TABU SEARCH WITH FULLY SEQUENTIAL PROCEDURE FOR SIMULATION OPTIMIZATION

A THESIS
SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL
ENGINEERING AND THE INSTITUTE OF ENGINEERING AND
SCIENCE OF BİLKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Savaş Çevik
August, 2003

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

Prof. İhsan Sabuncuoğlu (Principal Advisor)

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

Prof. Erdal Erel

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

Assoc. Prof. Mustafa Akgül

Approved for the Institute of Engineering and Science:

Prof . Mehmet Baray
Director of Institute of Engineering and Science

Prof. Kürşat Aydoğan
Director

ABSTRACT

TABU SEARCH WITH FULLY SEQUENTIAL PROCEDURE FOR SIMULATION OPTIMIZATION

Savaş Çevik

M.S. in Industrial Engineering

Advisor: Prof. İhsan Sabuncuoğlu

August,2003

Simulation is a descriptive technique that is used to understand the behaviour of both conceptual and real systems. Most of the real life systems are dynamic and stochastic that it may be very difficult to derive analytical representation. Simulation can be used to model and to analyze these systems. Although simulation provides insightful information about the system behaviour, it cannot be used to optimize the system performance. With the development of the metaheuristics, the concept simulation optimization has became a reality in recent years. A simulation optimization technique uses simulation as an evaluator, and tries to optimize the systems performance by setting appropriate values of simulation input. On the other hand, statistical ranking and selection procedures are used to find the best system design among a set of alternatives with a desired confidence level. In this study, we combine these two methodologies and investigate the performance of the hybrid procedure. Tabu Search (TS) heuristic is combined with the Fully Sequential Procedure (FSP) in simulation optimization context. The performance of the combined procedure is examined in four different systems. The effectiveness of the FSP is assessed considering the computational effort and the convergence to the best (near optimal) solution.

Keywords: Simulation Optimization, Ranking and Selection, Tabu Search, Fully Sequential Procedure.

ÖZET

SİMÜLASYONLA ENİYİLEME İÇİN TABU ARAMASI İLE BİRLEŞTİRİLMİŞ SIRALI SEÇİM METODU

Savaş Çevik
Endüstri Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. İhsan Sabuncuoğlu
Ağustos 2003

Simülasyon var olan veya tasarım aşamasındaki sistemlerin davranışlarını anlamak için kullanılan tanımlayıcı bir araçtır. Var olan sistemlerin çoğu dinamik ve rassal bir yapıya sahiptir. Bu durum sistemin analitik bir modelini çıkarmayı güçleştirebilir. Simülasyon bu tür sistemlerin modellenmesinde ve analiz edilmesinde kullanılabilir. Sistem davranışı hakkında çok yararlı bilgiler sağlasa da, simülasyon tek başına sistem performansını eniyilemede kullanılamaz. Son yıllarda, sezgisel yöntemlerin geliştirilmesiyle birlikte, simülasyonla eniyileme kavramı büyük önem kazanmıştır. Simülasyonla eniyileme teknikleri simülasyonu bir değerleme aracı olarak kullanır ve simülasyon girdi degerlerini uygun şekilde ayarlayarak sistemin performansını en iyilemeye calışır. Diğer taraftan, istatiksel sıralama ve seçim metodları belirli bir güven seviyesiyle alternatif sistemler içinden en iyi sistemi seçmek için kullanılırlar. Bu çalışmada, bu iki metodolojiyi birleştirdik ve ortaya çıkan hibrid metodun performansını inceledik. Tabu Araması, Tamamen Sıralı Seçim metoduyla simülasyonla eniyileme bağlamında birleştirildi. Ortaya çıkan metodun performansı dört farklı sistem üzerinde denendi. Tamamen Sıralı Seçim metodunun etkinligi hesapsal efor ve en iyi çözüme yakınsama göz önünde tutularak degerlendirildi.

Anahtar kelimeler: Simulasyonla eniyileme, sıralama ve seçim metodları, tabu araması, tamamen sıralı seçim metodu.

Arife ve Seniye ye,

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Prof. İhsan Sabuncuoğlu who supervised me through all stages of this research. He always encouraged me to go for this thesis.

I am also indebted to Prof. Erdal Erel and Assoc. Prof. Mustafa Akgül for their accepting to read and review this thesis and valuable comments.

I would also like to thank Prof. Barbaros Tansel for his support. He helped and supported this thesis to its end.

I want to express my gratefulness to my beloved family, they are the greatest. Especially my brother Barış for his patience! with me while we were playing Diablo II.

I want to thank to my friends Nur, Gürol, Batu, Halil, Arda, Sezgin, Ozan, Burhan, Ünal, İbrahim, Çağatay, Aykut, Sabri, Banu, Abdullah, Burhaneddin, Rabia, and Ömer for their friendship and morale support all the time. I especially want to memorize my friends Kutay, Osman, Deniz, Hasan, Hasan (Doğa's father), and Derya.

As a final word, I want to thank Conan The Barbarian and The Cranberries. They make me feel great.

May the force be with you.

TABLE OF CONTENTS

CHAPTER 1 ..... **1**

INTRODUCTION **1**

1.1. BASIC CONCEPTS 1

1.2. SIMULATION AS AN OPERATIONS RESEARCH (OR) TECHNIQUE 3

1.3. AIM OF THE STUDY 4

CHAPTER 2 ..... **9**

LITERATURE REVIEW **9**

2.1. INTRODUCTION 9

2.2. THE GENERAL STRUCTURE OF A SIMULATION OPTIMIZATION PROBLEM 9

2.2.1. Simulation Optimization Methodologies *11*

2.2.1.1. Gradient-based Search Methods 11

2.2.1.2 Stochastic Approximation 11

2.2.1.3. Response Surface Methodology (RSM) 13

2.2.1.4. Heuristic Methods 13

2.2.1.5. Statistical Methods 15

2.2.2. LITERATURE SURVEY SIMULATION OPTIMIZATION 15

2.3. RANKING AND SELECTION 22

2.3.1. Multiple Comparison Procedures (MCPs) *23*

2.3.2. Ranking and Selection Procedures *24*

2.3.2.1. Subset Selection 24

2.3.2.2. Indifference-zone Selection 24

2.3.3. Literature Survey Ranking and Selection *25*

CHAPTER 3 ..... **32**

PROPOSED STUDY **32**

3.1. METHODOLOGY 32

3.1.1. Tabu Search *33*

3.1.1.1. Tabu Algorithm 35

3.1.2. The Fully Sequential Procedure (FSP) *36*
3.1.2.1. The Fully Sequential Algorithm (Kim and Nelson (2000)) 36
3.2. EXPERIMENTAL SETTINGS 39
3.2.1. Manufacturing Problem *39*
3.2.2. Inventory Control Problem *41*
3.2.3. Job Shop Problem *43*
3.2.4. Three-stage Buffer Allocation Problem *46*

CHAPTER 4 48

EXPERIMENTAL RESULTS 48

4.1. MANUFACTURING PROBLEM 48
4.1.1. The Construction *48*
4.1.2. Results *50*
4.2. INVENTORY CONTROL PROBLEM 59
4.2.1. The Construction *59*
4.2.2. The Results *60*
4.3. JOB SHOP PROBLEM 68
4.3.1. The Construction *68*
4.3.2. The Results *68*
4.4. THREE-STAGE BUFFER ALLOCATION PROBLEM 72
4.4.1. The Construction *72*
4.4.2. The Results *73*

CHAPTER 5 76

CONCLUSIONS 76

REFERENCES 80

LIST OF TABLES

Table 1.1. The most known simulation optimization software packages, and supported simulation software. 7
Table 2.1. The summary of some studies in the literature. 21
Table 2.2. Some of the R&S procedures in the literature. 30
Table 3.1. Model parameters of the production problem. 40
Table 3.2. The parameters of the inventory control problem. 42
Table 3.3. The routings of the jobs. 44
Table 3.4. The distances between the stations (in feet). 44
Table 3.5. The mean processing times of the machines. 45
Table 3.6. The adjusted profits/costs of jobs/machines. 46
Table 4.1. The number of machines in the best solutions of the first nine iterations. 49
Table 4.2. The results of STS method. 50
Table 4.3. The results of TS+FSP method. 50
Table 4.4. The computational times of the methods. 53
Table 4.5. The performances of the solutions according to 100 replications. 54
Table 4.6. The results of TS+FSP method when $n_0 = 10$. 54
Table 4.7. The results of STS method with doubled processing times. 55
Table 4.8. The results of TS+FSP method with doubled processing times. 55
Table 4.9. The values of β_B and β_D parameters of related machines. 57
Table 4.10. The results of STS method with breakdowns. 57
Table 4.11. The results of TS+FSP method with breakdowns. 58
Table 4.12. The computational times of the methods. 58
Table 4.13. The long run performances of the best solutions found by both methods. 58
Table 4.14. The neighbours of the solution (-5,10). 60
Table 4.15. The results of STS method. 60
Table 4.16. The results of TS+FSP method. 61
Table 4.17. The computational times of the methods. 62
Table 4.18. The performances of the solutions based on 100 replications. 62
Table 4.19. The results of STS method with doubled number of neighbour solutions..... 64
Table 4.20. The results of TS+FSP method with doubled number of neighbour solutions..... 64
Table 4.21. The results of STS method with random shelf lives. 65
Table 4.22. The results of TS+FSP method with random shelf lives..... 65

Table 4.23. The performances of the solutions based on 100 replications. ……..… 66
Table 4.24. The results of STS method. ……..…….. 68
Table 4.25. The results of TS+FSP method. …….. 69
Table 4.26. The computational times of the methods. …….. 69
Table 4.27. The performances of the solutions based on 100 replications. ……... 70
Table 4.28. The results of STS method. ……..…….. 71
Table 4.29. The results of TS+FSP method. …….. 71
Table 4.30. The performances of the solutions based on 100 replications. ……... 72
Table 4.31. The results of STS method. ……..…….. 73
Table 4.32. The results of TS+FSP method. …….. 73
Table 4.33. The performances of the solutions based on 100 replications. …….. 74
Table 4.34. The computational times of the methods. …….. 74

LIST OF FIGURES

Figure 1.1. Simulation model of a system. …….. 2

Figure 1.2. Working of a simulation model...…….…….. 2

Figure 1.3. Simulation optimization model. …… 6

Figure 1.4. The comparison of an ordinary TS approach and our approach. …….. 8

Figure 2.1. Classification of Simulation Optimization Methodologies…… 12

Figure 3.1. The graph of $W_{ij}(r)$…….. 38

Figure 3.2. The outline of the production facility. …….…….. 39

Figure 3.3. The outline of the job shop problem. …… 43

Figure 3.4. The outline of the three-stage buffer allocation problem. ……. 46

Figure 4.1. The convergence of the methods when the initial solution is (1111111)….. 52

Figure 4.2. The convergence of the methods when the initial solution is (3 6 3 5 2 4 2). 52

Figure 4.3. The convergence of the methods when the initial solution is (2 4 2 4 2 4 2). 59

Figure 4.4. The convergences of the methods when the initial solution is (2,15). …….. 63

Figure 4.5. The convergences of the methods when the initial solution is (-1,5). …….. 63

Figure 4.6. The convergences of the methods when the initial solution is (2,10). …….. 67

Figure 4.7. The convergences of the methods when the initial solution is (4,5). …… 67

Figure 4.8. The convergences of the methods when the initial solution is (4 1 4 2 2 2). 70

Figure 4.9. The convergences of the methods when the initial solution is (2 2 2 2 2 2). 71

Figure 4.10. The convergences of the methods when the initial solution is (2 2 2 2 18). 75

CHAPTER 1

INTRODUCTION

1.1. Basic Concepts

Simulation is a very useful tool in understanding the behavior of both existing and conceptual systems. It is used in a wide variety of areas from manufacturing to military applications. Although, there are many definitions to simulation, the simplest one is “the imitation of life”. The aim of the simulation is to give insights, to provide information about the system being simulated. According to the simulation results (output or response), one can observe if the system operates as it is intended to be. The factors that affect its performance can be detected and by adjusting these factors, system performance may be improved to the desired level. It is also beneficial to simulate conceptual systems that are considered to build. A lot of information can be gathered from simulation output and analyzing this data, the conceptual system may be redesigned in order to improve the system performance.

The *Oxford English Dictionary* gives the following definition of simulation: “The technique of imitating the behaviour of some situation or process (whether economic, military, mechanical, etc.) by means of a suitably analogous situation or apparatus, especially for the purpose of study or personnel training”. Shannon (1975) defines simulation as “the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of the system”. Following figure illustrates a simulation model of a system:

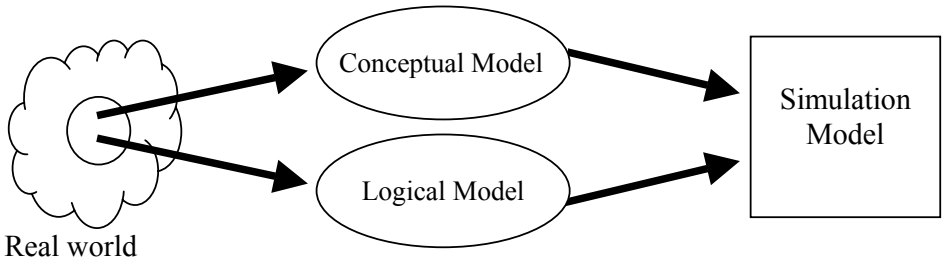


Figure 1.1. Simulation model of a system.

where *Conceptual Model* defines and integrates model elements (e.g., entities, processes, resources, queues, etc.), and *Logical Model* defines logical interactions between these elements (e.g., precedence relations, queuing strategies, etc.). After combining these separate models in to one model called *Simulation Model*, one can evaluate system performance and detect various effects that manipulate the output. Following figure shows working of a simulation model:

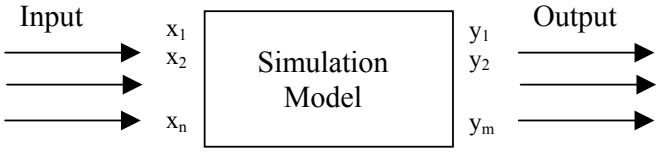


Figure 1.2. Working of a simulation model.

Simulation models offer a completely controllable environment. Every aspect of the system is controllable to the experimenter. Simulation models are flexible. When needed parameters and variables of the system can easily be changed. This is a very useful feature especially when employing "what-if" questions in order to improve the performance of the system. Finally, simulation time is completely independent from the real time in the way that one can speed up the simulation time to quickly access the simulation output, while the other can slow it down in order to be able to observe certain processes (zooming) in the system.

Simulation experiments are done according to prepared plans called experimental design. "A simulation experiment can be defined as a test or series of tests in which meaningful changes are made to the input variables of a simulation model so that we may observe and identify the reasons for changes in the output variables" (Carson and Maria (1997)). Experimental design is another crucial point in simulation. The factors that are considered to have effects on system performance are determined. After determination of these factors, simulation experiments are conducted for different values, which are defined as levels, of these factors. By

analyzing simulation output, one may find out the factors that are significant and extract the optimal levels for these factors. Significant factors and the levels associated with them may be used to redesign the system and improve the system performance.

1.2. Simulation as an Operations Research (OR) Technique

Consider a manufacturing facility that faces a decision making problem. There are two available options: to build another job shop or to rearrange the existing one in order to meet the increasing demand. This is a critical decision. If the facility decides to build another job shop where they actually can meet the increasing demand by rearranging their manufacturing environment (existing job shop), then they would have invested a lot of money in vain. Off course this is an undesirable situation. On the other hand, they may decide rearranging their facility and it may turn out that rearrangement fails to meet the demand. This is even worse because there will be an additional cost of loosing customer to the cost of rearrangement. The cost of loosing customer is much more crucial than the former. Making use of simulation can help to make decisions.

First, the two alternative systems are examined, and then conceptual and logical models of the alternatives are built to combine them in a simulation model. After the construction of the simulation models for both systems, input data (e.g., distribution functions of the inter-arrival time of demands, of the amount of demands, and of the processing times of the jobs etc.) analysis is conducted. This analysis is very important because simulation models are driven by input data, and using wrong or inadequate data may (actually does) lead unreliable output hence wrong decision. So, statistical analysis tools must be utilized for both input and output data analysis.

Assuming input data and simulation models are ready, simulation experiments are performed according to an experimental design to obtain the output data. Analyzing this data, one can decide which one of the alternatives is more appropriate. The manufacturing facility can now select the best alternative. Of course this simulation study comes with a certain cost, but this cost is not comparable to the cost of making wrong decision. From this point of view, simulation can be seen as an Operations Research, OR, technique. As with all other OR techniques, simulation is utilized to make the best decision among alternatives as in above example.

Simulation is superior in comparison to other OR techniques when dealing with stochastic systems those are too complex to derive an analytical (mathematical) model, which consist of the majority of the real life systems. Since analytical model is too hard to obtain if not impossible or even does not exist, classical deterministic OR techniques cannot be used to solve these problems. Some assumptions may be made in order to come up with an analytical model but this approach may divert us from the real problem.

Simulation is welcome to overcome this deficiency. As we describe it, it is simply the imitation of life and any real world system can be transformed into a simulation model. And once you have the simulation model, every aspects of the system can be inspected. By analyzing output data and employing some "what if" questions, the model can be easily modified. This allows researchers to redesign the system being simulated in order to improve the performance of the system.

Unfortunately, simulation does not provide optimal solution, which makes it a descriptive tool. This is the major drawback of the simulation when compared to other OR techniques. But it has many advantages that surpass this drawback. Furthermore, with the incredible advances in the computer science and technology and the emergence of the metaheuristics, the concept "simulation optimization" attracts many researchers and scientist during the last decade. A lot of papers have been published and are being published.

The main reason behind this attraction is, as we mentioned, almost any real life system has a stochastic nature that is hard or impossible to describe analytically, and one of the simplest ways to optimize these systems without being diverted from the very essence of the problem is to make use of simulation within a simulation optimization framework.

1.3. Aim of the Study

The aim of this study is to examine the effects of a Ranking and Selection (R&S) tool namely Fully Sequential Procedure (FSP) due to Kim and Nelson (2001), on the output of a heuristic search algorithm, Tabu Search, in the context of simulation optimization. The approach is simply embedding the FSP in TS.

Our motivation is to find better search directions in each iteration of TS. In TS a neighbourhood set of solutions to the current solution is created at each iteration.

The solutions in the neighbourhood are evaluated to find the best of them. Evaluation of the solutions is based on taking arbitrarily number of observations (replications). Instead, one can use a statistical technique to find the best among neighbours which may give better search directions.

We try to find out if employing the FSP increases the solution quality. If it does, is it worth increasing computational effort. How does the FSP affect the convergence behaviour of the search. And finally, if the FSP should be implemented to increase the efficiency of the search or not.

TS is a very effective global search algorithm, which is first introduced by Glover (1986). FSA is a recently developed ranking and selection tool, which reduces the computational effort dramatically when compared to conservative ranking and selection procedures. Detailed descriptions of these methods will be given in Chapter 3. In what follows, we will introduce the concepts of *simulation optimization* and *ranking and selection*.

Simulation optimization is defined as optimization of performance measures (e.g., throughput, waiting time in the system, and production cost or profit etc.) by adjusting model settings (input variables or decision variables) according to simulation output of previous settings. Another definition is "the process of finding the best input variable values from among all possibilities without explicitly evaluating each possibility" (Carson and Maria (1997)). Law and McComas (2000) define simulation optimization as "orchestration of the simulation of sequence of system configurations (each configuration corresponds to particular settings of the decision variables (factors)) so that a system configuration is eventually obtained that provides an optimal or near optimal solution."

The idea is using simulation as an evaluation function or an evaluator. First, simulate the system with current model settings, and then observe the output and take this data to an optimization algorithm. The algorithm analyses the output by means of the effects of the current model settings on the output. According to this analysis algorithm generates new model settings to simulate the system with this new model settings. The process repeats itself until a certain stopping condition is satisfied (e.g. a certain improvement has been made or pre-specified number of iterations has passed etc.) The following figure illustrates the logic of the simulation optimization model:

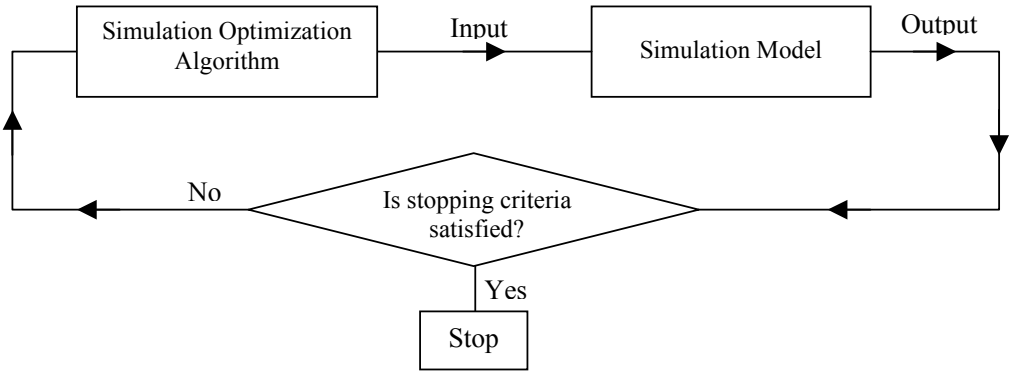


Figure 1.3. Simulation optimization model.

