Stochastic optimization in the design of flexible manufacturing systems: Simulated annealing versus stochastic quasigradient

by

Han-chung Wang

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Industrial and Manufacturing Systems Engineering
Major: Industrial Engineering

Iowa State University
Ames, Iowa
1993

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION 1

CHAPTER 2. REVIEW OF RELEVANT LITERATURE 6

An Overview of FMS 6

Monte Carlo Optimization 9

CHAPTER 3. PROBLEM DESCRIPTION AND FORMULATION 12

Hypothetical Flexible Manufacturing System 12

Problem Description 15

Scenario of the Hypothetical FMS 15

Objectives and Methodology 16

Problem Formulation 18

Throughput Optimization Model 19

Cost Optimization Model 19

Optimization Techniques 25

Input Parameters 25

CHAPTER 4. THE CONCEPT OF SIMULATED ANNEALING . 29

Introduction of Simulated Annealing 29

Metropolis Procedure 30

CHAPTER 5. MODIFIED SIMULATED ANNEALING 35
Statistically Measuring System Performance 35
Initial Bias Deletion 35
Confidence Interval Estimate 36
Batch Means Method 36
Comparison of Two System Designs 37
Tests Concerning Differences between Variances 37
Tests Concerning Differences between Means 38
Metropolis Procedure with Reward Process 38
CHAPTER 6. STOCHASTIC QUASIGRADIENT METHOD . . . 46
Introduction 46
The Algorithm 48
Projection 48
Calculation of Step Direction ν 48
Gradient Calculation 50
Normalization of Gradients 50
Choice of Step Size ρ 50
Stopping Criteria 52
Additional Features 52
CHAPTER 7. IMPLEMENTATION 55
Determination of Transient Period 56
Selection of an Appropriate Batch Size 58
Simulation Model 59
Implementation of Modified Simulated Annealing 61

Implementation of Stochastic Quasigradient Algorithm 63

CHAPTER 8. RESULTS AND ANALYSES 65

Summary of Experiment Results 66

Throughput Optimization with Simulated Annealing 67

Throughput Optimization with SQG 68

Cost Optimization with Simulated Annealing 68

Cost Optimization with SQG 76

Maximize System Throughput versus Minimize Total Cost 84

Simulated Annealing versus Stochastic Quasigradient 85

Summary and Conclusions 87

CHAPTER 9. CONCLUSIONS 89

BIBLIOGRAPHY 94

APPENDIX Modified Simulated Annealing 98

LIST OF TABLES

Table 3.1: Hypothetical FMS components 14

Table 3.2: Interarrival rate and batch size for each part type 17

Table 3.3: Distance between each component 17

Table 3.4: Mean of failure time and repair time for each machine . . . 18

Table 3.5: The numerical data of the cost parameters 28

Table 7.1: A list of design parameters and corresponding variables . . . 55

Table 8.1: Summary of the results – throughput optimization with SA . 69

Table 8.2: Summary of results – throughput optimization with SQG . . 70

Table 8.3: Summary of results – cost optimization with SA 77

Table 8.4: Summary of results – cost optimization with SQG 78

Table 8.5: A comparison of simulated annealing and SQG 86

LIST OF FIGURES

Figure 1.1: Interacting process between optimization algorithms and a simulation model 5

Figure 2.1: The content of FMS design 8

Figure 2.2: Classification of decision aids 10

Figure 3.1: Hypothetical flexible manufacturing system 13

Figure 3.2: The operations sequence of each product 15

Figure 3.3: Schematic of the proposed cost model 25

Figure 4.1: Simulated Annealing Algorithm 33

Figure 5.1: Unsavory situation if always accepting S_1 40

Figure 5.2: Unsavory situation if always accepting S_2 40

Figure 5.3: Policy made if S_3 is better than S_2 42

Figure 5.4: Policy made if S_3 is worse than S_2 42

Figure 5.5: System improving if S_3 is better than S_1 43

Figure 5.6: System worsening if S_3 is worse than S_1 43

Figure 5.7: The framework of modified simulated annealing 44

Figure 6.1: Gradient method 47

Figure 6.2: Stochastic quasigradient algorithm 54

Figure 7.1: Time in system responses versus simulated time 57

Figure 7.2: Smoothing the responses by using moving average with lag 15 57

Figure 7.3: Correlogram for time in system observations 60

Figure 7.4: Determining an appropriate batch size by using **FILTER** command 60

Figure 7.5: Linkage among procedures of simulated annealing 61

Figure 7.6: Linkage among procedures of SQG 64

Figure 8.1: Throughput optimization with SA ($t = 1.0$) 71

Figure 8.2: Throughput optimization with SA ($t = 1.5$) 71

Figure 8.3: Throughput optimization with SA ($t = 3.0$) 72

Figure 8.4: Throughput optimization with SA (*sample 1*) 72

Figure 8.5: Throughput optimization with SA (*sample 2*) 73

Figure 8.6: Throughput optimization with SQG ($s = 1.0$) 73

Figure 8.7: Throughput optimization with SQG ($s = 2.0$) 74

Figure 8.8: Throughput optimization with SQG ($s = 3.0$) 74

Figure 8.9: Throughput optimization with SQG (*sample 1*) 75

Figure 8.10: Throughput optimization with SQG (*sample 2*) 75

Figure 8.11: Cost optimization with SA ($t = 3.0$) 79

Figure 8.12: Cost optimization with SA ($t = 4.0$) 79

Figure 8.13: Cost optimization with SA ($t = 5.0$) 80

Figure 8.14: Cost optimization with SA (*sample 1*) 80

Figure 8.15: Cost optimization with SA (*sample 2*) 81

Figure 8.16: Cost optimization with SQG ($s = 1.0$) 81

Figure 8.17: Cost optimization with SQG ($s = 2.0$) 82

Figure 8.18: Cost optimization with SQG ($s = 3.0$) 82

Figure 8.19: Cost optimization with SQG (*sample 1*) 83

Figure 8.20: Cost optimization with SQG (*sample 2*) 83

CHAPTER 1. INTRODUCTION

Severe challenges in the market such as shorter product life cycles, shorter response time to market changes, increased variety and number of new products introduced, and the high quality required, have forced manufacturers to look critically at the option of using current emerging technologies to secure a competitive edge. One of these technologies is the concept of *flexible manufacturing systems* (FMS).

The first FMS installations began to appear in the US about 1967 [RANK 83]. The development and adoption of this concept have grown rapidly. FMS capabilities have shown promise of substantial economic advantages, e.g., reduced inventory, more rapid delivery of products, better quality control and less need for human intervention in operations. As a result, in the 1980s FMS became a national trend and several highly developed FMSs were installed. This is well-known as the first generation of manufacturing system evolution.

Somewhat similar and yet different from conventional manufacturing systems, an FMS is much more complex due to its dynamic behavior and the interrelations between its various components. It is clear that traditional techniques of designing and analyzing conventional manufacturing systems are not applicable [WHIT 85]. The design analysis and the use of flexible manufacturing systems involve some intricate operation research problems such as general FMS design problems, FMS scheduling

problems and FMS control problems [STEC 85]. In this study, our concern is the *FMS design problem*, i.e., determining the appropriate number of machine tools of each type in the system. Each of these must be accomplished as part of the design task. Since they all interact in complex ways, any allocation of resources for one manufacturing system will necessarily affect its system performance.

To develop an appropriate FMS design solution, we need a tool which permits the study of the dynamic behavior of an FMS. There exist several research papers on modeling and evaluating an FMS quantitatively. In general, they can be classified into two categories – analytical (e.g., mathematical programming) and non-analytical (e.g., simulation) models. Instead of using the mathematical calculation, a computer simulation model is utilized as a design tool. Simulation models enable one to describe an FMS design and to predict its performance. In addition, simulation is able to calculate the same measures of system performance for hypothetical system configurations as are used to judge real systems. Thus, FMS designers can try out various FMS configurations without actually building or controlling a system.

Because the FMS design problems involve the complex and subtle relationships among many interdependent design variables as well as the randomness from the interarrival rate of the workparts, operation time, machine failure and repair time, the FMS design problem can be regarded as a discrete type stochastic combinatorial optimization problem. A combination of those design variables is said to be a *system configuration.* For example, x_1 and x_2 are the only two decision variables with respect to the possible number of machine centers and the buffer size, then, given a value of each variable, (x_1, x_2) is said to be a system configuration, i.e., the system is configurated by x_1 machines and x_2 buffer size. All possible system configurations

comprise a solution space S. The solution space grows quickly as the number of design variables increases. One could imagine taking a very cautious approach such an exhaustive search, exploring carefully each configuration and making no firm decisions until the whole solution space has been explored. But such an approach would consume more time and money than is ordinarily available for the preliminary design. This leads to our interest in employing a search method for finding the solutions of the optimization problem. The system is simulated and the performance of the design is estimated. This performance estimate is then used by the search method to derive a new design configuration. The procedure continues iteratively until the search stops. This iterative process is called *simulation-optimization* or *Monte Carlo optimization* if the problem is stochastic (see Figure 1.1). The simulated annealing and stochastic quasigradient methods which have recently attracted much attention are adopted as the search methods to be investigated.

Simulated annealing is a randomized algorithm which searches for a globally optimal solution of deterministic combinatorial optimization problems. The name comes from an analogous procedure in statistical mechanics where the experiments are performed by careful annealing, i.e., first melting the material and then lowering the temperature slowly to obtain the desired crystalline structures. Simulated annealing has seen a number of applications in large combinatorial optimization problems. However, in most applications of simulated annealing the objective function is assumed to be deterministic. In order to apply simulated annealing to the FMS design problem, certain modifications have to be made in order to compensate for the stochastic nature of the problem.

Stochastic quasigradient methods (SQG) provide an alternative for solving those

functions where neither the functions or their gradients can be calculated analytically. Since direct calculation of the gradients is not possible, the alternative approach is to use statistical estimates of the gradients, then apply a monotonically constrained procedure to determine the step direction to drive the values of the decision variables to the optimal solution. A more detailed discussion about SQG is included in Chapter 6. In the case of FMS design, a few modifications on SQG have been made.

The primary effort of this research is to investigate and compare the applicability of simulated annealing and stochastic quasigradient methods to the improvement of the design of FMS. A hypothetical flexible manufacturing system is used as a testbed. We elected the *throughput* and the *total cost* of the system as two objective functions. Objective function estimates are furnished through a computer simulation model. Also due to the stochastic nature of the FMS, several statistical analysis procedures for the simulation output are included. In the study, we demonstrate how simulated annealing and SQG perform on the testbed and we summarize their performance in terms of *time complexity* and *quality of the solutions.*

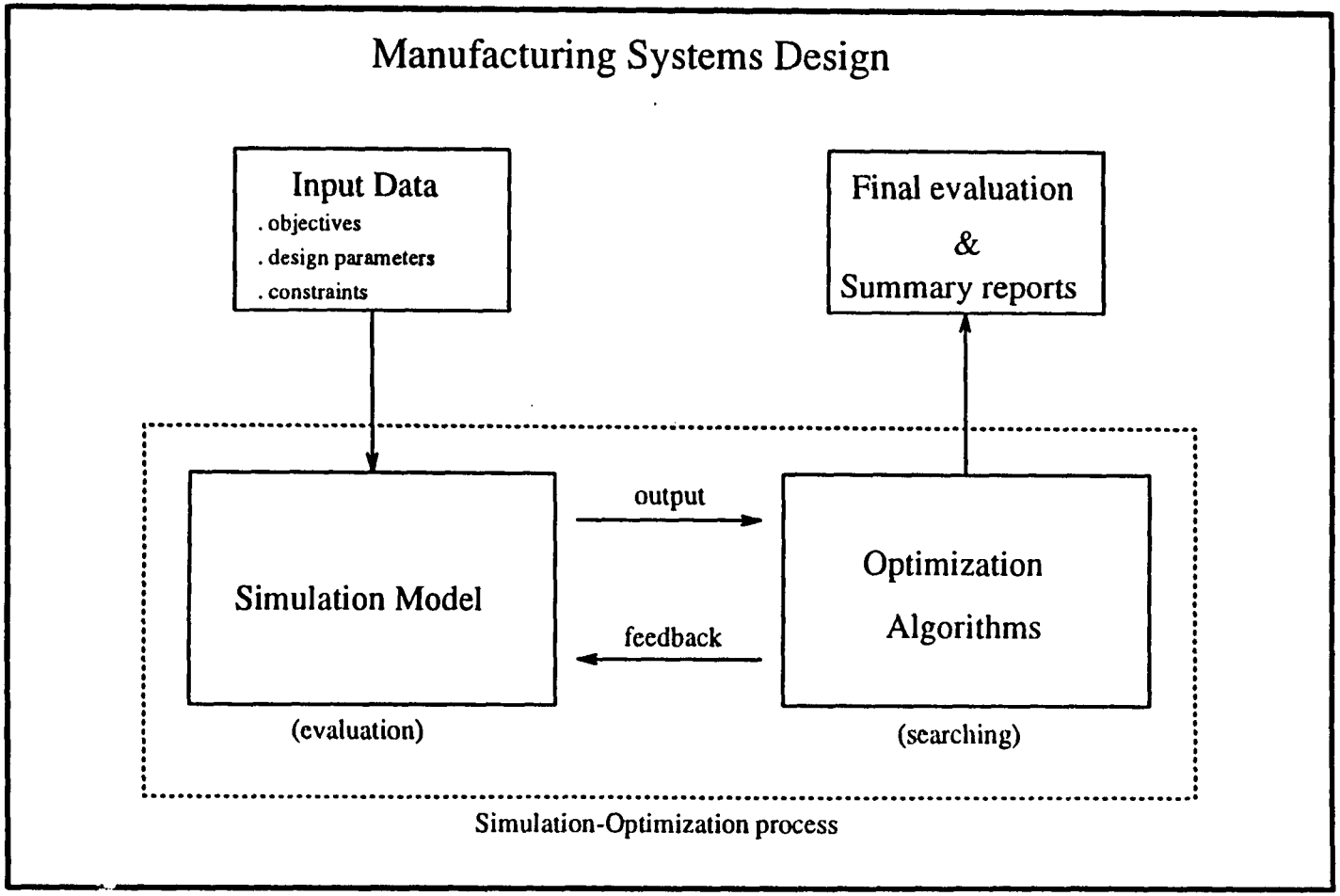

Figure 1.1: Interacting process between optimization algorithms and simulation model

CHAPTER 2. REVIEW OF RELEVANT LITERATURE

An Overview of FMS

To cope with the increasing varieties of products and the production changes caused by product-mix changes and introduction of new products, flexibility and productivity will be required as important factors in manufacturing. The Flexible Manufacturing System (FMS) is a system which can meet the requirements for both flexibility and productivity [HUTC 79]. The FMS concept was originally developed within the context of machining pieces of parts. The details on development of the basic concept of a flexible manufacturing system can be found in [RANK 83]. Basically, *an FMS consists of computer numerically controlled machines where production operations are performed, linked by a material handling system, operating as an integrated system under the direction and control of a central computer.*

Unlike conventional manufacturing systems, the problems associated with flexible manufacturing systems are far more difficult since it is more integrated and complex. These problems can be classified into four categories – *planning*, *scheduling*, *control*, and *design* [STEC 85]. Numerous studies have been conducted on those categories [KUSI 86]. As discussed in [KUSI 86], any decision made at the design stage has critical impact on the system management and operations since the design which finally emerges will impose constraints on how the FMS can or should be

controlled.

The design of an FMS begins with a survey of the manufacturing requirements of the products with a view to identifying the range of parts which should be produced on the FMS [KLAH 81]. Then the basic design concept must be established. There are two basic elements of FMS design – *product design* and *system design* (Figure 2.1) [KUSI 86]. In designing an FMS, there is a partial ordering to some of the decisions that have to be made, i.e., some decisions must precede others in time. [STEC 86] partitions these decisions into *initial specification decisions* and *subsequent implementation decisions.* [BUZA 86] also proposed a general process for the FMS design problem. They categorized the FMS design process into two hierarchical stages: establishing the *basic design* and the *detailed design.* At the basic design stage the function, capability and number of each type of work station, the type of material handling system and the type of storage should be determined. At the detailed design stage it will be necessary to determine such aspects as the required accuracy of machines, tool changing systems, etc. Then the number of transporting devices, the number of pallets, and the capacity of central and local storage must be determined.

In order to analyze and evaluate the behavior and performance of an FMS, models and performance measures for describing FMSs are needed. Moreover, models are useful to give the designer insight into the technical issues described above. Measures of performance such as utilization, system throughput, work in process, flow time, average queue length, return on investment and net present value are typically used as criteria for evaluating an FMS [SURI 85].

It is not likely that a single model can be used to fully describe its behavior since an FMS is a complex system, consisting of many interconnected components of

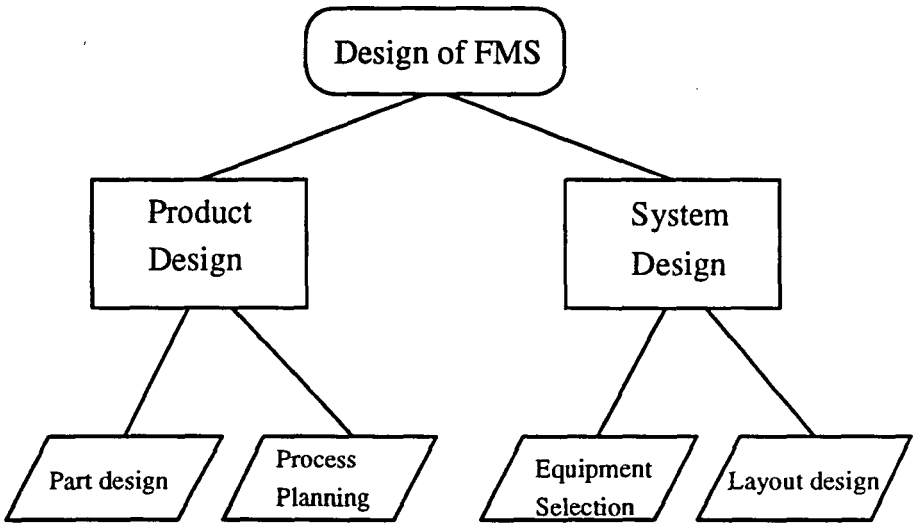

Figure 2.1: The content of FMS design

software and hardware. In general, those models for evaluating the performance of an FMS could be divided into two sets: *analytical* and *non-analytical models.* Analytical models always address a small subset of design issues. Thus while applying analytical models for an FMS design, we have to decide how to structure the problem first. That means they require simplifying assumptions in order to solve the problems. Many studies on the development of analytical models of FMS have used *queueing network* theories for performance evaluation and mathematical programming or control theory for deriving scheduling and operating procedures. [BUZA 86] provided a sufficient review of the recent work on the development of analytical models.

Beyond a certain level of detail, the mathematics of the analytical approaches becomes intractable. Consequently, non-analytical models such as *simulation* and *artificial intelligence* (AI) become the necessary tools. AI methods are based on expert simulation systems [SHAN 84]. Several applications of AI on manufacturing have

been addressed and published. Perhaps the most welcome technique for evaluating system design alternatives is computer simulation [CHEN 85]. *By simulating the dynamic behavior of a real system with a model, and analyzing data from the model, the characteristics of the real system can be identified. Moreover, more detailed and accurate estimates of the performance of an FMS can be obtained from simulation.* However, it takes more effort to develop a simulation model than utilizing an analytic model, and requires more analyses about the system. A number of high level simulation languages have been developed in recent years (e.g., SIMAN, SLAM and GPSS) to ease the task of developing the simulation model.

Monte Carlo Optimization

From the practical user's point of view, [SURI 85] specifically differentiates between *generative* (or prescriptive) models and *evaluative* (or descriptive) models as shown in Figure 2.2. Evaluative models measure the performance of a system design given a specific set of values for the decision variables. They can provide more insights about the system performance rather than decision variables. However, it may take a long time to find good values for the decision variables. The main evaluative models can be divided into five classes: (1) *static allocation models*, (2) *queueing network models*, (3) *simulation models*, (4) *perturbation models*, and (5) *Petri nets.* A detailed review of evaluative models can be found in the article by [SURI 85].

