10	2	3.24
	4	0.02235	0.02235	25	32	15.51	0.02235	25	7	2	5.99	0.02235	25	10	2	9.14
	5	0.02461	0.02462	24	32	3.99	0.02462	24	9	4	2.23	0.02462	24	10	4	2.76
	Avg	0.02078	0.02079	25	32	9.24	0.02079	25	8	3	4.06	0.02079	25	10	3	5.67

Appendix C2 Results of experimental studies for the problem set 5.100

Problem Setting	Instance	Desired Rate (N=50)	Binary Search				Tabu Search				Simulated Annealing					
			Achieved Rate	N	Total # of Iterations	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)
1	1	0.082002	0.082002	32	40	8.95	0.082002	32	16	6	9.58	0.082002	32	13	6	12.45
	2	0.074952	0.074952	27	36	5.52	0.074952	27	17	7	5.83	0.074952	27	15	7	6.55
	3	0.079517	0.079517	17	26	3.45	0.079517	17	17	7	4.79	0.079517	17	15	7	5.25
	4	0.079516	0.079516	9	18	1.86	0.079516	9	15	5	2.42	0.079516	14	3	3	1.45
	5	0.073379	0.073379	14	23	3.11	0.073379	14	13	3	3.65	0.073379	14	3	3	2.46
	Avg	0.077873	0.077873	20	29	4.58	0.077873	20	16	6	5.25	0.077873	21	10	5	5.63
2	1	0.075750	0.075750	47	55	14.63	0.075750	47	15	5	8.83	0.075750	47	10	5	9.35
	2	0.059465	0.059465	50	58	17.32	0.059465	50	12	2	7.10	0.059465	50	10	2	9.27
	3	0.063879	0.063879	47	55	21.01	0.063879	47	15	5	12.86	0.063879	47	10	5	12.73
	4	0.059568	0.059568	49	58	14.82	0.059568	49	15	5	8.36	0.059568	49	10	5	9.22
	5	0.083032	0.083032	50	58	30.42	0.083032	50	12	2	11.05	0.083032	50	10	2	17.64
	Avg	0.068339	0.068339	49	57	20.89	0.068339	49	14	4	9.64	0.068339	49	10	4	11.64
3	1	0.073273	0.073273	49	58	22.17	0.073273	49	15	5	12.11	0.073273	49	10	5	13.64
	2	0.066721	0.066721	50	58	34.65	0.066721	50	12	2	14.26	0.066721	50	10	2	18.42
	3	0.078925	0.078925	50	58	20.20	0.078925	50	12	2	8.25	0.078925	50	10	2	10.80
	4	0.056857	0.057020	27	45	16.33	0.056934	39	20	10	27.58	0.057020	27	17	11	29.70
	5	0.073089	0.073089	50	58	20.03	0.073089	50	12	2	8.30	0.073089	50	10	2	10.62
	Avg	0.069773	0.069806	45	55	22.68	0.069788	48	14	4	14.10	0.069806	45	11	4	16.64
4	1	0.065298	0.065298	50	58	33.46	0.065298	50	12	2	13.74	0.065298	50	10	2	17.44
	2	0.053414	0.053414	50	58	19.59	0.053414	50	12	2	8.00	0.053414	50	10	2	10.39
	3	0.072989	0.072989	50	58	26.40	0.072989	50	12	2	11.20	0.072989	50	10	2	13.98
	4	0.063962	0.064161	49	58	26.63	0.064161	49	15	5	15.02	0.064161	49	10	5	15.48
	5	0.059712	0.059712	50	58	33.18	0.059712	50	12	2	13.88	0.059712	50	10	2	17.50
	Avg	0.063075	0.063115	50	58	27.85	0.063115	50	13	3	12.37	0.063115	50	10	3	14.96



Appendix C2 Results of experimental studies for the problem set 5.100 (cont.)