Here are some application areas of simulation optimization: *Manufacturing systems*; one can build a simulation model for a specific production facility (e.g., job shop, assembly line etc.), and use this model in order to maximize the number of the finished jobs or products while minimizing the cost incurred in conjunction with a simulation optimization technique. *Supply chain systems*; simulation optimization can be used in order to minimize inventory levels and response times while maximizing fill rates. *Queuing systems*; waiting times of customers or jobs can be minimized or total number of served customers can be maximized by making use of simulation optimization. *Inventory control models*; one can use a simulation optimization method in order to determine optimal levels of s, re-order point, and S, order-up-to point, that minimize the cost which consists of ordering, holding, and shortage costs. There are many other application areas in addition to above ones.

For example, a manufacturing facility wants to optimize the number of machines in one of their job shops in order to maximize the throughput. The above approach, illustrated in Figure 1.3, may be used until the maximum throughput is reached i.e., adding one more machine does not improve the objective function. The number of machines at this point is the optimal solution. Actually, due to the stochasticity, is not the optimal but very close to optimal. This is a very simple example. When the size and the complexity of the problem increases, more sophisticated algorithms are needed to come up with near optimal solutions.

There are many simulation optimization algorithms in the literature. But the most commonly used algorithms are called metaheuristics, which includes Tabu Search (TS), Genetic Algorithm (GA), and Simulated Annealing (SA). The role of the metaheuristics in simulation optimization's popularity is unquestionable.

Many software developers for simulation modeling and analysis add a simulation optimization module into their software packages. Since simulation

optimization applications are widely used nowadays, software vendors want to make their product preferable to others. Following table summarizes the most known optimization packages and supported simulation software (adapted from Law and McComas (2002)):

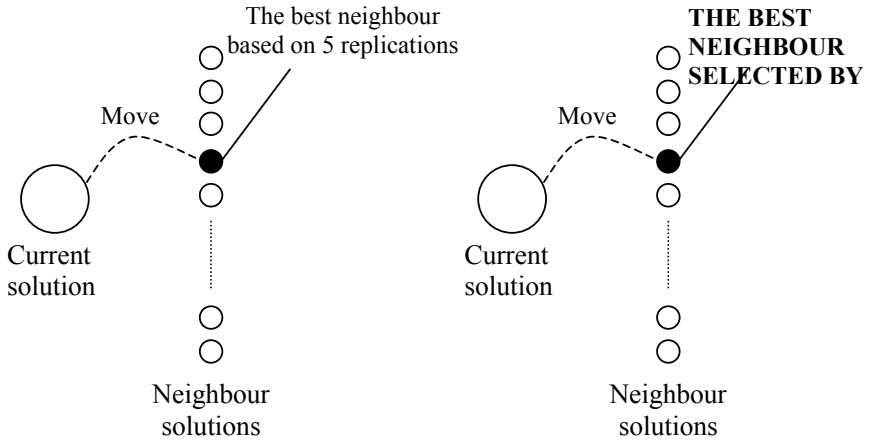
Table 1.1. The most known simulation optimization software packages, and supported simulation software.

Optimization Package	**Vendor**	**Simulation software supported**	**Search Strategies**
AutoStart	Brooks-PRI Automation	AutoMod, AutoSched	Evolution Strategies
Extend Optimizer	Imagine That	Extend	Evolution Strategies
OptQuest	Optimization Technologies	Arena, Flexim ED, Micro Saint, Pro-Model, QUEST, SIMUL8	Scatter Search, Tabu Search, Neural Networks
WITNESS Optimizer	Lanner Group	WITNESS	Simulated Annealing, Tabu Search

Ranking and Selection procedures are statistical tools that select the best alternative from a set of alternatives with a given confidence level. They can be grouped into two categories. The first one is Multiple Comparison Procedures (MCPs) and the other is Subset Selection and Indifference Zone Selection. In MCPs, alternative system designs are compared to each other and according to the comparison results the best system design or designs are determined. In Subset Selection a subset, which contains the best, is excluded from a set of alternatives, while in Indifference Zone Selection, the best alternative is selected. The detailed examinations of these methods will be given in Chapter 2.

Increasing attraction to simulation optimization area made researchers to seek new methodologies. One of these new methodologies is combining simulation optimization with ranking and selection. A couple of papers have been published related to this topic. A simulation optimization technique and a ranking and selection procedure can be used in conjunction in two ways. One is using ranking and selection procedure after a simulation optimization study. The elite solutions encountered by the search can be further inspected by the accompanying ranking and selection algorithm thus increasing the solution quality. Since the simulation optimization search already took observations from these elite solutions, there is no need for ranking and selection procedure to perform first stage sampling. This

approach may increase the solution quality with little extra computational effort. The other way, which we will implement in this study, is using ranking and selection algorithm within the search. This approach may lead to search quickly converge the best (near optimal) solution thus reducing the computational effort. The following figure illustrates our approach:



An ordinary TS approach Our approach

Figure 1.4. The comparison of an ordinary TS approach and our approach.

In the following chapter, we will describe the simulation optimization and ranking and selection methodologies and review the studies in the literature related to both topics. In Chapter 3, we will introduce our approach and give the details of the experimental study including descriptions of the various system designs in which we implement our methodology. The results of the experimental study will be given in Chapter 4. Our conclusions will be formed in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

2.1. Introduction

In this chapter we will introduce the basics of the simulation optimization and ranking and selection. And then some studies in the literature related to both topics will be summarized respectively. We first start with a general structure of the simulation optimization problem. Then we describe the simulation optimization methodologies in short. A summary of the studies in the literature will follow. After describing the ranking and selection concept and methodologies we will review the literature.

2.2. The General Structure of a Simulation Optimization Problem

A simulation optimization problem is defined, as in all other optimization problems, by decision variables, an objective function, and constraints.

Decision variables

Realizations of decision variables i.e., values of variables, directly affect the system's response. A complete set of decision variables is called a solution. The aim of the simulation optimization is to find the best set of decision variables (the best solution), which optimizes the objective function. For example, s, re-order point, and S, order-up-to point, are decision variables in an inventory control problem.

Objective function

Objective function is the function that is wanted to be optimized. It may be simply one of the performance measures (e.g., number of finished jobs, waiting time in the system, and cycle time, makespan etc.) or it may be represented as a linear or non-linear function of decision variables. In inventory control problem, the objective function is the total cost function, which consists of ordering, holding, and shortage costs. Note that, in this example, decision variables, (s,S), are not visible in the representation of the objective function but still the objective function, the cost function, is a function of decision variables.

Constraints

There are two types of constraints: qualitative and quantitative. Quantitative constraints may be linear or non-linear combinations of the decision variables. For example, *S*, order-up-to point, must be lower than some upper bound due to capacity limitations. On the other hand, some constraints cannot be represented mathematically. For example, a finished part on a machine blocks the machine until succeeding buffer has an empty room or an AGV (or forklift) unloads the part. Another example is the dispatching rule that is used in a queuing problem. The Shortest Processing Time (SPT) rule cannot be expressed mathematically. Actually this is a good representation of the power of the simulation.

In general a simulation optimization problem is represented as:

$$\min_{x \in C}(\max)\ E[f(x)]$$

where x is the solution vector, i.e., $x = [x_1, x_2, ...x_n]$, and C is the set of quantitative constraints. The qualitative constrains are represented in the simulation model. Note that we use the expectation of the objective function instead of the function itself in the formula. This is because the function itself cannot be calculated due to stochastic nature of the problem. One can only estimate it. A solution cannot be hundred percent said better than another one (of course, if not one of the solutions clearly inferior to the other one), instead one is said better than the other according to a confidence level. Normally high-level confidence is desired which may cause very exhaustive simulation optimization study, i.e., longer runs and/or more replications. Furthermore, if the number of the alternative solutions is large the simulation optimization study may become intractable.

2.2.1. Simulation Optimization Methodologies

Simulation optimization methods can be grouped into six main categories. Figure 1.4 illustrates these categories (adapted from Carson and Maria (1997)).

2.2.1.1. Gradient-based Search Methods

"Methods in this category estimate the response function gradient (f) to assess the shape of the objective function and employ deterministic mathematical programming techniques" (Carson and Maria (1997)). "Two major factors in determining the success of these methods are reliability and efficiency" (Azadivar (1999)). Since a simulation optimization problem has a stochastic nature there will be an error in estimating the gradient. If this error is large, then this may lead the search to the wrong directions. This is why reliability is a major factor in determining the success of the methods. On the other hand, the efficiency of a gradient estimation method can be measured by the required number of function evaluations (replications) to be able to estimate the gradient. Since the simulation experiments are expensive, less number of required replications means more efficiency. When the size and the complexity of the problem increase, efficiency becomes more important. Some of the gradient based search methods are: finite difference estimates, perturbation analysis, likelihood ratio estimates, and frequency domain analysis. One can refer to Carson and Maria (1997) and Adizavar (1999) for brief explanations of these methods.

2.2.1.2 Stochastic Approximation

"Stochastic approximation methods refer to a family of recursive procedures that approach to the minimum or maximum of the theoretical regression function of a stochastic response surface using noisy observations made on the function. These are based on the original works of Robins and Monro (1951) and Kiefer and Wolfowitz (1952)" (Adizavar (1998)). These methods use a recursive formula iteratively in order to find the optimal solution. The number of observations required in each iteration increases when the number of decision variables increases. Stochastic approximation methods slowly converge to the optimum and suffer from the lack of good stopping rules. Furthermore, they have difficulties with handling constraints.

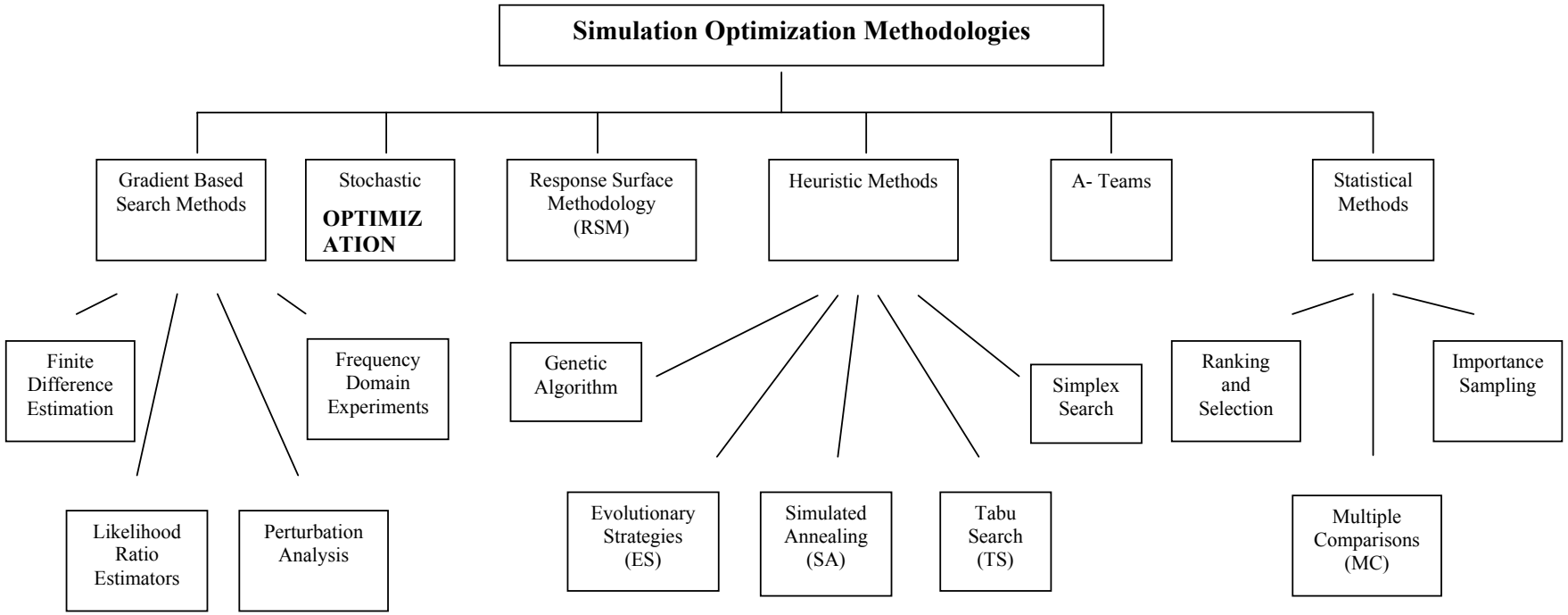


Figure 2.1. Classification of Simulation Optimization Methodologies (Carson and Maria (1997)).

2.2.1.3. Response Surface Methodology (RSM)

"Response Surface Methodology is a procedure for fitting a series of regression models to the output variable of a simulation model (by evaluating it several input variable values) and optimizing the resulting regression function" (Carson and Maria (1997)). "The process usually starts with a first order regression function and after reaching the vicinity of the optimum, higher degree regression functions are utilized" (Avdizavar (1998)). When compared to gradient estimation methods RSM is more efficient in terms of the required number of replications. On the other hand, if the complexity of the objective function (thus the response surface) increases, i.e. sharp ridges, flat valleys, RSM may become inefficient because of the relatively large errors in the fitted regression function.

2.2.1.4. Heuristic Methods

With the development of the heuristic methods attraction to the simulation optimization field has been increased. Because of the exploration and exploitation features, heuristics are very efficient global search strategies. The most known heuristic methods are:

2.2.1.4.1. Genetic Algorithms (GA)

Genetic algorithm is analogous to the biologic evolution. GA was developed by Holland (1992). Its DNA determines the fitness of an organism, which is defined as the ability to survive in its environment. A DNA can be represented as a string of values. An offspring's DNA consists of two parts. One part that inherits from its parents and the other is due to mutation. The idea behind the GA is to increase the overall fitness of the population by inheriting good features (traits) of the parents to the next generations.

The DNA of a population member can be thought as a solution, thus member as a solution and the population as a solution space. Each value in the DNA string represents a decision variable. The fitness of a solution (member) is determined by an objective (evaluation) function. Creation of an offspring (a new solution) is subject to biological operators. Where the *crossover operator* takes different parts of

parents' DNAs and brings them together to build offspring's DNA, the *mutation operator* randomly selects a position (a decision variable) in this new string and changes its value according to a pre-specified probability. There are also *selection* and *reproduction* operators. After a certain number of generations (iterations), the solution(s) with the best fitness value is (are) selected as the optimal. GA is noted for robustness in searching complex spaces and is best suited for combinatorial problems.

2.2.1.4.2. Simulated Annealing (SA)

Simulated Annealing process is analogous to the physical annealing process. It was introduced by Metropolis et. al. (1953). The key feature of this process is the temperature, T. SA starts with an initial solution and an initial temperature value. This temperature value remains same for a certain number of iterations and gradually decreases until the pre-determined final temperature is reached. At each iteration, a neighbor solution is generated and evaluated. If any improvement is made then the neighbor solution replaces the current solution. If no improvement is made then the neighbor solution may still be accepted as the current solution with a probability, which is a function of T. The reason behind this move is to avoid being trapped by the local optima. When T decreases the acceptance probability of non-improving solution decreases.

2.2.1.4.3. Tabu Search (TS)

Tabu Search can be classified as a neighborhood search. It is developed by Glover (1989). TS starts with an initial solution, and a neighborhood set, a subset of solution space, is created at each iteration. Each solution in this set is evaluated and the best one is selected as the new current solution if it is not classified as tabu. The old solution is classified as tabu and added to the tabu solutions list. Tabu solutions cannot be selected as a new solution for a certain number of iterations, which is called tabu tenure. At each iteration tabu tenure is decreased by 1. When tenure reaches zero the solution is removed from the tabu list. The search continues until a stopping criterion is satisfied.

Apart from these heuristics, Evolution Strategies, Nelder and Mead's Simplex Search, and Complex Search, an extension of Simplex Search, are also used in simulation optimization applications.

2.2.1.5. Statistical Methods

Most of the statistical simulation optimization techniques are Ranking and Selection Techniques (R&S). We will be inspecting these techniques in Section 2.3. In the following section we summarize some of the simulation optimization studies in the literature.

2.2.2. Literature Survey Simulation Optimization

In Carson and Maria (1997) a general review of the simulation optimization methods in the literature is given. The simulation optimization methods are classified into six main categories, which are Gradient based Search Methods, Stochastic Optimization, Response Surface Methodology, Heuristic Methods, A-teams, and Statistical Methods. Brief explanations of these methods can be found in the paper. Some of the examples of the simulation optimization applications and software are also mentioned.

Olafsson and Kim (2002) made a broad introduction to simulation optimization concept. General problem setting of a simulation optimization problem, i.e. decision variables, objective function, and constraints, is discussed. Brief information on some simulation optimization techniques for both continuous decision variables and discrete decision variables cases can be found in the paper. A couple of simulation optimization software is mentioned to stress out the increasing usage and popularity of the simulation optimization techniques in practice.

Fu (2001) summarized most of the major approaches and briefly described the most known software implementations in question-answer (Q&A) formatted tutorial paper.

Law and Kelton (2002) presented a tutorial, which is an introductory level, to simulation optimization. A simulation optimization problem is introduced. Experimental results of two commercial optimization packages applied to this

problem are illustrated. A table, which shows popular optimization software packages, supported software, and utilized strategies, is also included in the paper.

Azadivar (1999) addressed some specific issues related to decision variables, objective function, and constraints. Several problem classifications, i.e., according to single objective versus multi-criteria or continuous decision variables versus discrete decision variables, are mentioned. Brief descriptions of some simulation optimization approaches including gradient based approaches and heuristic search strategies are given. Discussion on multi-criteria optimization and on-parametric optimization are added to the paper.

Abspoel et. al. (2000) developed an optimization strategy that is based on a series of linear approximate sub problems. Each sub problem is built according to the outcomes of simulation experiments. A D-optimal designs of experiments is used to plan the simulation experiments. Stochasticity in constraints and objective function is dealt with explicitly using safety indices. Two text problems including a simulation based four-station production flow line problem are presented to illustrate proposed strategy.

Lee et. al. (1999) proposed an algorithm that searches the effective and reliable alternatives satisfying the target values of the system to be designed through a single run in a relatively short time period. The algorithm estimates an autoregressive model and constructs mean and confidence interval for evaluating the objective function obtained by small amount of data. The algorithm is applied to an (s,S) inventory control problem. Experimental results are illustrated in the paper.

Olafsson and Shi (1998) developed a new simulation based optimization method called Nested Partitions (NP) method. The method generates a Markov chain thus solving the optimization problem becomes equivalent to maximizing the stationary distribution of this Markov chain over certain states. The method may therefore be considered a Monte Carlo Sampler that samples from the stationary distribution. It is also shown in the paper that Markov chain converges geometrically fast to the true stationary distribution.

Olafsson and Shi (1999) analyzed a new simulation based optimization method that draws from two recent stochastic optimization methods: Nested Partitions, which is an adaptive sampling approach, and ordinal optimization. The new method guarantees global convergence under certain conditions. Furthermore, for certain problems, the method has exponential convergence rate characteristics, which is

shown by using ordinal optimization perspective. New conditions, under which asymptotic convergence holds, are derived and practical guidelines for determining the sampling effort in each iteration are provided.

Pichitlamken and Nelson (2002) proposed an optimization-via-simulation algorithm that combines Shi and Olafsson's (2000) Nested Partitions (NP) method with Sequential Selection with Memory (SSM) method due to Pichitlamken and Nelson (2001), and Hill Climbing (HC) algorithm. A numerical example on three-stage buffer allocation problem is presented. Comparisons with other optimization algorithms such as Simulated Annealing (SA), Random Search (RS), and Nested Partitions are also illustrated in the paper.

Pichitlamken and Nelson (2001) proposed a ranking and selection algorithm, Sequential Selection with Memory (SSM), very similar to Kim and Nelson's (2001) Fully Sequential Algorithm (FSA) in order to use in simulation optimization context. Idea is using SSM in a neighborhood search to find the best solution among neighbors. The algorithm uses a statistical selection approach to ensure the best selection with a certain probability (confidence level), while it makes use of memory, i.e. encountered solutions so far, to reduce the computational effort. A numerical example and comparisons to a few other selection approaches are presented in the paper.

Brady and McGarvey (1998) integrated the heuristic search methods with a simulation model to improve the operating performance of a pharmaceutical manufacturing laboratory. The problem is allocating small set of operators to a large set of test machines. A very detailed simulation model is used in conjunction with some heuristics namely Simulated Annealing (SA), Genetic Algorithm (GA), Tabu Search (TS), and Frequency Based Heuristic in order to improve operating performance of the laboratory, which can be defined in terms of work in process, operator efficiency, and operator balance. Dramatic improvements are achived up to nearly 16% with different heuristics.