In contrast, generative models are used to find good candidate decisions with respect to the objective functions by applying optimization techniques (e.g., LP/NLP), i.e., using the simulation to measure the performance of the system and integrating the optimization techniques, the generative models can be used to search for an

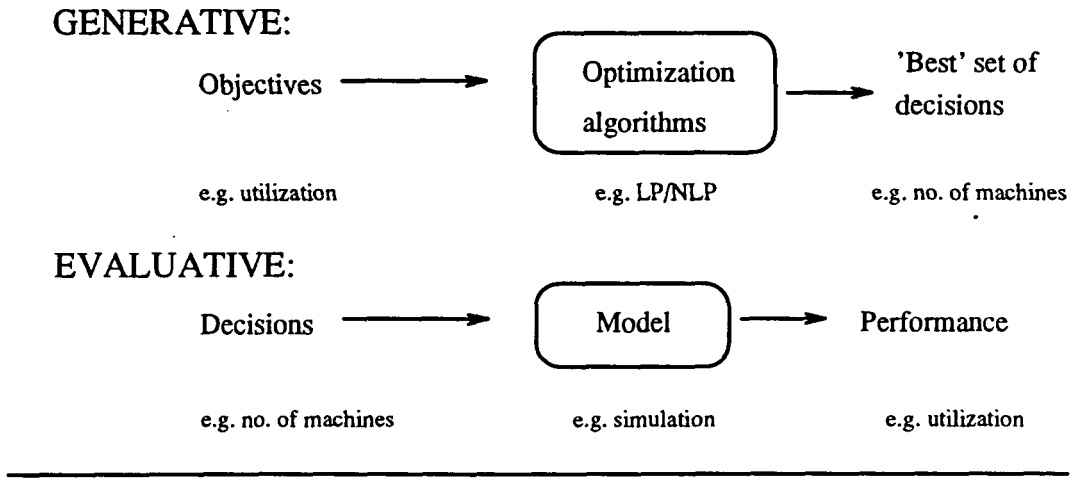

Figure 2.2: Classification of decision aids

optimal design. Such a process is known as *Monte Carlo optimization.*

Monte Carlo optimization or stochastic optimization is an optimization method which involves the use of computer simulation. It is concerned with general optimization problems under uncertainty. Several optimization techniques such as stochastic approximation method, simulated annealing, response surface methodology, and genetic algorithms have been studied and applied. A general survey of optimization methods for stochastic optimization can be found in papers by [GLYN 86b] and [MEKE 87].

During recent years, applying Monte Carlo optimization to manufacturing systems design has gained a lot of attention. [SURI 87] developed a single run optimization approach to maximize the throughput of closed loop asynchronous flexible assembly systems. [HO 79] proposed a gradient method to optimize the buffer size of production lines. The use of stochastic quasigradient methods (SQG) to maxi-

mize the production rate of asynchronous flexible assembly systems and to solve the portfolio problem has been conducted by [LIU 88] and [GEMM 88]. [ERMO 83] has surveyed several variants of the SQG algorithm on many different problem types. He formulated problems into into four groups: general stochastic programming problems, resource problems, stochastic minimax problems and nonlinear programming problems and has given the specific results.

[BULG 88] proposed a modified simulated annealing algorithm with simulation to optimize the buffer size in automatic assembly systems. [LAAR 92] used simulated annealing to solve the job shop scheduling problem. Many papers about the applications of simulated annealing can be found such as the routing and location problems [GOLD 86], part ordering/release problem in an FMS [LEE 91], quadratic assignment problems [WILH 87], etc. They showed the performance of simulated annealing tends to be good if the parameters were "well-controlled".

CHAPTER 3. PROBLEM DESCRIPTION AND FORMULATION

Hypothetical Flexible Manufacturing System

The selected flexible manufacturing system for implementing Monte Carlo optimization is shown in Figure 3.1, where the physical layout of the system is depicted. This model is derived from a case presented in [FUCH 88] and has been modified to fit our own needs.

The components of the system are described as shown in Table 3.1. All manual work is performed in the setup area in which workpieces are mounted onto fixtures, reoriented as necessary during the machine cycle, and removed when completed. NC deburring machines and coordinate measuring machines are also installed for deburring operations and inspections.

An additional process is needed to give pre-work for the corresponding main operation as the workparts cycle through the system (e.g., it is necessary to go through process #1 before entering setup area). A predetermined sequence of manufacturing processes is assigned to each workpiece as well as AGV for transportation.

Workparts enter the system in batches and are transported in the system according to the following description. Parts enter the process area being processed first and await the availability of the required FMS components (setup area, machining center, inspection station, deburring station) specified on the predetermined manufacturing

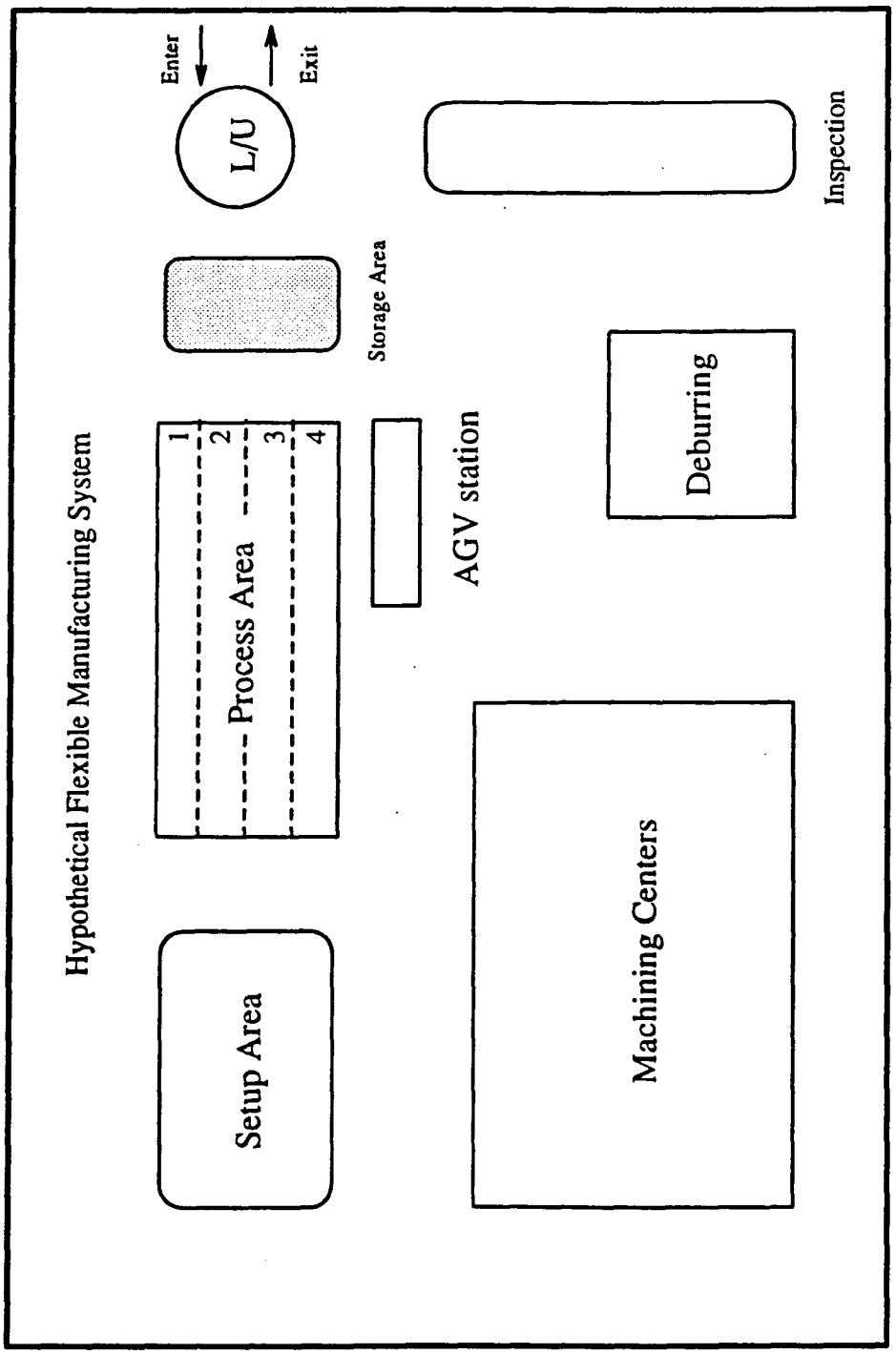

Figure 3.1: Hypothetical flexible manufacturing system

Table 3.1: Hypothetical FMS components

Component #	Corresponding description
1	Loading/Unloading area
2	Process #1
3	Process #2
4	Process #3
5	Process #4
6	Setup station
7	Machining station
8	Deburring station
9	Inspection station
10	AGV maintenance area

sequence. The first main operation for all parts requires a move to the manual setup. For this activity the parts claim an AGV and move to the setup area where the parts are put into a fixture on a pallet. At the completion of the setup operation, the parts move back to the process area and the next main operation in the predetermined sequence is indexed to the parts, usually the machining operation.

The parts are processed and stay in the process area until a machining center is available, and at this moment the parts claim an AGV and are transported to the free machining center. They remain at the machining center for the duration of operation time. After the machining operation, the parts again claim an AGV and return to the process area. At the same time, the next required operation is issued to the parts. The parts circulate in this way until all operations specified in a predetermined sequence are accomplished. At the completion of all operations the parts exit the system.

	Operations Sequence
Part A	1 - 7 - 6 - 7 - 8 - 9 - 1
Part B	1 - 7 - 8 - 9 - 1
Part C	1 - 7 - 8 - 6 - 7 - 8 - 9 - 1

Figure 3.2: The operations sequence of each product

Problem Description

Scenario of the Hypothetical FMS

Assume there are three different kinds of parts yielded from this system, namely **A**, **B** and **C**. Each of the parts enters the system based on the *Poisson process* with different interarrival rates and lot sizes (see Table 3.2).

After entering the system, the workparts go through the system with an assigned sequence of operations (see Figure 3.2, the number represents the corresonding component.) and require different amounts of processing time at each operation. The processing time at the setup station is said to be *uniformly* distributed over a given range since it is a manual operation, and the operation time at the process station has a *gamma* distribution; the others are said to be *deterministic* due to the high level of automation in the FMS.

Workparts are transported through the system via AGVs. The AGV is regarded as the major device of the material handling system. Three AGV units are deployed

and they travel at a constant speed of *300* distance units per time unit. *Shortest Path First* (SPF) criteria is provided when an AGV is signaled to pick up two or more parts at the same time. The distance between each component is given in Table 3.3.

Machine failures are also considered in our study. It is assumed that the time between machine failures and the time for repairing are *exponentially* distributed. Table 3.4 gives the time between machine failures and the time for repairing.

Objectives and Methodology

Given the hypothetical flexible manufacturing system and operating information described above, a design problem arises in determining the appropriate resource capacity such as:

- *Capacity for process #1*,
- *Capacity for process #2*,
- *Capacity for process #3*,
- *Capacity for process #4*,
- *The number of workers allocated at the setup area*,
- *The number of machines allocated at the machining station*,
- *The number of robots allocated at the deburring station*, and
- *The number of tools allocated at the inspection station*,

Table 3.2: Interarrival rate and batch size for each part type

Part type	Interarrival rate	Batch size
A	6	6
B	4	4
C	8	3

Table 3.3: Distance between each component

Component #	1	2	3	4	5	6	7	8	9	10
1		25	25	25	25	200	300	325	150	35
2			0[a]	0	0	150	250	300	150	30
3				0	0	150	250	300	150	30
4					0	150	250	300	150	30
5						150	250	300	150	30
6							200	150	310	160
7								250	220	220
8									470	330
9										100

[a]Indicating the distance is ignored.

Table 3.4: Mean of failure time and repair time for each machine

Equipment	mean of failure time	mean of repair time
Machining center	250	41.0
Deburring robot	303	52.0
Measuring Machine	406	34.0

which will result in optimization of system performance. Throughput and total cost of the system are modeled as performance criteria and simulation-optimization methodology is used to solve the problem. The application of modified simulated annealing and SQG methods are investigated.

Problem Formulation

This problem is formulated as follows:

$\max F(\mathbf{X})$ or $\min F(\mathbf{X})$

s.t. $\mathbf{X} \in \mathbf{C}$

where $\mathbf{X}$ is a vector $(x_1, x_2, ..., x_n)$ representing a *system configuration*, x_i represents the size of component i, $i = 1, ..., n$, and $F(\mathbf{X})$ is a performance criteria or objective function for a given $\mathbf{X}$, which is obtained through simulation. $\mathbf{C}$ represents a feasible set of decisions for components, and it could be the lower and upper bounds of the capacity of each component.

Throughput Optimization Model

Several types of performance measures for evaluating FMS alternatives have been presented and utilized; for example, utilization, work in process, flow time, average queue length, net present value, system throughput, and so on. The most commonly used performance measure has been *throughput* or *production rate.* There exist many studies related to the maximization of throughput while evaluating FMS design alternatives. In our study, we also utilize the system throughput as a performance measure and it can be defined as the number of parts produced per unit time, and formulated in the following.

Production Rate (units/hour) = (1 / *estimated cycle time in minute*) $\times$ 60

The cycle time is defined as the interval of time between two finished parts. The use of this measure may result in an arbitrarily large FMS design in order to achieve the highest system throughput. In other words, the optimal design solutions will tend to increase the resource capacities infinitely. However, constraints on design variables such as capital or plant size is always considered in the FMS design phase. Therefore, based upon certain constraints, system throughput can also be used to measure the performance of a particular FMS design. Instead of maximizing system throughput, economic analysis will be introduced to avoid the situation described above.

Cost Optimization Model

FMS encourages technological change and innovation. However, such a system requires extensive capital investment which is a discouraging factor in accepting FMS. Therefore, in order to make proper decisions, the user needs to carefully analyze the

cost and benefit of FMS while adopting it.

Two categories of cost elements can be identified, *tangible* and *intangible* cost elements . Tangible costs are the ones that easily or with varied degrees of difficulty are quantifiable. Examples include the costs such as equipment, labor, and inventory. Intangible costs are difficult to quantify. These are the costs that are neglected by traditional cost-accounting methods and decision makers who are dedicated to using these traditional methods. Examples include reduction in lead time, market flexibility, and increased learning.

Since intangible costs are difficult to quantify and building a very accurate FMS cost model is not the main purpose in our study, a cost model is built which only includes tangible costs. This can be utilized without sacrificing the quality of the investigation of our proposed approaches.

Assuming the basic configuration of the FMS is given, the cost elements considered in our model consist of equipment costs, floor space cost, labor cost, repair cost, estimated balking cost, estimated work-in-process inventory holding cost, estimated cost of not achieving the desired utilization rate and estimated cost of not meeting the production goal. The first four cost elements are one-time costs. The last four cost elements can be referred as *operational costs*, which depends primarily on the nature of interactions between system components. It should be noted that operational costs are incurred in every time period. Therefore, their impact over the horizon of the analysis may be significant even though the operational cost in a specific time period is not large.

It is necessary for different cost elements to have a common unit of measure. For our model, we will use *estimated total cost* as a unit of measure. The equation of the

total cost appears as follows and appropriate systems can then be chosen based on minimal total cost at the required production volumes.

Total Cost = Equipment cost + Labor cost
+ Floor space cost + Repair cost
+ Estimated work-in-process inventory holding cost
+ Estimated balking cost
+ Estimated cost of not achieving the desired utilization rate
+ Estimated cost of not meeting production goal

Equipment Cost All manufacturing systems require a capital investment of hardware of some magnitude. It includes initial cost and salvage value of equipment, spare parts, maintenance, installation, etc. For cost accounting purposes, this expense is usually created in a formal manner, often called annualized cost.

With a certain rate of return and equipment cost, the annualized cost can be formed as,

$\mathbf{AEC} = (\textit{equipment cost}) \times \mathbf{A/P}$

A/P: the annualized cost factor

The equation above defines the annualized cost of each equipment cost. For our cost calculation purpose, it is desirable to scale annualized cost in terms of hours. When dividing such cost by the annual hours, the equipment cost becomes:

$$e = \frac{\mathbf{AEC}}{\textit{Annual-hours}}$$

The total equipment cost depends on the number of components in the system. Therefore, it can be written as follows.

$\mathbf{EQUIPMENT} = \sum_{i=1}^{n} e_i$

e_i: equipment cost of component i

n: the number of components

Labor Cost This is direct labor and overhead cost of operating and maintaining the equipment. In our model, this can be stated as,

$$\textbf{LABOR} = l \times r$$

l: the number of workers needed
r: labor hour rate

For example, if there are three workers assigned to the setup area and the labor hour rate is 10($/hour), then totally 30($/hour) of the labor cost is incurred.

Floor Space Cost It depends upon the total area of floor space needed in the system. The larger the system is, the more floor space is needed. The total needed area can be estimated by the number of resources, buffer size, system layout, and so on. It is assumed that system layout is given and the floor space needed is determined only by the capacity of the resource and buffer size. Therefore, the total floor space cost is formulated as,

$$\textbf{FLOOR} = (\sum_{i=1}^{n} s_i N_i + B \times b) \times \lambda$$

s_i: floor space factor for component i (*area/capacity*)
b: floor space factor for buffers(*area/buffer units*)
B: Buffer size
N_i: capacity of component i
n: the number of components
λ: average floor space cost factor($/area)

For example, suppose there is only one component (n=1) in the system (e.g., machining station) and ten buffers (B=10) is allocated, given three machines (N_1=3), s_1=1.5 (i.e., each machine occupies 1.5 area units of the floor space), b=1 (i.e., each

buffer has 1 area unit of the floor space) and λ=5(\$/area), then totally a \$72.50 floor space cost occurs.

Repair Cost Repair cost occurs when equipment breaks down and needs to be fixed. The average frequency of repair during a certain time period is estimated and the average repair cost of each component is given. The total repair cost can be formulated as follows.

$\mathbf{REPAIR} = \sum_{i=1}^{n} k_i \times f_i$

k_i: average frequency of repair for component i
f_i: average repair cost per occurrence for component i

Estimated Balking Cost Since the capacity of the storage area is limited, incoming orders will balk when the storage area is full (see Figure 3.1). It can be regarded as the cost of losing business. To determine such costs, the average number of orders balking and the associated cost should be known. This is written as follows.

$\mathbf{BALK} = (\textit{average number balking}) \times (\textit{balking cost})$

Work-in-process Inventory Holding Cost Value is added to a product as it goes through each operation. It is actually the opportunity cost of money being tied up in the inventory because that money is not invested elsewhere. To determine the total in process inventory holding cost, the estimated amount of work-in-process inventory, the time length in the system and the holding cost factor should be known. The total in-process inventory holding cost can be computed as follows.

$\mathbf{WIP} = \sum_{i=1}^{m} h_i \times w_i \times t_i$

h_i: holding cost factor for part i

w_i: estimated in-process inventory of part i
t_i: estimated time in the system of part i
m: the number of types of parts

Estimated Penalty Cost of not Achieving the Desired Utilization Rate Efficiency of each component is considered from a managerial point of view. Simulation can give good estimates of the utilization for each component in the system. If the estimated utilization rate of a component is less than required, the penalty cost of under utilization is placed in the total cost. The estimated penalty cost of not achieving the desired utilization rate is formulated as in the following expression.

$$\textbf{UNDER UTILIZATION} = \sum_{i=1}^{n} (U_i - Ue_i) \times \mu_i$$

Ue_i: estimated utilization rate of component i
U_i: required utilization rate of component i
μ_i: penalty cost rate of component i if under utilized

If the estimated utilization rate is greater than the desired rate, the penalty cost is zero.