Problem Setting	Instance	Desired Rate (N=50)	Binary Search				Tabu Search				Simulated Annealing					
			Achieved Rate	N	Total # of Iterations	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)
5	1	0.036759	0.036759	14	23	6.35	0.036759	14	13	3	6.19	0.036759	14	3	3	4.26
	2	0.029006	0.029006	34	43	8.58	0.029006	34	15	5	7.07	0.029006	34	10	5	6.69
	3	0.037133	0.037133	5	10	1.34	0.037133	5	6	6	1.75	0.037133	5	6	6	1.81
	4	0.030850	0.030850	17	26	3.87	0.030850	17	17	7	5.82	0.030850	17	11	7	5.07
	5	0.033691	0.033691	7	15	1.19	0.033691	7	15	5	1.48	0.033691	7	8	6	1.67
	Avg	0.033488	0.033488	15	23	4.27	0.033488	15	13	5	4.46	0.033488	15	8	5	3.90
6	1	0.019655	0.019655	17	26	9.84	0.019655	17	17	7	14.24	0.019655	17	11	7	11.95
	2	0.021452	0.021452	32	40	7.85	0.021452	32	16	6	7.92	0.021452	32	13	6	7.36
	3	0.033825	0.033825	49	58	36.85	0.033825	49	15	5	18.08	0.033825	49	10	5	20.51
	4	0.028550	0.028550	49	58	17.32	0.028550	49	15	5	9.64	0.028550	49	10	5	9.98
	5	0.018001	0.018001	47	55	11.25	0.018001	47	15	5	7.24	0.018001	47	10	5	6.99
	Avg	0.024297	0.024297	39	47	16.62	0.024297	39	16	6	11.42	0.024297	39	11	6	11.36
7	1	0.031343	0.031343	50	58	46.57	0.031343	50	12	2	18.75	0.031343	50	10	2	25.02
	2	0.023497	0.023497	39	48	10.47	0.023497	39	17	7	9.52	0.023497	39	15	7	10.83
	3	0.037226	0.037226	49	58	22.29	0.037226	49	15	5	11.34	0.037226	49	10	5	12.65
	4	0.031018	0.031018	44	52	9.72	0.031018	44	14	4	6.18	0.031018	44	15	6	8.36
	5	0.033869	0.033869	49	58	11.82	0.033869	49	15	5	6.79	0.033869	49	10	5	6.99
	Avg	0.031391	0.031391	46	55	20.17	0.031391	46	15	5	10.52	0.031391	46	12	5	12.77
8	1	0.014675	0.014675	49	58	34.07	0.014675	49	15	5	17.33	0.014675	49	10	5	19.27
	2	0.018924	0.018924	50	58	36.18	0.018924	50	12	2	14.20	0.018924	50	10	2	19.06
	3	0.025171	0.025171	50	58	28.41	0.025171	50	12	2	11.23	0.025171	50	10	2	14.80
	4	0.023489	0.023489	50	58	58.41	0.023489	50	12	2	21.40	0.023489	50	10	2	30.84
	5	0.025001	0.025004	49	58	17.16	0.025004	49	15	5	10.38	0.025004	49	10	5	9.95
	Avg	0.021452	0.021453	50	58	34.85	0.021453	50	13	3	14.91	0.021453	50	10	3	18.78

Appendix C3 Results of experimental studies for the problem set 5.200

Problem Setting	Instance	Desired Rate (N=100)	Binary Search				Tabu Search				Simulated Annealing					
			Achieved Rate	N	Total # of Iterations	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)
1	1	0.082002	0.082002	32	56	20.09	0.082002	32	22	7	17.43	0.082002	32	10	4	12.87
	2	0.074952	0.074952	27	37	7.91	0.074952	27	23	8	11.29	0.074952	27	15	6	11.03
	3	0.079517	0.079517	17	27	8.42	0.079517	17	21	6	8.19	0.079517	17	10	5	10.31
	4	0.079516	0.079516	9	19	4.31	0.079516	9	21	6	5.87	0.079516	9	10	6	5.79
	5	0.073379	0.073379	14	24	7.64	0.073379	14	21	6	8.83	0.073379	14	8	5	11.25
	Avg	0.077873	0.077873	20	33	9.67	0.077873	20	22	7	10.32	0.077873	20	11	5	10.25
2	1	0.073524	0.073524	82	91	80.34	0.073524	82	22	7	54.66	0.073524	82	15	6	50.81
	2	0.059610	0.059610	99	109	81.29	0.059610	99	21	6	38.20	0.059610	99	15	6	44.24
	3	0.063879	0.063879	47	56	21.65	0.063879	47	20	5	17.60	0.063879	47	10	5	15.32
	4	0.069585	0.069585	54	64	56.36	0.069585	54	22	7	41.50	0.069585	54	10	5	30.87
	5	0.064190	0.064190	94	197	163.82	0.064190	94	20	5	37.72	0.064190	94	10	5	32.81
	Avg	0.066158	0.066158	75	103	80.69	0.066158	75	21	6	37.94	0.066158	75	12	5	34.81
3	1	0.094845	0.094845	100	111	250.55	0.094845	100	17	2	85.21	0.094845	100	10	3	80.25
	2	0.068029	0.068029	99	109	72.40	0.068029	99	21	6	34.60	0.068029	99	15	6	45.19
	3	0.095359	0.095359	97	106	90.17	0.095359	97	21	6	39.45	0.095359	97	15	6	50.78
	4	0.086525	0.086525	99	109	236.64	0.086525	99	21	6	120.29	0.086525	99	15	6	151.43
	5	0.073168	0.073168	87	97	80.89	0.073168	87	21	6	43.76	0.073168	87	15	6	48.13
	Avg	0.083585	0.083585	96	106	146.13	0.083585	96	20	5	64.66	0.083585	96	14	5	75.16
4	1	0.070071	0.070071	100	109	154.66	0.070071	100	17	2	55.49	0.070071	100	10	2	56.71
	2	0.056413	0.056426	99	109	86.27	0.056426	99	21	6	40.97	0.056426	99	15	6	50.40
	3	0.078401	0.078401	100	109	123.46	0.078401	100	17	2	45.16	0.078401	100	10	2	47.08
	4	0.066478	0.066489	99	109	117.58	0.066489	99	21	6	53.80	0.066489	99	15	6	52.45
	5	0.062756	0.062756	100	109	165.33	0.062756	100	17	2	59.37	0.062756	100	10	2	62.84
	Avg	0.066824	0.066829	100	109	129.46	0.066829	100	19	4	50.96	0.066829	100	12	4	53.90

Appendix C3 Results of experimental studies for the problem set 5.200 (cont.)