Finke et. al. (2002) combined Tabu Search (TS) with simulation to develop a scheduling procedure for an automated steel plate fabrication facility in order to minimize earliness/tardiness penalties. The performance of the procedure is evaluated by comparisons to the optimal solutions for small problem instances and to a good heuristic for longer problems, TS allowed the incorporation of more realistic

constraints on system operation. Experimentation and results are presented in the paper.

Joines et. al. (2002) addressed the critical decision problems of "How much to order" and "How often to order" in a supply chain environment. A genetic algorithm is developed to optimize these system parameters. The quality of the results depends on the performance measure that is optimized. The deficiencies of using traditional performance measures are discussed and a new genetic algorithm methodology is developed to overcome these limitations.

Baretto et. al. (1999) applied the Linear Move and Exchange Move Optimization (LEO), which is based on a Simulated Annealing (SA) algorithm designed for solving hard combinatorial optimization problems, to a manufacturing problem. The problem description and results are presented in the paper. The paper also demonstrates the effectiveness and the versatility of the algorithm.

Baesler and Sepulveda (2000) introduced a new approach to solve multi objective simulation optimization problems. The approach integrates a simulation model with a genetic algorithm and a goal programming model. The genetic algorithm is modified to perform the search considering the mean and the variance of the responses. This new approach is able to lead the search towards a multi objective solution.

Altiparmak et. al. (2002) developed an artificial neural network (ANN) metamodel for simulation model of asynchronous assembly system. This metamodel is used in conjunction with Simulated Annealing (SA) to optimize the buffer sizes in the system. Experimental results are presented in the paper.

Humprey and Wilson (1998) developed a variant of Nelder and Mead's (NM) Simplex Search procedure, Revised Simplex Search (RSS), for simulation optimization. This new search method is designed to avoid the weaknesses, which can be stated as excessive sensitivity of starting values, being trapped by local optima, lack of robustness, and lack of computational efficiency, of some other direct search methods. A simulation study conducted to compare RSS to NM and RS9 (a simplex search procedure recently proposed by Barton and Ivey (1990)) based on separate factorial experiments for selected performance measures. Experimental results show that the improved performance of RSS with marginally increased computational effort.

Gupta and Sivakumar (2002) combined discrete event simulation and various techniques, which are used to deal with multi objective optimization such as weighted aggregation approach, global criterion method, minimum deviation method, and compromise programming, in order to generate optimal schedules for semiconductor manufacturing where there are more than one objectives to satisfy including cycle time, machine utilization, and due date accuracy. First, the job shop scheduling problem is modeled and the problem is divided in to simulation clock based lot selection sub problems. Then at each decision point in simulated time, a Pareto optimal lot is selected using the techniques mentioned above. Results show that how these techniques work effectively in solving the multi objective scheduling problem using discrete event simulation.

Sivakumar (1999) developed a discrete event simulation based "on-line near-real time" dynamic scheduling and optimization system to optimize the cycle time and asset utilization in semiconductor test manufacturing. The system has been implemented at a semiconductor back-end site. The impact of the system includes the achievement of very good cycle time, improved machine utilization, and more predictable and highly repeatable manufacturing performance.

Schruben (1997) introduced a new simulation optimization approach that takes advantage of the ability to run simultaneous replications of different experimental factor settings in a single run. Different time scales for the events corresponding to different design points can be used. In this manner, the run can focus on factor settings that are likely to be optimal and feasible. An example is presented using a penalty function to dilate event times to find the cycle time constrained capacity of a queue.

Lee et. al. (1997) developed a simulation optimization technique exploring a new paradigm called the "reverse simulation". The paper focuses on the method of on-line determination of steady state, which is a very important issue in reverse simulation optimization, and the construction of a reverse simulation algorithm with expert systems. The algorithm employs the Lyapunov exponent of Chaos Theory to determine the steady state of the system and an optimal state. M/M/s queuing model is chosen to illustrate the algorithm. Experimental results show that obtained number of servers by the algorithm corresponds to the theoretical value.

Neddermeijer et. al. (2000) developed a framework for automated optimization of stochastic simulation models using RSM. The framework is especially intended

for simulation models where the calculation of the corresponding stochastic response function is very expensive or time consuming. Many choices that have to be made in development of an automated RSM algorithm are described in the framework.

Angun et. al. (2002) modified RSM in the way that the determination of the search directions. Classical RSM locally fits first order polynomials in the first stages of the search, and then uses Steepest Descent (SD) strategy, which is scale dependent, to determine search direction. A scale independent search strategy, Adapted Steepest Descent (ASD), is derived which accounts for covariance between components of the local gradient. Monte Carlo experiments show that ASD gives a better search direction compared to SD. In multi objective analogous, interior point methods and binary search are used to derive scale independent search direction. Monte Carlo experiments show that a neighborhood of the true optimum can be reached in a few runs. Experimental results are presented in the paper.

Marito and Lee (1997) presented a simulation optimization approach for finding a dynamic dispatching priority in a stochastic job shop environment under the presence of multiple identical jobs. The key ingredients of the approach are: an efficient processing time based dispatching rule, simulation model of a job shop, and a mechanism to fake (or modify) job processing times based on the information of job slack obtained from simulation. An overall approach to fake processing times is described and alternative strategies for algorithm design are identified in the paper. Experimental results are illustrated.

Rogers (2002) applied a commercial simulation optimization tool, OptQuest, to manufacturing system design and control problems. After a brief introduction to both general simulation optimization concept and OptQuest for Arena, implementation of the software in tackling with *sequence dependent setup problem* for a production facility, and *optimal order acceptance/rejection problem* in a make to order environment is reported in detail. Results and conclusions are presented in the paper. Table 2.1. summarizes the studies :

Table 2.1. The summary of some studies in the literature.

Authors	The Paper
Carson and Maria (1997)	Overview of methodologies.
Olafsson and Kim (2002)	Overview of methodologies, some problem settings, and software.
Fu (2001)	Question and answer (Q&A) formatted tutorial.
Law and Kelton (2002)	An introductory level tutorial and some software applications.
Azadivar (1999)	General review of methodologies.
Abspoel et. al. (2000)	New methodology based on a series of linear approximate sub problems.
Lee et. al. (1999)	New algorithm based on estimating an autoregressive model.
Olafsson and Shi (1998)	New method Nested Partitions (NP).
Olafsson and Shi (1999)	New approach that combines NP and ordinal optimization.
Pichitlamken and Nelson (2001)	New Ranking and Selection approach, Sequential Selection with Memory (SSM).
Pichitlamken and Nelson (2002)	New approach, which is a combination of NP, SSM, and Hill Climbing .
Brady and McGarvey (1998)	Application of heuristic search methods to optimize the operating performance of pharmaceutical manufacturing facility.
Finke et. al. (2002)	Application of Tabu Search (TS) to minimize the earliness/tardiness penalties in a steel plate fabrication facility.
Joines et. al. (2002)	Application of Genetic Algorithm (GA) in a supply chain environment.
Baretto et. al. (1999)	Application of Linear Move and Exchange Move Optimization (LEO) to a manufacturing problem.
Baesler and Sepulveda (2000)	New approach integrates GA with Goal Programming to solve multi objective optimization problems.

Authors	The Paper
Altiparmak et. al.	Application of Simulated Annealing (SA) in conjunction with an artificial Neural Network (ANN) metamodel, to an asynchronous assembly system in order to optimize the buffer sizes.
Humprey and Wilson (1998)	New approach, Revised Simplex Search (RSS), which is a variant of Nelder and Mead's (NM) Simplex Search procedure.
Gupta and Sivakumar (2002)	New approach that combines discrete event simulation and various techniques, which are used to deal with multi objective optimization.
Sivakumar (1999)	New approach, dynamic scheduling and optimization system, to optimize the cycle time and asset utilization in semiconductor test manufacturing.
Schruben (1997)	New approach based on simultaneous replications of different experimental factor settings.
Lee et. al. (1997)	New approach based on Reverse Simulation.
Neddermeijer et. al. (2000)	Framework for automated optimization of stochastic simulation models using Response Surface Methodology (RSM).
Angun et. al. (2002)	Modification of RSM.
Marito and Lee (1997)	New approach for finding a dynamic dispatching priority in a stochastic job shop environment
Rogers (2002)	Application of OptQuest, to manufacturing system design and control problems

2.3. Ranking and Selection

One of the most important areas that simulation is used is comparing alternative system designs. For example, suppose two layout designs for a production facility are being considered. Decision maker wants to know which design is better. One can make use of simulation in order to compare the designs and select the best one among. Of course this task must be performed carefully to avoid the possibility of

selecting the wrong system. Even if the simulation study is performed perfectly, an appropriate method must be chosen to compare the systems using simulation output. This issue is very important. Because of the stochastic nature of simulation, arising from randomness, one can never be sure which system is better with certainty. So, appropriate statistical methods must be used to distinguish the best system from the other alternatives within a given confidence level.

There are a lot of statistical methods to compare the alternative system designs. When we look at the literature, these procedures can be divided into two main categories. The first category is Multiple Comparison Procedures (MCPs) and the second is Ranking and Selection (R&S) procedures.

2.3.1. Multiple Comparison Procedures (MCPs)

These procedures basically construct confidence intervals, with the desired confidence level, around the differences of two systems' performance measures and try to give insights about the systems' performances with respect to each other.

The most known procedure is due to Tukey. (Goldsman and Nelson (1998)). The procedure requires identical, independent, and normally distributed outputs from each system. The procedure does pair wise comparison, takes difference between two alternatives and constructs a confidence interval to see the magnitude and the direction of the difference. $k(k\text{-}1)/2$ confidence intervals are formed for k alternatives.

Instead of comparing each alternative with the others, one can compare each system with the best of the remaining systems thus reducing the number of confidence intervals. This kind of comparison is called Multiple Comparisons with the Best (MCB). "The first MCB procedures were developed by Hsu." (Goldsman and Nelson (1998))

Another type of multiple comparison procedures is called Multiple Comparisons with the Control (MCC). In this approach alternative systems are compared to a control (or a default) system. Thus we only need to construct k-1 confidence intervals. This kind of situation may arise when we compare alternative designs to an existing system. "MCC procedures are well known for the case when the variances across systems are equal and the data are normal (Goldsman and Nelson (1998)).

Although MCPs give insights when comparing alternatives, they do not either provide much information or select the best. But we can detect the systems worth examining further by using MCPs. At this point of view they can be considered as selection procedures.

2.3.2. Ranking and Selection Procedures

2.3.2.1. Subset Selection

In subset selection approach, we form a subset of alternative systems which includes the best system. The cardinality of the subset depends on the procedure used. It can be random-size or pre-determined. The most known method is Gupta's single stage procedure (Goldsman and Nelson (1998)). The method assumes that simulation outputs are independent, balanced (equal number of observations from each system), and normally distributed with common (unknown) variance. "Gupta and Huang" proposed a similar procedure for the unbalanced case" (Goldsman and Nelson (1998)).

2.3.2.2. Indifference-zone Selection

The indifference-zone procedures select the best system among alternatives with pre-determined confidence level. The term indifference-zone, δ, comes from user specified parameter that indicates practically significant difference. This means, if a system's expected value for a given performance measure is at least δ amount better than the others then the system is considered as the best. If the differences between expected values of two or more systems are within the indifference zone (less than δ) then this means there is no practically significant difference between systems and one of them can be selected as the best.

Most of the indifference-zone procedures are two-stage procedures. In the first stage, sample variances are calculated from simulation output for each system. Then using a simple formula, that accounts for these sample variances, and a user specified indifference zone parameter, and a statistical constant that is a function of number of systems, and desired confidence level, the required sample sizes are calculated. In the second stage, more replications are performed according to the required sample sizes. After required replications are taken new sample means are calculated and one of the

systems is selected as the best by looking the sample means of each system. If the largest is better then the system with the largest sample mean is selected.

The most known indifference zone procedures are due to Rinott and due to Dudewicz and Dalal (Goldsman and Nelson (1998)). Both methods are two-stage procedures and assume normality and independence across systems. The main difference is, in the second stage, unlike Rinott's, Dudewicz and Dalal's procedure uses the weighted averages. Nelson and Matejcik proposed procedures those can handle dependence across systems. (Goldsman and Nelson (1998))

"Matejcik and Nelson established a fundamental conjunction between indifference-zone selection and MCB by showing that most indifference-zone procedures can simultaneously provide MCB confidence intervals with the width of the intervals corresponding to the indifference zone." (Goldsman and Nelson (1998)) There are several combined procedures: Rinott+MCB, NM+MCB, and Bonferroni+MCB.

The multinomial selection approach is an another kind of indifference-zone procedures.(Goldsman and Nelson (1998)) This approach tries to select the system that is most likely to have the best response. Let p_i be the probability that system *i* will produce the best response from a given observation from each system. "The goal is to select the best system with a given confidence level whenever the ratio of the best to the second-best p_i is greater than some user specified constant, say $\theta>1$. The indifference constant θ can be regarded as the smallest ratio worth detecting" (Goldsman and Nelson (1998)). Bechhofer, Elmaghraby and Morse (BEM) proposed a single stage procedure that uses this approach. "There is also more efficient but more complex method due to Bechhofer and Goldsman (BG)" (Goldsman and Nelson (1998)).

2.3.3. Literature Survey Ranking and Selection

Goldsman (1983) introduced some common ranking and selection terminology and procedures. Some additional references for more complicated procedures are given. Indifference-zone approach, subset selection approach, and other approaches are explained. Discussion on R&S procedures in simulation applications is included.

Goldsman and Nelson (1998) presented a review of screening, selection, and multiple comparisons procedures that are used to compare system designs via

computer simulation. Screening large number of system designs, selecting the best system, and comparing all systems to a standard are the main topics of the paper.

Goldsman et. al. (1999) presented a review paper in the area of ranking and selection available to practicing engineers and management scientists.

Nelson (1993) used the multiple comparisons with the best (MCB) procedures to analyze simulation experiments that employ common random numbers (CRNs).

Matejcik and Nelson (1993) proposed three procedures that combine indifference-zone selection and multiple comparison inference. The first method uses Rinott's indifference-zone procedure then constructs MCB confidence intervals. This method requires independence across systems. The second and the third methods allow dependence across systems. The second one uses Clark and Yang's indifference-zone selection procedure than constructs MCB confidence intervals. The third method is due to Nelson and Matejcik and works as the same way as the others. The importance of this paper comes from that it shows indifference-zone procedures can be used in conjunction with MCB.

Haynes et. al. (1997) conducted a robustness study. A new family of distributions called *g*-and-*k* distributions, which may be used to approximate a wide class of distributions and allow effectively controlling skewness and kurtosis through independent parameters, are used in the study. The frequentist selection rules are found robust to small changes in the distributional shape parameters g and k. The study can be used to assess robustness and to develop procedures to allow for non-normality and also to understand the effects of non-normality on selection procedures.

Matejcik and Nelson (1995) developed two-stage sampling procedures to compare a small number of stochastic systems. The procedures are MCB procedures and require independence and normality. They also allow experimenter to specify the desired precision in advance. The paper includes guidelines for experiment design and an illustrative example.

Inoue and Chick (1998) compared the Bayesian Approach and the frequentist approaches in the literature. First, Bayesian Approach for both known and unknown precision is introduced. Then, a bayesian model is constructed for multiple systems for dependent and independent cases. Finally, comparison of Bayesian Approach to classical approaches, under the normality assumption for both dependent and independent cases, is illustrated. Although Bayesian Approach produced better

results, the differences between the proposed approach and the other approaches are not significant.

Ahmed and Alkhamis (1999) presented a new iterative method that combines the simulated annealing method and the ranking and selection procedures for solving discrete stochastic optimization problems.

Olafson (1999) developed a new algorithm for simulation based optimization where the number of alternatives is finite but very large. The method combines the Nested Partitions (NP) method for global optimization and Rinott's two-stage procedure.

Goldsman and Marshall (1999) modified Rinott's procedure. Instead of using classical variance estimators, variance estimators arising from the method of Standardized Time Series (STS), are used. STS variance estimators have more degrees of freedom according to Batch Means (BM) variance estimators. On the other hand STS variance estimators require more sample sizes to achieve the desired probability of correct selection. The paper stresses out this trade-off between STS and BM variance estimators.

Morrice et. al. (1999) conducted a sensitivity analysis on a ranking and selection procedure for making multiple comparisons of systems that have multiple performance measures. The procedure combines Multiple Attribute Utility (MAU) theory with ranking and selection. The analysis focused on the weights generated by the MAU procedure. Implementation of the analysis, on a simulation model of a large project that has six performance measures is illustrated. The impact of the sensitivity analysis on the results of the ranking and selection procedure is also discussed.

Kim and Nelson (2001) developed a new ranking and selection procedure, Fully Sequential Procedure (FPS), for indifference-zone selection. The motivation of the procedure is eliminating apparently inferior systems at the early stages of the experimentation thus reducing the computational effort. The procedure only requires normality and can handle dependence across systems. Actually, it is shown that inducing dependence increases the efficiency of the procedure. The results of different configurations for varying number of systems are presented in the paper. Comparisons to some existing procedures are also given.

Goldsman et. al. (2000) presented two ranking and selection procedures for use in steady state simulation experiments. Both procedures require independent and

normally distributed data. The procedures are extensions of Rinott's procedure and Fully Sequential Procedure (FSP). They were modified to handle steady-state simulation. Experimental design and summary of the results are presented in the paper.

Chen and Kelton (2000) proposed a two stage selection procedure called an Enhanced Two-Stage Selection (ETSS) procedure. The main difference is required sample size for each system in the second stage is determined by both the variances of the sample means and the differences of the sample means of alternative designs. The procedure is compared to Rinott's procedure. Several experiments under different conditions are performed to show the procedure's validity. Results of the experiments are presented in the paper.

Nelson et. al. (2000) addressed the problem of selecting the best system when the number of alternatives is too large that ranking and selection procedures may require too much computation to be practical. A new approach, combining screening procedures with indifference-zone procedures, is proposed. A combined procedure may eliminate inferior systems at the first step and thus reduces the number of alternatives for attached indifference-zone procedure. Computational effort may be dramatically reduced according to stand-alone indifference-zone procedure. A general theory for constructing combined screening and indifference-zone procedures is presented. Several combined procedures are proposed. An empirical evaluation study and some results of the study are also given.

Nelson and Goldsman (2000) considered the problem of comparing finite number of systems with respect to a single system (standard). The goal is to find out if systems better than the standard exist, and if so, to determine the best of alternatives. Two-stage experiment design and analysis procedures are proposed. The analysis is based on variety of scenarios including dependence across systems. A couple of methods for estimating the critical constants required by the proposed procedures are provided. A portion of an extensive empirical study and demonstration of one of the procedures are presented in the paper.

Hedlund and Mollaghasemi (2001) developed a methodology based on a genetic algorithm in conjunction with an indifference-zone selection procedure under common random numbers (CRN). The method is applied to a stochastic mathematical model. Results are presented in the paper.

Chen (2001) discussed the validity of using common random numbers (CRN) with two-stage selection procedures to improve the probability of correct selection. An experimental study is performed using several procedures including Rinott's procedure and an Enhanced Two-Stage Selection (ETSS) procedure. It is shown that when CRN was employed, the procedures returned better results compared to independent case. Experimental design and the results are presented in the paper.

Chen (2002) brought a conservative adjustment to ETSS procedure to increase the probability of correct selection. An experimental study is conducted and the results, which show the efficiency of the adjustment, are presented in the paper. Comparisons to Rinott's procedure and original ETSS procedure are also given.

Boesel et. al. (2002) considered the problem of finding the best system when the number of systems is large and initial samples from each system have already been taken. This situation may be encountered when a heuristic search procedure has been applied in a simulation optimization context. The true best system may not be the system that the search procedure indicates because of the stochastic variation. Some statistical procedures that return the best system encountered by the search with a pre-specified probability are developed. An empirical study and the results are presented in the paper.

A summary of the R&S procedures is given in Table 2.2:

Table 2.2. Some of the R&S procedures in the literature.

Procedure	Equal variance	Known variance	Balanced first sampling	Dependence	Normality	# of stages	Type
Rinott (1978)	No	No	Yes	No	Yes	2	IZ
Dudewicz and Dalal()	No	No	Yes	No	Yes	2	IZ
Clark and Yang (1986)	No	No	Yes	Yes	Yes	2	IZ
Nelson and Matejcik (1993)	Yes	No	Yes	Yes	Yes	2	IZ
Nelson and Matejcik Two-stage MCB (1995)	No	No	No	No	Yes	2	MCB
BT (Bechhofer, Turnbull) (1978)	Yes	No	Yes	No	Yes	2	IZ+MCB
BEM (Bechhofer, Elmaghraby, Morse) (1959)	No	No	Yes	No	Yes	1	MSA
BG (Bechhofer, Goldsman) (1986)	No	No	Yes	No	Yes	Seq.	MSA
AVC (Miller, Nelson, Reilly) (1998)	No	No	Yes	Yes	Yes	1	MSA
Nelson and Goldsman (2000)	No	No	Yes	No	Yes	2	CWS
Goldsman and Marshall (Rinott+STS) (1999)	No	No	Yes	No	Yes	2	IZ

Procedure	Equal variance	Known variance	Balanced first sampling	Dependence	Normality	# of stages	Type
ETSS (2000)	No	No	Yes	No	Yes	2	IZ+SS
Fully Sequential Procedure (2000)	No	No	Yes	Yes	Yes	Seq.	IZ
Screen-to-the-best (Nelson et. al.) (2000)	No	No	Yes	Yes	Yes	1	SS
Combined Procedure (Nelson et. al.) (2000)	No	No	Yes	No	Yes	2	IZ+SS
Group Screening (Nelson et. al.) (2000)	No	No	Yes	No	Yes	2	IZ+SS
Tukey ()	Yes	No	No	No	Yes	1	MCP
Gupta (1956)	Yes	No	Yes	No	Yes	1	SS

IZ : Indifference-zone selection	MSA : Multinomial Selection Approach	MCP : Multiple Comparison Proc.
SS : Subset Selection	MCB : Multiple Comparisons with the Best	CWS : Comparison with Standard
Note : "Yes" means that the method requires (or allows in "dependence" case) the feature that reads in the column head, while "No" means the opposite.		