Estimated Cost of not Meeting the Production Goal Since the model is to minimize total cost of the system during a certain time period with a desired production volume rather than to maximize the production rate only, a penalty cost for not meeting the production goal is included in the cost model. The formulation is shown below

$$\textbf{UNDER PRODUCTION} = (\textit{required throughput - estimated throughput}) \times (\textit{penalty cost of underproduction})$$

It is assumed that if the estimated throughput is greater than the goal, the company will only produce as much as needed and no penalty cost is applied.

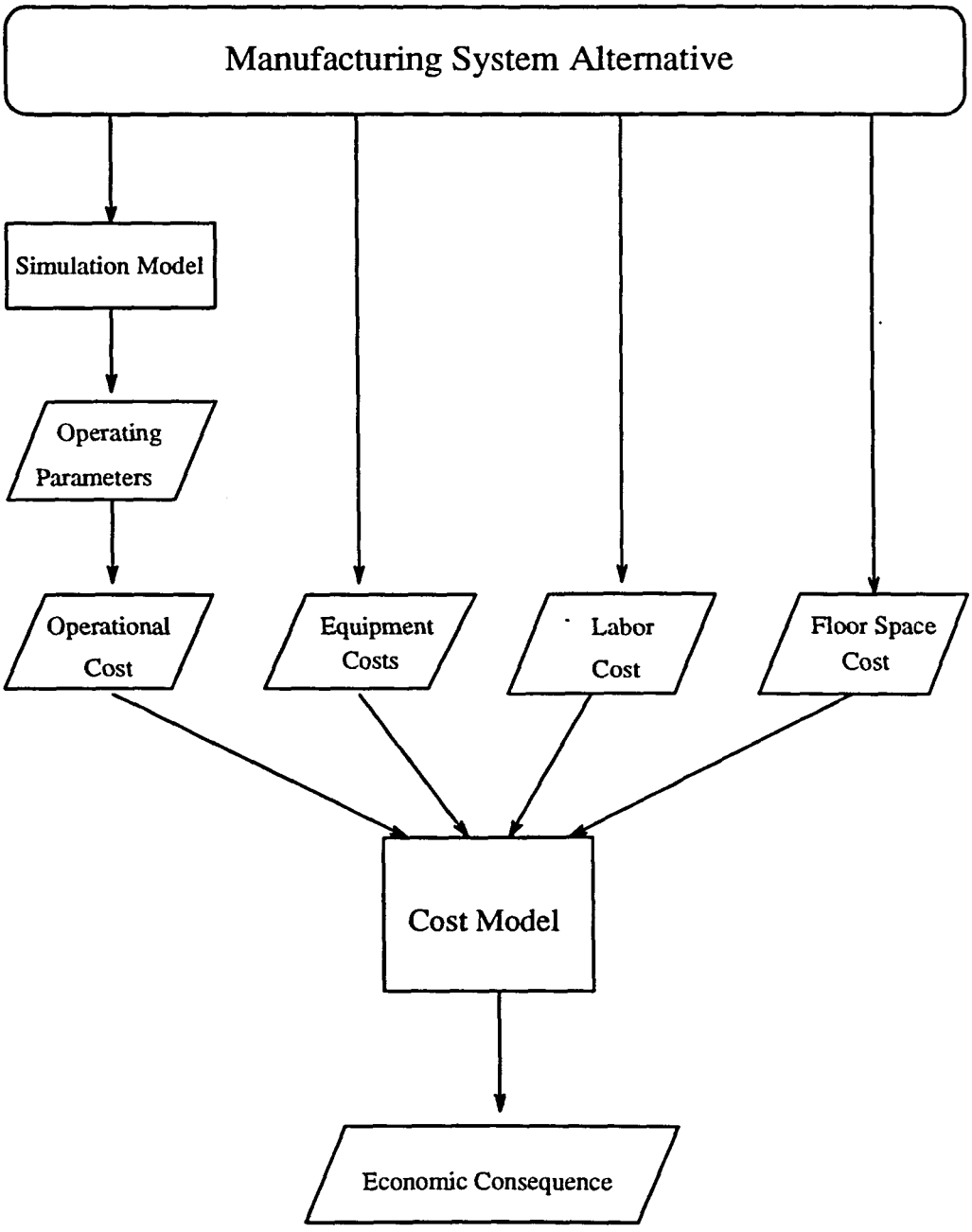

Figure 3.3: Schematic of the proposed cost model

Optimization Techniques

Due to the dynamic behavior of an FMS, we use a simulation model to measure the system performance. The combination of various FMS components is said to be a system configuration. Practically, the possible system configurations of an FMS are finite. Suppose the only decision variable is N_m, the possible number of machine centers. Then there is N_m possible system configurations. Suppose another variable is included, N_q, which denotes the possible number of deburring robots. The size of system configuration space now grows to $N_m \times N_q$. In general, an FMS has more than two components and the feasible range for each component can be large. As a result, the decision space of configurations can be quite large. The problem then becomes how to find a system configuration S_i which yields the best performance value for a proposed FMS from a large configuration space. The simulated annealing algorithm as well as SQG are selected to search for the system configuration which produces the best system performance.

Input Parameters

We have already depicted the hypothetical flexible manufacturing systems, the throughput optimization model and the cost model. In this section, we will discuss the required inputs to the model. There are two types of input parameters: *system parameters* and *cost parameters*. The system parameters contain the interarrival rate, the number of the components, the capacity of each component, the number of pallets, the number of AGVs, the speed of an AGV, the processing time at each station, machine failure rate, the time to repair the machines and the planned production rate. The numerical data for these parameters are given in Table 3.2 through 3.4.

The cost parameters include the cost of each facility, the unit cost of floor space, labor rate, repair cost, inventory holding cost rate, balking cost per unit, the cost of under-production and the penalty rate of the under-utilized equipment. The numerical values of the cost parameters are listed in Table 3.5. The cost values have been *normalized* or *scaled down* rather than using the large cost values. This makes it easier to calculate and express the total cost of the system.

Table 3.5: The numerical data of the cost parameters

Items		Value	Items		Value
Under production		5.5	Balking loss		0.005
Holoding costs(h_i):			Repair costs(f_i):		
	Part A	0.0015		machine center	0.197
	Part B	0.0011		deburring robot	0.101
	Part C	0.0019		Inspection tool	0.247
Resource costs(e_i):			Under util.(μ_i):		
	Process #1	0.103		Process #1	0.03
	Process #2	0.103		Process #2	0.03
	Process #3	0.103		Process #3	0.03
	Process #4	0.103		Process #4	0.03
	Workers	0.39		Workers	0.045
	Machines	0.68		Machines	0.08
	Robots	0.43		Robots	0.05
	Inspection tools	0.81		Inspection tools	0.1
Floor Space(λ)		0.03			

CHAPTER 4. THE CONCEPT OF SIMULATED ANNEALING

Introduction of Simulated Annealing

As mentioned before, simulated annealing is one of the techniques for application to combinatorial types of optimization problems. It was introduced by [KIRK 83]. The basic idea of the simulated annealing approach was inspired from statistical mechanics and motivated by an analogy to the behavior of physical systems of annealing in solids. In mechanics, it is concerned with how to coerce a solid into a low energy state. A low energy state usually means a highly ordered state such as a crystal lattice. To achieve this, the material is annealed: heated to a temperature that allows many atomic realignments, then cooled carefully, slowly until the material freezes into a good solid. The simulated annealing technique uses an analogous set of controlled cooling procedures for nonphysical optimization problems, to transform a poor solution into an optimal, or near optimal solution.

Knowing how simulated annealing applies to optimization problems, we first need to understand the *iterative improvement* or *local optimization.* Iterative improvement is the most common framework used in heuristic methods of multivariate optimization. It can be seen as a special case of simulated annealing. In iterative improvement, one starts with the system in an initial configuration C_i, then rearranges it until an improved configuration C_j is found. *(We have to specify a method for re-*

arranging the current configuration. The set of configurations which can be obtained in one such step from a current configuration C is called the neighborhood of C.) C_j then becomes the starting point for further rearrangement. This process terminates when no further improvements can be found, i.e., it will be *locally optimal* in that none of its neighbors has a *better* configuration.

The inherent limitation in iterative improvement is that this process may converge to a local optimal solution and there is no way to escape from these unattractive local optima. Since iterative improvement procedures may get *stuck* at local optima, some tricks need to be found to increase the probability of getting the solutions reasonably close to the unknown global optimum. Simulated annealing is an approach that attempts to avoid having the improvement process become entrapped in poor local optima by allowing an occasional hill-climbing move. This is achieved by using the *Metropolis procedure.*

Metropolis Procedure

The Metropolis procedure is the heart of the simulated annealing which is performed under the effects of a random number, current and next states, and *temperature (a special term used in simulated annealing).* At each step a new state of the system is generated from the current state via the *neighborhood function.* If, for instance, the cost associated with the new state is lower than or equal to the cost of the current state, the move from current state to new state would be accepted, i.e., the new state becomes the current state. On the other hand, if the cost of the new state is higher than the current state's, the acceptance of this move would be based on the comparison between a random number generated uniformly in the interval

[0, 1] and the probability,

$exp(-(C_j - C_i)/t_s)$

where,

C_j = cost of the new state,

C_i = cost of the current state,

t_s = control parameter (temperature) at iteration s,

s = iteration number , s = 1,2,...

Note $exp(-(C_j - C_i)/t_s)$ will be a real number between 0 and 1. $(C_j - C_i)$ is so-called uphill and T_s is positive.

For high temperatures, the uphill move has a very high likelihood of being accepted, and this acceptance probability decreases as the temperature declines. In addition, for a fixed temperature t_s small uphill moves have a higher probability of acceptance than do larger ones. It should be apparent at this point that simulated annealing is just iterative improvement done at a sequence of finite temperatures with the Metropolis procedure for accepting or rejecting a generated trial move instead of an *improvement-only* rule. Now, as was pointed out by Kirkpatrick et al., the two most important issues in implementing the simulated annealing approach are those of the *annealing schedule* and *the test for equilibrium* (steady-state) at each temperature in the foregoing algorithm.

The annealing schedule is a less well-defined concept. It can be considered as the sequence of temperatures and the amount of time to reach equilibrium at each temperature. For different applications, there are different approaches for determining the sequence of temperatures in the annealing schedule. Those can range from constant ratio to various non-linear ratios. A method of establishing the cooling

schedule for optimal annealing was provided by [HAJE 86]. A theoretical analysis of annealing schedules also can be found in [MITR 86]. In our study, the constant ratio method for decreasing temperatures will be used,

$t_i = t_{i-1} \times \rho$

where, i= 1,2, ... ,s and $0 < \rho < 1$.

At each temperature, the simulation must proceed long enough for the system to reach the thermal equilibrium (steady state). However, there is no well-defined method to test for equilibrium. Based on a number of experiments in our study, a steady state is said to be reached after a preset number of configurations have been rejected consecutively or a given maximum number of iterations have occurred at each temperature.

Given a schedule of temperatures, $T = \{ t_1,t_2,\ldots,t_{m-1},t_m \}$ with $t_1 > t_2 > \ldots > t_{m-1} > t_m$, the general simulated annealing algorithm can be outlined as shown in Figure 4.1 (adopted from [MITR 86]).

Since the appearance of simulated annealing, it has been applied to many *deterministic* multivariate combinatorial optimization problems in diverse areas: design of integrated circuits, network theory, image processing, quadratic assignment problem, graph partition problem, scheduling, and so on. It has proved to be a useful and reasonably general tool for deterministic combinatorial problems even though its efficiency is not always as great as one might hope. Unfortunately, most of the applications with simulated annealing are designed for deterministic and not stochastic problems. When one extends the problem range to stochastic combinatorial problems where the value of the system's configurations can only be estimated, there are few methods which offer any hope of solving these complex problems. [BULG 88]

```
==========================================================
Begin;
(* given the initial state j0 and initial value for the parameters T0*)
    X := j0;
    s := 0;
    While (stopping criterion is not satisfied)
       begin
          While (inner loop criterion is not satisfied)
             begin
                j := Generate(X);
                if (Accept(Cj, Cx, Ts))
                X:=j;
             end;
          Ts+1 := Update(Ts);
          s := s + 1;
       end;
End;
Accept(Cj, Ci, T)
begin
(* return 1 if the cost variation passes a test. T is the control parameter*)
    ΔCij := Cj - Ci;
    y := Min [ 1, exp(-ΔCij/T) ] ;
   r := Random(0,1);
(* Random is a function returning a random number uniformly distributed in the
interval[0,1] *)
    if ( r ≤ y)
       return(1);
    else
       return(0);
end;
==========================================================
```

Figure 4.1: Simulated Annealing Algorithm

extended the application of simulated annealing to the domain of Monte Carlo optimization by proposing a modified simulated annealing algorithm to optimize buffer sizes in automatic assembly systems. They included statistical analysis procedures in the evaluation of the objective function. In the following chapter we will discuss a modified version of simulated annealing which allows for its application to stochastic problems.

CHAPTER 5. MODIFIED SIMULATED ANNEALING

Most of the optimization problems arising in the real world are stochastic. To those problems, Monte Carlo Optimization techniques are applied. One of our goals is to integrate an extension of the simulated annealing algorithm with discrete event simulation of a manufacturing system to search for an optimal system design. In order to integrate the simulated annealing algorithm into Monte Carlo optimization procedures, some modifications are necessary.

Statistically Measuring System Performance

The need for statistically measuring system performance is based on the observation that the output data from a simulation exhibit random variability because a random number is used to produce the values of the input variables. Since the nature of the output data is *stochastic*, it is necessary to have statistical analysis procedures for the simulated annealing algorithm. We formally state these additional procedures in the following sections.

Initial Bias Deletion

The initial conditions of the system at time 0 may influence the output data. If not chosen well, the specified initial conditions will have an especially deleterious ef-

fect when attempting to estimate the steady-state performance of a simulation model. Several ways were developed for reducing the estimators' bias caused by unrealistic initial conditions. The simplest and most general technique for determining initial conditions is a *graphical procedure.* The idea here is to use a graphical approach to estimate the *warmup period.* Then we start to collect output data only after the warmup period is completed and continue until the simulation is terminated.

Confidence Interval Estimate

Consider the comparison of two alternative system designs. It is desired to have an interval estimate of system performance. A valid interval estimation requires a method to estimate the variance of the point estimator in a relatively unbiased fashion. However, since the sequence of output observations is not independent, we can not apply classical methods of statistics directly. However, if a modified approach such as the *replication method* or *batch means method* is used to meet the independence requirement, we can then use classical statistical techniques to estimate the variance and confidence interval.

Batch Means Method

The batch means method parallels the method of independent replications used for terminating systems. However, instead of replicating the simulation with a defined initial starting condition, it divides the sequence of data from a single run into subsequences (or batches) of data that are approximately independent of each other. A batch of data then is treated as though it were an independent replication of the system. It is known that if the batch size is sufficiently large, successive batches

will be approximately independent. Therefore, classical statistical analysis can be performed. The idea here is to choose an appropriate batch size to obtain an independent sample and then statistical analysis is conducted to measure the system performance.

Comparison of Two System Designs

The heart of this modified version of simulated annealing is to make a comparison among the system designs based on whether or not the values of their objective functions indicate they are statistically significantly different at each iteration. To draw valid conclusions, we must resort to the statistical procedures for testing significant differences. The additional testing procedures we incorporated into modified simulated annealing for comparing two system designs are described as follows.

Tests Concerning Differences between Variances

By performing an F test, we can test the equality of the variances of two normal samples. If it can be reasonably assumed that both variances are equal, we can compute a single estimate of variance S_p^2 as follows.

$$S_p^2 = \frac{(n_1-1)S_1^2+(n_2-1)S_2^2}{n_1+n_2-2} \qquad \text{where,}$$

S_1, S_2 = estimated sample variances

n_1, n_2 = sample size

This procedure pools the data from both samples. Thus, S_p^2 is known as a *pooled variance.* If our assumption of equality of variances is not reasonable, then we can not obtain a single estimate of variance by simply pooling the sample variances. The

appropriate calculation then becomes

$$S_s^2 = \frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}$$

which is known as a *separated variance.*

Tests Concerning Differences between Means

To test the hypothesis that there is a significant difference between two designs, we set a confidence interval on the difference in population means of the performance measures $\mu_1 - \mu_2$. This confidence interval can be constructed from differences in sample means, $\overline{Y}_1 - \overline{Y}_2$, a single estimate of variance (S_p^2 or S_s^2) derived from both samples and the t-value (*we assume sample size is small*). Depending on the position of this confidence interval relative to zero, we conclude:

1. There is no significant difference between two system designs if it contains zero.

2. These two system designs are statistically different if it does not contain zero.

The question of *how to decide whether or not to accept the new system configuration to be the current one when they are statistically equal* arises. In this case, the iteration is ignored and more analyses are needed as follows.

Metropolis Procedure with Reward Process

Regard S_1 as the current configuration. We could acquire a new configuration, say S_2, by perturbing S_1 via a specified *neighborhood function.* After performing the procedures which test whether or not S_1 and S_2 are statistically significantly different, we would have 3 different cases:

Case 1. S_1 and S_2 are significantly different and S_2 is better than S_1. Clearly, we accept S_2 to be the current configuration.

Case 2. S_1 and S_2 are significantly different and S_2 is worse than S_1. Then we apply the Metropolis procedure to give a certain probability for allowing acceptance of S_2.

Case 3. S_1 and S_2 are not significantly different, i.e., they are the same statistically.

The problem occurs in **Case 3**. If we always eliminate S_2 and keep S_1, we may significantly increase the number of simulation replications required. In addition, when all the neighbors of S_1 are the same as S_1 statistically, the algorithm will get stuck at S_1, never moving further (as Figure 5.1). If we always accept S_2 and the successors of S_2 are getting worse than S_2 but have no significant differences (as Figure 5.2), the algorithm will proceed in the wrong direction undetected. Therefore, we present a *reward process* for supporting the simulated annealing algorithm in order to make a decision of *when to accept* S_2 as follows:

Step 1. Keep the current configuration S_1.

Step 2. Regard S_2 as current configuration (actually not) and perturb S_2 to get next new configuration, say S_3.

Step 3. Evaluate S_3 as usual.

Step 4. Compare the performances of S_2 and S_3.

Step 5. With the comparison of S_2 and S_3, do

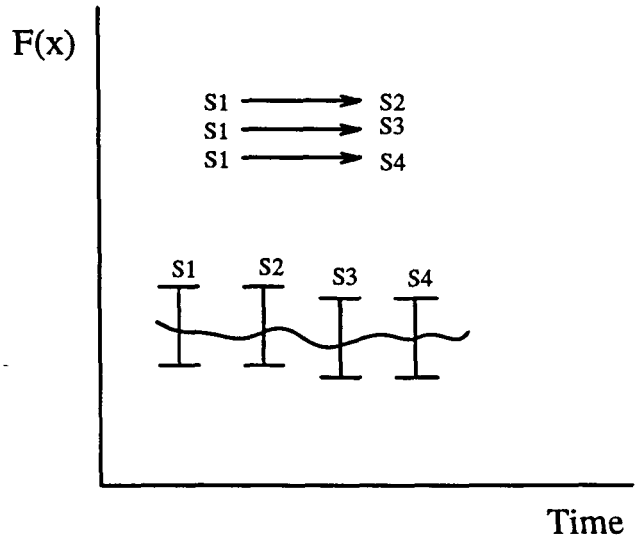

Figure 5.1: Unsavory situation if always accepting S_1

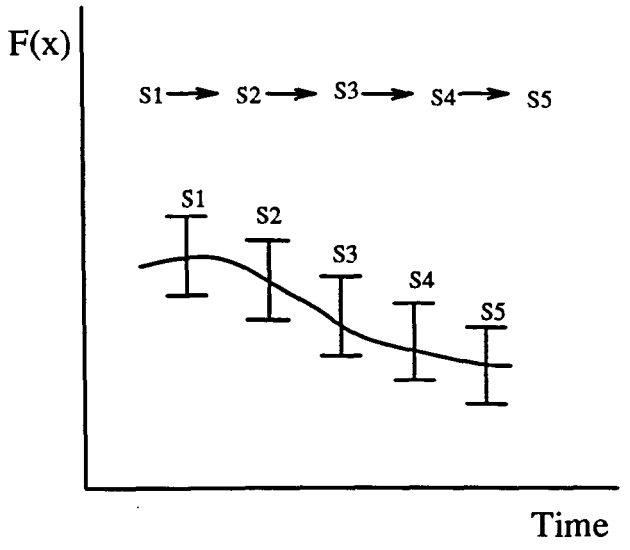

Figure 5.2: Unsavory situation if always accepting S_2

Case 1. If S_3 is significantly different from S_2 and S_3 is better than S_2, then accept the configurations S_2 and S_3, and let S_3 to be the new current configuration (as Figure 5.3). Return.