Problem Setting	Instance	Desired Rate (N=100)	Binary Search				Tabu Search				Simulated Annealing					
			Achieved Rate	N	Total # of Iterations	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)
5	1	0.036759	0.036759	14	24	11.57	0.036759	14	21	6	14.76	0.036759	14	10	5	13.35
	2	0.029006	0.029006	34	44	12.96	0.029006	34	21	6	15.05	0.029006	34	10	5	12.15
	3	0.037133	0.037133	5	11	3.71	0.037133	5	22	7	4.91	0.037133	5	8	7	5.07
	4	0.030850	0.030850	17	27	7.05	0.030850	17	21	6	8.97	0.030850	17	9	5	7.80
	5	0.033691	0.033691	7	16	4.29	0.033691	7	22	7	5.23	0.033691	7	9	7	5.02
	Avg	0.033488	0.033488	15	24	7.92	0.033488	15	21	6	9.78	0.033488	15	9	6	8.68
6	1	0.019655	0.019655	17	27	14.51	0.019655	17	21	6	17.91	0.019655	17	10	5	16.77
	2	0.028621	0.028621	64	74	40.33	0.028621	64	22	7	31.64	0.028621	64	15	6	28.20
	3	0.033833	0.033833	74	84	72.93	0.033833	74	21	6	37.94	0.033833	74	15	6	49.53
	4	0.028572	0.028572	89	99	61.06	0.028572	89	23	8	39.05	0.028572	89	15	7	39.09
	5	0.018001	0.018001	47	56	13.57	0.018001	47	20	5	11.17	0.018001	47	10	5	9.56
	Avg	0.025736	0.025736	58	68	40.48	0.025736	58	21	6	27.54	0.025736	58	13	6	28.63
7	1	0.031408	0.031408	92	379	659.63	0.031408	92	34	19	133.53	0.031408	92	15	6	116.97
	2	0.023497	0.023497	39	49	13.54	0.023497	39	22	7	13.98	0.023497	39	15	6	13.01
	3	0.037226	0.037226	49	59	27.66	0.037226	49	21	6	20.55	0.037226	49	15	6	24.32
	4	0.031018	0.031018	44	53	11.48	0.031018	44	20	5	9.98	0.031018	44	10	4	8.77
	5	0.033877	0.033877	74	84	31.31	0.033877	74	21	6	19.42	0.033877	74	15	6	21.32
	Avg	0.031405	0.031405	60	125	148.72	0.031405	60	24	9	39.49	0.031405	60	14	6	36.88
8	1	0.014685	0.014685	84	94	74.99	0.014685	84	21	6	41.51	0.014685	84	15	6	44.80
	2	0.019281	0.019281	99	109	138.79	0.019281	99	21	6	61.84	0.019281	99	15	6	73.32
	3	0.025784	0.025784	99	109	123.49	0.025784	99	21	6	52.65	0.025784	99	15	6	65.32
	4	0.023520	0.023520	64	74	102.79	0.023520	64	22	7	62.12	0.023520	64	15	6	72.92
	5	0.025074	0.025074	97	106	77.36	0.025074	97	21	6	39.87	0.025074	97	15	6	45.37
	Avg	0.021669	0.021669	89	98	103.48	0.021669	89	21	6	51.60	0.021669	89	15	6	60.35

Appendix C4 Results of experimental studies for the problem set 10.100

Problem Setting	Instance	Desired Rate (N=50)	Binary Search				Tabu Search				Simulated Annealing					
			Achieved Rate	N	Total # of Iterations	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)
1	1	0.08694	0.08694	13	34	129.15	0.08694	13	17	7	144.89	0.08694	13	8	3	129.92
	2	0.07338	0.07338	38	46	904.85	0.07338	38	14	4	665.61	0.07338	38	10	3	742.16
	3	0.06442	0.06442	13	21	138.69	0.06442	13	13	3	248.68	0.06442	13	5	3	147.19
	4	0.07554	0.07554	23	32	258.63	0.07554	23	14	4	325.46	0.07554	23	9	4	244.45
	5	0.07907	0.07907	28	64	173.71	0.07907	28	21	11	252.00	0.07907	29	10	5	113.40
	Avg	0.07587	0.07587	23	39	321.01	0.07587	23	16	6	324.84	0.07587	23	8	4	275.42
2	1	0.05596	0.05602	48	57	710.80	0.05596	48	15	5	1279.37	0.05597	49	10	5	478.53
	2	0.06253	0.06266	48	57	650.63	0.06266	48	15	5	677.13	0.06266	48	10	5	485.85
	3	0.06730	0.06776	38	45	582.16	0.07183	47	17	7	749.69	0.07281	44	10	4	618.81
	4	0.05925	0.05925	48	57	1096.30	0.05925	48	16	6	664.08	0.05925	48	10	5	754.16
	5	0.06936	0.06936	48	102	1780.16	0.06936	48	17	7	949.01	0.06936	48	10	5	745.20
	Avg	0.06288	0.06301	46	64	964.01	0.06381	48	16	6	863.86	0.06401	47	10	5	616.51
3	1	0.07214	0.07214	48	57	408.61	0.07214	48	15	5	302.91	0.07214	48	10	5	286.74
	2	0.06911	0.06911	43	52	523.49	0.06911	43	16	6	427.25	0.06911	43	10	5	409.00
	3	0.08597	0.08618	53	113	3113.34	0.08618	53	16	6	1501.83	0.08589	53	10	5	1232.49
	4	0.06365	0.06365	48	57	1099.02	0.06365	48	15	5	908.58	0.06365	48	10	5	790.36
	5	0.06660	0.06660	48	57	311.77	0.06660	48	15	5	220.13	0.06660	48	10	5	202.43
	Avg	0.07149	0.07154	48	67	1091.25	0.07154	48	15	5	672.14	0.07148	48	10	5	584.20
4	1	0.05029	0.13110	13	57	289.86	0.17492	16	13	3	176.61	0.22174	24	13	9	496.83
	2	0.04972	0.05134	78	87	2289.82	0.05134	78	16	6	1073.19	0.05134	78	10	4	1041.88
	3	0.04304	0.04389	28	37	277.34	0.04389	28	15	5	352.90	0.04389	28	10	3	354.11
	4	0.05610	0.05610	38	46	415.24	0.05610	38	14	4	363.65	0.05610	38	10	5	367.60
	5	0.05589	0.05631	4	6	108.95	0.05631	4	5	5	120.45	0.05631	4	6	6	169.18
	Avg	0.05101	0.06775	32	47	676.24	0.07651	33	13	5	417.36	0.08587	34	10	5	485.92