CHAPTER 3

PROPOSED STUDY

3.1. Methodology

As we stated in the first chapter, the aim of this study is to investigate the effects of the Fully Sequential Procedure (FSP) when embedded in Tabu Search (TS). TS, when applied in simulation optimization context, uses simulation as an evaluator. In each iteration, a solution is selected as the best among the neighbours. This best solution is used as a starting point for next iterations, i.e., the new neighbourhood set is generated according to this solution. The efficiency of the search is directly related to selecting the best (or near best) solution in each step. Selecting the best solution in each step makes the search to converge quickly to the optimal (or near optimal) solution. Hence, the computational effort is reduced.

An ordinary TS algorithm uses arbitrary number of replications when evaluating the neighbours to find the best among them. This is, take an arbitrary number of replications, say n, from each alternative, and select the solution with largest average value (in a maximization problem) as the best. Due to stochasticity, this approach may lead the search to the wrong directions or it may make the search delayed especially when n is small. It is even worse when the size and the complexity of the problem increase.

At this point, we think if a Ranking and Selection (R&S) algorithm, which guarantees the best selection with a certain probability, is used then the efficiency of the search might be improved. We expect better solutions with decreased number of iterations. On the other hand, wrong selection in any iteration does not always mean the search will not end up with a good (near optimal) solution. If this is the case then employing a R&S algorithm is useless. And considering the additional computational effort, it is certainly beneficial not to employ it. This is a trade-off and we will be examining this trade-off in the following sections. The next two sections describe the TS and FSP respectively, while the remaining sections present analysis and results of our approach when applied on a various system designs.

3.1.1. Tabu Search

Tabu Search is an iterative search heuristic due to Glover (1986). It is designed to solve combinatorial optimization problems. Classical optimization techniques are inefficient to solve these problems because of the computational intractability of the problems when the size of the problem gets larger. To overcome this difficulty heuristic methods were developed. Although these methods do not guarantee the optimal solution, they provide good (close to optimal) solutions. TS is one of the most known and the most efficient of these methods. It has many application areas including production scheduling, location allocation, telecommunication, routing, and graph optimization.

The key features of TS are intensification and diversification strategies, and its utilization of memory (history). Intensification is, as the name implies, to intense the search around promising or good solutions to improve the objective function value, while the diversification means to direct the search to the new compromising regions to avoid being trapped by the local optima.

Memory plays very important role in implementing these strategies. Recency based (short term) memory keeps track of recent solutions. Recently encountered solutions are classified as tabu, and for a certain number of iterations, which is called *tabu tenure*, the search cannot move into those solutions. This prevents the search from going into a cycle. Frequency based (long-term) memory, on the other hand, records solutions encountered by the search. The frequent solutions according to this memory may be penalized to diversify the search. Also the search may be restarted to

thoroughly search the neighbourhood of the good (elite) solutions in the long-term memory.

If a solution is tabu then this solution is ignored by the search even it is the best solution in the neighbourhood. There are a couple of ways to make tabu classifications. One of them is using a part of a solution e.g., one of the solution variables, and another one is using the solution itself (the whole solution). In the former case tabu list is designed to record any part of the solution (any solution variable). To determine whether a solution is tabu or not, solution variables are compared to counterparts in the tabu list. For example, suppose a solution consists of five solution variables, $x = (x_1, x_2, x_3, x_4, x_5)$. And tabu list at some iteration of the search is formed as (just illustration purposes):

Variable	Value	Tenure
x_1 or 1	5	3
x_2 or 2	3	2
x_3 or 3	3	3
x_4 or 4	1	2
x_5 or 5	2	1

This means if variable x_1 of a solution is equal to 5 then that solution becomes tabu (classified as tabu). And this situation continues for the next three iterations since tabu tenure equals to 3. Similar explanations are valid for the other variables. Note that actually tabu list has more elements than it is shown in the table. In the latter case in which the tabu classification is made by looking the whole solution, the picture of the tabu list becomes the following:

Tabu Solution	Tenure
(5,2,3,4,4)	5
(5,3,3,3,4)	4
(4,3,3,3,3)	3
(4,4,3,3,3)	2
(3,4,4,3,4)	1

The solution (5,2,3,4,4) is tabu and following 5 iterations the search cannot move into this solution. But there are no restrictions for the variables in the solution. For example, the solution (5,2,3,4,3) is not tabu. The last variable of the solution

prevents it from being tabu. This kind of tabu classification is less restrictive than the former one.

Tabu status of a solution can be overridden according to some criterion. This is called *aspiration criterion*. There are several types of aspiration criterion. For example, if the objective function value of the tabu solution is better than the best solution encountered by the search so far then the tabu status of the solution is overridden. If all solutions in the neigbourhood are tabu then tabu status of the best solution is overridden.

Another important issue in TS is the generation of the neighbourhood set. The efficiency of the search can be improved by implementing clever neighbourhood generation algorithms that make use of memory. By utilizing memory better search directions can be found, and neighbourhood generation algorithm may focus on these directions.

3.1.1.1. Tabu Algorithm

Here is the Tabu Search algorithm.

Step1 : Start with an initial $x \in X$ and let $x_{best} = x$. Set the iteration counter, $c = 0$, and empty the tabu list, $T = \varnothing$.

Step2 : If stopping criteria is satisfied, i.e., simple iteration count or a certain improvement has been made etc., then stop and return x_{best}.

Step3 : Generate $N(x)$, neighbourhood set to the current solution. Evaluate the neighbours and select the best, x_{new}, if it is not classified as tabu, i.e., it is not in the tabu list, $x_{new} \notin T$. If it is in the tabu list, but satisfies *aspiration criterion*, $f(x_{new}) > f(x_{best})$ (in a maximization problem), then still select the solution as the new current solution. Otherwise select the second best solution as the new current solution.

Step4 : If $f(x_{new}) > f(x_{best})$ then set $x_{best} = x_{new}$.

Step5 : Increase iteration count, update tabu list, *T*, by adding x_{new} to the list, and decreasing the tabu tenures of the solutions in the tabu list. If tabu tenure of a solution is zero then remove the solution from the list. Goto **Step2**.

x denotes solution vector,

X denotes solution space,

T denotes tabu list,

x_{best} denotes best solution so far,

$f(x)$ denotes objective function, and

$N(x)$ denotes neighbourhood set of x.

One can refer to Glover (1989) and Glover and Laguna (2002) for better understanding of TS.

3.1.2. The Fully Sequential Procedure (FSP)

The Fully Sequential Procedure (FSP) is a sequential, indifference-zone Ranking and Selection (R&S) algorithm proposed by Kim and Nelson (2000). It is designed to reduce computational effort by eliminating clearly inferior alternatives at the early stages of the experimentation. The procedure only assumes normally distributed data. It can handle unknown and unequal variances, and dependence across systems. Actually it is shown that employing Common Random Numbers (CRN) increases the efficiency of the procedure.

3.1.2.1. The Fully Sequential Algorithm (Kim and Nelson (2000))

The algorithm is directly excerpted from Kim and Nelson (2000).

Setup: Select confidence level $1-\alpha$, indifference zone δ and first stage sample size $n_0 \geq 2$. Calculate η and c as described below.

Initialization: Let $I = \{1,2,...,k\}$ be the set of systems still in contention, and let $h^2 = 2c\eta \times (n_0 - 1)$.

Obtain n_0 observations X_{ij}, $j = 1,2,...,n_0$ from each system $i = 1,2,...,k$.

For all $i \neq l$ compute

$$S_{il}^2 = \frac{1}{n_0 - 1}\sum_{j=1}^{n_0}\left(X_{ij} - X_{lj} - \left[\overline{X}_i(n_0) - \overline{X}_l(n_0)\right]\right)^2$$

the sample variance of the difference between systems *i* and *l*. Where $\overline{X}_i(n_0) = \frac{1}{n_0}\sum_{j=1}^{n_0} X_{ij}$, X_{ij} denotes the *j*th independent observation from the system *i*. Let

$$N_{il} = \left\lfloor \frac{h^2 S_{il}^2}{\delta^2} \right\rfloor$$

where $\lfloor \cdot \rfloor$ indicates truncation of any fractional part, and let

$$N_i = \max_{l \neq i} N_{il}$$

Here $Ni+1$ is the maximum number of observations that can be taken from system *i*. If $n_0 > \max_i N_i$ then stop and select the system with the largest $\overline{X}_i(n_0)$ as the best. Otherwise set the observation counter $r = n_0$ and go to **Screening**.

Screening: Set $I^{old} = I$. Let

$$I = \left\{ i : i \in I^{old} \text{ and } \overline{X}_i(r) \geq \overline{X}_l(r) - W_{il}(r), \forall l \in I^{old}, l \neq i \right\}$$

where

$$W_{il}(r) = \max\left\{0, \frac{\delta}{2cr}\left(\frac{h^2 S_{il}^2}{\delta^2} - r\right)\right\}$$

Notice that $W_{il}(r)$, which determines how far the sample mean from system *i* can drop below the sample means of the other systems without being eliminated, decreases monotonically as the number of replications *r* increases.

Stopping Rule: If $|I| = 1$, then stop and select the system whose index is in *I* as the best. Otherwise, take one additional observation $X_{i,r+1}$ from each system $i \in I$ and set $r = r + 1$.

If $r = \max_i N_i + 1$, then stop and select the system whose index is in *I* and has the largest $\overline{X}_i(r)$ as the best. Otherwise go to **Screening**.

(Notice that the stopping rule can also be $|I| = m > 1$ if it is desired to find a subset containing the best, rather than the single best.)

Constants: The constant *c* may be any nonnegative integer, with standard choices being $c = 1,2$; these values are standard in the sense that they were used by Hartmann (1991), and that η is easy to compute when $c = 1 \text{ or } 2$. We evaluate different choices later in the paper and argue that $c = 1$ is typically the best choice. The constant η is the solution to the equation

$$g(\eta) \equiv \sum_{l=1}^{c} (-1)^{l+1} \left(1 - \frac{1}{2}\mathrm{I}(l = c)\right)\left(1 + \frac{2\eta(2c-l)l}{c}\right)^{-(n_0-1)/2} = \frac{\alpha}{k-1}$$

where I is the indicator function. In the special case that $c = 1$ we have the closed-form solution

$$\eta = \frac{1}{2}\left[\left(\frac{2\alpha}{k-1}\right)^{-2/(n_0-1)} - 1\right].$$

At screening process each alternative is compared to the remaining alternatives in the set. If it is inferior to any of the other alternative then it is eliminated from the set. If it survives from the comparisons it stays in the set for further inspection. Comparisons are done in the following manner (in a maximization problem). If a system's sample mean is bigger than or equal to the any other system's sample mean minus a function value called W_{xy} then this system passes the comparison. Otherwise it is eliminated. For example, let $\overline{X}_i(r) = 20$ be the sample mean of system *i*, and let $\overline{X}_j(r) = 22$ be the sample mean of system *j*, and $W_{ij}(r) = 3$. If $\overline{X}_i(r) \geq \overline{X}_j(r) - W_{ij}(r)$ then the system *i* passes the test, and it is compared with other systems in the same manner. If it passes all the tests then it stays in the set for further inspection. Since $20 > 22 - 3 = 19$ system *i* passes the test. If $W_{ij}(r) = 1$ then the system *i* could not pass the test, and it would be eliminated by system *j*.

$W_{ij}(r)$ function, which is used in pairwise comparisons, monotonically decreases as the number of replications *r* increases. This function can be thought as a range (or window). The length of the range decreases with every additional iteration, and it becomes difficult for a system, which is not the true best, to stay in the set. When the length of the range reaches zero, if there are more than one alternative in the set then the alternative with the largest sample mean selected as the best. Statistically there is no difference between the alternatives in the set and any of them can be the true best. But, in most cases, the best is found before the length of the range ($W_{ij}(r)$) reaches zero. Following figure illustrates the graph of $W_{ij}(r)$:

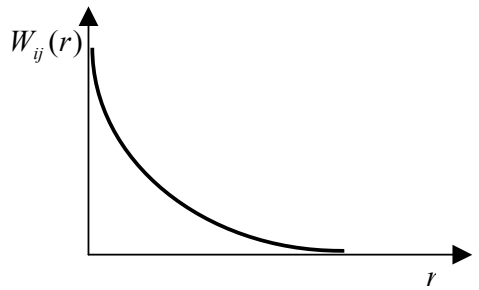


Figure 3.1. The graph of $W_{ij}(r)$

It can be seen that the $W_{ij}(r)$ quickly decreases in the early stages to eliminate the clearly inferior systems. Towards the end it slowly decreases to get rid of the possibility of wrong decision. The alternatives those survive towards the end of the stages should be carefully compared.

The validity of the procedure has been proven in the paper. Design of the procedure including choice of *c*, whether or not to use CRN, and the effect of the batch size is examined. The results of the experimental study performed to compare FSP to two other R&S algorithms namely Rinott's (1978) procedure and 2SP proposed by Nelson et. al. (2000), are illustrated in the paper.

3.2. Experimental Settings

3.2.1. Manufacturing Problem

Our first problem is a production problem that is introduced by Law and McComas (2002). Since this problem is a test problem in simulation optimization area, we decided to start implementing our methodology with this problem. Following figure illustrates the outline of the production facility.

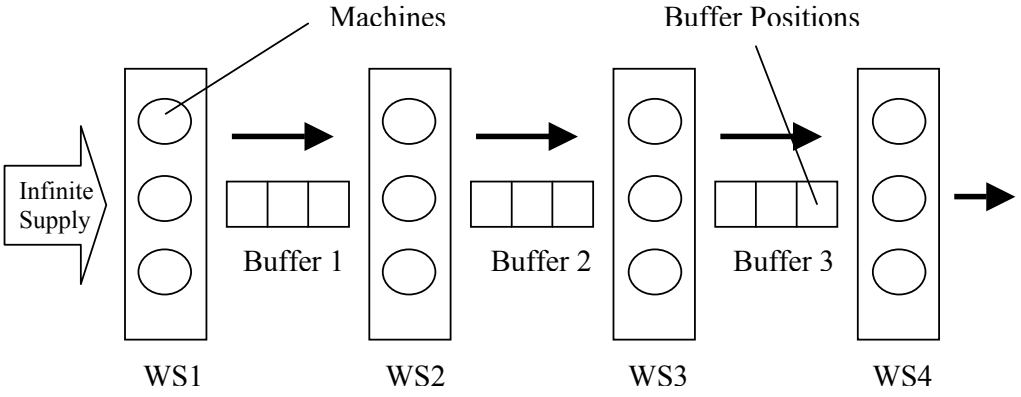


Figure 3.2. The outline of the production facility.

As seen in the Figure 3.2. the production facility consists of four work stations and three buffers located between workstations. The machines at a specific workstation are identical. The number of machines at each workstation may be different from others. The processing times of the machines at each workstation are exponentially distributed with means (0.3333, 0.5, 0.2, 0.25) respectively. There is an infinite supply in front of the first workstation. The parts enter system from workstation 1 and leave from workstation 4.

A part enters the system as soon as if there is an available machine at workstation 1. When a part completes its processing, it is transferred to succeeding buffer if there is enough room (position) for the part. Otherwise it blocks the machine. The machine remains blocked until the part moves. A part that is waiting in the buffer moves the succeeding workstation if there is an available machine at that workstation. If not then the part waits in the buffer until a machine at succeeding workstation becomes available. The transfer times are negligible.

The objective is to maximize the profit where each machine costs $250 and each buffer costs $10. We earn $2 per finished part. The objective (profit) function is formed as the following:

P=finished parts*2 - total number of machines*250 - total number of buffer positions*10

The decision variables are the number of machines at each station and the number of buffer positions in each buffer. Let $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ denotes a solution. The odd numbered variables $x_1, x_3, x_5,$ and x_7 represent the number of machines at workstations 1, 2, 3, and 4 respectively. And the even numbered variables $x_2, x_4,$ and x_6 represent the number of buffer positions in buffer 1, 2, and 3. The objective function then can be formulated as follows:

$$P(x) = \text{finished parts} \cdot 2 - (x_1 + x_2 + x_3 + x_4) \cdot 250 - (x_2 + x_4 + x_6) \cdot 10$$

Although there are no limitations on the machine number at any workstation or the number of buffer positions in any buffer we put upper bounds on these variables for practical reasons. The maximum number of machines at any workstation will be 3 and the maximum number of buffer positions in any buffer will be 8. These numbers are based on direct observations from Law and McComas (2002).

We developed a simulation model of the above problem to estimate the objective function. The model takes decision variables $x_i, i = 1,2,..,7$ as an input and gives the total profit. Model parameters are represented in Table 3.1:

Table 3.1. Model parameters of the production problem.

Parameter	Value			
Run length (Hours)	920			
Warm-up period (Hours)	240			
Mean Processing Times (Hours)	WS1	WS2	WS3	WS4
	0.3333	0.5	0.2	0.25

Although this simulation model can evaluate the solutions and give the output for a specific solution, it has nothing to do with optimization. It has to be used in conjunction with an optimization algorithm. TS can use this model as an evaluator in optimizing the objective function. At each iteration of TS, alternative solutions (neighbours) are evaluated using this model and the best solution is selected according to the simulation output.

3.2.2. Inventory Control Problem

Our second problem is a single item single location continuous control inventory problem. We wanted to apply our methodology on different kind of system designs. Since inventory control problems are very common in real life, we decided to implement our methodology on this problem. It is taken from Yuksel (2000). The stored items are perishable. This means, after a time period, which is called *shelf life*, the items perish and become useless. The shelf life of the goods can be fixed or random. (*s*,*S*) policy is used to decide when to order and how much to order.

The inventory starts with *S* amount of goods. When a demand occurs, it is immediately met if there is enough amount of the item to meet the demand. Otherwise, the amount that could not be met is backordered. If the inventory level drops below a certain point *s*, re-order point, then an order is placed to raise the inventory level to *S*, order up to point. The order arrives after a certain time period, which is called *lead-time*. The backordered demands are met first when the order arrives. The items that fill their shelf lives are disposed. The items that are put inventory at the same time have the common shelf life. On the other hand, this shelf life could be fixed or random.

There are four types of cost incurred managing the inventory. The first one is the *ordering cost*, which is fixed and occurs when an order is placed. The second one is the *holding cost*. There is a unit cost of items during the time they are stored in the inventory. The third one is the *backorder cost*. If a demand cannot be met then it brings a certain amount of cost. Furthermore, it will cause the loss of customers' goodwill. The backorder cost is time weighted. This means, a unit of backordered item will bring a certain amount of cost per unit time until it is met. The fourth type of cost is the *perishing cost*. When an item in the inventory completes its lifetime

then it becomes useless and is disposed. There is a cost per perished item. Assumptions of the inventory model are described below:

- Single item single location,
- Fixed or random shelf life (the shelf life of an item could be distributed exponentially or it could be gamma distributed),
- The shelf life of all items in a batch is same,
- The lead time is positive,
- The FIFO rule applies,
- (*s*,*S*) policy is used.

Inter-demand times are gamma distributed with the shape parameter $\alpha_{id} = 0.04$ and the coefficient of variation parameter $\beta_{id} = 2$. The demand quantities have a uniform distribution within the range U[0.5,1]. The shelf life of an item is 0.5 time units in fixed case, and it is exponentially distributed with mean $\mu_{sl} = 0.5$ in random case. The lead time is 1 time unit. The ordering cost is determined as \$50. The holding cost is taken \$1 unit per item. The time weighted backorder cost is set to \$2 units per item. The perishing cost is \$5 units per item. The following table summarizes the parameters of the system:

Table 3.2. The parameters of the inventory control problem.

Parameters	Value
Inter-demand times	Gamma(0.04,2)
Demand quantities	Uniform[0.5,1]
Shelf life	0.5 or Exponential(0.5)
Lead time	1
Ordering cost (\$)	50
Holding cost (\$)	1
Backorder cost (\$)	2
Perishing cost (\$)	5
Simulation run length	1000 served customers
Warm-up period	100 served customers

The objective is to minimize the total cost function. The total cost function is formed as:

TC=Ordering cost + Holding cost + Backorder cost + Perishing cost.