Case 2. If S_3 is significantly different from S_2 and S_3 is worse than S_2, then accept the configuration S_2 and S_2 becomes the current configuration, and perform the Metropolis procedure on S_2 and S_3 (as Figure 5.4). Return.

Case 3. If S_3 is not significantly different from S_2, then recall S_1 and compare S_3 with S_1.

case 3a. If we find S_3 is the same as or better than S_1, this indicates a trend that the system is improving (as Figure 5.5), so we accept S_2 to be current configuration. Since S_2 is now the current configuration instead of S_1 and we know that S_2 and S_3 are not significantly different, we need to determine whether or not to accept S_3 i.e. S_2 becomes S_1 and S_3 becomes S_2, so we go directly to step 1 of this procedure.

case 3b. If S_3 is worse than S_1, this is an indication that our system is worsening (as Figure 5.6), then ignore S_2 and S_3, and return to S_1. Return.

The purpose of adding these procedures is to provide a way for simulated annealing to operate in a stochastic environment. The outline for this modified version of simulated annealing is given in Figure 5.7.

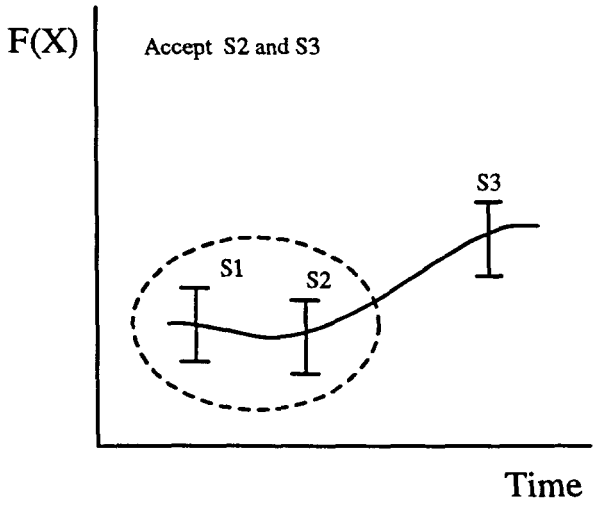

Figure 5.3: Policy made if S_3 is better than S_2

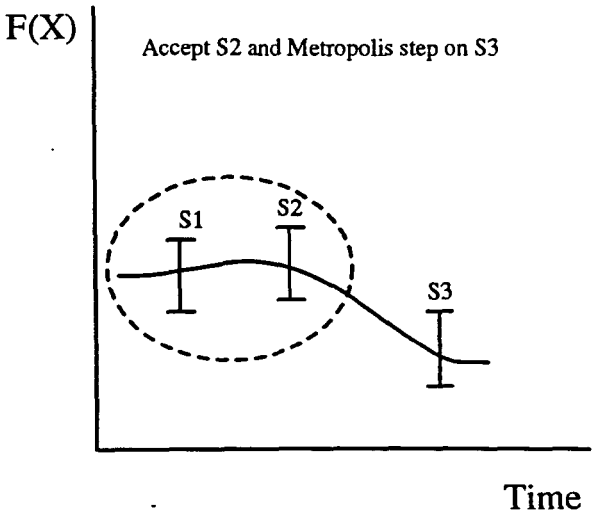

Figure 5.4: Policy made if S_3 is worse than S_2

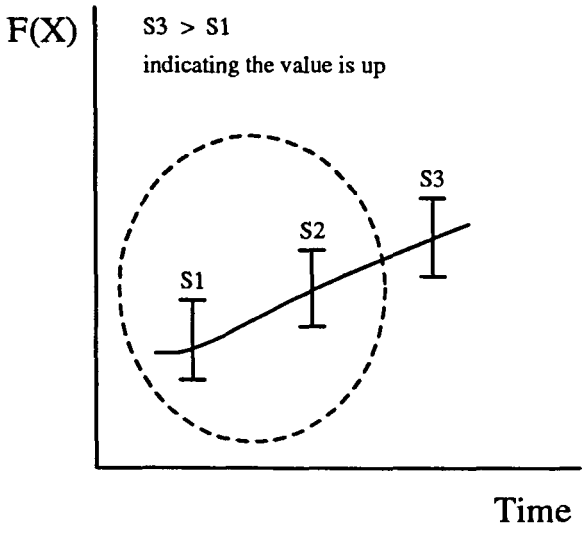

Figure 5.5: System improving if S_3 is better than S_1

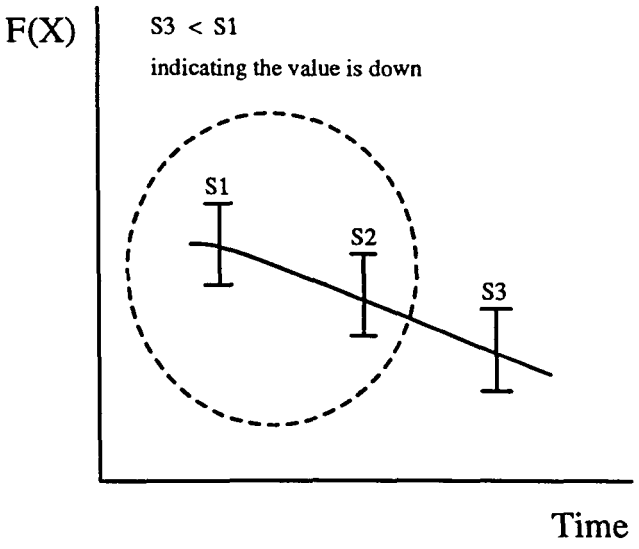

Figure 5.6: System worsening if S_3 is worse than S_1

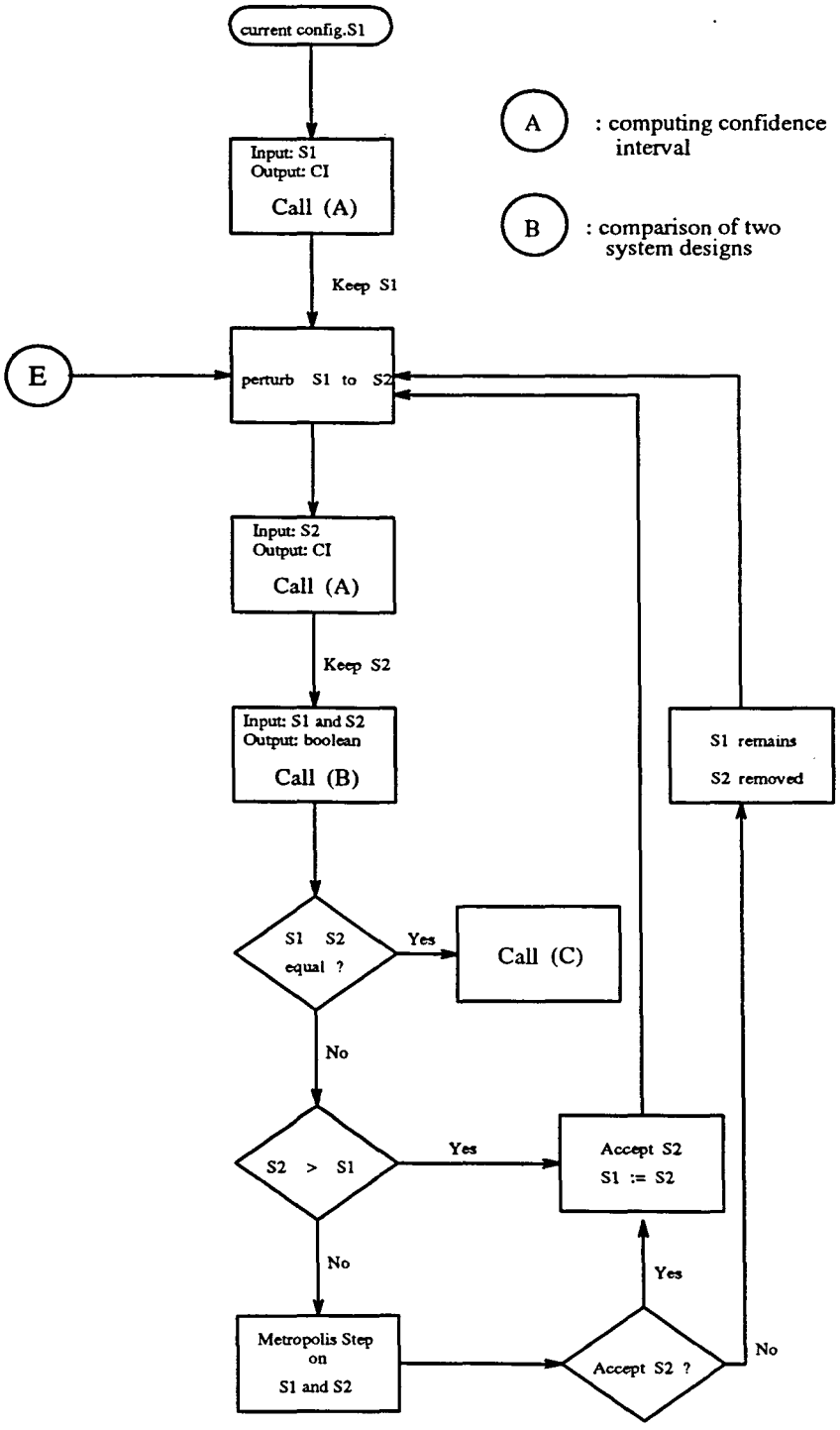

Figure 5.7: The framework of modified simulated annealing

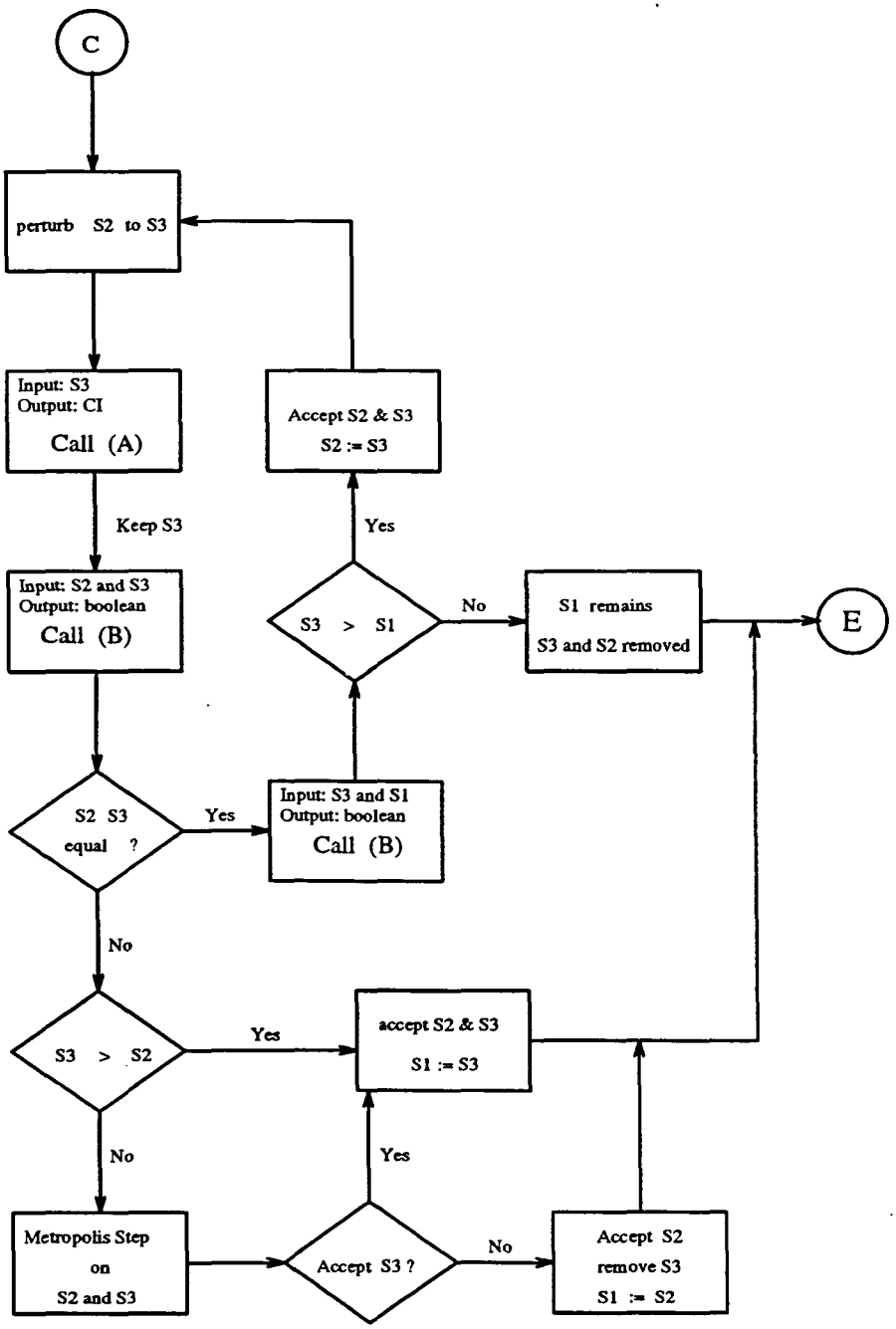

Figure 5.7 (Continued)

CHAPTER 6. STOCHASTIC QUASIGRADIENT METHOD

Consider a one variable minimization problem such as show in Figure 6.1. Suppose that x_1 is the current approximation to the optimal solution x^*. In order to approach x^*, we need additional information about the function value near x_1. Therefore, we evaluate the function value by increasing x_1 by a step size l and then estimate the gradient of the function at x_1. If the gradient appears to be negative, a move of a certain distance to the *right* is made. Otherwise a move to the *left* is made (e.g. x_2). We can see this process moves the current approximation to another approximation that is closer to the optimal point. Stochastic quasigradient methods utilize a similar process.

Introduction

Stochastic quasigradient (SQG) is an algorithmic procedure for solving general stochastic constrained optimization problems with nondifferentiable, nonconvex functions. These methods allow us to solve optimization problems with objective functions and constraints of such a complex nature that it is impossible to formulate and calculate the precise values of these functions. The concept of this approach is to use *statistical estimates* for the value of the function and its derivates rather than the unknown exact values.

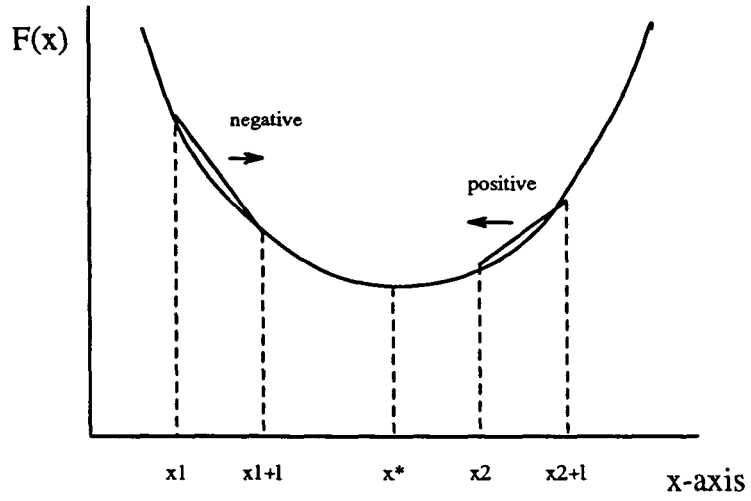

Figure 6.1: Gradient method

Consider the following minimization problem,

$$min F(\mathbf{X})$$

$$F(\mathbf{X}) = E_{\omega}[f(\mathbf{X}, \omega)]$$

$$\text{s.t. } \mathbf{X} \in \mathbf{C}$$

$\mathbf{X}$ is a vector of decision variables to be optimized, $\mathbf{C}$ is a set of constraints, and ω is a random variable belonging to the appropriate probability space. The stochastic function $F(\mathbf{X})$ can not be analytically formulated due to its complexity. Clearly, in this case the gradient of $F(\mathbf{X})$ may not exist and can not be calculated by deterministic approaches. Therefore, we use simulation methods to estimate the expected value of $f(\mathbf{X}, \omega)$ and quasigradients which are statistical estimates of the gradients obtained by using finite-difference methods. We will illustrate some finite-difference methods and the basic SQG algorithm in the following sections.

The Algorithm

The stochastic quasigradient algorithm moves from one feasible point to another as follows.

$$\mathbf{X}^{s+1} = \Pi_{\mathbf{C}}[\mathbf{X}^s - \rho_s \nu^s]$$

$$\mathbf{X},\ \rho,\ \nu \in \mathbf{R}^n$$

where the $\mathbf{X}^s$ is the current approximation to the optimal solution; ρ_s is the step size which determines the magnitude of the move; ν^s is a random step direction, i.e., an estimate of the gradient direction at the current point $\mathbf{X}^s$; s is the iteration number and $\Pi_{\mathbf{C}}$ is the projection operator which keeps the x values within the constraint set $\mathbf{C}$.

Projection

The projection operator keeps x within its constraints by finding the closest value z to x if x is out of bounds. It takes the form:

$$\Pi_{\mathbf{C}}(x) = \arg\min_{z \in \mathbf{C}} \parallel x - z \parallel$$

For example, if x is defined within the integer range [1,10], the projection operator will set x to 10 if x is greater than the upper bound of the range.

Calculation of Step Direction ν

The step direction may be a statistical estimate of the gradient of the function $\mathbf{F}(X)$. We define ν^s as the quasigradient estimated by ξ^s such that

$$E(\xi^s \mid X^1, X^2, \cdots, X^s) = F_x(X^s) + a^s = \nu^s$$

where $a^s \to 0$ as $s \to \infty$. In this case, ν^s is called *stochastic quasigradient* of the function $F(X)$.

There are several methods which can be used to find an estimate of the gradient such as finite-difference approximations, analogues of random search methods, etc. The most common method used is the finite-difference approximation. There are two types of finite-difference approximation method: *forward finite difference approximation* (FFD) and *central finite difference approximation* (CFD).

Forward finite-difference approximation

The FFD method can be written in the form:

$$\xi^s = \sum_{i=1}^{n} \frac{f(X^s + \delta_{si} e_i, \omega_1^{si}) - f(X^s, \omega_2^{si})}{\delta_{si}} e_i$$

where ξ^s is the statistical estimate of the step direction of the function $F(X)$ at iteration s, δ_{si} is the step size for i-th variable, ω_1^{si} and ω_2^{si} are random variables generated for interation s and e_i are unit basis vectors from $\mathbf{R}^n$

Central finite-difference approximation

The CFD method has the form:

$$\xi^s = \sum_{i=1}^{n} \frac{f(X^s + \delta_{si} e_i, \omega_1^{si}) - f(X^s - \delta_{si} e_i, \omega_2^{si})}{2\delta_{si}} e_i,$$

where the notations are the same as for the FFD method.

Comparing these two methods, the CFD method requires twice the number of function value estimates than does the FFD method. However, experience indicates that the CFD method seems to be able to obtain more precise step directions.

Gradient Calculation

Consider a n-variable problem. The estimation of gradients are performed *one variable at a time.* That is, the value of one variable is increased by an amount δ_{si}, and then the estimate of the gradient for this variable is made using FFD. The variable is then reset to its original value before the gradient for another variable is estimated. Therefore, as the number of variables increases, the number of finite difference gradient estimations becomes larger. For example, if central finite difference is applied to a 5-variable problem, then a total of 11 computations are necessary. In addition, with $s = 50$, a total of 550 computations would be made. This becomes very costly in terms of time required. Thus, the forward finite difference approximation method is utilized in our application in order to reduce the time required by about one half.