Appendix C4 Results of experimental studies for the problem set 10.100 (cont.)

Problem Setting	Instance	Desired Rate (N=50)	Binary Search				Tabu Search				Simulated Annealing					
			Achieved Rate	N	Total # of Iterations	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for	CPU (sec.)
5	1	0.03089	<b>0.03089</b>	6	14	72.38	<b>0.03089</b>	6	15	5	110.45	<b>0.03089</b>	6	8	6	134.00
	2	0.02444	<b>0.02444</b>	12	21	39.05	<b>0.02444</b>	12	14	4	57.47	<b>0.02444</b>	12	7	5	51.78
	3	0.02340	<b>0.02340</b>	8	17	89.76	<b>0.02340</b>	8	15	5	121.52	<b>0.02340</b>	8	6	5	147.27
	4	0.02347	<b>0.02347</b>	8	17	61.09	<b>0.02347</b>	8	15	5	95.53	<b>0.02347</b>	8	6	5	104.83
	5	0.03700	<b>0.03700</b>	28	37	142.33	<b>0.03700</b>	28	17	7	202.83	<b>0.03700</b>	28	10	4	151.60
	Avg	<b>0.02784</b>	<b>0.02784</b>	12	21	80.92	<b>0.02784</b>	12	15	5	117.56	<b>0.02784</b>	12	7	5	117.90
6	1	0.01849	<b>0.01849</b>	23	32	282.88	<b>0.01849</b>	23	14	4	326.56	<b>0.01849</b>	23	10	4	300.24
	2	0.02347	<b>0.02343</b>	23	45	889.73	<b>0.02343</b>	23	30	29	886.30	<b>0.02343</b>	23	16	11	531.35
	3	0.02322	<b>0.02322</b>	48	57	464.83	<b>0.02322</b>	48	15	5	318.26	<b>0.02322</b>	48	10	5	319.25
	4	0.02892	<b>0.02892</b>	48	57	1298.27	<b>0.02892</b>	48	15	5	856.80	<b>0.02892</b>	48	10	5	922.63
	5	0.02505	<b>0.02505</b>	48	57	739.89	<b>0.02505</b>	48	15	5	478.50	<b>0.02505</b>	48	10	5	522.10
	Avg	<b>0.02383</b>	<b>0.02382</b>	38	50	735.12	<b>0.02382</b>	38	18	10	573.28	<b>0.02382</b>	38	11	6	519.11
7	1	0.02165	<b>0.02165</b>	43	52	169.79	<b>0.02165</b>	43	16	6	168.51	<b>0.02165</b>	43	10	5	135.36
	2	0.02169	<b>0.02169</b>	48	101	775.23	<b>0.02169</b>	48	15	5	352.30	0.02169	50	10	2	281.11
	3	0.02167	<b>0.02167</b>	18	27	48.25	<b>0.02167</b>	18	15	5	69.78	<b>0.02167</b>	18	9	4	53.57
	4	0.02558	<b>0.02558</b>	43	52	1092.27	<b>0.02558</b>	43	16	6	787.72	<b>0.02558</b>	43	10	5	793.01
	5	0.01396	<b>0.01396</b>	23	32	45.43	<b>0.01396</b>	23	14	4	58.53	<b>0.01396</b>	23	9	4	48.02
	Avg	<b>0.02091</b>	<b>0.02091</b>	35	53	426.19	<b>0.02091</b>	35	15	5	287.37	<b>0.02091</b>	35	10	4	262.21
8	1	0.01747	<b>0.01744</b>	43	52	211.68	<b>0.01744</b>	43	21	11	365.57	<b>0.01744</b>	43	10	5	174.74
	2	0.01280	<b>0.01280</b>	43	101	947.54	<b>0.01780</b>	43	27	17	804.16	<b>0.01280</b>	43	10	3	390.64
	3	0.03790	<b>0.03790</b>	48	104	1275.82	<b>0.03790</b>	48	16	6	515.81	<b>0.03790</b>	48	10	3	533.24
	4	0.01818	<b>0.01818</b>	48	56	612.83	<b>0.01818</b>	48	16	6	615.39	<b>0.01818</b>	48	10	5	614.58
	5	0.01610	<b>0.01610</b>	49	58	590.80	<b>0.01610</b>	49	15	5	397.63	<b>0.01610</b>	49	10	5	379.50
	Avg	<b>0.02049</b>	<b>0.02048</b>	46	74	727.73	<b>0.02148</b>	46	19	9	539.71	<b>0.02048</b>	46	10	4	418.54