The decision variables are re-order point *s*, and order up to point *S*. A simulation model was constructed to estimate the total cost function. We try to optimize the objective (the cost) function using two different approaches as in the previous problem. These approaches were STS and TS+FSP respectively. Our aim is to compare these methods and examine the effects of the FSP on TS.

3.2.3. Job Shop Problem

Our third problem is a job shop model. The model is taken from Law and Kelton (2000). Job shop production environments are part of many production facilities. We thought, it would be appropriate to apply our methodology on a job shop problem in order to be more realistic. There are five workstations and one input/output station in the shop. The machines in a particular station are identical. Jobs are transferred by forklift truck(s) from one station to another. The following figure illustrates the outline of the shop:

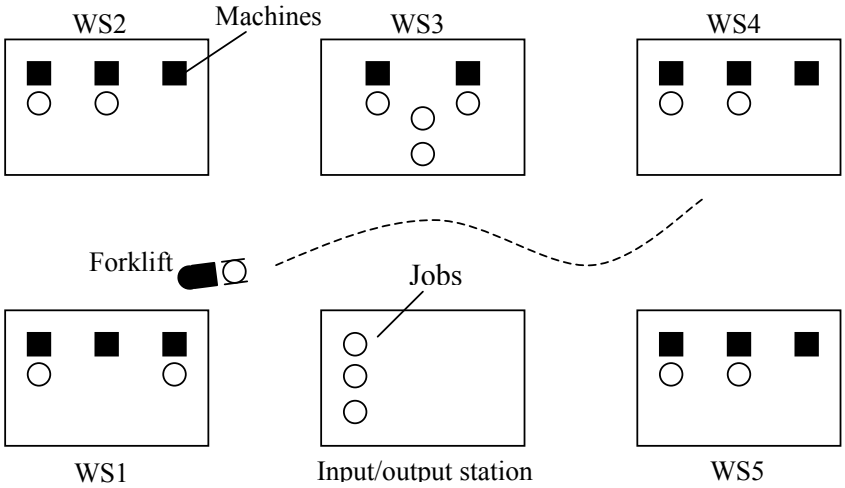


Figure 3.3. The outline of the job shop problem.

Jobs enter and leave the system through the station 6. The inter-arrival times of the jobs are exponentially distributed with mean $\mu_a = 1/15$ hours. There are three types of jobs namely job 1, job 2, and job 3 with probabilities 0.3, 0.5, and 0.2 respectively. The jobs require a number of operations to be done. The number of operations depends on the type of the job. Each operation must be done at a specified workstation in a specified order. The routings of the jobs for each job type are illustrated in the following table:

Table 3.3. The routings of the jobs.

Job Type	Workstations
1	3-1-2-5
2	4-1-3
3	2-5-1-4-3

According to above table, a job of type 1 has to travel 3rd, 1st, 2nd, and 5th stations respectively in order to finish its processing and to leave the system. Of course, since jobs enter the system from station 6, the job first has to be moved to station 1 from station 6. Likely when the job finishes its processing at station 5, it has to be moved to station 6 to leave the system. A job is moved from one station to another by a forklift truck. A forklift truck moves at a constant speed of 5 feet per second. When a forklift becomes available, if there is more than one jobs waiting to be moved then the forklift processes the requests in increasing order of the distance between the forklift and the job. In other words, shortest distance first rule applies. The distances between stations are given in Table 3.4. When a forklift finishes moving a job to a station, it stays at that station if there is not any waiting job to be moved. On the other hand, when a job arrives at a station, if there is no available machine to process the job then the job joins a single FIFO queue at that station. Similarly, if there is not any forklift waiting in the 6th station, arriving jobs join a queue at the 6th station.

Table 3.4. The distances between the stations (in feet).

Stations	1	2	3	4	5	6
1	0	150	213	336	300	150
2	150	0	150	300	336	213
3	213	150	0	150	213	150
4	336	300	150	0	150	213
5	300	336	213	150	0	150
6	150	213	150	213	150	0

The processing time of a particular machine is a gamma random variable with shape parameter of 2 whose mean depends on the job type and the workstation to which the machine belongs. The mean processing times for each job type and each operation is given in the following table:

Table 3.5. The mean processing times of the machines.

Job Type	Mean processing time
1	0.25 - 0.15 - 0.10 - 0.30
2	0.15 - 0.20 - 0.3
3	0.15 - 0.10 - 0.35 - 0.20 - 0.20

When a machine finishes processing a job, it remains blocked until a forklift removes the job.

We built a simulation model for this system design to use it as an evaluator in TS. A simulation run lasts 920 hours where the first 120 hours constitute the warm-up period.

The decision variables are the number of machines at the workstations and the number of forklifts. Let $x = (x_1, x_2, x_3, x_4, x_5, x_6)$ denote a solution where $x_1, x_2, x_3, x_4,$ and x_5 represent the number of machines at each workstation respectively, and x_6 represents the number of forklifts. Our objective is to maximize the total profit function. We consider two cases with regard to the profit function. In the first case, all finished jobs are treated to be same and each has a common outcome, which is \$2. Similarly each machine's cost is same and set to \$250, where a forklift truck costs \$50. According to this setting the objective function is formed as:

$$P = \text{finished jobs} \cdot 2 - (x_1 + x_2 + x_3 + x_4 + x_5) \cdot 250 - x_6 \cdot 50$$

In the second case, however, outcome of each job depends on its type and total processing time. This means, if the total processing time of a job is bigger than the other one then the outcome of the first job will be bigger than the second. For example, the total processing time of a job 3 is 1 hour, where it is 0.8 hours for a job 1. Let the outcome of the job 3 be \$2 then the outcome of the job1 will be 0.8·2=\$1.6. Additionally, we adjust the machine costs according to the total processing time at each workstation. We can calculate the total processing time for each workstation, and then calculate the machine costs according to these times. For example, the total processing times are $0.25 + 0.3 + 0.2 = 0.75$ and $0.15 + 0.2 + 0.35 = 0.70$ for workstations 3 and 1 respectively. If we set the cost of a machine at the workstation 3 equal to \$250 then the cost of a machine at workstation 1 will be $0.7 \cdot 250 / 0.75 = \$233.33$. The adjusted profit and cost values for jobs and machines are given in the following table:

Table 3.6. The adjusted profits/costs of jobs/machines.

Job Types	Profit ($)	Machines at	Cost ($)
Job 1	1.6	Workstation 1	233.33
Job 2	1.3	Workstation 2	83.33
Job 3	2	Workstation 3	250
		Workstation 4	116.66
		Workstation 5	133.33

According to this table the total profit function can be formed as:

$$P = \text{finished job 1s} \cdot 1.6 + \text{finished job 2s} \cdot 1.3 + \text{finished job 3s} \cdot 2 - (x_1 \cdot 233.33 + x_2 \cdot 83.33 + x_3 \cdot 250 + x_4 \cdot 116.66 + x_5 \cdot 133.33) - x_6 \cdot 50$$

3.2.4. Three-stage Buffer Allocation Problem

Our fourth problem is very similar to the first problem. It is taken from Pichitlamken and Nelson (2002). The system consists of three workstations and two buffers between them. Following figure illustrates outline of the system:

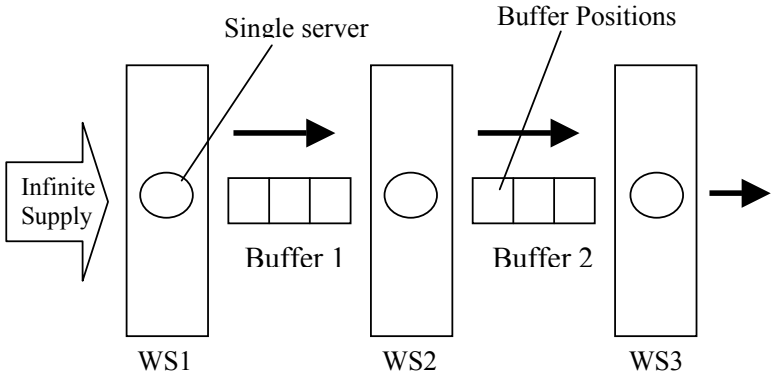


Figure 3.4. The outline of the three-stage buffer allocation problem.

Each workstation has a single machine, which has an exponentially distributed processing time with rate μ_i, i = 1,2, and 3. This is the main difference from the first problem. Apart from that, the two systems operate exactly the same way.

However, our decision variables and objective function are changed. This time our objective is to maximize the throughput, which is defined as the average output of the system per unit time. The decision variables are the service rates and the buffer sizes. Let $x = (x_1, x_2, x_3, x_4, x_5)$ denote a solution then the x_1, x_2, x_3 are the service rates for workstations 1,2, and 3 respectively, where the x_4 and x_5 are the number of buffer positions in each buffer. There are also some constraints over decision variables. These are:

$$x_1 + x_2 + x_3 \leq 20$$
$$x_4 + x_5 \leq 20$$
$$-x_4 - x_5 \leq -20$$
$$1 \leq x_i \leq 20, \quad i = 1,2,3$$
$$1 \leq x_i \leq 20, \quad i = 4,5$$
$$x_i \in Z^+$$

The simulation run length is set to 2050 parts are served. The warm-up period is the first 2000 parts. This means the throughput is calculated over 50 parts.

The results of the experimental study covering these four systems will be illustrated in the following chapter.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1. Manufacturing Problem

4.1.1. The Construction

We developed a TS algorithm to optimize above problem. A whole solution is decided to be tabu since it is less restrictive. Tabu list records whole solution instead of any solution variable. If a solution is in tabu list then it becomes tabu and it is ignored by the search. We decided tabu tenure to be 5, and the maximum number of iterations to be 30. If tabu tenure is too small then most probably the search will go into a cycle. Since a solution that becomes tabu is a very good solution around that neighbourhood, the search will select that solution as the best after a few iterations later and it will go into a cycle. On the other hand, if tabu tenure is too large then the search may be diverted from the neighbourhood of the real best solution. This causes the search to be delayed, which reduces the efficiency. Thus we intuitively thought 5 would be a good choice considering the maximum number of iterations is 30. We observed that in most of the experiments the TS algorithm found the best solution far before the 30^{th} iteration. So there was no reason to keep searching beyond the 30^{th} iteration. Our stopping criterion is reaching maximum number of iterations. Our neighbourhood strategy is the following. At first we fix the number of buffer positions for a certain number (9) of iterations. Our aim is to find a good combination of the number of machines. At each iteration, the search records the

number of machines $x_1, x_3, x_5,$ and x_7 of the best solution. At the end of the ninth iteration, the most frequent machine numbers are calculated. Since these numbers are the most frequent they together constitute a good combination of the number of machines. For the next nine iterations we fix the number of machines and try to find a good combination of the number of buffer positions in each buffer. For example, suppose the following table shows the number of machines in the best solutions of first nine iterations:

Table 4.1. The number of machines in the best solutions of the first nine iterations.

Iteration	WS1	WS2	WS3	WS4
1	2	2	2	2
2	2	3	2	2
3	3	3	3	2
4	3	3	2	2
5	3	3	2	3
6	3	3	2	2
7	3	3	3	3
8	3	3	2	2
9	3	3	2	2

The most frequent numbers are 3,3,2,2 for the workstations respectively. For the following nine iterations this combination will be fixed, and the search will try to find a good combination of the number of the buffer positions.

We coded the simulation model in Borland Delphi 6.0 with the accompanying TS algorithm. There are two versions of the code. One version takes arbitrarily and pre-specified number of replications to evaluate the alternatives. We call this version Standard Tabu Search (STS). The other version uses FSP to evaluate the alternatives. We call this version TS with FSP (TS+FSP).

The number of replications to evaluate the neighbours (alternatives) was set to 5 in STS. The number of replications for the first stage sampling n_0 for FSP was set to 5. The indifference zone parameter δ was set to \$10. If this parameter is too small then the computational effort increases dramatically. On the other hand, if it is too large then the probability of the wrong selection increases. Five experiments for each version with different initial solutions and with different random numbers were made. CRN was employed in both versions. The experiments were conducted on a

PC that has a Pentium 4 1.7 GHz processor with 256 MB RAM. The operating system on the computer is Microsoft Windows 2000 5.00.2195.

4.1.2. Results

Following tables show the results of the STS and TS+FPS, respectively:

Table 4.2. The results of STS method.

STS						
Initial solution	Best Solution	Value ($)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications
1 1 1 1 1 1 1	3 8 3 7 2 6 2	5935	161	5	10	1690
3 3 3 4 2 3 2	3 7 3 7 2 6 2	5914	134	5	10	1505
3 6 3 5 2 4 2	3 7 3 7 2 5 2	5945	133	5	10	1505
2 4 2 4 2 4 2	3 8 3 7 2 5 2	5776	161	5	10	1790
3 8 3 8 3 8 3	3 7 3 7 2 4 2	5946	134	5	10	1465

Table 4.3. The results of TS+FSP method.

TS+FSP (indifference zone=\$10, $n_0 = 5$)							
Initial solution	Best Solution	Value ($)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications	
						n_0	Additional
1 1 1 1 1 1 1	3 7 3 7 2 4 2	5912	205	20	11	1850	851
3 3 3 4 2 3 2	3 8 3 7 2 5 2	5884	188	27	17	2695	1882
3 6 3 5 2 4 2	3 8 3 7 2 4 2	5873	159	26	13	2010	964
2 4 2 4 2 4 2	3 7 3 7 2 5 2	5897	188	27	11	2040	2311
3 8 3 8 3 8 3	3 6 3 7 2 5 2	5889	160	22	15	1710	944

Best solution column represents the best solution found by the search. *Value* column shows the objective function value of the best solution in dollars. *Number of visited solutions* column represents the number of distinct solutions that are encountered by the search. *Number of replications* column represents the number of replications that were taken from each alternative to find the best among them. In TS+FSP this number varies because FSP keeps taking replications one at a time from each alternative until finding the best. *Iteration number* column shows the number of iteration that the best solution updated for the last time. This means after that iteration the search could not find a better solution. *Total number of replications* column represents the total number of replications that were taken up to *iteration number*. In TS+FSP table this column is further divided into two columns. The first column represents the total number of the replications that were taken during the first

stage sampling, while the second column represents the total number of the additional replications taken during the screening process.

When we look at the results we observe similarities in the solutions. The combination of the number of machines is same '3,3,2,2' in all of the solutions. The number of buffer positions '7' in buffer 2 is also common. The only difference is in the number of buffer positions in buffer 1 and 3. Solution values that are found by STS seem better than TS+FPS's. Furthermore, when we compare the total number of replications, STS is much better than TS+FPS concerning computational effort. Actually total numbers of replications represented in the tables are exaggerated. Since we use same random numbers within each experiment, objective function value of a specific solution will not change throughout the search. So there is no need to evaluate solutions encountered before in STS. This is, record the objective function value of any solution encountered by the search, if this solution is needed to evaluate in the next iterations, instead of taking five replications again use the recorded value. This situation is a little bit different in TS+FSP case. Above approach can be used in first stage sampling since it has no differences with STS case. But when it comes to take additional replications, recorded values cannot be used. One additional replication must actually be taken. The first row of the total number of replications column then will be at most $161 \times 5 = 805$ in STS case. It will be at most $5 \times 205 + 851 = 1861$ in TS+FSP case.

When we compare the methods according to the number of visited solutions, STS is better than the other. It scanned smaller portion of the solution space to end up with good solutions compared to TS+FSP.

The methods almost similarly converged to the best solution (near optimal). In all of the experiments the best solution was found at 10^{th} iteration in STS case. This number varied in TS+FSP between 10 and 18. The reason that the methods found the best solutions around 10^{th} iteration lies in our neighbourhood generation algorithm. The algorithm tries to find a good combination of the number of the machines at the workstations for the first 9 iterations. After finding the combination, the number of buffer positions is determined. In a few number of iterations after 9^{th} iteration, the methods find the best solution. Figure 4.1. and Figure 4.2. show the convergence of the methods for different initial solutions. We add STS-10 case in which the evaluations are based on 10 replications into the picture:

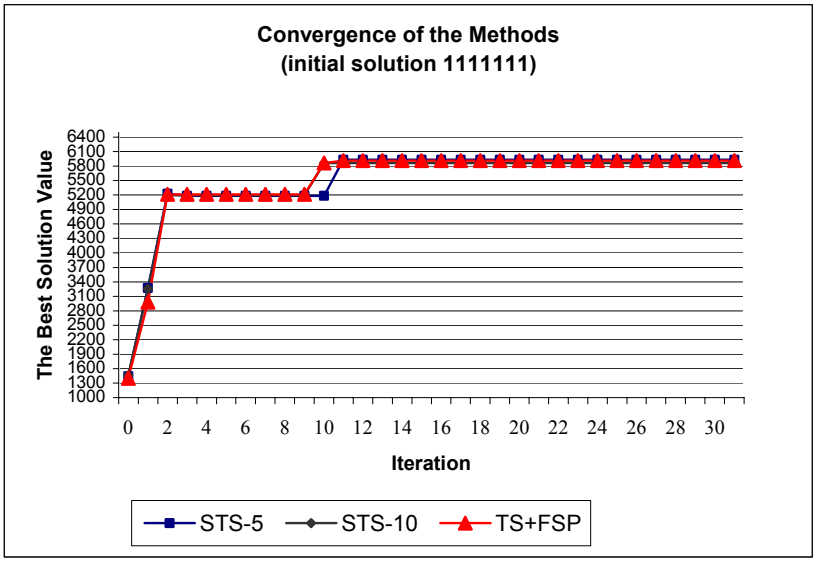


Figure 4.1. The convergence of the methods when the initial solution is (1111111).

The convergences of the methods are almost same. This means, the idea of using TS+FPS to quickly converge to the best solution is proven wrong for this problem. The methods jumped to similar solutions at every iteration. But, STS used only five replications where TS+FSP used much more replications to statistically ensure the best selection.

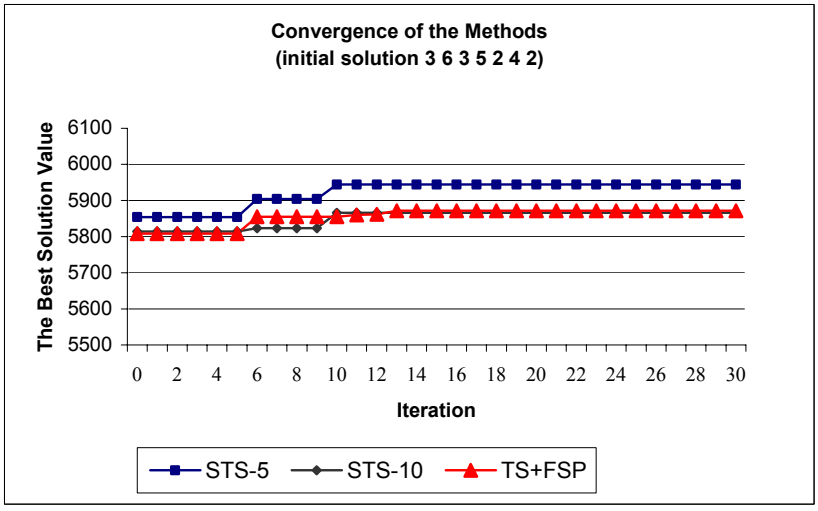


Figure 4.2. The convergence of the methods when the initial solution is (3 6 3 5 2 4 2).

The picture is not different for the initial solution 3 6 3 5 2 4 2. This time we see STS-5 overestimated the best solution. Probably, the random numbers that were used caused this overestimation. When the number of replications increased i.e., in STS-10 and in TS+FSP it seems the overestimation was removed.

When we look at the computational times that were spent by the methods, we observe that the STS method outperformed the TS+FSP. Table 4.4 shows the computational times in seconds for different experiments:

Table 4.4. The computational times of the methods.

Initial Solution	STS		TS+FSP		
	Simulation Time	Non-simulation Time	Simulation Time	Non-simulation Time	Screening Time
1 1 1 1 1 1 1	232.57	0.08 (0.03%)	423.53	0.12 (0.03%)	184.57 (43.5%)
3 3 3 4 2 3 2	231.70	0.06 (0.03%)	473.78	0.08 (0.02%)	210.47 (44.4%)
3 6 3 5 2 4 2	259.05	0.12 (0.05%)	471.85	0.11 (0.02%)	205.76 (43.6%)
2 4 2 4 2 4 2	250.66	0.09 (0.04%)	655.43	0.20 (0.03%)	392.93 (59.9%)
3 8 3 8 3 8 3	273.47	0.08 (0.03%)	556.87	0.17 (0.02%)	282.53 (50.7%)

The sum of the *Simulation Time* and the *Non-simulation Time* columns gives the total search time. The simulation time represents the time that is spent for simulation within the search. The non-simulation time represents the time that is spent for the other operations such as generation of neighbours, comparison of solutions, and maintaining the tabu list etc. The *Screening Time* indicates the total time that is spent for the screening process including the simulation time of additional replications. For example, in the TS+FSP method, the first experiment lasted $423.53 + 0.12 = 423.65$ seconds. The 184.57 seconds of this time were spent for screening process. Note that the values within parenthesis are the percentages of related times over total search time.