Normalization of Gradients

Sometimes it is useful to *normalize* the estimate of the gradient [ERMO 84]. The normalized version would be

$$\nu_s = \frac{\nu_s}{|\nu_s|}$$

As a result, the quasigradient only appears to be positive or negative.

Choice of Step Size ρ

We trigger the algorithm with the initial step size. If the current point is far away from the optimal point, we need a large step size. As the algorithm approaches the optimal region, a smaller step size is needed in order to converge to the solution. Generally speaking, there are three different methods for determining the step size

of each iteration : *interactive selection*, *automatic adjusting*, and *pre-determined step size sequence.*

Interactive selection

[ERMO 83] considered this approach as the best strategy. We change the step size based on examination of the changes of the function estimates at several iterations. There are three different cases :

1. When the changes appear to be too small or insignificant, we will enlarge the step size.

2. When the function estimates result in random jumps, we will reduce the step size.

3. Or when the function estimates are steadily decreasing (or increasing), we will not change the step size

This method requires an experienced user and remains somewhat of an art. The major disadvantage of the interactive procedure is the time needed for the user to operate the the program.

Automatic adjusting procedure

When it is impractical to use an interactive approach, an automatic adjusting method can be used. This procedure uses information obtained from the previous iterations to determine whether or not to adjust the step size for the successive iterations. It consists of calculating the *ratio of improvement* of the function value to distance traveled or path length. If the ratio is less than a preset threshold, the step size is modified by using $\rho_{s+1} = \beta\rho_s$, where β is a step size multiplier, usually $\beta \in (0,1)$ and ρ_{s+1} is the step size for iteration $s+1$.

Pre-determined step size sequence

A pre-determined step size sequence is a special case of the automatic adjusting procedure. As with the cooling schedule of simulated annealing, there is no well defined approach for selecting the step size sequence. One is to choose the sequence by using the following relation:

$$\rho_s = \rho_{s-1} \times \alpha$$

where, s = 1,2,...,m and $0 < \alpha < 1$.

When using this procedure, we can only expect convergence to a local optimum.

Stopping Criteria

There is no well defined stopping criteria for the SQG method. Several possible practical stopping criteria include stopping after *a set number of iterations*, when the *improvement* in the function value is small enough, or when the *step size* is less than some minimum value. We will terminate the algorithm when a set number of iterations is reached or the improvement in the function value is less than a preset value, which ever occurs first.

Additional Features

Again, due to the stochastic nature of the problem, we can not neglect statistical analysis procedures. The additional statistical analysis procedures are similar to those used in simulated annealing such as, *initial bias deletion*, *batch means method*, *confidence interval estimate* and *comparison of two function values*. We then include these procedures to modify the forward finite difference approximation method as follows:

Step 0. Set $k = 0$ and keep $f(\mathbf{X}, \omega)$

Step 1. Estimate $f(\mathbf{X} + \delta_i e_i, \omega)$

Step 2. **If** $k <$ M (number of replications allowed) do

2a. Compare $f(\mathbf{X} + \delta_i e_i, \omega)$ with $f(\mathbf{X}, \omega)$

2b. If statistically the same, then $k := k + 1$, increase δ_i, go to Step 1.

else, go to Step 3.

Else go to Step 3

Step 3. Calculate the quasigradient for x_i; $k := 0$; Return.

In conclusion, with the modification of the forward finite difference approximation, the stochastic quasigradient algorithm can be pseudo-coded as Figure 6.2.

```
==========================================================
Begin;
   X := j0; (given initial state)
   s := 0; (count for iteration)
   δs := δ0 ;(initial step size of FFD)
   ρs := ρ0 ;(initial step size of algorithm)
   C0 := f(X,ω);
   While (stopping criterion is not satisfied)
      begin
        Repeat
          begin
            X = j0; (restore the original X)
            Ci := f(X + δiei,ω);
            if (NotSame(Ci, C0))
               νsi := Gradient(Ci,C0); (calculate gradient one at a time)
            else
                  if (k ≤ M)
                     k := k + 1 (how many times it appears to be the same)
                     δi = δi + V (increase the step size in FFD)
                  else
                     k := 0; (reset to initial value)
                     νsi := Gradient(Ci, C0);
                  endif
            endif
          end;
        until (all of the variables)
        X := Projection(X - νsi ρs);
        j0 := X; (keep X)
        C0 := f(X,ω);
        s := s + 1; (increment the iteration number)
        while (necessary) do ρs := α ρs−1; (change step size)
        while (necessary) do δs := β δs−1; (change step size in FFD)
      end;
End;
==========================================================
```

Figure 6.2: Stochastic quasigradient algorithm

CHAPTER 7. IMPLEMENTATION

The application of discrete parameters Monte Carlo optimization to the FMS design problem was implemented. The hypothetical flexible manufacturing system under this study was described in Chapter 3. In the design phase, given the capacities of AGVs and pallets (they are assumed to be large enough to support the system), the design parameters which needed to be optimized are listed in Table 7.1.

Table 7.1: A list of design parameters and corresponding variables

Design variables	Description
x_1	Capacity for process #1
x_2	Capacity for process #2
x_3	Capacity for process #3
x_4	Capacity for process #4
x_5	The number of workers allocated at the setup area
x_6	The number of machines allocated at the machining station
x_7	The number of robots allocated at the deburring station
x_8	The number of tools allocated at the inspection station

A combination of these system design parameters represents a system configuration. We use two different objective functions. One is to *maximize the number of parts produced per unit time*, the other is to *minimize the total production cost*. First, the initial bias period and appropriate batch size are determined after a number of pilot simulation runs are made. Secondly, both of the algorithms (modified simulated

annealing and SQG) are invoked by giving an initial system configuration and the required inputs; then the simulation is triggered to obtain the estimated values of the objective functions. The algorithms will be terminated when the specified stopping criteria are satisfied.

Determination of Transient Period

One of the methods to resolve the initial bias problem is to discard those observations recorded during the transient phase of the simulation. This approach necessitates selecting a truncation point t at which all the observations recorded before t are excluded. The outcome of the simulation is then based upon the observations recorded after the truncation point t.

The simplest, most practical and probably best approach for selecting a truncation point is *visual* determination, i.e., selecting a truncation point from the simulation response plots over the simulation time. In systems with large fluctuations in the response, we can improve this process by first using a moving-average approach to smooth the response. A moving average is constructed by calculating the arithmetic average of the k most recent observations at each data point in the data set. We utilize the SIMAN Output Processor to accomplish this. The SIMAN Output Processor is an interactive program which operates in a post processing mode to help us analyze the data generated from a simulation model. It provides two types of data analysis – *Graphics* and *Statistics*.

We randomly choose several system configurations for conducting graphical output analysis. The plots shown in Figure 7.1 and 7.2 are created by using the Output Processor on a data set generated from a simulation run.

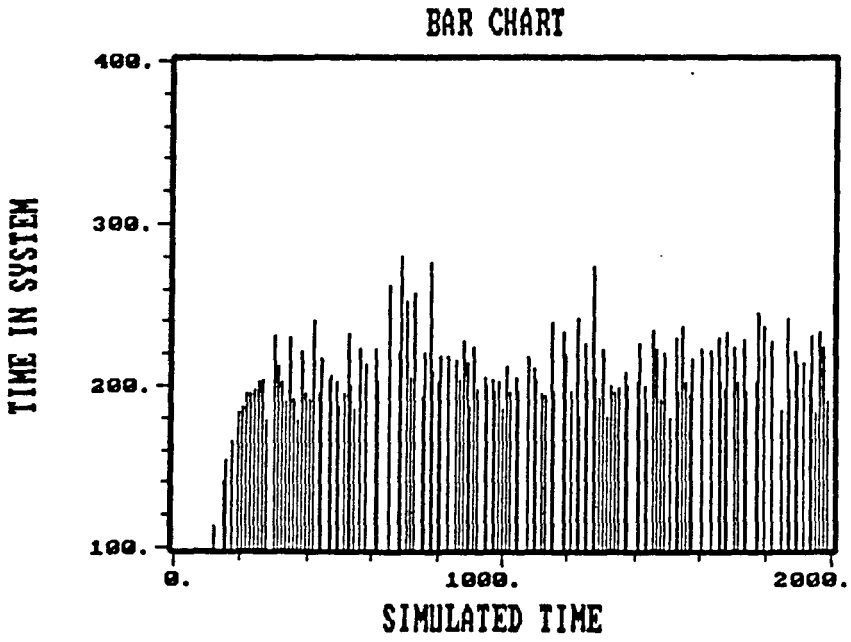

Figure 7.1: Time in system responses versus simulated time

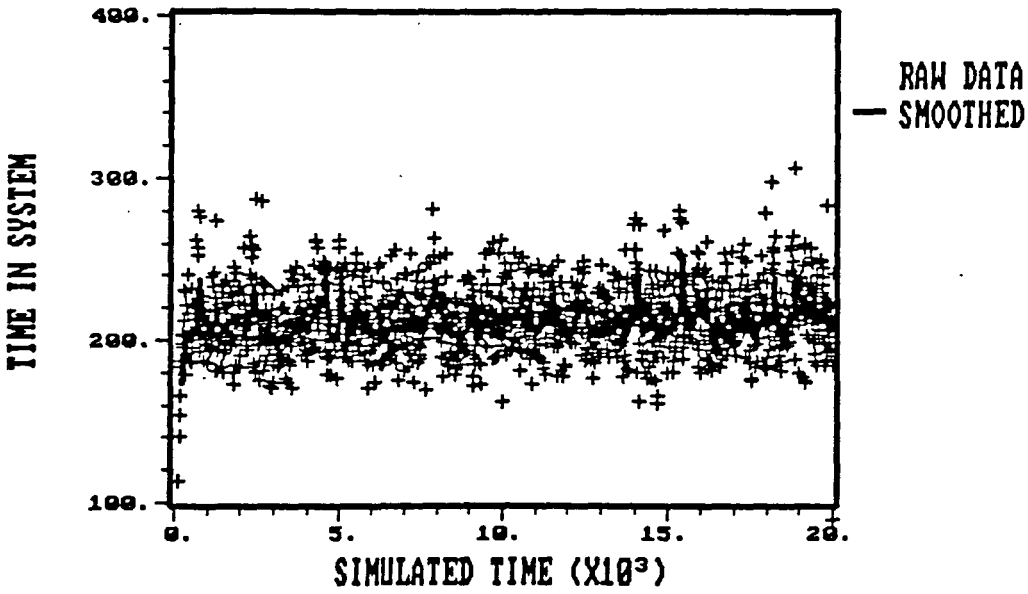

Figure 7.2: Smoothing the responses by using moving average with lag 15

Figure 7.1 shows the *Time In System* responses within the run time period; Figure 7.2 was obtained by using the moving average method. We use the 15 most recent observations to compute the moving average at each point. The size of the moving average was determined by experimenting with several increasing values beginning with 10 observations.

We have experimented with several system configurations in order to estimate the truncation point t. After performing these experiments, *we determined that* 1000 *time units as the transient period and it is assumed that this value can be used regardless of the system configuration.*

Selection of an Appropriate Batch Size

The key to making the batch method work well is to select an appropriate batch size. As mentioned before, if the batches are sufficiently large, the means of the two adjacent batches will be approximately independent, even though the observations at the end of batch j are correlated with the observations at the beginning of batch $j+1$. The size of the batch required to achieve this independence is a function of the correlation structure for the system response since the observations are correlated within the run. Therefore, the most practical way to select the batch size is to examine a *correlogram*, which is a plot of the sample correlations. Again we utilize the Output Processor to make a correlogram.

Figure 7.3 shows a typical correlogram generated from simulation data recorded for the measure of *time in system* for each part. As can be seen from the correlogram, the correlation in the data is significant for lag 1 (the distance between observations being compared) up to lag 5, but is relatively small for a lag of 8 or more. This

simply means that time in system for one part significantly influences the time in system for the parts that closely follow it but is less significant on parts that follow 10 or more later in the sequence.

Although we have used a fixed number of discrete observations per batch, the batch means method can also be applied to time persistent data by defining the batch as a fixed duration of time in which case the number of observations per batch is a random variable. The same analysis procedure is used in time persistent data to support the determination of batch size. The **FILTER** command helps us to gain the necessary information for determining the appropriate batch size as shown in Figure 7.4.

Several experiments were done upon different system configurations. A 800 time unit duration was elected to be the batch size. *For each system configuration generated in each simulation run, we assume the appropriate batch size is constant.*

Consequently, a simulated time of 9000 time units for each simulation replication was selected due to the summation of 1000 time units for warmup period and 8000 time units for 10 batches (each batch size is equal to 800 time units).

Simulation Model

The framework of the simulation model for the hypothetical flexible manufacturing system was developed by using SIMAN – the general purpose simulation language. Several pilot simulation runs and modifications were conducted to ensure that the simulation model behaves as desired.

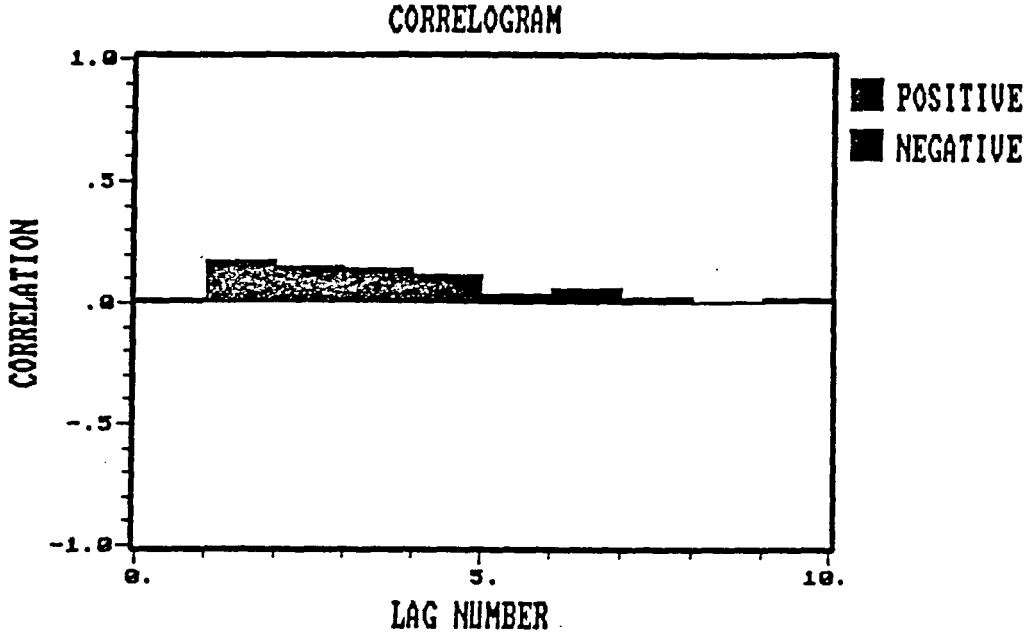

Figure 7.3: Correlogram for time in system observations

```
> FILTER, 12, TIME IN SYSTEM, T/1000, T/800:

     FILTERS SUMMARY : TIME IN SYSTEM
 ------------------------------------------------

     INITIAL TIME TRUNCATED              1000.
     TIME SPANNED PER BATCH              800.0
     NUMBER OF BATCHES                   23
     TRAILING TIME TRUNCATED             590.3
     EST. OF COV. BETWEEN BATCHES        0.2102
 ------------------------------------------------
```

Figure 7.4: Determining an appropriate batch size by using **FILTER** command

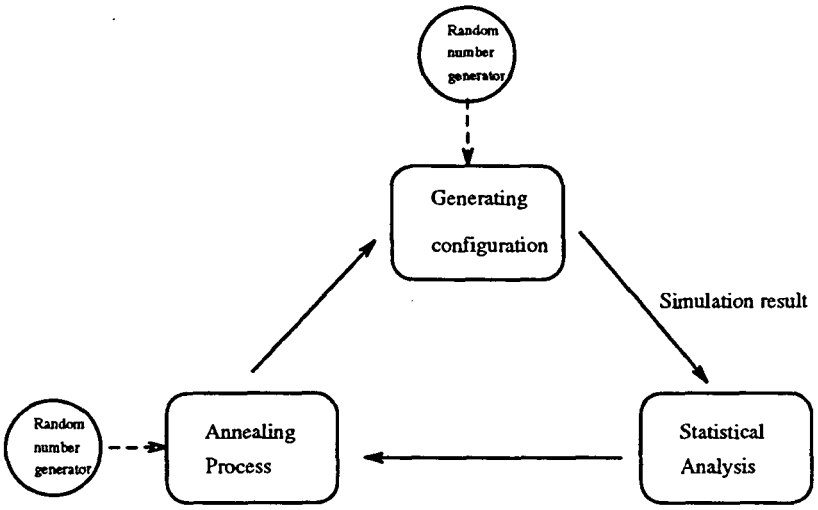

Figure 7.5: Linkage among procedures of simulated annealing

Implementation of Modified Simulated Annealing

After determining the truncation point and the batch size, they were included in the simulated annealing as a constant. The modified version of simulated annealing was written in Microsoft FORTRAN v4.0 and was designed to contain four main procedures (see Figure 7.5):

1. *Generating system configurations* includes an interactive input environment for importing an initial configuration and the required parameters, and a subroutine for determining the next configuration via a neighborhood function.

2. *Statistical analysis* computes the confidence interval of the output data generated from the simulation run and makes a comparison of two system configurations with a 95% significance level.

3. *Annealing process* performs the Metropolis step, checks the equilibrium stage and updates the temperature based on the cooling schedule.

4. *Random number generator* uniformly produces random numbers in the interval [0,1] by using the multiplicative linear congruential method.

In the algorithm, the generation of the next configuration is done by fetching it from the set of the current configuration's neighbors. We use a random number to decide which neighbor of the current configuration is to be fetched. For example, when a generated random number is less than 0.5, it reduces the resource capacity by a pre-determined amount; when a generated random number is greater than 0.5, it increases the resource capacity by a pre-determined amount. In other words the neighborhood function is a function of a random number. We are also given a set of constraints which are lower and upper bounds of resource capacity. When any resource capacity of the system configuration is beyond the constraints, it is pulled back to the feasible capacity which is nearest to the bounds.

The Metropolis step is performed based upon the cooling schedule. When the system reaches the equilibrium state at a given temperature, the temperature is reduced via a cooling function. The cooling function that we use for updating the temperature parameter has the form $t_i = t_{i-1} \times \rho$, $0 < \rho < 1$. As discussed before, the procedure needs to proceed long enough at each temperature to let the system reach the equilibrium state. It is said that when a preset number of generated system configurations has been rejected consecutively or the total number of iterative runs at a temperature exceeds a specified maximum number, *the system has reached the equilibrium state.*

The algorithm runs iteratively. We terminate the algorithm after reaching a maximum number of iterations at which a very low temperature has been reached and where we can regard the system as *frozen.*

Implementation of Stochastic Quasigradient Algorithm

SQG was also coded with Microsoft Fortran v4.0 and can be divided into three parts :

- *Configuration generation procedure* moves the current configuration to another feasible configuration according to the quasigradients and makes an appropriate change to the step size if necessary.
- *Gradient calculation procedure* calculates the statistical estimates of the gradient of the stochastic function by forward finite difference approximation.
- *Statistical analysis procedure* includes the calculation of confidence intervals and the comparison of two system configurations with a 95% significance level .

Given a system configuration, the move to the next configuration depends on the step size and quasigradients. Due to the discrete characteristics of the variables, quasigradients obtained from FFD method are normalized. As a result, the quasigradient only appears to be 0, -1 or 1, i.e., it only indicates the direction to move. For example, if the quasigradient is 0, the value of the variable is not changed, or if the quasigradient is 1, the value of the variable is reduced while handling the minimization problems.