Appendix C5 Results of experimental studies for the problem set 10.200

Problem Setting	Instance	Desired Rate (N=100)	Binary Search				Tabu Search				Simulated Annealing					
			Achieved Rate	N	Total # of Iterations	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)
1	1	0.08694	0.08694	13	21	168.44	0.08694	13	21	6	247.13	0.08694	13	12	6	261.05
	2	0.07338	0.07338	38	47	968.68	0.07338	38	18	3	866.90	0.07338	38	10	3	890.29
	3	0.06442	0.06442	13	22	186.16	0.06442	13	23	8	425.13	0.06442	13	13	8	373.81
	4	0.07554	0.07554	23	33	359.38	0.07554	23	19	4	434.06	0.07554	23	10	4	448.78
	5	0.07907	0.07907	28	37	149.45	0.07907	28	36	21	190.68	0.07907	30	10	5	157.62
	Avg	0.07587	0.07587	23	32	366.42	0.07587	23	23	8	432.78	0.07587	23	11	5	426.31
2	1	0.05682	0.05684	83	252	4756.88	0.05683	87	21	6	1076.00	0.05684	89	15	10	1699.47
	2	0.06398	0.06407	98	108	3259.61	0.06407	98	21	6	1851.57	0.06407	98	15	6	2129.57
	3	0.06836	0.07281	41	84	1286.43	0.06836	100	21	6	2085.34	0.06836	100	15	7	2355.93
	4	0.05929	0.05929	83	93	1025.67	0.05929	83	22	7	1003.66	0.05929	83	10	4	1012.86
	5	0.07228	0.07228	98	108	4225.02	0.07228	98	24	9	2411.39	0.07228	98	15	7	2879.19
	Avg	0.06415	0.06506	81	129	2910.72	0.06417	93	22	7	1685.59	0.06417	94	14	7	2015.40
3	1	0.07394	0.07394	98	108	1434.35	0.07394	98	21	6	653.14	0.07394	98	15	6	865.72
	2	0.06911	0.06911	48	58	701.42	0.06911	48	20	5	594.17	0.06911	48	10	5	529.79
	3	0.07087	0.07087	58	78	2981.95	0.07087	68	17	2	966.84	0.07087	68	10	2	1334.97
	4	0.06506	0.06506	98	108	3127.48	0.06506	98	21	6	1426.11	0.06506	98	15	6	1981.70
	5	0.06802	0.06803	98	108	1403.25	0.06802	98	21	6	723.08	0.06803	98	15	7	945.58
	Avg	0.06940	0.06940	80	92	1929.69	0.06940	82	20	5	872.67	0.06940	82	13	5	1131.55
4	1	0.05423	0.13110	13	47	481.18	0.10816	41	48	33	3277.72	0.13110	13	25	22	1497.85
	2	0.05134	0.05134	78	87	2684.64	0.05134	78	21	6	1654.57	0.05134	78	15	6	1987.15
	3	0.05790	0.05790	73	83	2467.56	0.05790	73	22	7	1598.45	0.05790	73	15	6	1856.71
	4	0.05916	0.05916	98	108	2506.76	0.05916	98	21	6	1567.89	0.05916	98	15	6	1834.65
	5	0.05798	0.05798	58	68	2967.82	0.05798	58	21	6	1884.34	0.05798	58	15	6	2105.55
	Avg	0.05612	0.07150	64	79	2221.59	0.06691	70	27	12	1996.59	0.07150	64	17	9	1856.38

Appendix C5 Results of experimental studies for the problem set 10.200 (cont.)