When we look at the table, we observe that most of the simulation times were doubled in the TS+FSP method compared to the STS method. That is an expected result when we think the results in Table 4.2. and Table 4.3. The TS+FSP method took much more replications than the other. This caused the search to last much longer than the STS. Looking closer one can observe the parallelism between these tables and the Table 4.4. Non-simulation time values seem negligible. On the other hand, screening times took almost the half of the total search time. The reason behind this situation is, most probably, that the smallness of the indifference zone parameter.

Although the above picture clearly shows that STS performed better than TS+FSP, one more analysis has to be done to be more definite. Since the solutions found by STS are results of only five replications, the objective function values of these solutions may be misleading. We took 100 replications of each solution founded by both methods. The following table represents the results:

Table 4.5. The performances of the solutions based on 100 replications.

Solution (STS)	Value ($)	Standard Deviation	Solution (TS+FPS)	Value ($)	Standard Deviation
3 8 3 7 2 6 2	5861	123.4	3 7 3 7 2 4 2	5863	120.86
3 7 3 7 2 6 2	5861	122.93	3 8 3 7 2 5 2	5863	121.27
3 7 3 7 2 5 2	5865	122.39	3 8 3 7 2 4 2	5861	119.74
3 8 3 7 2 5 2	5863	121.27	3 7 3 7 2 5 2	5865	122.39
3 7 3 7 2 4 2	5863	120.86	3 7 3 7 2 6 2	5861	122.93

The values of the solutions are very close to each other. Although in some cases solutions found by TS+FPS seem better, the difference between values are insignificant. In other words, it is not worth additional computational effort.

We know that if we increase the first stage sampling size n_0, FSP performs better. So we decided to increase n_0 to 10 to see if TS+FSP could find better solutions. The following table shows the results of the five experiments:

Table 4.6. The results of TS+FSP method when $n_0 = 10$

TS+FSP (indifference zone=\$10, $n_0 = 10$)							
Initial solution	Best Solution	Value ($)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications	
						n_0	Additional
1 1 1 1 1 1 1	3 8 3 7 2 5 2	5908	152	13	14	4501	51
3 3 3 4 2 3 2	3 7 3 7 2 5 2	5887	134	10	10	3210	77
3 6 3 5 2 4 2	3 7 3 7 2 5 2	5891	158	10	10	3210	123
2 4 2 4 2 4 2	3 8 3 7 2 4 2	5893	187	11	13	4590	152
3 8 3 8 3 8 3	3 7 3 7 2 4 2	5896	160	10	10	3130	185

The picture is not different than the previous one. Only required additional computation dramatically reduced at the cost of doubling n_0. Decreasing the indifference zone parameter can further increase the efficiency of the FSP. This also will increase the computational effort. But, since STS already found very good solutions, this approach seems meaningless.

Another point that should be stressed out is, as we said before, due to stochasticity one cannot be a hundred percent sure if a solution is optimal or not. It is clear from the tables that different random numbers direct us different solutions. Fortunately, these solutions are very close to each other and one of them can be selected as the best. Of course one can further inspect these solutions to find the real best among them, but most probably the gain will be insignificant considering the additional computation.

As a conclusion TS+FSP did not perform better than STS contrary to our expectation. We thought simplicity of the problem caused that result. Even five replications are enough to distinguish good solutions from others. Only source of variability of the system is processing times of the machines. We thought if we could increase the system variability then this might have made it difficult to distinguish good systems taking small number of replications. First we decided to increase the processing time variability. We doubled the mean processing times of the machines at each workstation. The following tables illustrates the results:

Table 4.7. The results of STS method with doubled processing times.

STS						
Initial solution	Best Solution	Value ($)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications
1 1 1 1 1 1 1	3 7 3 7 2 4 2	1630	161	5	10	1750
3 3 3 4 2 3 2	3 6 3 7 2 4 2	1620	160	5	10	1605
3 6 3 5 2 4 2	3 6 3 7 2 4 2	1591	133	5	10	1755
2 4 2 4 2 4 2	3 6 3 5 2 4 2	1583	187	5	30	5040
3 8 3 8 3 8 3	3 6 3 5 2 4 2	1574	134	5	10	1265

Table 4.8. The results of TS+FSP method with doubled processing times.

TS+FSP (indifference zone=\$10, $n_0 = 5$)							
Initial solution	Best Solution	Value ($)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications	
						n_0	Additional
1 1 1 1 1 1 1	3 6 3 7 2 4 2	1603	178	14	10	1715	323
3 3 3 4 2 3 2	3 6 3 6 2 4 2	1605	160	18	12	1875	510
3 6 3 5 2 4 2	3 6 3 5 2 4 2	1629	158	5	>30	-	-
2 4 2 4 2 4 2	3 7 3 5 2 5 2	1606	179	55	28	4955	4101
3 8 3 8 3 8 3	3 6 3 7 2 4 2	1603	134	12	10	1715	1537

When we doubled the processing times the objective function values are reduced as expected. The solutions found by both methods still seem very close to each other. But, TS+FSP method requires much more replications to come up with these solutions. In the third experiment, TS+FSP method could not find a better solution from the initial. There may be two explanations for this situation. The first one is the initial solution is actually a very good solution. And the second one is, the FSP takes additional replications (more than 5) causing the sample means of the solutions are reduced to their real levels. Since the initial solution is evaluated based on 5 replications, the sample mean of the initial solution will be greater than the

other solutions that are evaluated during the search. This situation causes the search could not find a better solution.

The results make us to conclude that there is no need to take more than five replications to distinguish the best system thus the FSP becomes useless again. Another way to increase the system variability is to add breakdowns for the machines. We added breakdowns to the machines in the following manner.

Since this system is a conceptual system, there is no available information about the distributions of the machine up times and down times. We used *busy time* approach described in Law and Kelton (2000) to model the machine breakdowns. According to this approach "the amount of the machine busy time before a failure has a gamma distribution with shape parameter $\alpha_B = 0.7$ and the scale parameter β_B to be specified. The machine down time (or repair time) has a gamma distribution with shape parameter $\alpha_D = 1.4$ and a scale parameter β_D to be determined." β_B and β_D are calculated as the following:

$$\beta_B = \frac{e \cdot \mu_D}{0.7(1-e)} \qquad \beta_D = \frac{\mu_D}{1.4}$$

where "the efficiency e is defined to be the long run proportion of potential processing time (i.e., parts present and machine not blocked) during which the machine actually processing parts." It is calculated as:

$$e = \frac{\mu_B}{\mu_B + \mu_D}$$

where $\mu_B = E(\mathrm{B})$ mean amount of machine busy time before a failure and $\mu_D = E(\mathrm{D})$ is estimate of machine down time.

We calculated these parameters by taking five replications of the model. We used the solution (3 8 3 7 2 4 2) as an input for the model. Because it is one of the good (near best) solutions and it is more appropriate than any unrealistic solution as (1 1 1 1 1 1 1). The following table shows the β_B and β_D for each machine:

Table 4.9. The values of β_B and β_D parameters of related machines.

Machine	Efficiency	β_B	β_D
WS1 Machine1	0.685	5,180	1,190
WS1 Machine2	0.679	5,040	1,190
WS1 Machine3	0.680	5,072	1,190
WS2 Machine1	0.991	394,107	1,785
WS2 Machine2	0.990	366,007	1,785
WS2 Machine3	0.989	323,990	1,785
WS3 Machine1	0.658	2,757	0,714
WS3 Machine2	0.542	1,691	0,714
WS4 Machine1	0.789	6,709	0,892
WS4 Machine2	0.692	4,015	0,892

Except for the machines at workstation 2, obtained results are not realistic. Machines up times before a failure are too small which is not the case in real life. Only for the machines at workstation 2 the parameters seem logical. So we added breakdowns for the machines at workstation 2. After a second thought we decided to add breakdowns for the machines at workstation 1 with the same parameters that were for the machines at workstation 2. It is logical because the parts enter the system from workstation 1, breakdowns at this station may affect the system variability.

We performed same experiments in previous case but this time with breakdowns. The following tables illustrate the results of STS and TS+FSP respectively:

Table 4.10. The results of STS method with breakdowns.

STS (with breakdowns)						
Initial solution	Best Solution	Value ($)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications
1 1 1 1 1 1 1	3 8 3 7 2 5 2	5799	161	5	10	1690
3 3 3 4 2 3 2	3 8 3 7 2 4 2	5818	134	5	10	1510
3 6 3 5 2 4 2	3 7 3 7 2 5 2	5804	133	5	10	1510
2 4 2 4 2 4 2	3 7 3 7 2 5 2	5842	161	5	10	1790
3 8 3 8 3 8 3	3 6 3 7 2 6 2	5796	134	5	10	1415

Table 4.11. The results of TS+FSP method with breakdowns.

TS+FSP (indifference zone=\$10, $n_0 = 5$)							
Initial solution	Best Solution	Value (\$)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications	
						n_0	Additional
1 1 1 1 1 1 1	3 8 3 7 2 4 2	5845	206	37	17	2120	3882
3 3 3 4 2 3 2	3 8 3 6 2 5 2	5832	161	37	12	1875	1296
3 6 3 5 2 4 2	3 7 3 7 2 4 2	5837	186	29	17	2550	2443
2 4 2 4 2 4 2	3 7 3 7 2 6 2	5845	188	22	13	2295	2891
3 8 3 8 3 8 3	3 7 3 7 2 5 2	5871	161	23	11	1700	1586

As it is seen from the tables, solutions are very close to each other as in the previous case. Concerning computational effort we can clearly say STS is better than TS+FPS. It is understood from the results that adding breakdowns did change the variability of the system. Because, the additional computational effort was increased according to the previous case. The FSP needed to take much more additional replications to select the best solution. This situation is echoed in the computational times table. Table 4.12. shows the computational times for this case:

Table 4.12. The computational times of the methods.

Initial Solution	STS		TS+FSP		
	Simulation Time	Non-simulation Time	Simulation Time	Non-simulation Time	Screening Time
1 1 1 1 1 1 1	301.98	0.12 (0.04%)	691.28	0.18 (0.03%)	385.4 (55.7%)
3 3 3 4 2 3 2	307.36	0.03 (0.01%)	622.71	0.33 (0.05%)	309.7 (49.7%)
3 6 3 5 2 4 2	318.04	0.06 (0.02%)	782.29	0.23 (0.03%)	461.1 (58.9%)
2 4 2 4 2 4 2	324.89	0.03 (0.01%)	1065.46	0.34 (0.03%)	737.6 (69.2%)
3 8 3 8 3 8 3	316.81	0.03 (0.01%)	950.58	0.25 (0.03%)	610.7 (64.2%)

The screening times increased compared to the case without breakdowns. This is a direct result of increasing system variability by adding breakdowns to the machines. The long run performances of the best solutions found by both methods are shown below:

Table 4.13. The long run performances of the best solutions found by both methods.

Solution (STS)	Value (\$)	Standard Deviation	Solution (TS+FPS)	Value (\$)	Standard Deviation
3 8 3 7 2 5 2	5779	133.93	3 8 3 7 2 4 2	5779	132.42
3 8 3 7 2 4 2	5779	132.42	3 8 3 6 2 5 2	5773	133.45
3 7 3 7 2 5 2	5779	133.68	3 7 3 7 2 4 2	5778	132.73
3 7 3 7 2 5 2	5779	133.68	3 7 3 7 2 6 2	5777	133.94
3 6 3 7 2 6 2	5770	134.95	3 7 3 7 2 5 2	5779	133.68

The convergence of the methods was similar with previous case. Figure 4.3. shows the convergence of the methods for a specific initial solution:

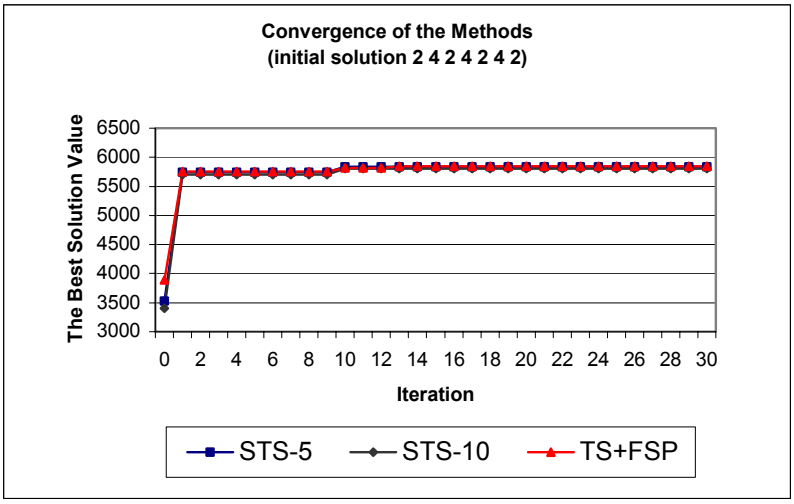


Figure 4.3. The convergence of the methods when the initial solution is (2 4 2 4 2 4 2).

It seems all three methods converged to the best solution in same precision. Again TS+FSP needed to take much more replications than the others.

In sum, we conclude that STS performed better than TS+FPS and our claim is failed for this problem.

4.2. Inventory Control Problem

4.2.1. The Construction

We used similar TS algorithm to the one that was used in the previous problem. The only difference is the neighbourhood generation algorithm. In this case, we properly decreased and then increased one of the decision variables while remaining the other fixed to generate neighbours around the current solution. In other words, the decision variables were made to vary in a range with respect to each other. The number of neighbours was set to 81. Adjusting the length of the range this number can be changed. For example, let the current solution be (-5,10) and range equal to 4. To generate the neighbours to this solution, first the algorithm fixes $s = -5$ then S is increased and decreased as a multiple of 0.25 in the range $10 - 4 \cdot 0.25 \le S \le 10 + 4 \cdot 0.25$. The neighbours of the solution (-5,10) according to this setting are (-5,9), (-5,9.25), (-5,9.5), (-5,9.75), (-5,10), (-5,10.25), (-5,10.5),

(-5,10.75), (-5,11). In the following steps s is varied within the range $-5-4\cdot 0.25 \le s \le -5+4\cdot 0.25$ and above process is repeated for every s. The following table shows the neighbours of the solution (-5,10):

Table 4.14. The neighbours of the solution (-5,10).

s	S								
-6	9	9.25	9.5	9.75	10	10.25	10.5	10.75	11
-5.75	9	9.25	9.5	9.75	10	10.25	10.5	10.75	11
-5.5	9	9.25	9.5	9.75	10	10.25	10.5	10.75	11
-5.25	9	9.25	9.5	9.75	10	10.25	10.5	10.75	11
-5	9	9.25	9.5	9.75	10	10.25	10.5	10.75	11
-4.75	9	9.25	9.5	9.75	10	10.25	10.5	10.75	11
-4.5	9	9.25	9.5	9.75	10	10.25	10.5	10.75	11
-4.25	9	9.25	9.5	9.75	10	10.25	10.5	10.75	11
-4	9	9.25	9.5	9.75	10	10.25	10.5	10.75	11

We chose the whole solution to be tabu. The tabu tenure was set to 5 iterations. The simulation model and TS algorithm was coded in Borland Delphi 6.0. We applied the methods for both fixed and random shelf lives. The CRN was employed. The indifference zone was set to \$200. This time we set the indifference zone parameter value relatively high to avoid the unnecessary computational effort. The confidence level was set to %95. The procedures were applied for 5 different initial solutions each with different random numbers.

4.2.2. The Results

The following tables show the results of the first set of experiments where the shelf life was fixed to 0.5 time unit and the lead time was set to 1 time unit:

Table 4.15. The results of STS method.

STS						
Initial solution	Best Solution	Value (\$)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications
2 , 15	-10.25, 24	2791	840	5	16	6160
2 , 10	-10 , 23.5	2767	910	5	17	6545
2 , 5	-10.25 , 23.75	2779	1187	5	23	8855
4 , 5	-10 , 23.75	2773	1219	5	26	10010
-1 , 5	-10.25 , 24.25	2793	1180	5	22	8020

Table 4.16. The results of TS+FSP method.

TS+FSP (indifference zone=\$200, $n_0 = 5$)							
Initial solution	Best Solution	Value (\$)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications	
						n_0	Additional
2 , 15	-10.25 , 23.75	2784	868	5	19	7315	60
2 , 10	-10 , 23.5	2781	928	6	16	6160	90
2 , 5	-9.25 , 23	2774	1189	6	23	8855	7631
4 , 5	-10 , 24.25	2777	1242	5	28	10780	7190
-1 , 5	-9.75 , 24.25	2785	1123	5	27	10395	7585

The results of the procedures are very close to each other. The objective function values of related solutions are almost same. The difference between them is insignificant. It can be observed from Table 4.16. that the required number of replications that must be taken to distinguish the best system is 5 in most of the cases, and it is 6 in the remaining. This means, FSP did not need to take additional replications to select the best. This result may be interpreted as taking 5 replications is enough to find out the best system among the alternatives, and there is no need for further inspection. In this case, additional computational effort that is spent by FSP (e.g., calculating variances of the differences of the systems, screening process etc.), becomes useless. Why to spend time and money where one can select the best alternative by simply taking 5 replications.

When we look at the *additional* column under *the total number of replications* header, we observe the number of additional replications in the first two rows are very small when compared to remaining three rows. The initial solutions of these experiments caused this huge difference. Since the initial solutions are far from the best solution, their cost function values are incredibly high. For example, the total cost is \$72490 for the solution (5,2). At the early stages of the search (e.g., first two or three iterations) the costs are still too high because the solutions at these iterations are in the neighbourhood of the initial solution. Since indifference zone $\delta = \$200$ is too small compared to these high costs, FSP requires much more additional replication to select the best among these high cost valued solutions. When the cost values become reasonable ranges at the later iterations of the search, the required number of additional replications is reduced to zero. In fact 7559 out of 7631 additional replications were taken in the first six iterations of the third experiment. This situation leads us to the following conclusion. If the initial solution is relatively close to the best solution then the computational effort is incredibly reduced.

The numbers of visited solutions are close for both methods. This can be interpreted as at each iteration both methods selected the similar or same solutions as the current solution. Since neighbours were formed according to these solutions, the procedures span the similar portions of the solution space.

When we compare the total number of replications, STS seems performed better than TS+FSP. Except one case (the second experiment), TS+FSP requires much more replications than STS. As for the computational times, the picture did not chance much according to the previous problem. Table 4.17. shows the computational times for both methods:

Table 4.17. The computational times of the methods.

Initial Solution	STS		TS+FSP		
	Simulation Time	Non-simulation Time	Simulation Time	Non-simulation Time	Screening Time
2 , 15	49.1	2.87 (5.53%)	49.3	3.32 (6.31%)	0.06 (0.12%)
2 , 10	49.2	2.90 (5.57%)	49.3	3.48 (6.59%)	0.09 (0.17%)
2 , 5	53.4	2.95 (5.23%)	111.7	3.58 (3.16%)	57.78 (50.15%)
4 , 5	53.4	2.93 (5.21%)	87.8	3.57 (3.91%)	33.67 (36.84%)
-1 , 5	53.4	2.99 (5.31%)	93.5	3.59 (3.70%)	39.57 (40.74%)

Note that the resemblance to the Tables 4.15 and 4.16. Different from the previous problem the non-simulation times were a little increased. The neighbourhood generation algorithm might cause this situation. The algorithm has many loops within loops to generate the neighbours. This might have taken a while. Again we need to look at the long run performances of the solutions to be able to truly compare the solutions. We took 100 replications of each solution. The following table shows the performances of the system based on 100 replications:

Table 4.18. The performances of the solutions based on 100 replications.

Solution (STS)	Value ($)	Standard Deviation	Solution (TS+FPS)	Value ($)	Standard Deviation
-10.25, 24	2822	54.66	-10.25 , 23.75	2820	51.36
-10 , 23.5	2820	57.23	-10 , 23.5	2820	57.23
-10.25 , 23.75	2820	51.36	-9.25 , 23	2819	57.15
-10 , 23.75	2820	58.30	-10 , 24.25	2821	55.95
-10.25 , 24.25	2819	55.78	-9.75 , 24.25	2834	64.33

According to these performances the solution (-10 , 23.75) can be selected as the best (near optimal). But it seems there is no significant difference between the most of the solutions.

Our results show that the convergences of the methods are almost same. Figure 4.4. and Figure 4.5. show the convergences of the methods for different initial solutions. Note that STS-10 method, in which evaluations are based on 10 replications, is added to the picture.