The step size determines the amount of distance to move. Each decision variable has a corresponding step size. For example, if a variable represents the buffer size,

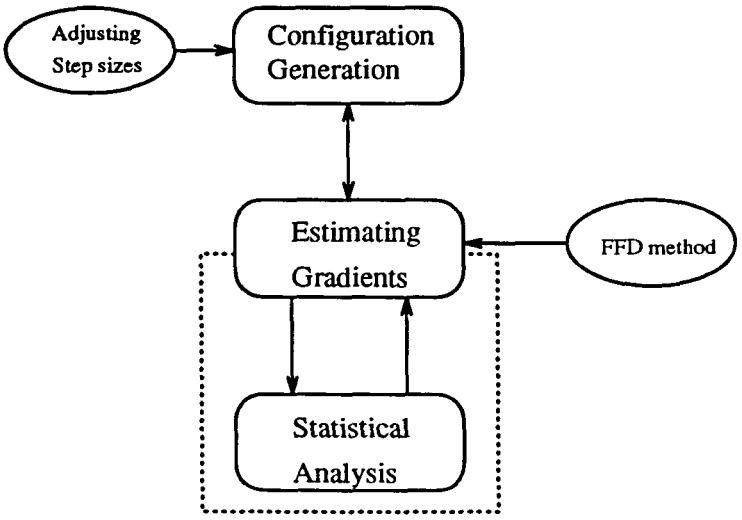

Figure 7.6: Linkage among procedures of SQG

a reasonable step size corresponding to buffer size is used. Initially, the step sizes are large for both the algorithm and FFD. The step sizes are then reduced gradually by the preset sequence until the step sizes reach the minimum value allowed. In our case, the minimum step size allowed is 1 since the resource capacity is an integer and at least one.

We define the change of the system performance as $\frac{|\sum_{i=1}^{n} F_{i+1}-F_i|}{n-1}$. At each iteration, five previous runs were kept in order to calculate the change upon the system performance. If the change appears to be less than what we expect, the algorithm is terminated. Additionally, a counter is set to count the iteration number. If the total number of iterations is greater than the maximum iteration number allowed, the algorithm is terminated.

CHAPTER 8. RESULTS AND ANALYSES

We have addressed the FMS design problem, Monte Carlo optimization methodology, modified simulated annealing, stochastic quasigradient method, the development of a simulation model, and implementation aspects in the previous chapters. As shown before, there are several factors that could affect the performance of simulated annealing such as starting point, initial control parameter, cooling schedule, neighborhood function, time to reach equilibrium state at each temperature and stopping criteria. There are no well-defined approaches to find the preferred starting conditions. Similarly, there are also many factors that could affect the results when using the SQG method; for example, the starting point, initial step size in the algorithm, initial step size in FFD, method of adjusting step sizes during the run, and stopping criteria. Again, SQG is also not easy to select a good starting condition.

As a result, we have done several experiments on both methods utilizing different values for those factors. It was interesting to see how those factors influence the algorithms' behavior. However, it is not our main purpose to have a detailed analysis about the effects of these factors. The application and comparison of simulated annealing and SQG to the FMS design problem is our main concern. Based on our experiences, preferred starting conditions for both simulated annealing and SQG were chosen from pilot experiments. They are shown as follows.

Simulated annealing

starting point: (1, 1, 1, 1, 1, 1, 1, 1)[1]
cooling schedule: $t_i = t_{i-1} \times 0.9$
stopping criteria: *25 maximum iterations allowed*

Stochastic quasigradient

starting point :(1, 1, 1, 1, 1, 1, 1, 1)
step size in FFD : 1.0
adjusting step size : $s_i = s_{i-1} \times 0.9$
stopping criteria : *25 maximum iterations allowed*

With the starting conditions determined, a series of experiments were performed in order to examine the application of the simulated annealing and SQG methods and to make a comparison of the two methods. The results are shown in a later section. Discussions of a throughput optimization model and a cost optimization model are given in the following section. Finally, a comparison of simulated annealing and SQG is addressed.

Summary of Experiment Results

Since we have two models and two optimization methods, we categorize the experiments into four classes : (1) throughput optimization with simulated annealing, (2) throughput optimization with SQG, (3) cost optimization with simulated annealing and (4) cost optimization with SQG. In addition, in order to examine the effects of random variates, the random number seeds were changed and the experiments for four classes were replicated. The pairs of experiments are differentiated by *sample 1*

[1]This expression represents the combination of the design variables or a system configuration. A system configuration is denoted as the form of (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) and the meaning of each scalar can be referred to Table 7.1; for example, x_1=1 represents one tool is allocated at process #1.

and *sample 2*, i.e., experiments in sample 1 used the same random number seed, similarly in sample 2.

Throughput Optimization with Simulated Annealing

As defined, system throughput is the number of parts produced per unit time , which is the reciprocal of expected cycle time for the system. Simulated annealing was applied at three levels of initial temperature, $t = 1.0$, $t = 1.5$ and $t = 3.0$. The results are summarized in Table 8.1. The outcome of each experiment associated with the different initial temperatures are plotted in Figure 8.1, 8.2 and 8.3. These give us a graphical view of how the algorithm responded. Figures 8.4 and 8.5 combine the plots for all three temperatures by sample. These plots help us examine how the temperature parameters affect the algorithm's response.

Due to the sophisticated interrelations between an FMS's components, there does not appear to be a unique optimal design. As we can see in Table 8.1 , the experiments generated several different good system configurations. Those configurations seem to have the same system throughput, approximately *12.5* (*units/hr*). Using different seeds affects the transient response of the algorithm during the run (see Figures 8.1, 8.2 and 8.3). However, each instance reached a solution with approximately the same throughput.

Observation of Figures 8.4 and 8.5 seems to indicate that a smaller initial temperature causes faster convergence. The faster convergence is caused by a decrease in the probability of *uphill* movement due to smaller initial temperatures. However, the smaller initial temperature increases the risk of convergence to a local optima.

Throughput Optimization with SQG

SQG was applied to the same model with the same system inputs. In this class, we conducted experiments with three different initial step sizes, $s = 1.0$, $s = 2.0$ and $s = 3.0$. The results are summarized in Table 8.2. As with simulated annealing, different good system configurations were obtained and each appeared to have the same system throughput, approximately *12.5* (*unit/hr*).

Figures 8.6 through 8.8 show the plots for the experiments associated with each step size. In this case, the use of different seeds did not seem to affect the performance of SQG and did not cause much change in the response surfaces. The effects of the initial step size on SQG's responses can be seen in Figures 8.9 and 8.10. In this case, $s = 2.0$ appears to be the best initial step size because it allows the algorithm to approach the optimal solution space more quickly. This phenomenon fits our claim in Chapter 5 – "when the response point is close to the near optimal solution space, a smaller step size is needed.".

Cost Optimization with Simulated Annealing

In this class, the cost optimization model was investigated. The total cost function was developed in Chapter 3, and simulated annealing was applied to solve the cost model. As before, three initial temperature parameters were utilized in the experiments. The results associated with temperature parameters are graphically shown in Figures 8.11 through 8.15. A summary of the solutions are listed in Table 8.3.

As shown in Table 8.3, several design alternatives were obtained from the different runs, each providing good system performance. The alternatives consistently gave good cost values, approximately *17.3* (*$/hr*). Even with different seeds and ini-

Table 8.1: Summary of the results – throughput optimization with SA

Experiment	Sample	System configuration[a]	95%CI (units/hr)	Run time(mins)
t_0[b] = 1.0	#1	(8, 10, 6, 6, 10, 7, 10, 6)[c]	12.5643±0.1476	1721
	#2	(8, 9, 6, 4, 6, 9, 5, 8)	12.2642±0.1612	1619
t_0 = 1.5	#1	(8, 10, 4, 9, 9, 9, 8, 6)	12.8900±0.0905	1278
	#2	(10, 8, 7, 4, 8, 10, 7, 8)	12.5628±0.2727	1173
t_0 = 3.0	#1	(9, 9, 9, 5, 8, 10, 10, 9)	12.3622±0.1923	1231
	#2	(7, 10, 4, 4, 6, 10, 7, 6)	12.9098±0.2205	1168

[a] It is denoted as the form of (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) (refer to Table 7.1).
[b] Initial temperature
[c] Each scalar represents the capacity allocated to the corresponding component.

Table 8.2: Summary of the results – throughput optimization with SQG

Experiment	Sample	System configuration[a]	95%CI (units/hr)	Run time(mins)
s_0[b] = 1.0	#1	(10, 10, 6, 5, 9, 10, 7, 8)[c]	12.6508±0.1615	1463
	#2	(8, 10, 6, 5, 7, 10, 7, 8)	12.6170±0.2279	1604
s_0 = 2.0	#1	(6, 8, 5, 4, 7, 8, 5, 9)	12.4015±0.2151	1755
	#2	(8, 10, 10, 4, 8, 10, 8, 9)	12.5179±0.2257	1566
s_0 = 3.0	#1	(9, 9, 9, 6, 8, 8, 9, 9)	12.4452±0.1476	1721
	#2	(9, 10, 8, 9, 10, 8, 10, 7)	12.3773±0.2873	1619

[a]It is denoted as the form of (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) (refer to Table 7.1).
[b]Initial step size
[c]Each scalar represents the capacity allocated to the corresponding component.

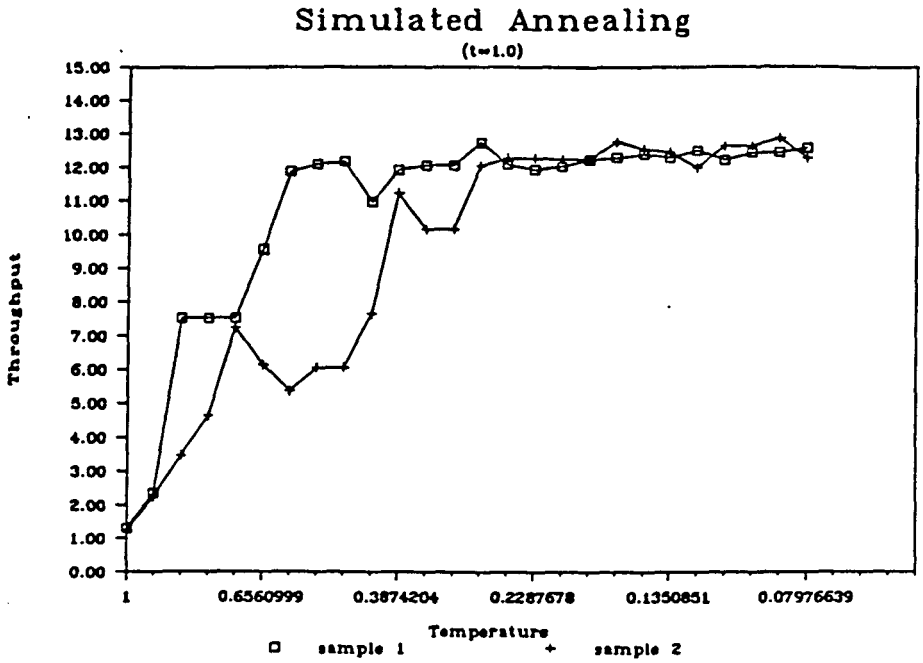

Figure 8.1: Throughput optimization with SA ($t = 1.0$)

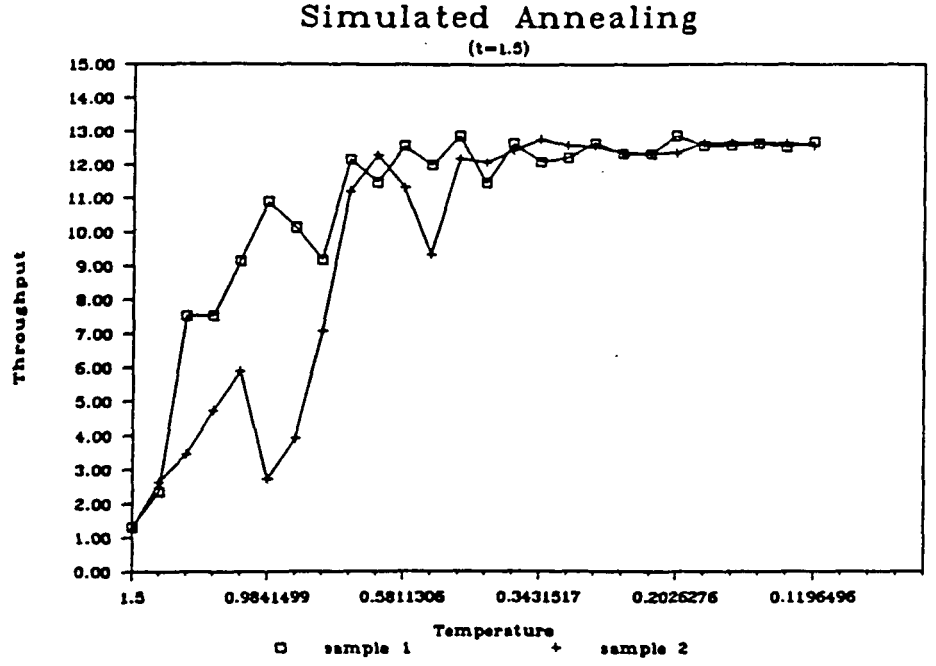

Figure 8.2: Throughput optimization with SA ($t = 1.5$)

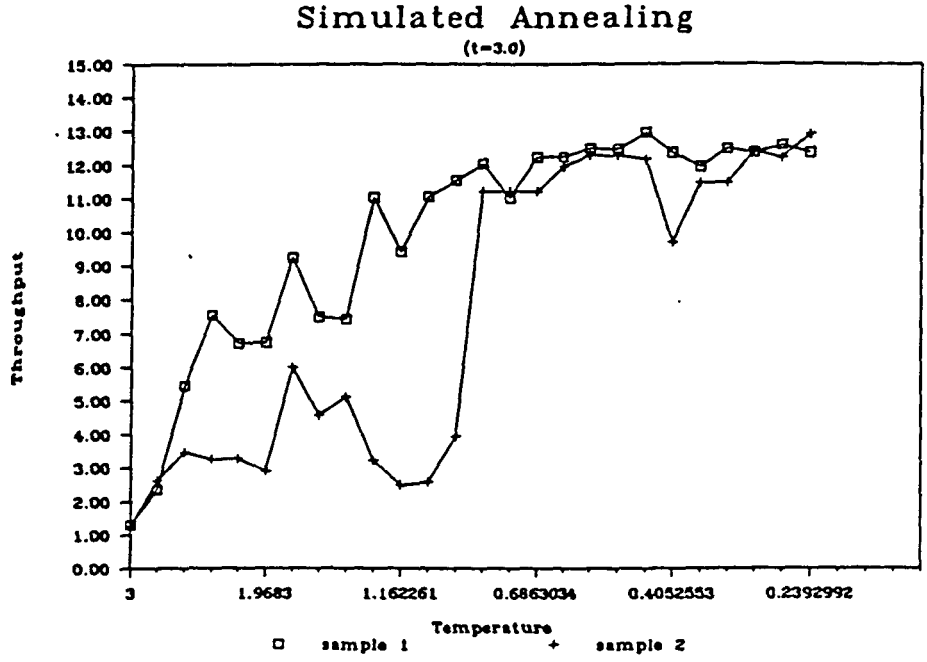

Figure 8.3: Throughput optimization with SA ($t = 3.0$)

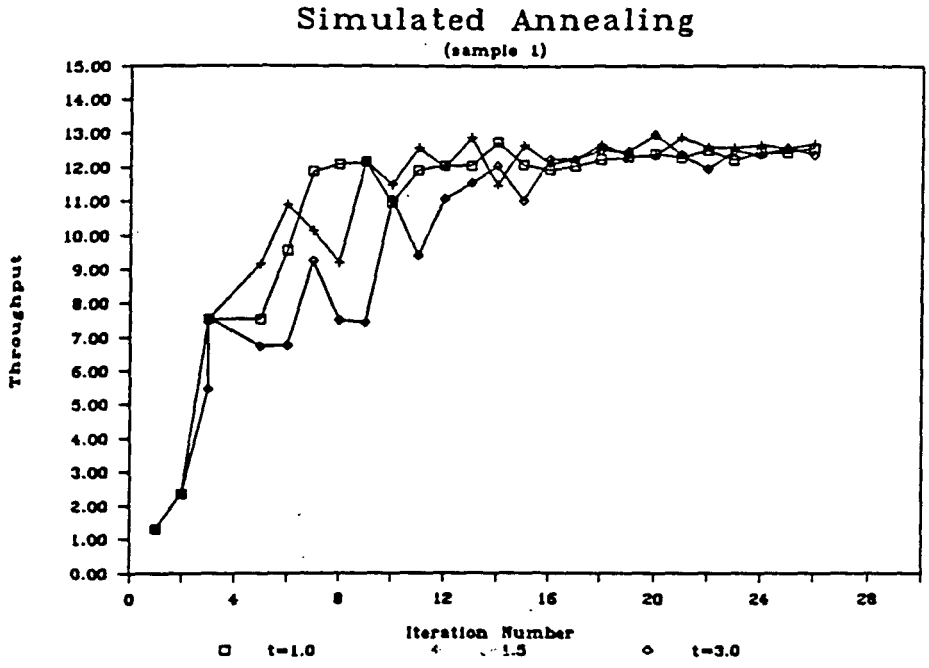

Figure 8.4: Throughput optimization with SA (*sample 1*)

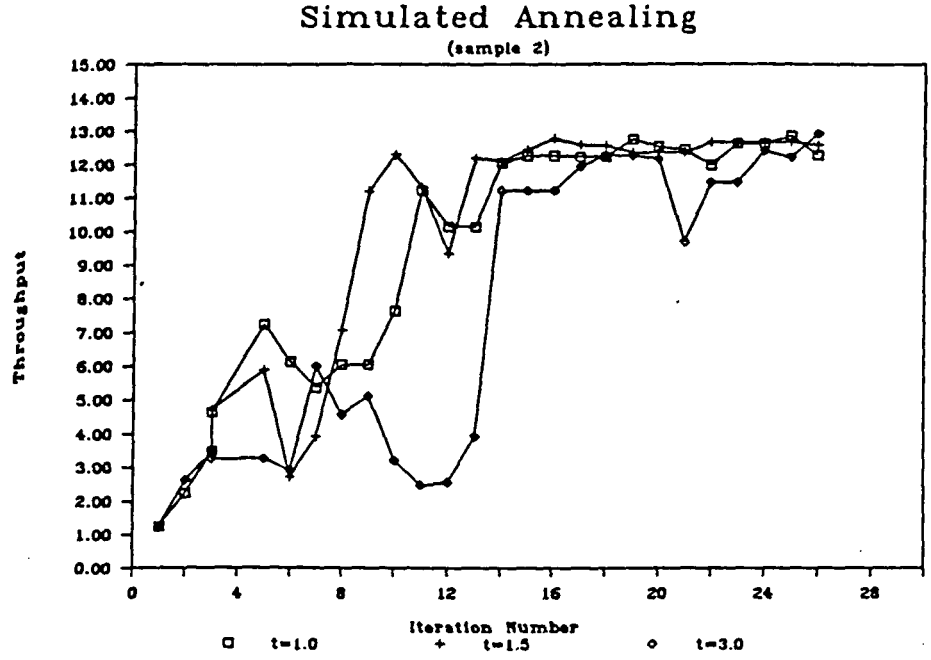

Figure 8.5: Throughput optimization with SA (*sample 2*)