Problem Setting	Instance	Desired Rate (N=100)	Binary Search				Tabu Search				Simulated Annealing					
			Achieved Rate	N	Total # of Iterations	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for	CPU (sec.)
5	1	0.03089	0.03089	6	15	155.19	0.03089	6	8	8	248.11	0.03089	6	10	8	263.66
	2	0.02444	0.02444	12	22	78.61	0.02444	12	21	6	123.85	0.02444	12	10	6	113.93
	3	0.02769	0.02769	8	18	101.20	0.02769	8	22	7	157.89	0.02769	8	8	7	158.67
	4	0.02347	0.02347	8	18	102.16	0.02347	8	22	7	172.43	0.02347	8	8	7	171.43
	5	0.03700	0.03700	28	61	288.66	0.03700	28	20	5	330.95	0.03700	28	10	4	306.54
	Avg	0.02870	0.02870	12	27	145.16	0.02870	12	19	7	206.65	0.02870	12	9	6	202.85
6	1	0.01849	0.01849	23	33	472.84	0.01849	23	19	4	510.09	0.01849	23	10	4	651.91
	2	0.02454	0.02455	13	29	381.69	0.02457	13	21	6	403.98	0.02457	13	18	15	850.50
	3	0.02364	0.02364	98	108	1894.61	0.02364	98	21	6	935.95	0.02364	98	15	6	1218.30
	4	0.02893	0.02893	73	83	2418.18	0.02893	73	22	7	1543.93	0.02893	73	15	6	1761.57
	5	0.02507	0.02507	98	108	2482.15	0.02507	98	21	6	1171.84	0.02507	98	15	6	1492.72
	Avg	0.02413	0.02414	61	72	1529.89	0.02414	61	21	6	913.16	0.02414	61	15	7	1195.00
7	1	0.02166	0.02166	58	68	359.33	0.02166	58	23	8	397.32	0.02166	58	10	5	275.86
	2	0.02175	0.02175	98	108	1730.67	0.02175	98	21	6	858.47	0.02175	98	15	6	1149.02
	3	0.02167	0.02167	18	28	83.58	0.02167	18	21	6	130.49	0.02167	18	13	6	127.37
	4	0.02558	0.02558	43	53	1162.78	0.02558	43	20	5	1015.34	0.02558	43	10	5	901.12
	5	0.01396	0.01396	23	33	79.23	0.01396	23	19	4	106.28	0.01396	23	10	4	102.12
	Avg	0.02092	0.02092	48	58	683.12	0.02092	48	21	6	501.58	0.02092	48	12	5	511.10
8	1	0.01751	0.01751	58	86	579.59	0.01751	58	24	9	318.83	0.01751	58	10	5	344.60
	2	0.01361	0.01361	98	108	1846.84	0.01361	98	21	6	832.56	0.01361	98	15	6	1188.77
	3	0.01754	0.01754	98	108	2060.08	0.01754	98	21	6	788.88	0.01754	99	10	5	778.42
	4	0.01908	0.05198	53	72	1683.77	0.05198	53	22	7	812.56	0.05198	53	10	5	796.65
	5	0.01661	0.01661	98	108	2368.73	0.01661	98	21	6	813.33	0.01661	98	15	6	1348.45
	Avg	0.01687	0.02345	81	96	1707.80	0.02345	81	22	7	713.23	0.02345	81	12	5	891.38

Appendix C6 Results of experimental studies for the problem set 10.400

Problem Setting	Instance	Desired Rate (N=200)	Binary Search				Tabu Search				Simulated Annealing					
			Achieved Rate	N	Total # of Iterations	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)
1	1	0.08694	0.08694	13	23	381.70	0.08694	13	32	17	658.34	0.08694	14	13	9	650.18
	2	0.07338	0.07338	38	48	1189.39	0.07338	38	20	5	1464.62	0.07338	38	10	5	1336.19
	3	0.06442	0.06442	13	23	344.46	0.06442	13	26	11	929.65	0.06442	13	12	11	785.76
	4	0.07554	0.07554	23	33	577.50	0.07554	23	24	9	1043.04	0.07554	23	14	9	965.27
	5	0.07907	0.07907	28	65	394.93	0.07907	28	25	10	488.97	0.07907	29	15	6	403.82
	Avg	0.07587	0.07587	23	38	577.60	0.07587	23	25	10	916.92	0.07587	23	13	8	828.24
2	1	0.05703	0.05720	124	340	12379.43	0.05704	157	50	46	9105.03	0.05704	140	40	40	11747.86
	2	0.06462	0.06462	188	198	14621.69	0.06462	188	19	4	3362.46	0.06462	188	10	4	3484.95
	3	0.05225	0.05225	118	330	8254.40	0.05225	118	25	10	1814.55	0.05225	118	15	6	2050.34
	4	0.05929	0.05929	83	94	2347.65	0.05929	83	22	7	1386.41	0.05929	83	10	4	1183.07
	5	0.07273	0.07274	153	284	19200.56	0.07274	153	29	14	7441.57	0.07274	153	15	6	5013.38
	Avg	0.06118	0.06122	133	249	10545.51	0.06119	140	29	16	4622.00	0.06119	136	18	12	4695.92
3	1	0.07476	0.07476	198	209	6183.84	0.07476	198	21	6	1657.66	0.07476	198	15	6	2239.66
	2	0.06911	0.06911	48	59	872.26	0.06911	48	21	6	928.48	0.06911	48	15	6	1028.82
	3	0.07087	0.07087	58	79	3256.76	0.07087	68	18	3	1245.67	0.07087	68	10	3	1553.15
	4	0.06578	0.06578	198	209	6789.45	0.06578	198	21	6	2383.43	0.06578	198	15	6	3284.68
	5	0.06889	0.06889	198	209	6541.66	0.06889	198	21	6	1613.45	0.06889	198	15	6	2397.51
	Avg	0.06988	0.06988	140	153	4728.79	0.06988	142	20	5	1565.74	0.06988	142	14	5	2100.76
4	1	0.05735	0.13110	13	29	733.78	0.05737	194	25	10	4232.44	0.13632	158	20	13	5913.80
	2	0.05441	0.05441	124	134	3224.05	0.05441	124	25	10	1674.95	0.05441	124	20	12	1769.45
	3	0.04924	0.04924	198	209	5376.66	0.04924	198	21	6	2000.32	0.04924	198	15	6	2373.12
	4	0.05993	0.05993	198	209	5679.43	0.05993	198	21	6	2501.34	0.05993	198	15	6	2857.11
	5	0.06415	0.06415	198	209	6798.31	0.06415	198	21	6	2365.41	0.06415	198	15	6	2543.30
	Avg	0.05701	0.07176	146	158	4362.45	0.05702	182	23	8	2554.89	0.07281	175	17	9	3091.36



Appendix C6 Results of experimental studies for the problem set 10.400 (cont.)