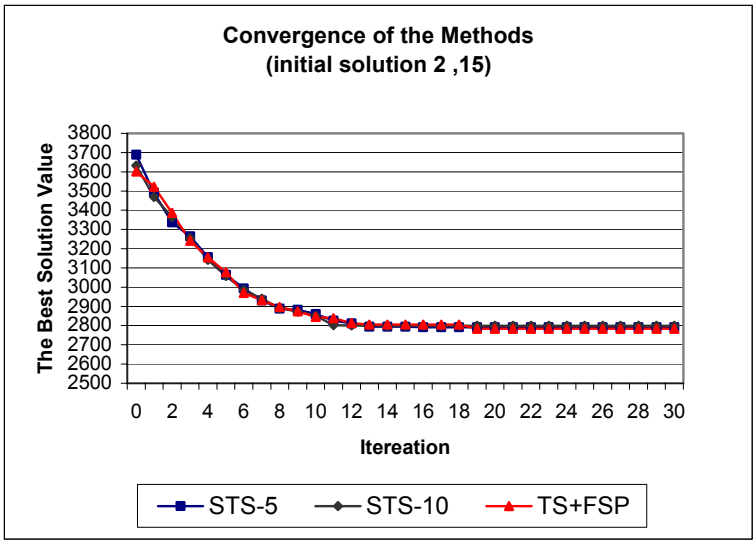


Figure 4.4. The convergences of the methods when the initial solution is (2,15).

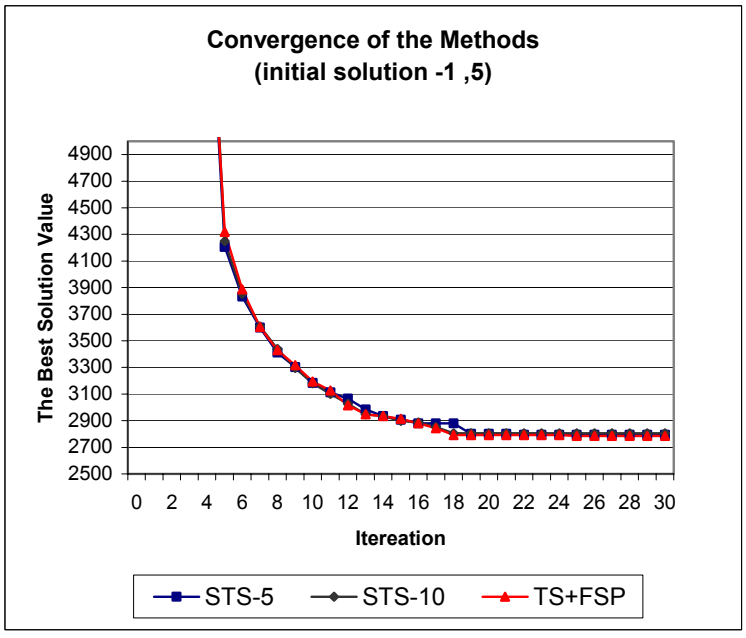


Figure 4.5. The convergences of the methods when the initial solution is (-1,5).

In the first graph the methods starts to converge after 10^{th} iteration, while in the second they starts to converge after 16^{th} iteration. Considering the different initial solutions this is an expected result.

It seems FSP procedure could not meet our expectations for this problem instance either. It is strange that the best solution can be selected based on five replications. Different from the previous problem FSP did not require additional replications apart from first stage sampling. This result indicates, indeed, even only the small number of replications is enough to distinguish the best alternative among others.

Before conducting experiments with random shelf life, we wanted to examine the results when the size of neighbour was increased (doubled). We adjusted our neighbourhood generation algorithm such that it doubles the size at each iteration according to the previous case. The following tables illustrates the results:

Table 4.19. The results of STS method with doubled number of neighbour solutions.

STS						
Initial solution	Best Solution	Value ($)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications
2 , 15	-10.25, 24	2791	1214	5	10	8250
2 , 10	-10 , 23.5	2767	1381	5	10	8250
2 , 5	-10.25 , 23.75	2779	1850	5	15	12730
4 , 5	-10 , 23.75	2773	1853	5	15	12730
-1 , 5	-10.25 , 24.25	2793	1781	5	15	12730

Table 4.20. The results of TS+FSP method with doubled number of neighbour solutions.

TS+FSP (indifference zone=\$200, $n_0 = 5$)							
Initial solution	Best Solution	Value ($)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications	
						n_0	Additional
2 , 15	-10.25 , 24.75	2719	1397	6	10	8250	117
2 , 10	-10.25 , 24.25	2724	1530	6	13	10725	94
2 , 5	-10.25 , 24	2726	1784	5	14	11550	11807
4 , 5	-9.75 , 24.5	2725	1929	5	17	14025	10069
-1 , 5	-9.25 , 23.25	2738	1774	5	15	12375	9532

The solutions found by the STS method are not different than the previous case. On the other hand, as expected, the numbers of visited solutions are increased since the search evaluates more solution points. Doubling the number of neighbours made the search quickly converge to the best according to the previous case. We can observe from the tables that the search found the best solutions at the earlier iterations compared to the previous case. This is also an expected result. Since more number of neighbours are evaluated, the search can find the best solution at the early iterations. However, the computational effort increases because of the same reason.

Furthermore, doubling the number of neighbours make the search to evaluate clearly inferior solutions which also increases the computational effort.

Although the solutions that were found by TS+FSP method are a little different than the previous case, there is no significant difference between the solutions. Again the method converged quickly than the previous case at the cost of increased computational effort. As a result, keeping the number of solutions at a reasonable level is beneficial considering the computational effort.

We performed same experiments with random shelf life. The following tables illustrate the results:

Table 4.21. The results of STS method with random shelf lives.

STS						
Initial solution	Best Solution	Value ($)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications
2 , 15	-10.25 , 24	2993	1090	5	18	7110
2 , 10	-11.75 , 24.25	2984	1045	5	20	7900
2 , 5	-10.25 , 23.25	3022	1171	5	30	11850
4 , 5	-11.25 , 24.25	2985	1368	5	28	11060
-1 , 5	-11 , 24	2994	1251	5	23	9085

Table 4.22. The results of TS+FSP method with random shelf lives.

TS+FSP (indifference zone=\$200, $n_0 = 5$)							
Initial solution	Best Solution	Value ($)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications	
						n_0	Additional
2 , 15	-11.75 , 24.5	3016	1019	15	23	9085	4537
2 , 10	-11 , 24.25	3004	1029	13	18	7110	2945
2 , 5	-12.25 , 25	3004	1292	14	29	10455	15521
4 , 5	-11.25 , 24.75	2982	1347	20	30	11850	11757
-1 , 5	-11.75 , 25.5	3005	1181	21	25	9875	14709

The solutions that were found by either method seem close to each other. The objective function values were a little raised according to the previous case. The computational effort was increased in TS+FSP method with regard to fixed shelf life case. This can be explained by additional variability that was added by introducing random shelf lives. As expected, FSP procedure required more replications to distinguish the best system. But the picture was not change. STS still ended up with good solutions in spite of the increased variability. It is confusing that one needs to

take huge number of replications in order to statistically ensure the best selection, where actually taking five replications is quite enough to select the best alternative.

One can claim that the main reason behind this situation is utilizing TS. But, since we used same random numbers within each experiment, it is not different gathering for example, 1090 solutions (the number of visited solutions by STS method in the first experiment) in to a pool and to evaluate them based on 5 replications to select the best solution among them. In this point of view FSP or any other R&S procedure become meaningless. Of course, we cannot generalize this statement but the two problems that we examined so far make us to come this conclusion.

To be fair we again looked the long run performances of the solutions and the convergences of the methods. Table 4.23. shows the long run performances of the solutions and the following graphs illustrate the convergences of the methods:

Table 4.23. The performances of the solutions based on 100 replications.

Solution (STS)	Value ($)	Standard Deviation	Solution (TS+FPS)	Value ($)	Standard Deviation
-10.25 , 24	3079	104.70	-11.75 , 24.5	3058	82.55
-11.75 , 24.25	3045	71.39	-11 , 24.25	3061	95.34
-10.25 , 23.25	3064	90.41	-12.25 , 25	3052	64.19
-11.25 , 24.25	30.51	94.57	-11.25 , 24.75	3060	101.74
-11 , 24	30.54	83.21	-11.75 , 25.5	3071	95.52

As it can be observed the long run performances of the solutions are close to each other. The solution (-11.25,24.25) seems a little better than the others. Thus, it can be selected as the best solution. Figure 4.6. and Figure 4.7. illustrates the convergence of the methods for different initial solutions:

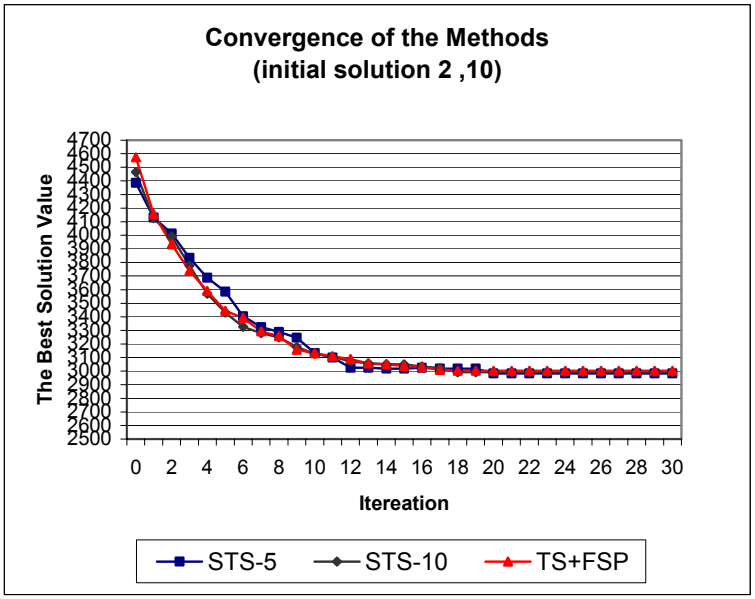


Figure 4.6. The convergences of the methods when the initial solution is (2,10).

The convergences of the methods are almost same. After 12th iteration methods starts to converge to the best solution.

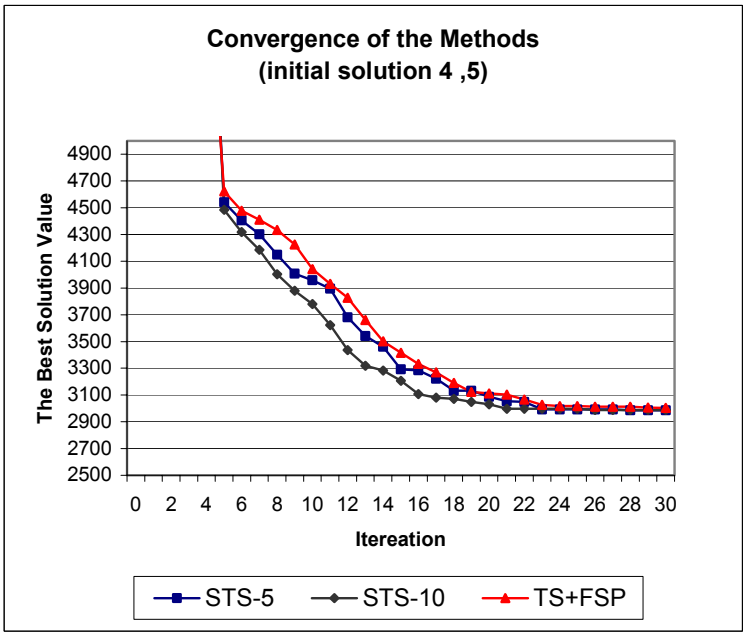


Figure 4.7. The convergences of the methods when the initial solution is (4,5).

In this sample, the methods start to converge to the best solution at the late iterations of the search. The distance of the initial solution from the best (near optimal) solution can explain this situation. Although the methods converge similarly, this figure does not have the smoothness of the other examples.

4.3. Job Shop Problem

4.3.1. The Construction

Similar TS algorithm was used to optimize the problem. At each iterartion, the neighbourhood generation algorithm generates 64 neighbours. The algorithm is based on decreasing and increasing solution variables by 1, respectively. For example, let (2 2 2 2 2 2) be a solution. The two neighbours of this solution are (1 1 1 1 1 1) and (1 1 1 1 1 3). As it seen, the last solution variable is increased and decreased by 1 while the others are decreased by 1. If we continue, (1 1 1 1 3 1) and (1 1 1 1 3 3) are other two neighbours. At final step, the neighbours will be (3 3 3 3 3 1) and (3 3 3 3 3 3). This can be thought as a loop in loop. We chose the whole solution to be tabu. The tabu tenure was set to 5 iterations.

The simulation model and TS algorithm was coded in Borland Delphi 6.0. We applied the methods for both fixed and random shelf lives. The CRN was employed. The indifference zone was set to $200. The confidence level was set to %95. The procedures were applied for 5 different initial solutions each with different random numbers.

4.3.2. The Results

As we stated in the third chapter, there are two cases with regard to the objective function. Following tables show the results of the first case where the profits of the jobs equal to each other and the costs of the machines at different workstations are the same:

Table 4.24. The results of STS method.

STS						
Initial solution	Best Solution	Value ($)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications
4 1 4 2 2 2	7 2 9 6 5 3	16569	806	5	5	1585
6 3 6 4 6 5	7 2 9 6 5 2	16660	748	5	3	955
2 2 2 2 2 2	7 3 9 6 5 3	16552	886	5	7	2220
3 2 4 3 2 1	7 2 8 6 4 3	16851	764	5	4	1270
8 6 8 6 8 5	7 3 9 6 5 4	16447	748	5	3	955

Table 4.25. The results of TS+FSP method.

TS+FSP (indifference zone=\$150, $n_0 = 5$)							
Initial solution	Best Solution	Value (\$)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications	
						n_0	Additional
4 1 4 2 2 2	7 2 9 6 5 3	16552	646	5	5	1815	0
6 3 6 4 6 5	7 2 9 6 5 2	16678	584	5	3	955	0
2 2 2 2 2 2	7 3 9 6 5 3	16531	926	5	7	2215	0
3 2 4 3 2 1	7 2 8 6 4 5	16918	836	5	4	1270	0
8 6 8 6 8 5	7 3 9 6 5 4	16434	784	5	3	955	0

The solutions found by both methods are almost same. The only difference is in the fourth experiment. But the difference is only in the number of forklifts. The numbers of machines are same. When we look at the values they are close to each other as a result. Note that the two procedures found the best solution at the same iterations. The numbers of replications are same for all experiments. Another point is, the FSP did not require to take additional replications. Since we selected the indifference zone parameter as \$150, the FSP could distinguish the best system from the first stage sampling and at screening process inferior solutions were eliminated at once. If we had selected the indifference zone parameter as a smaller amount than the above one then the FSP would have required to take additional replications.

It seems taking five replications is enough to select the best system for this problem instance too. Employing FSP did not help to improve either the solution quality or convergence to the best. The computational times are still favourable to the STS method. Table 4.26. shows the computational times:

Table 4.26. The computational times of the methods.

Initial Solution	STS		TS+FSP		
	Simulation Time	Non-simulation Time	Simulation Time	Non-simulation Time	Screening Time
4 1 4 2 2 2	13079.8	6.89 (0.05%)	16372.6	492.8 (2.92%)	543.5 (3.22%)
6 3 6 4 6 5	11662	6.43 (0.05%)	17396.4	449.6 (2.63%)	467.2 (2.61%)
2 2 2 2 2 2	17010.3	2.82 (0.02%)	19977.5	494.6 (2.41%)	513.6 (2.50%)
3 2 4 3 2 1	13994.8	7.74 (0.05%)	18752.1	437.7 (2.28%)	436.4 (2.27%)
8 6 8 6 8 5	10338.9	2.93 (0.03%)	13358.7	595.8 (4.26%)	606.4 (4.34%)

Simulation time of this problem is much longer than the others. Thus, it has the longest total search time. Strangely, the non-simulation time in the TS+FSP method was incredibly increased compare to STS. The following table shows the long run performances of the solutions found by both methods:

Table 4.27. The performances of the solutions based on 100 replications.

Solution (STS)	Value ($)	Standard Deviation	Solution (TS+FPS)	Value ($)	Standard Deviation
7 2 9 6 5 3	16598	221.23	7 2 9 6 5 3	16598	221.23
7 2 9 6 5 2	16645	215.39	7 2 9 6 5 2	16645	215.39
7 3 9 6 5 3	16348	223.81	7 3 9 6 5 3	16348	223.81
7 2 8 6 4 3	16850	165.37	7 2 8 6 4 5	16837	168.34
7 3 9 6 5 4	16298	225.84	7 3 9 6 5 4	16298	225.84

The solution (7 2 8 6 4 3) can be selected as the best solution since it has a better objective function value with lower standard deviation. Figure 4.8. and Figure 4.9 show the convergences of the methods for different initial solutions:

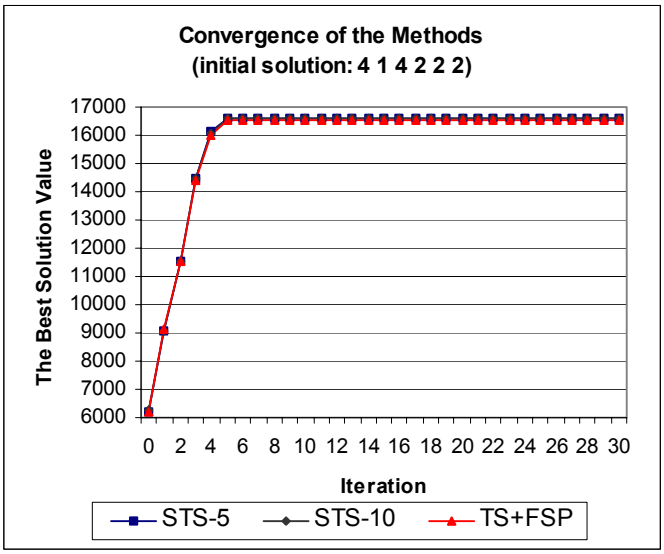


Figure 4.8. The convergences of the methods when the initial solution is (4 1 4 2 2 2).

The methods converge exactly in the same fashion. After three or four iteration all methods seem to reach the best solution. This is an expected result since STS and TS+FSP took same number of replications.

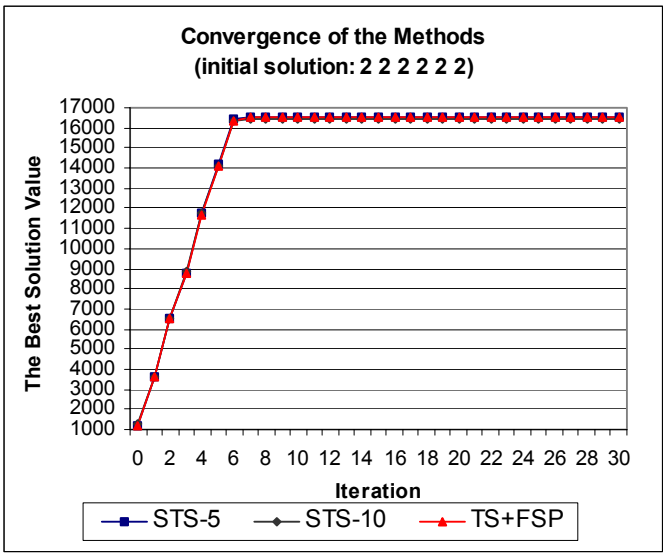


Figure 4.9. The convergences of the methods when the initial solution is (2 2 2 2 2 2).

The figure is not change for this instance either. Only the methods converged a little late according to the previous one. The distance between the initial solution and the best solution caused this situation.

The following tables show the results of the second case where the profits of the finished jobs and the costs of the machines calculated based on the total processing times:

Table 4.28. The results of STS method.

STS						
Initial solution	Best Solution	Value ($)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications
4 1 4 2 2 2	7 2 9 6 5 3	12788	1086	5	5	1585
6 3 6 4 6 5	7 2 9 6 5 2	12848	1036	5	3	955
2 2 2 2 2 2	7 3 9 6 5 3	12862	888	5	7	2215
3 2 4 3 2 1	7 2 8 6 4 3	12991	1120	5	4	1270
8 6 8 6 8 5	7 3 9 6 5 2	12871	1032	5	3	955

Table 4.29. The results of TS+FSP method.

TS+FSP (indifference zone=150$, $n_0 = 5$)							
Initial solution	Best Solution	Value ($)	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications	
						n_0	Additional
4 1 4 2 2 2	7 2 9 6 5 3	12834	806	5	25	12815	0
6 3 6 4 6 5	7 2 9 6 5 2	12860	704	5	3	955	0
2 2 2 2 2 2	7 3 9 6 5 3	12845	776	5	7	2215	0
3 2 4 3 2 1	7 2 8 6 4 3	13036	844	5	4	1370	0
8 6 8 6 8 5	7 3 9 6 5 2	12830	816	5	3	955	0

Both methods found same solutions. Again taking five replications is enough to select the best system. Except from the first experiment, the best solutions were found at the same iterations. The random numbers might have caused this situation. It seems at the early iterations of the search, a solution's expected value was estimated high due to random numbers, and it was selected as the best solution. After that it took a while that the search to find the best solution. We need to look at long run performances of the solutions to distinguish the real best. The following table shows the performances of the solutions based on 100 replications:

Table 4.30. The performances of the solutions based on 100 replications.