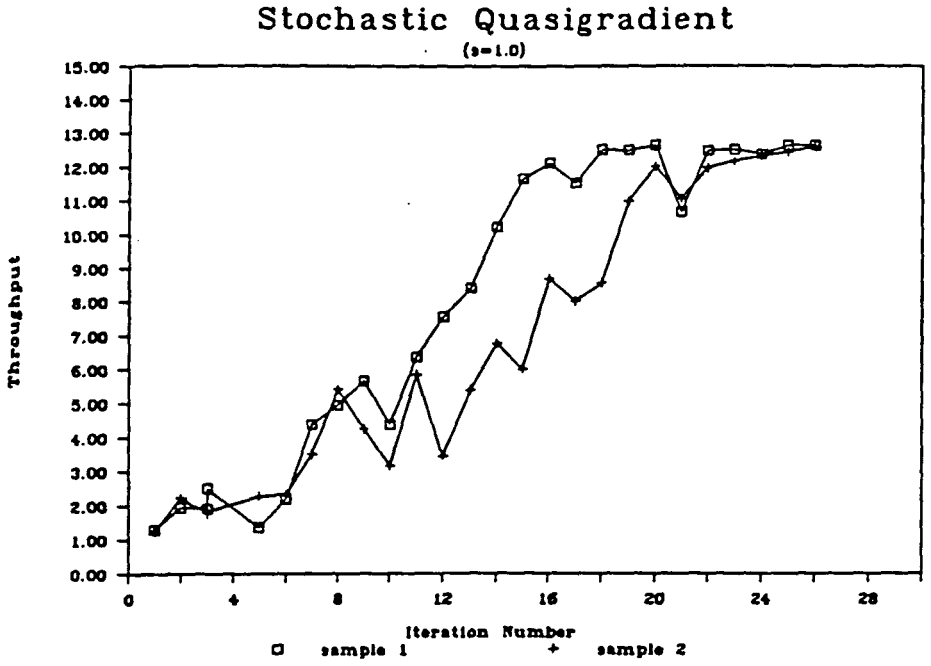

Figure 8.6: Throughput optimization with SQG ($s = 1.0$)

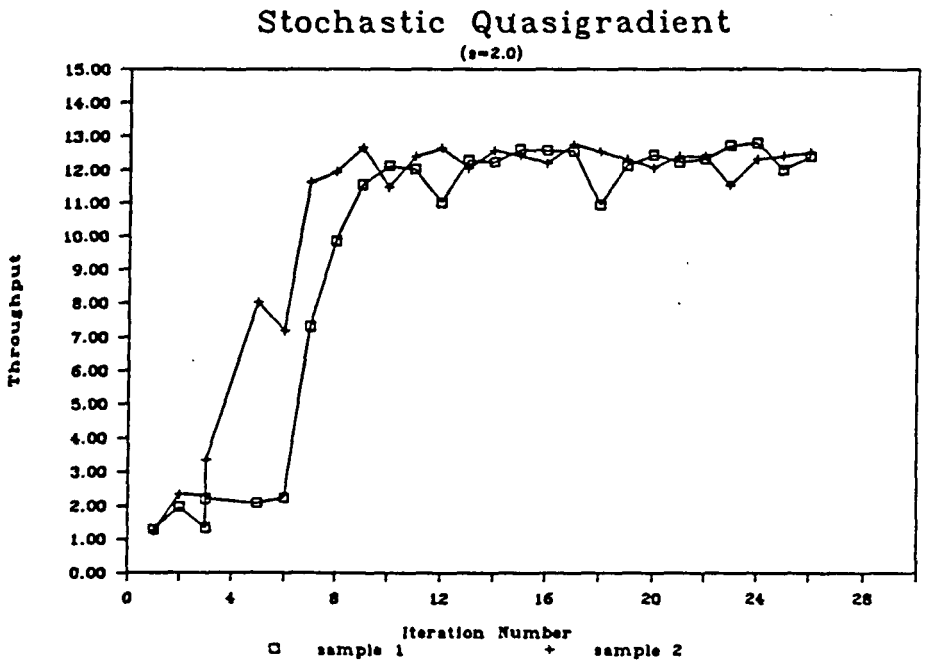

Figure 8.7: Throughput optimization with SQG ($s = 2.0$)

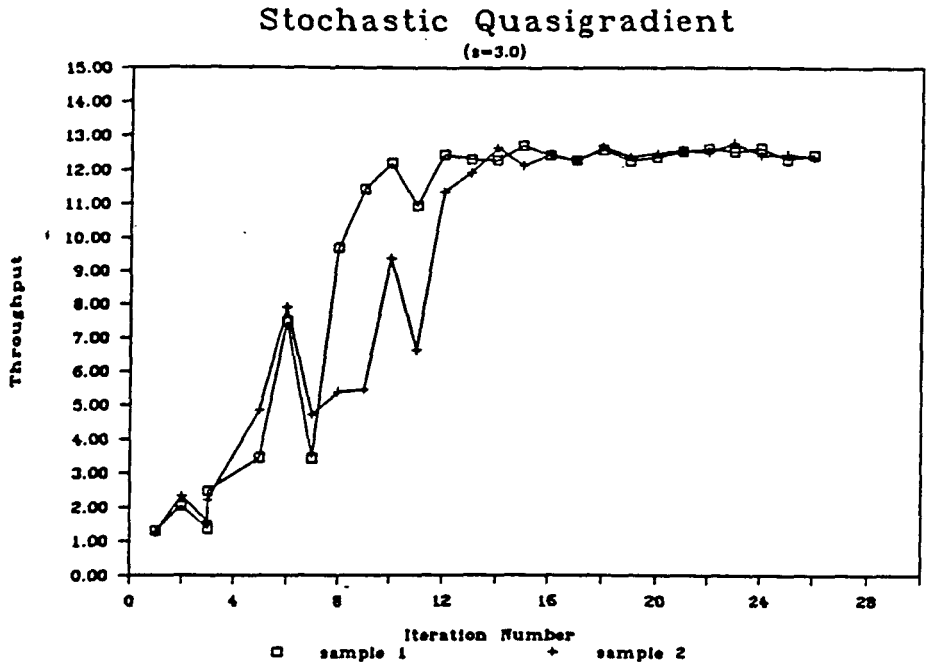

Figure 8.8: Throughput optimization with SQG ($s = 3.0$)

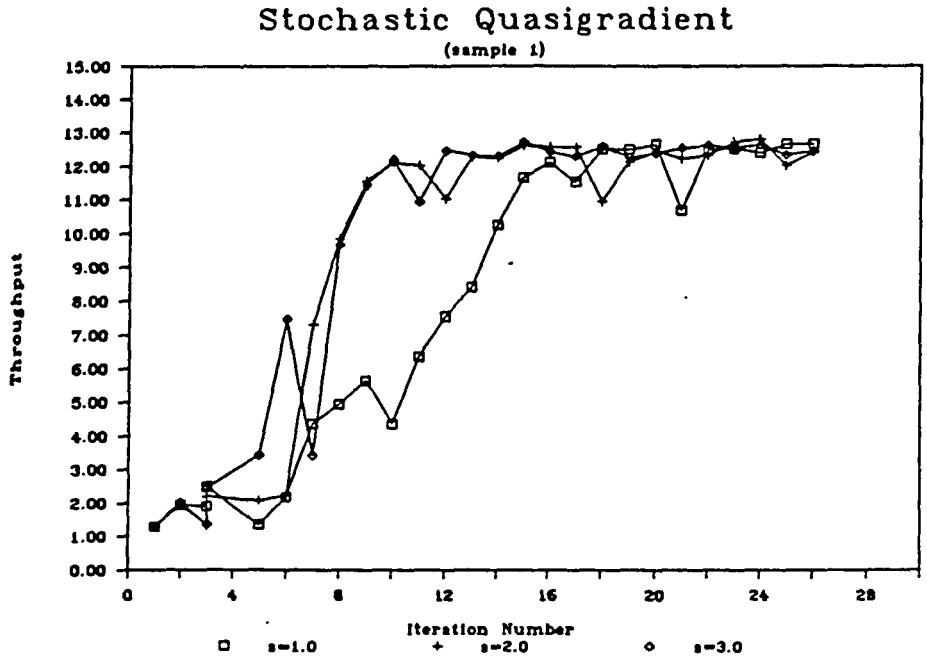

Figure 8.9: Throughput optimization with SQG (*sample 1*)

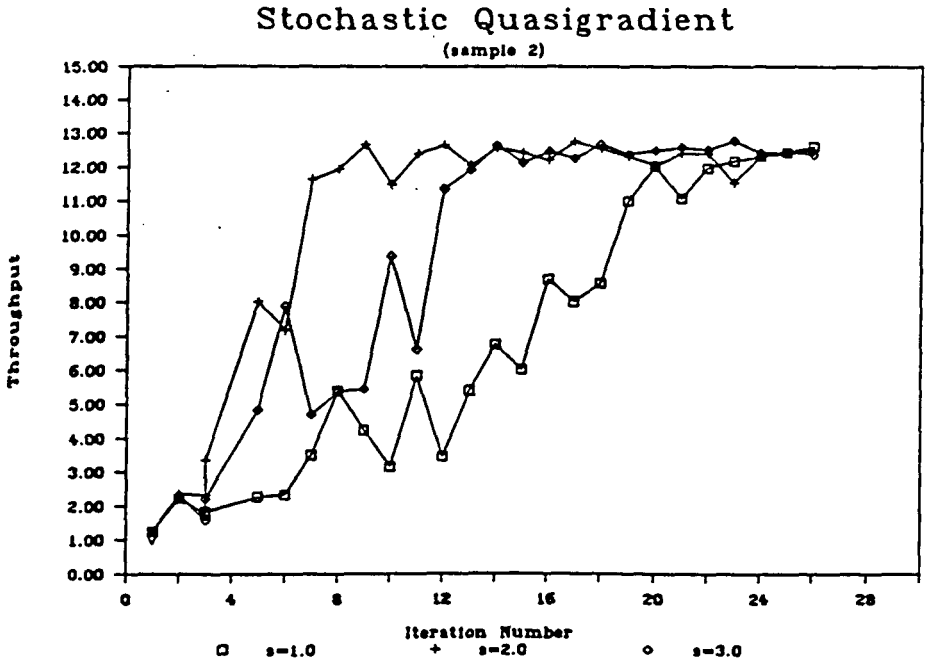

Figure 8.10: Throughput optimization with SQG (*sample 2*)

tial temperature parameters, simulated annealing behaved very well and converged to a good solution very quickly (Figures 8.11 through 8.15).

Cost Optimization with SQG

With the same cost model and system inputs, experiments were performed by using the SQG method with three initial step sizes, $s = 1.0$, $s = 2.0$ and $s = 3.0$. The following table and figures give us a clear view of experiment results in this class.

SQG did not perform as well as simulated annealing in the cost optimization case (Table 8.4). The best solution of these experiments gave us a total cost value of *27.164* (*$/hr*). This value is higher than the worst solution obtained in simulated annealing. Moreover, the variation of the final results is quite large. The total cost values ranged from 27.164 to 52.032. This is an indication that SQG is not very reliable in this case.

As seen in Figures 8.16, 8.17 and 8.18, SQG seemed not to converge well. With larger initial step sizes, $s = 2.0$ and $s = 3.0$, the responses of SQG tended to vary widely during the run. This implies that the initial step size was too large. This phenomenon also matches our claim in chapter 5 – " if the step size was too large, then the responses of SQG tend to move back and forth.".

Therefore, either using a smaller initial step size or using more strict stopping criteria is necessary. However, the smallest step size we can have is *1.0* since resource capacity must be an integer value . As shown in Figure 8.16, when $s = 1.0$ the algorithm seems to converge but to a poor local optimum, approximately *38.0* (*$/hr*).

Table 8.3: Summary of the results – cost optimization with SA

Experiment	Sample	System configuration[a]	95%CI ($/hr)	Run time(mins)
t_0[b] = 3.0	#1	(10, 9, 9, 10, 8, 9, 6, 4)[c]	17.8038±0.3287	1527
	#2	(6, 7, 6, 5, 10, 7, 5, 4)	17.1225±0.2056	1460
t_0 = 4.0	#1	(4, 9, 8, 8, 10, 8, 5, 4)	17.3123±0.2913	1536
	#2	(6, 6, 10, 8, 8, 8, 5, 4)	17.2531±0.1793	1503
t_0 = 5.0	#1	(5, 10, 8, 4, 7, 9, 4, 4)	17.5131±0.3011	1843
	#2	(4, 7, 4, 6, 6, 8, 9, 4)	17.5360±0.2491	1630

[a]It is denoted as the form of (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) (refer to Table 7.1).
[b]Initial temperature
[c]Each scalar represents the capacity allocated to the corresponding component.

Table 8.4: Summary of the results – cost optimization with SQG

Experiment	Sample	System configuration[a]	95%CI ($/hr)	Run time(mins)
s_0[b] = 1.0	#1	(2, 6, 7, 4, 3, 4, 2, 3)[c]	37.5100±0.4038	249
	#2	(2, 4, 4, 3, 4, 4, 3, 2)	38.3647±0.3804	129
s_0 = 2.0	#1	(4, 5, 3, 3, 5, 7, 4, 3)	27.1646±0.4109	525
	#2	(5, 8, 4, 6, 6, 7, 4, 3)	27.4053±0.3407	546
s_0 = 3.0	#1	(4, 5, 2, 5, 4, 6, 4, 2)	37.7470±0.4517	592
	#2	(10, 6, 7, 3, 7, 4, 10, 1)	52.0320±0.3974	102

[a]It is denoted as the form of (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) (refer to Table 7.1).
[b]Initial step size
[c]Each scalar represents the capacity allocated to the corresponding component.

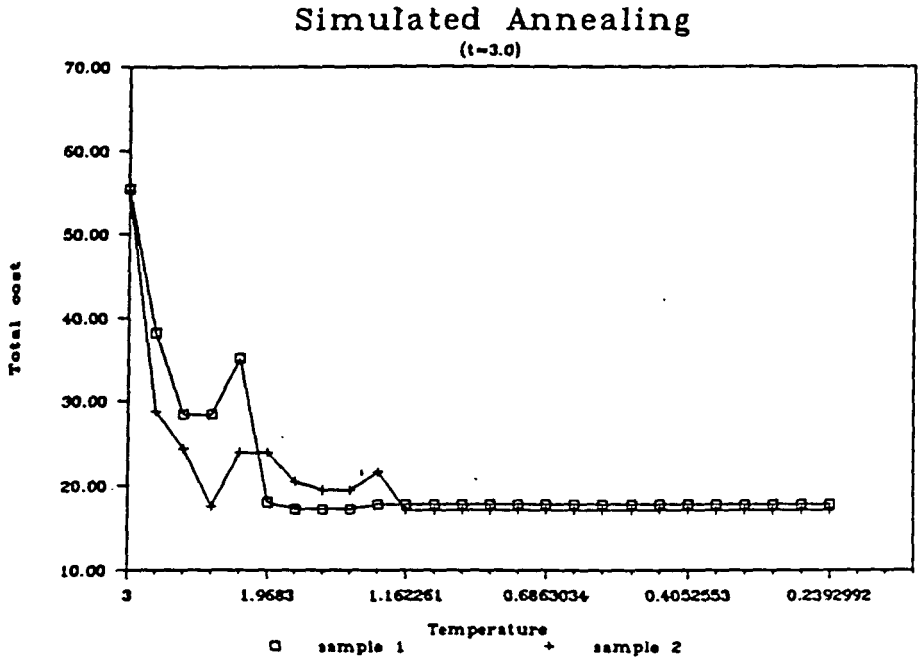

Figure 8.11: Cost optimization with SA ($t = 3.0$)

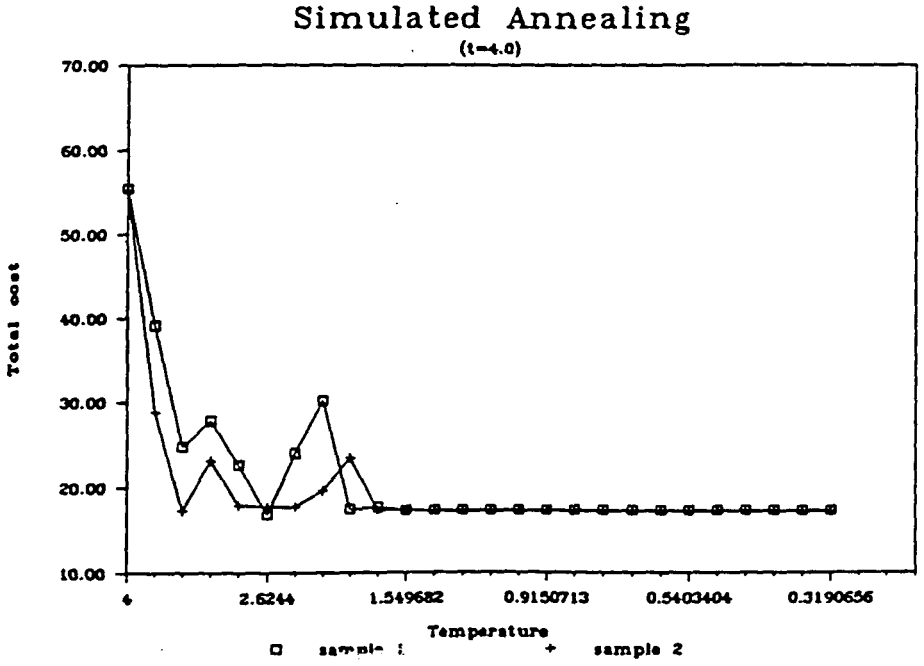

Figure 8.12: Cost optimization with SA ($t = 4.0$)

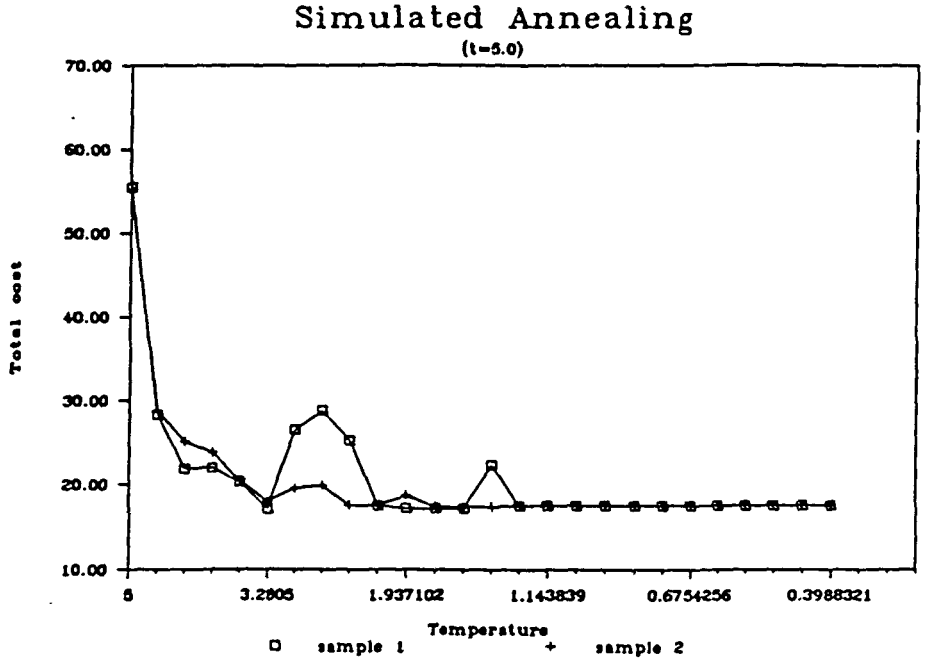

Figure 8.13: Cost optimization with SA ($t = 5.0$)