Problem Setting	Instance	Desired Rate (N=200)	Binary Search				Tabu Search				Simulated Annealing					
			Achieved Rate	N	Total # of Iterations	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for	CPU (sec.)
5	1	0.03089	0.03089	6	16	349.83	0.03089	6	26	11	649.63	0.03089	6	13	11	655.06
	2	0.02444	0.02444	12	23	248.57	0.02444	12	25	10	399.53	0.02444	12	14	10	397.96
	3	0.02769	0.02769	8	19	280.27	0.02769	8	26	11	460.90	0.02769	8	12	11	457.57
	4	0.02347	0.02347	8	19	215.50	0.02347	8	26	11	439.33	0.02347	8	12	11	438.27
	5	0.03700	0.03700	28	39	500.65	0.03700	28	22	7	814.20	0.03700	28	13	8	794.32
	Avg	0.02870	0.02870	12	23	318.96	0.02870	12	25	10	552.72	0.02870	12	13	10	548.64
6	1	0.01849	0.01849	119	332	8967.45	0.01849	119	35	20	6134.80	0.01849	130	20	12	5774.05
	2	0.02104	0.02104	83	94	2390.76	0.02104	83	22	7	1110.60	0.02104	83	15	6	1256.76
	3	0.02371	0.02371	123	134	3224.05	0.02371	123	24	9	2294.92	0.02371	123	15	6	2324.70
	4	0.02893	0.02893	73	84	2432.50	0.02893	73	22	7	1716.74	0.02893	73	15	6	1867.54
	5	0.02507	0.02507	98	109	2475.61	0.02507	98	21	6	1398.82	0.02507	98	15	6	1681.29
	Avg	0.02345	0.02345	99	151	3898.07	0.02345	99	25	10	2531.18	0.02345	101	16	7	2580.87
7	1	0.02171	0.02171	153	164	2886.10	0.02171	153	23	8	1561.59	0.02171	153	15	6	1496.09
	2	0.02177	0.02177	158	169	5221.89	0.02177	158	22	7	2105.10	0.02177	158	15	6	2330.49
	3	0.02167	0.02167	18	29	204.30	0.02167	18	25	10	368.13	0.02167	18	15	10	354.88
	4	0.02558	0.02558	43	54	1404.05	0.02558	43	19	4	1332.59	0.02558	43	10	4	1332.20
	5	0.01396	0.01396	23	34	232.83	0.01396	23	24	9	383.40	0.01396	23	13	9	379.39
	Avg	0.02094	0.02094	79	90	1989.83	0.02094	79	23	8	1150.16	0.02094	79	14	7	1178.61
8	1	0.01755	0.01755	110	321	8923.12	0.01755	184	33	18	3588.02	0.01755	188	10	5	1804.60
	2	0.01391	0.01391	198	209	8666.31	0.01391	198	21	6	2383.44	0.01391	198	15	6	3282.95
	3	0.01768	0.01768	198	209	8258.33	0.01768	198	21	6	1955.49	0.01768	198	15	6	2437.90
	4	0.01947	0.01947	198	209	8042.54	0.01947	198	21	6	2194.33	0.01947	198	15	6	2443.73
	5	0.01670	0.01670	193	219	8069.42	0.01670	193	22	7	2083.24	0.01670	193	15	6	2501.95
	Avg	0.01706	0.01706	179	233	8391.94	0.01706	194	24	9	2440.90	0.01706	195	14	6	2494.23