Solution (STS)	Value ($)	Standard Deviation	Solution (TS+FPS)	Value ($)	Standard Deviation
7 2 9 6 5 3	12796	166.55	7 2 9 6 5 3	12796	166.55
7 2 9 6 5 2	12843	160.99	7 2 9 6 5 2	12843	160.99
7 3 9 6 5 3	12712	169.90	7 3 9 6 5 3	12712	169.90
7 2 8 6 4 3	12989	134.22	7 2 8 6 4 3	12989	134.22
7 3 9 6 5 2	12764	165.54	7 3 9 6 5 2	12764	165.54

The solution (7 2 8 6 4 3) is proven to be the best since it has the highest performance and the lowest standard deviation.

4.4. Three-stage Buffer Allocation Problem

4.4.1. The Construction

We used similar TS algorithm with the previous problem. The neighbour generation algorithm is also same. Of course, the neighbours that do not satisfy the constraints are excluded from the neighbourhood set. The whole solution is chosen to be tabu as in the previous three problems. The tabu tenure was set to 5 iterations.

The simulation model and TS algorithm was coded in Borland Delphi 6.0. The CRN was employed. The indifference zone was set to 0.5. The confidence level was set to %90. And the first stage sample size n_0 was set to 4. These parameters were taken from Pichitlamken and Nelson (2002). The procedures were applied for 5 different initial solutions each with different random numbers.

4.4.2. The Results

The following tables illustrates the results of both methods:

Table 4.31. The results of STS method.

STS						
Initial solution	Best Solution	Value	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications
2 2 2 2 18	7 6 7 9 11	6.12	632	4	7	1320
4 5 4 3 17	6 7 7 7 13	6.53	486	4	4	740
10 5 5 8 12	6 7 7 11 9	6.23	468	4	5	664
5 5 10 2 18	6 7 7 10 10	6.25	442	4	8	1060
8 7 5 2 18	6 7 7 11 9	5.68	407	4	3	400

Table 4.32. The results of TS+FSP method.

TS+FSP (indifference zone=0.5, $n_0 = 4$)							
Initial solution	Best Solution	Value	Number of visited solutions	Number of Replications	Iteration Number	Total number of replications	
						n_0	Additional
2 2 2 2 18	6 7 7 9 11	6.15	589	10	21	3008	475
4 5 4 3 17	6 7 7 12 8	6.69	518	5	13	1940	1277
10 5 5 8 12	7 6 7 7 13	6.52	474	4	1	132	0
5 5 10 2 18	7 7 6 9 11	6.18	522	9	13	1744	627
8 7 5 2 18	7 6 7 8 12	6.11	416	17	8	1044	365

Although the best solutions seem close to each other, it is not proper to say that the solutions are almost same and there is no significant difference between them. The solutions must be further inspected. As in the previous problems we looked long run performances of the systems by taking 100 replications of each solution. Table 4.27. illustrates the results. When we look at the iteration columns of both table we observe that the STS method found the best solution at the earlier iterations of the search according to the TS+FSP method in most of the experiments. Only in the third experiment the TS+FSP method performed better than the STS. As for the total number of replications STS is far superior to the TS+FSP. The FSP requires additional replications to select the best in most of the experiments except the third one. It seems the FSP did not work as we expected for this problem instance too. Although it provides good solutions, these solutions are not superior to the solutions that are found by taking relatively small number of replications. At this point of view the FSP as a R&S tool is not worth employing in the way we implement it.

Table 4.33. The performances of the solutions based on 100 replications.

Solution (STS)	Value	Standard Deviation	Solution (TS+FPS)	Value	Standard Deviation
7 6 7 9 11	5.76	0.64	6 7 7 9 11	5.81	0.68
6 7 7 7 13	5.74	0.67	6 7 7 12 8	5.83	0.66
6 7 7 11 9	5.84	0.67	7 6 7 7 13	5.70	0.63
6 7 7 10 10	5.84	0.68	7 7 6 9 11	5.91	0.69
6 7 7 11 9	5.81	0.68	7 6 7 8 12	5.74	0.63

The solution (7 7 6 9 11) is the best solution (near optimal) since the objective function value of the solution is better than the others and standard deviations are very close to each other. Table 4.34. shows the computational times in seconds:

Table 4.34. The computational times of the methods.

Initial Solution	STS		TS+FSP		
	Simulation Time	Non-simulation Time	Simulation Time	Non-simulation Time	Screening Time
2 2 2 2 18	81.99	1.56 (1.87%)	84.14	1.57 (1.83%)	10.53 (12.28%)
4 5 4 3 17	65.44	1.57 (2.35%)	93.31	1.72 (1.81%)	27.97 (29.43%)
10 5 5 8 12	68.07	1.47 (2.11%)	79	1.60 (1.99%)	3.47 (4.30%)
5 5 10 2 18	66.18	1.56 (2.30%)	80.5	1.55 (1.89%)	18.11 (22.06%)
8 7 5 2 18	73.40	1.47 (1.96%)	104.89	1.49 (1.40%)	28.44 (26.73%)

We observe that the TS+FSP method requires more time than STS to find the best solution. Since the best solutions are close to each other the method with less computational effort is preferable. The non-simulation times are higher than the first (manufacturing) problem. This is again because of the loopy structure of the neighbourhood generation algorithm. As a final analysis, Figure 4.10. shows the convergences of the methods for the initial solution (2 2 2 2 18):

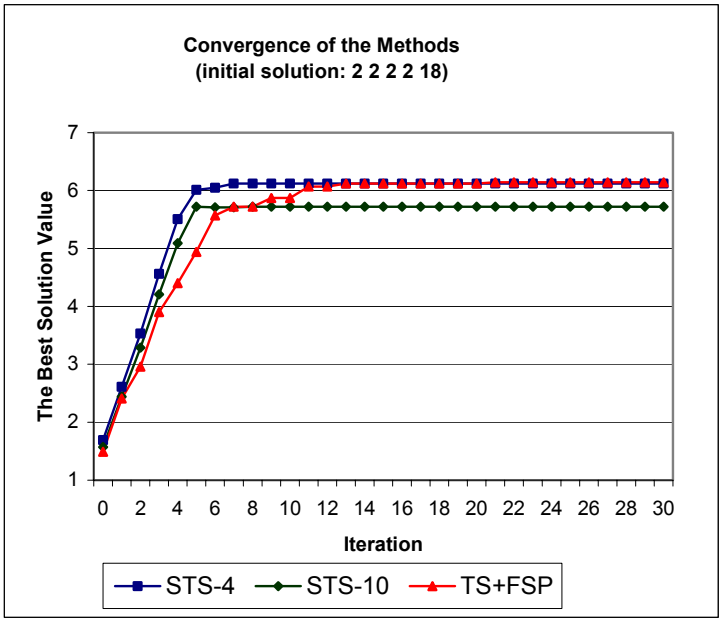


Figure 4.10. The convergences of the methods when the initial solution is (2 2 2 2 18).

We can observe that STS methods converged faster than the TS+FSP. On the other hand STS-4 and TS+FSP overestimated the best solution, which may be caused by random numbers. Both methods took less number of replications compared to STS-10.

CHAPTER 5

CONCLUSIONS

In this study, we combined a simulation optimization methodology with a ranking and selection procedure. Simulation optimization has become a very active research area in recent years with the development of metaheuristics. Since many real life systems could not be represented analytically, classical optimization methods become inefficient. Fortunately, simulation optimization offers a wide variety of solutions to this problem using the power of simulation. On the other hand due to the stochastic nature of the systems, one cannot a hundred percent sure if a solution is optimal. Instead it is said close to optimal.

Tabu Search (TS) is one of the most known and most used metaheuristic in the simulation optimization area. The intensification and diversification strategies made it a very efficient global search algorithm. Another important feature of TS is its making use of memory. The short and long termed memory structures prevent the search from going into cycles and being trapped by local optima.

Ranking and Selection (R&S) methods are used to compare the alternative system designs and to select the best design or a subset of alternatives, which contains the best. They are statistical methods and guarantee the best selection with a given confidence level. At some point of view, R&S procedures can be thought as a simulation optimization methodology especially when the size of the solution space is relatively small. Classical R&S procedures such as Rinott's two-stage procedure are very conservative such that they require a huge number of observations to select

the best alternative. On the other hand, newly developed sequential R&S procedures such as the Fully Sequential Procedure (FSP) are very efficient since they incredibly reduce the computational effort to distinguish the best.

We decided to combine these two methodologies, TS and the FSP to further improve the performance of the search. Our idea is embedding the FSP into TS in the following manner. At each iteration of TS a neighbourhood set is generated and the neighbours are evaluated to find out the best among them by taking an arbitrary number of observations. Instead one can use the FSP to select the best among neighbours. Our motivation is quickly converge to the best solution utilizing the FSP hence increasing the efficiency of the search.

We tested our claim on four different system designs. A manufacturing problem, an inventory control problem, a job shop problem, and a three-stage buffer allocation problem. Five different experiments with different initial solutions and different random numbers were designed and conducted for all of the problem instances. However, the results showed that employing the FSP did not improve either solution quality or efficiency of the search.

In the manufacturing problem, although our methodology found good solutions its computational effort was much more (about two times) than the standard TS. Interestingly, STS, which uses only 5 replications, could end up with very good solutions removing the need for FSP to improve the solution quality. The convergences of the methods were almost same. Thus employing the FSP did not increase the efficiency but add further computational efforts. We thought the simplicity of the problem caused this situation and decided to increase the variability by increasing processing times and adding breakdowns for the machines. In both cases, the results did not change. Taking five replications was still sufficient to come up with good solutions.

In the inventory control problem, the picture was the same, i.e., employing the FSP did not bring much concerning the efficiency of the search. Since we decided to keep the indifference zone parameter relatively large the additional computational effort was incredibly reduced compared to the manufacturing problem. The FSP did not require additional replications apart from the first stage sampling, which was quite enough to select the best solution.

In the job shop problem, both methods found same solutions except for one case. The methods took same number of replications and found the best solution at

the same iteration. In other words, there was no difference between the two methods. Again it was useless to employ the FSP.

In the three stage buffer allocation problem, TS+FSP method found a good solution (probably the best) that the STS method could not find. But the computational effort was too much since the indifference zone parameter was small. Since the solutions found by both method seemed close to each other there was no need to employ the FSP.

It seems our claim is failed for all of the problem instances. Although it is known that taking small number of replications is not enough to compare the alternative system designs and to select the best one, our results indicate the contrary. Even when we increased the variability, 5 replications were enough to select the best system. This might be caused by the simplicity of the test problems. But the more complex real life problems are combinations of these relatively small problem instances. We think, even the complexity of the problem increases, the best solution can be found with less amount of computational effort than the FSP or any other R&S procedure would spend.

Although R&S procedures distinguish the best solution among a set of alternatives, their computational effort is too high. The FSP is one of the most efficient R&S procedures since it can find the best alternative with smaller number of observations compared to conservative R&S methods. This is why we preferred the FSP to other R&S methods to use in our methodology. However, our results showed that it is actually inefficient in the way that taking relatively small number of replications is enough to distinguish the best system. So we think, the presence and the future of the R&S procedures must be reviewed. Their efficiency must be tested on very large and complex simulated systems. And instead of comparing them to other R&S techniques, they must be compared to taking small number of replications.

On the other hand, TS has proven to deserve its reputation. It performed very well and found very good (near optimal) solutions with or without FSP. Using more intelligent neighbourhodd generation algorithms or more intelligent memory structures may further increase its efficiency.

In sum, we understand combining TS and the FSP in the way we did were not a good idea. But, maybe, the FSP can be used after an exhaustive simulation optimization search to further inspect the elite solutions that are encountered by the

search similar to our long run performance analysis. Since a number of replications have already been taken during the search, these observations can be used as a first stage sampling to the FSP thus reducing the computational effort.

REFERENCES

Abspoel, S. J., Etman, L. F. P., Vervoot, J., Roda, J. E., (2000). Simulation Optimization of Stochastic Systems with Integer Variables by Sequential Linearization, *Proceedings of the 2000 Winter Simulation Conference*, J. A. Joines, R. R. Barton, K. Kong, and P. A. Fishwick eds.

Ahmed M.A., Alkhamis, T. M., (2002), Simulation-based optimization using simulated annealing with ranking and selection, *Computers & Operations Research*, 29, 2002.

Angun, E., Kleijnen, J. P. C., Hertog, D. D., Gurkan, G., (2002). Response Surface Methodology Revisited, *Proceedings of the 2002 Winter Simulation Conference*.

Azadivar, P., (1999). Simulation Optimization Methodologies, *Proceedings of the 1999 Winter Simulation Conference*, P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Ewans eds.

Baretto, M. R. P., Chwif, L., Eldabi, T., Paul, R. J., (1999). Simulation Optimization with Linear Move and Exchange Move Optimization Algorithm, *Proceedings of the 1999 Winter Simulation Conference*.

Baesler, F. F., Sepulveda, J. A., (2000). Multi Response Simulation Optimization Using Genetic Search within a Goal Programming Framework, *Proceedings of the 2000 Winter Simulation Conference*.

Brady X., McGarvey, Y., (1998). Heuristic Optimization Using Computer Simulation: A Study of Staffing Levels in a Pharmaceutical Manufacturing Laboratory, *Proceedings of the 1998 Winter Simulation Conference*, D. J. Mederios, E. F. Watson, J. S. Carson, and M. S. Manivannan eds.

Carson, Y., Maria, A., (1997). Simulation Optimization: Methods and Applications, *Proceedings of the 1997 Winter Simulation Conference*, S. Andradottir, K.J. Healy, D.H. Withers, B.L. Nelson eds.

Chen, E. J., Kelton, W. D., (2000). An Enhanced Two-Stage Selection Procedure, *Proceedings of Winter 2000 Simulation Conference*.

Chen, E. J., (2001). Using Common Random Numbers for Indifference-Zone Selection, *Proceedings of the 2001 Winter Simulation Conference*.

Chen, E. J., (2002). Using Common Random Numbers for Indifference-Zone Selection, *Proceedings of the 2002 Winter Simulation Conference*.

Finke D. A., Mederios, D. J., Traband, M. T., (2002). Shop Scheduling Using Tabu Search and Simulation, *Proceedings of the 2002 Winter Simulation Conference*.

Fu, M. C., (2001). Simulation Optimization, *Proceedings of the 2001 Winter Simulation Conference,* B. A. Peters, J. S. Smith, D. J. Modeiros, and M. W. Rohrer eds.

Glover F., (1986). Future paths for integer programming and links to artificial intelligence, *Computers & Operations Research*, vol. 5.
Glover, F., (1989). Tabu Search Part I, *ORSA Journal on Computing*, Vol 1 No 3 Summer 1989.

Glover, F., Laguna M., (2002). *Handbook of Applied Optimization Tabu Search*, P. M. Pardalos and M. G. C. Resende eds, Oxford Academic 2002.

Goldsman, D., (1983). Ranking and Selection in Simulation, *Proceedings of the 1983 Winter Simulation Conference*, Roberts, S., Banks, J., Schmeiser, B., eds.

Goldsman, D., Nelson B. L. (1998). Comparing Systems via Simulation, *Handbook of Simulation,* Edited by Jerry Banks, John Wiley & Sons, Inc.

Goldsman, D., Nelson, B. L., (1998). Statistical Screening, Selection, and Multiple Comparison Procedures in Computer Simulation, *Proceedings of the 1998 Winter Simulation Conference*, Mederios, D.J., Watson, E. F., Carson, J. S., Manivannan, M. S., eds.

Goldsman, D., Nelson, B. L., (1999). A Ranking and Selection Project: Experiences From A University-Industry Collaboration, *Proceedings of the 1999 Winter Simulation Conference*, Farrington, P.A., Nembhard, H.B., Sturrock, D. T., Ewans, G. W., eds.

Goldsman, D., Marshall, W. S., (1999). Selection Procedures With Standardized Time Series Variance Estimators, *Proceedings of the 1999 Winter Simulation Conference*.

Goldsman, D., Marshall, W. S., (2000). Ranking and Selection for Steady-State Simulation, *Proceedings of Winter 2000 Simulation Conference*, Joines, J.A., Barton, R. R., Kang, K., Fishwick, P.A., eds.

Gupta, A. K., Silvakumar, A. I., (2002). Simulation Based Multi Objective Schedule Optimization in Semiconductor Manufacturing, *Proceedings of the 2002 Winter Simulation Conference*.

Haynes, A. M., MacGillivray, H. L.,Mengersen, K.L., (1997). Robustnes of ranking and selection rules using generalised g-and-k distributions, *Journal of Statistical Planning and Inference*, 65 1997.

Hedlund, H. E., Mollaghasemi, M., (2001), A Genetic Algorithm and an Indifference-Zone Ranking and Selection Framework for Simulation Optimization, *Proceedings of the 2001 Winter Simulation Conference*, Peters, B. A., Smith, C. S., Mederios, D. J., Rohrer, M.W., eds.

Humprey, D. G., Wilson, J. R., (1998). A Revised Simplex Search Procedure for Stochastic Simulation Response Surface Optimization, *Proceedings of the 2001 Winter Simulation Conference*.

Inoue K., Chick, S. E., (1998). Comparison of Bayesian and Frequentist Assessments of Uncertainty for Selecting the Best System, *Proceedings of the 1998 Winter Simulation Conference*.

Joines, J. A., Gupta, D., Gokce, M. A., King. R. E., Kay, M. G., (2002). Supply Chain Multi Objective Simulation Optimization, *Proceedings of the 2001 Winter Simulation Conference*.

Kim, S. H., Nelson, B. L., (2001). A Fully Sequential Procedure for Indifference-Zone Selection in Simulation, ACM TOMAS, in press.

Law, A. M., McComas, M. G., (2002). Simulation Optimization, *Proceedings of the 2002 Winter Simulation Conference*.

Lee, Y. H., Park, J. K., Kim, Y. B., (1997). Single Run Optimization Using the Reverse Simulation Method, *Proceedings of the 1997 Winter Simulation Conference*.

Lee, Y. H., Park, K. J., Kim, T. G., (1999). An Approach for Finding Discrete Variable Design Alternatives Using a Simulation Optimization Method, *Proceedings of the 1999 Winter Simulation Conference*.

Matejcik, F. J., Nelson B. L., (1993). Simultaneous Ranking, Selection and Multiple Comparisons for Simulation, *Proceedings of the 1993 Winter Simulation Conference*, Ewans, G.W., Mollaghasemi, M., Russel, E. C., Biles, W. E., eds.

Matejcik, F. J., Nelson B. L., (1995). Two-Stage Multiple Comparisons With The Best For Computer Simulation, *Operations Research*, Vol. 43, No 4, pp. 633-640.

Morito, S., Lee, K. H., (1997). Efficient Simulation Optimization of Dispatching Priority with Fake Processing Times, *Proceedings of the 1997 Winter Simulation Conference*.

Morrice D. J., Butler, J., Mullarkey, P., Gavirneni, S., (1999). Sensitivity Analysis in Ranking and Selection for Multiple Performance Measures, *Proceedings of the 1999 Winter Simulation Conference*.

Neddermeijer, H. G., van Oortmarssen, G. J., Piersma, N., Dekker, R., (2000), A Framework for Response Surface Methodology for Simulation Optimization, *Proceedings of the 2000 Winter Simulation Conference*.

Nelson, B. L., (1993). Robust Multiple Comparisons Under Common Random Numbers, *ACM Transactions on Modeling and Computer Simulation*, Vol. 3, No 3.

Olafsson, S., Shi, L., (1998). Stopping Criterion for a Simulation Based Optimization Method, *Proceedings of the 1998 Winter Simulation Conference*.

Olafsson, S., (1999). Iterative Ranking and Selection for Large Scale Optimization, *Proceedings of the 1999 Winter Simulation Conference*.

Olafsson, S., Shi, L., (1999). Optimization via Adaptive Sampling and Regenerative Simulation, *Proceedings of the 1999 Winter Simulation Conference*.

Olafsson, S., Kim, J., (2002). Simulation Optimization, *Proceedings of the 2002 Winter Simulation Conference*, E. Yucesan, C. –H. Chen, S. L. Snowdon, and J. M. Charnes eds.

Pichitlamken, J., Nelson, B. L., (2002). A Combined Procedure for Optimization via Simulation, *Proceedings of the 2002 Winter Simulation Conference*.

Pichitlamken, J., Nelson, B. L., (2001). Selection-of-the-Best Procedures for Optimization via Simulation, *Proceedings of the 2001 Winter Simulation Conference*.

Rogers, P. (2002). Optimum Seeking Simulation in the Design and Control of Manufacturing Systems: Experience with OptQuest for Arena, *Proceedings of the 2002 Winter Simulation Conference*.

Shannon, R.E., (1975). Systems Simulation the art and science, Prentice-Hall, 1975.

Schuruben, L. W., (1997). Simulation Optimization Using Simultaneous Replications and Event Time Dilation, *Proceedings of the 1997 Winter Simulation Conference*.

Sivakumar, A. I., (1999). Optimization of Cycle Time and Utilization in Semiconductor Test Manufacturing Using Simulation Based, On-Line, Near-Real Time Scheduling System, *Proceedings of the 1999 Winter Simulation Conference*.

Yuksel, B., (2000). An inventory model for randomly perishing goods, The Department of Industrial Engineering, Ankara 2000.