Appendix C7 Results of experimental studies for the problem set 20.200

Problem Setting	Instance	Desired Rate (N=100)	Tabu Search					Simulated Annealing				
			Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)
1	1	0.07349	<b>0.07984</b>	35	22	7	7316.12	<b>0.07984</b>	35	10	5	15673.98
	2	0.08056	<b>0.08056</b>	35	18	3	6714.03	<b>0.08056</b>	35	10	5	15768.88
	3	0.07924	<b>0.07924</b>	25	22	7	3533.75	<b>0.07924</b>	25	10	5	3166.35
	4	0.10068	<b>0.10068</b>	65	22	7	7368.36	<b>0.10068</b>	65	15	6	13665.39
	5	0.07167	<b>0.07167</b>	65	22	7	5048.66	<b>0.07167</b>	65	10	5	12870.34
	Avg	<b>0.08113</b>	<b>0.08304</b>	48	21	6	<b>5666.20</b>	<b>0.08304</b>	48	11	5	<b>12228.99</b>
2	1	0.04606	<b>0.06495</b>	13	9	9	4897.94	<b>0.06495</b>	13	26	26	24780.50
	2	0.02106	<b>0.02106</b>	95	22	7	8970.16	<b>0.02106</b>	95	10	5	27880.45
	3	0.00642	<b>0.00775</b>	10	8	8	3094.13	<b>0.00775</b>	10	26	26	16287.12
	4	0.03307	<b>0.14200</b>	16	19	4	1666.93	<b>0.07375</b>	16	37	35	32218.58
	5	0.06737	<b>0.06737</b>	95	22	7	7654.32	<b>0.06737</b>	95	10	5	31563.91
	Avg	<b>0.03480</b>	<b>0.06063</b>	46	16	7	<b>5256.70</b>	<b>0.04698</b>	46	22	19	<b>26546.11</b>
3	1	0.06596	<b>0.06666</b>	84	22	7	15563.24	<b>0.06706</b>	95	15	10	24679.21
	2	0.06657	<b>0.07953</b>	93	36	21	14789.21	<b>0.08204</b>	104	20	12	32728.48
	3	0.06933	<b>0.10256</b>	10	8	8	5110.89	<b>0.07128</b>	12	26	26	51066.71
	4	0.06896	<b>0.06896</b>	95	22	7	16785.56	<b>0.06896</b>	95	10	5	35768.87
	5	0.07574	<b>0.07574</b>	95	22	7	18945.67	<b>0.07574</b>	95	10	5	38756.22
	Avg	<b>0.06770</b>	<b>0.07301</b>	80	22	10	<b>14238.91</b>	<b>0.07869</b>	75	16	12	<b>36599.90</b>
4	1	0.00822	0.04284	10	6	6	1995.65	0.04284	10	18	18	9489.94
	2	0.04593	0.04973	75	21	6	6746.21	0.09997	66	15	9	14729.71
	3	0.06137	0.06137	95	22	7	16681.57	0.06137	95	10	5	25748.99
	4	0.08784	0.08839	95	22	7	21843.12	0.08839	95	10	5	28955.95
	5	0.06164	0.06179	45	18	3	1506.31	0.06179	45	10	5	2938.84
	Avg	<b>0.05300</b>	<b>0.06082</b>	64	18	6	<b>9754.57</b>	<b>0.07087</b>	62	13	8	<b>16372.69</b>

Appendix C7 Results of experimental studies for the problem set 20.200 (cont.)

Problem Setting	Instance	Desired Rate (N=100)	Tabu Search					Simulated Annealing				
			Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)	Achieved Rate	N	Total # of Iterations	# of Iterations for Convergence	CPU (sec.)
5	1	0.02800	<b>0.02800</b>	10	7	7	3978.64	<b>0.02800</b>	10	7	7	3332.24
	2	0.02575	<b>0.02575</b>	85	21	6	22118.11	<b>0.02575</b>	85	15	6	24115.50
	3	0.02666	<b>0.02666</b>	25	19	4	3999.72	<b>0.02666</b>	25	10	5	3597.76
	4	0.02610	<b>0.02610</b>	15	22	7	3700.42	<b>0.02610</b>	15	8	6	3093.40
	5	0.02872	<b>0.02872</b>	15	22	7	4999.47	<b>0.02872</b>	15	8	6	3828.02
	Avg	<b>0.02705</b>	<b>0.02705</b>	30	18	6	<b>7759.27</b>	<b>0.02705</b>	30	10	6	<b>7593.38</b>
6	1	0.00441	<b>0.00441</b>	55	24	9	6109.73	<b>0.00441</b>	55	10	5	3194.51
	2	0.02104	<b>0.02104</b>	95	22	7	5879.65	<b>0.02104</b>	95	10	5	4875.33
	3	0.01724	<b>0.01724</b>	25	19	4	4611.94	<b>0.01724</b>	25	10	5	4267.50
	4	0.02337	<b>0.02337</b>	95	22	7	24501.72	<b>0.02337</b>	95	10	5	12655.98
	5	0.02073	<b>0.02073</b>	95	22	7	8674.23	<b>0.02073</b>	95	10	5	5745.41
	Avg	<b>0.01736</b>	<b>0.01736</b>	73	22	7	<b>9955.45</b>	<b>0.01736</b>	73	10	5	<b>6147.75</b>
7	1	0.02141	<b>0.02212</b>	62	22	7	12549.68	<b>0.02164</b>	65	15	6	12270.80
	2	0.02226	<b>0.02226</b>	95	22	7	39711.75	<b>0.02226</b>	95	10	5	49832.82
	3	0.02284	<b>0.02284</b>	95	39	24	34771.29	<b>0.02284</b>	96	10	5	26093.93
	4	0.02218	<b>0.02219</b>	58	30	15	25328.18	<b>0.02218</b>	62	10	5	25979.42
	5	0.02352	<b>0.06072</b>	33	19	4	9787.22	<b>0.05828</b>	32	19	13	29870.25
	Avg	<b>0.02244</b>	<b>0.03003</b>	69	26	11	<b>24429.62</b>	<b>0.02944</b>	70	13	7	<b>28809.44</b>
8	1	0.00820	<b>0.00820</b>	95	22	7	12205.36	<b>0.00820</b>	95	10	5	9836.94
	2	0.01666	<b>0.01666</b>	95	22	7	18678.65	<b>0.01666</b>	95	10	5	15468.34
	3	0.02127	<b>0.02127</b>	95	22	7	16435.23	<b>0.02127</b>	95	10	5	20547.66
	4	0.02382	<b>0.02382</b>	95	22	7	20456.17	<b>0.02382</b>	95	10	5	26395.45
	5	0.02453	<b>0.02453</b>	75	22	7	11356.78	<b>0.02453</b>	75	15	9	14454.14
	Avg	<b>0.01889</b>	<b>0.01889</b>	91	22	7	<b>15826.44</b>	<b>0.01889</b>	91	11	6	<b>17340.51</b>