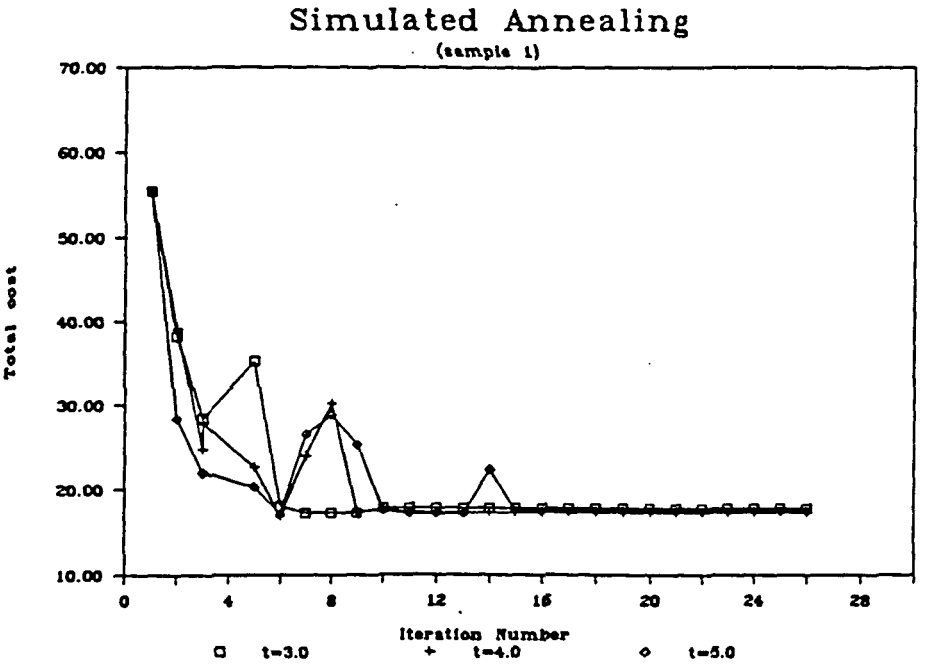

Figure 8.14: Cost optimization with SA (*sample 1*)

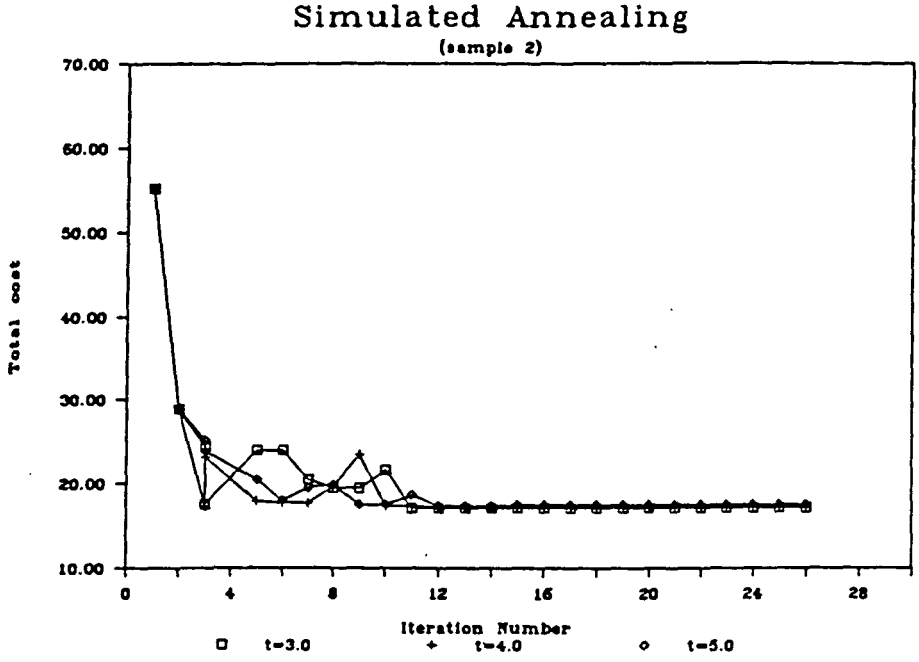

Figure 8.15: Cost optimization with SA (*sample 2*)

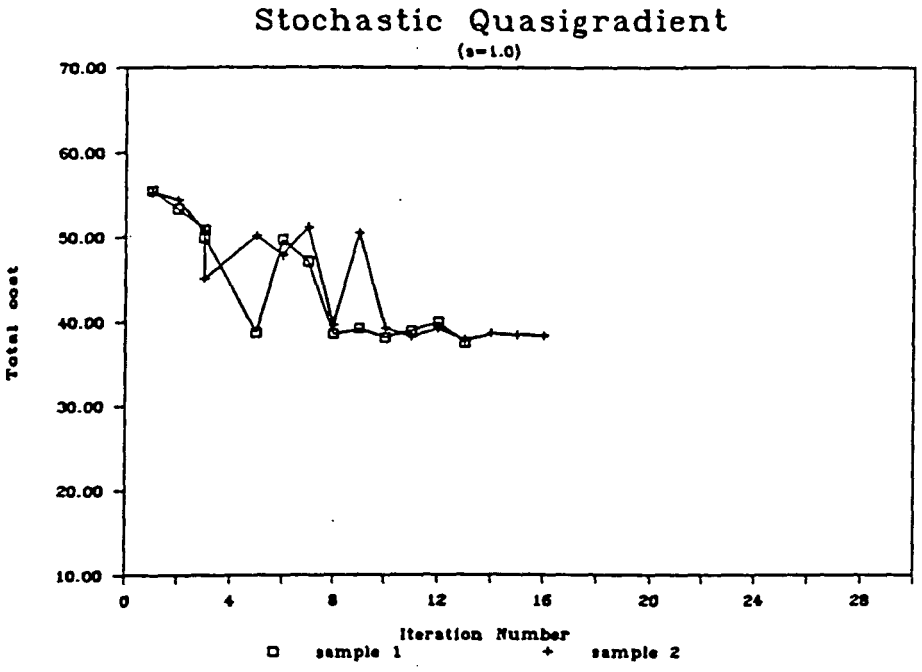

Figure 8.16: Cost optimization with SQG ($s = 1.0$)

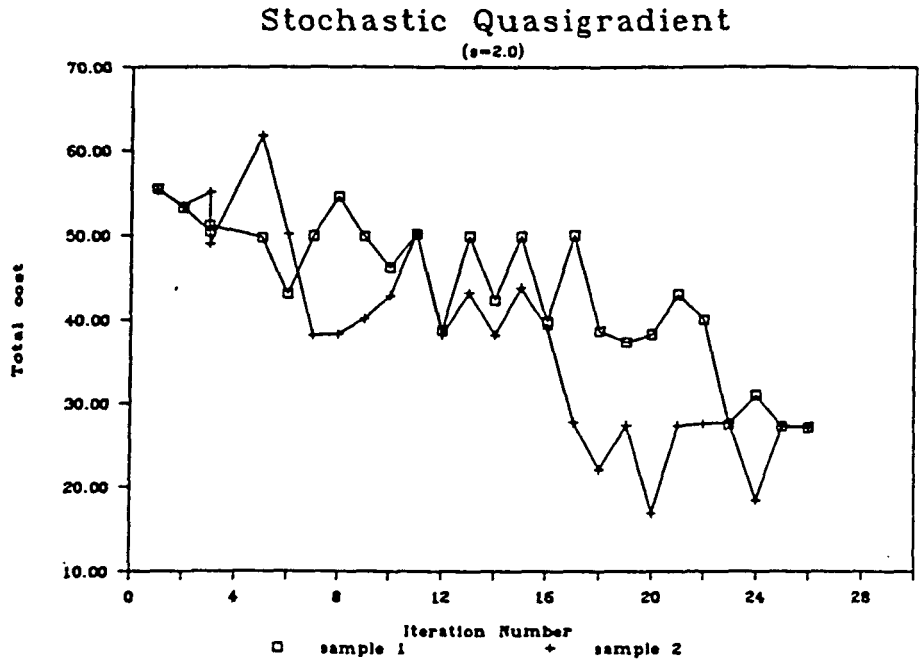

Figure 8.17: Cost optimization with SQG ($s = 2.0$)

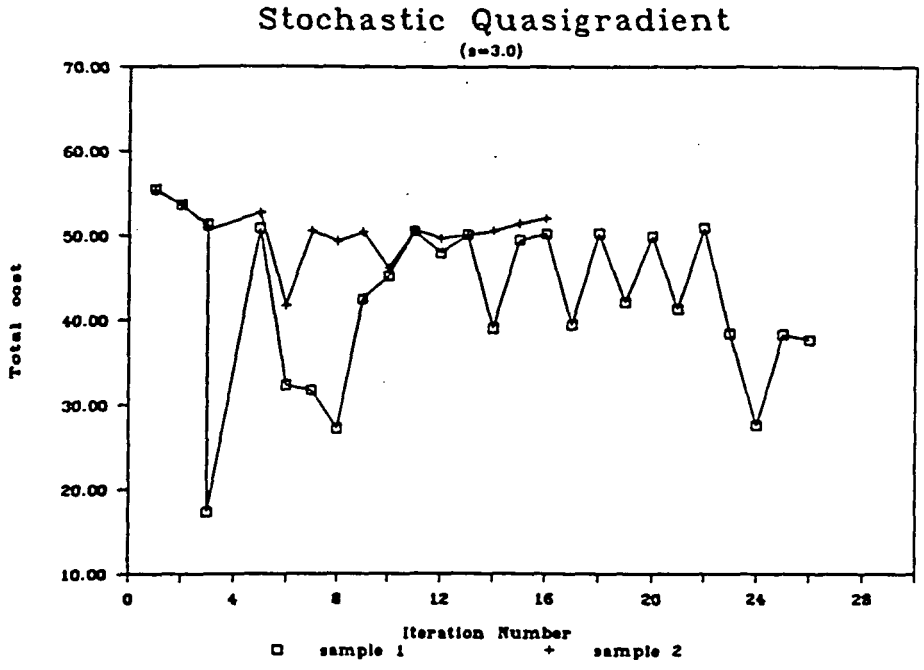

Figure 8.18: Cost optimization with SQG ($s = 3.0$)

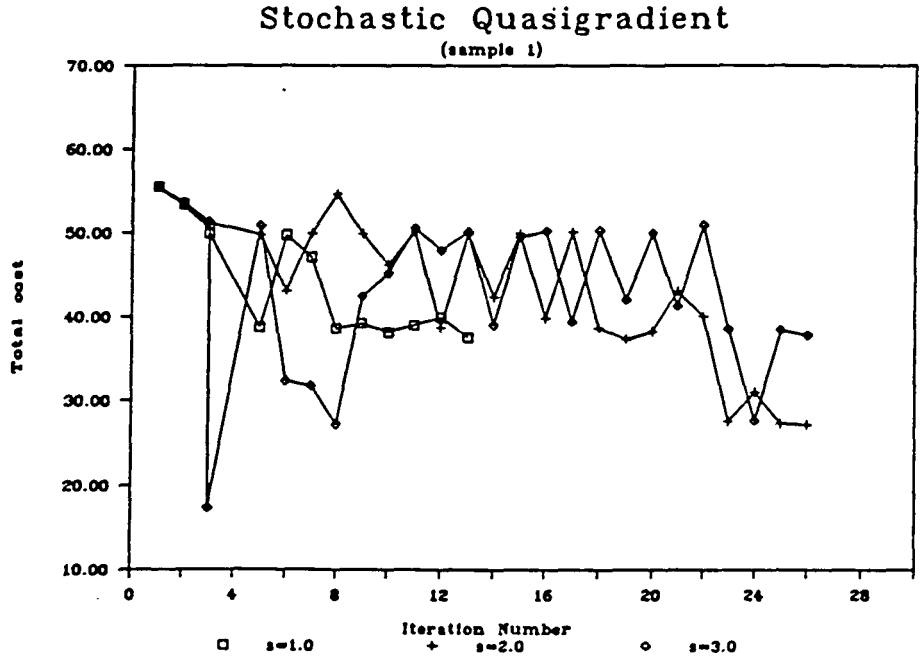

Figure 8.19: Cost optimization with SQG (*sample 1*)

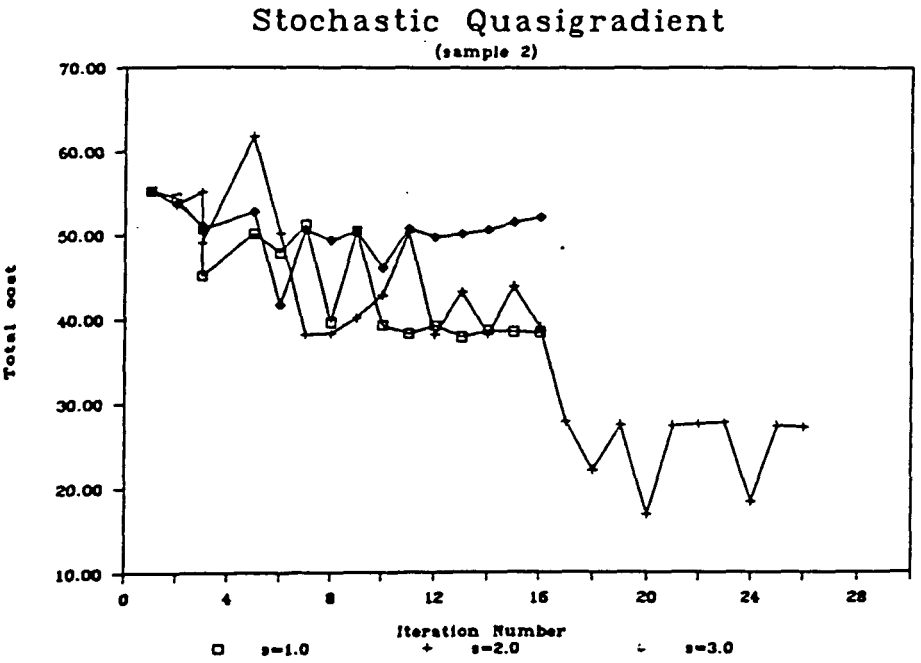

Figure 8.20: Cost optimization with SQG (*sample 2*)

Maximize System Throughput versus Minimize Total Cost

There are several kinds of performance measures used for evaluating the design and control of FMS. System throughput or production rate is the most commonly used one. Researchers and designers are likely to utilize throughput as an objective function while they analyze or design an FMS since it is simple and easy to measure.

However, for any manufacturer to stay in business, it is more likely necessary to minimize the total production cost associated with the system. Consequently, the economic analyses of manufacturing systems became more necessary and important. There are two difficulties with this approach. First, it is not easy to obtain the specific economic parameters for a given optimization problem. Secondly, unlike traditional accounting methods, some cost effects of FMS are difficult to quantify. Therefore, in order to investigate the performance of simulated annealing and SQG on a cost minimization model, the simplified cost model of Chapter 3 was developed.

By using throughput as the performance measure, the algorithms tended to choose larger scale systems. As we expected, the algorithms have overallocated the resource capacities in order to maintain the highest system throughput (see Table 8.1 and 8.2). In the real world, this could cause very low resource utilization. We learned that if there are no proper constraints to bind the resource capacities, the algorithms could reach an unrealistic large system. This is a disadvantage associated with the throughput optimization model.

By introducing total cost as the objective function, this effect can be avoided. As we can see in Table 8.3, the results appeared to be smaller scale systems while maintaining the desired throughput. This is because the cost factors on resources pull down the the number of resource while minimizing the total cost. By referring to

the cost parameters provided in Table 3.5, the more valuable resources have a smaller amount of capacity allocated, as we expected.

In conclusion, selection of an appropriate performance measure is a very important matter before designing or evaluating a particular FMS. It is really dependent upon the type of problem. Different performance measures result in different design alternatives. The tradeoff should be analyzed carefully.

Simulated Annealing versus Stochastic Quasigradient

The application of simulated annealing and SQG methods on FMS design problems was investigated. We have learned they are very different algorithms (Chapter 4, 5 and 6). Simulated annealing was designed for deterministic combinatorial optimization problems and it has been proven that it could converge to the global optimum as the number of iterations approaches infinity; the SQG method was designed for application to nondifferentiable performance functionals with continuous decision variables and the proof of convergence can be found in [ERMO 83]. As a result, a number of heuristic modifications on simulated annealing and SQG are necessary to the model (Chapter 5 and 6).

A series of experiments were performed and a comparison of simulated annealing and SQG was made. The results are summarized in Table 8.5.

In both models, simulated annealing appears to provide the best solutions. The SQG method worked very well in the throughput optimization model, providing consistent results with simulated annealing. However, it was much less effective when applied to the cost optimization model. The increased complexity of the response surface caused the poor performance of SQG.

Table 8.5: A comparison of simulated annealing and SQG

Model	Method	Performance Measure	Ave. Running Time
Max. throughput			
	SA[a]	(12.5922, 0.2652)[b]	(1365.17, 241.8)[c]
	SQG	(12.5016, 0.1136)	(1621.33, 106.0)
Min. cost			
	SA	(17.4234, 0.2438)[d]	(1583.17, 139.6)
	SQG	(36.7039, 9.1314)	(357.17, 222.6)

[a]Simulated annealing algorithm
[b](average production rate, standard deviation)
[c](average running time, standard deviation)
[d](average total cost, standard deviation)

The above analyses only considered resulting objective function values as a measure of each method performance. However, the length of time required to run the particular method is also an important factor. In the system throughput optimization model, the SQG method is clearly the most time consuming, with an average run of 1621.33 minutes on PC 286 (see Table 8.5). The larger run times are due to the modifications of the algorithm in which the step size in the FFD calculation is increased each time there is no statistical significance between two system configurations. This increase continues until the step size is out of the feasible range. This modification of the FFD calculation causes a lot of extra replications to be made in order to find a gradient.

Simulated annealing, however, included a reward process. This process helps the algorithm to reduce the number of unnecessary replications by offering feedback to

the algorithm (see Chapter 5). As a result, the efficiency is increased and simulated annealing needs less time to perform than SQG.

In contrast, in the cost optimization model simulated annealing seems to need more time to run than SQG. This is caused by an increase in the run length at each temperature needed to reach the equilibrium state.

Summary and Conclusions

We have demonstrated the application and comparison of modified simulated annealing and SQG upon FMS design. System throughput and total cost models were constructed as test platforms. The method that appears to be the *best* for our application in terms of time complexity and solution quality is simulated annealing.

In the throughput optimization model, the SQG method performed as well as simulated annealing but less effective in the cost model since the response surface of the cost model is more complicated than the throughput model.

In the study, we are involved with a discrete parameter Monte Carlo optimization problem. Several modifications to both methods are required since neither is explicitly designed for this case. However, the modified versions of simulated annealing and SQG appear to be reasonable and we are optimistic about the capability of either for analyzing optimization problems for certain complex manufacturing systems.

Due to intricate interrelations among FMS components, a unique optimal system configuration is not known. As we observed, several good system configurations were obtained using each optimization method. More strict decision criteria and analyses are necessary to make a final design.

As we expected, using system throughput as the objective function, the solutions

tend to be larger systems. This tendency can be prevented by using the total cost of the system as the performance measure.

Using different seeds did not seem to affect the ability of convergence for either method. However, it does tend to have more effects on the response of simulated annealing rather than the SQG method. This is because randomness is utilized in the neighborhood function and Metropolis procedure of simulated annealing.

CHAPTER 9. CONCLUSIONS

Flexible manufacturing systems are increasingly popular in many areas of manufacturing. Installation of an FMS normally requires a substantial investment, and thus careful analysis is necessary before making a definite commitment. In the previous chapters, a simulation model and the application of two types of stochastic optimization methods to the FMS design problem were presented: modified simulated annealing and stochastic quasigradient. In addition, two types of performance measures were examined. Conclusions regarding each of these areas are addressed as follows.

FMS generally consist of many interrelating components and thus it is rather difficult to estimate their performance. Discrete event simulation provides a perfect tool to study and analyze FMS, especially in the design phase. The system is analyzed and modeled, and a computer program is written which describes the performance of the system as a function of one or more parameters of the design.

If the design performance is sufficiently quantitative such that this performance can be represented by a real-valued function of the simulation results, then the design process reduces to the problem of refining the values of the design parameters to yield the best simulated system performance. Instead of using a trial and error process, direct search methods of optimization to the problem are employed: specifically,

simulated annealing and SQG. The advantages are:

1. A systematic approach is taken to the optimization of the design parameters.

2. A computer algorithm generally runs more efficiently in terms of both time and resources.

However, the suitability of application of simulated annealing and SQG is really dependent on the type of problem. A number of heuristic modifications on both methods are necessary, and due to the stochastic nature of simulation, statistical analysis procedures can not be excluded while evaluating the system performance.

We have performed several experiments regarding the different types of optimization algorithms and performance measures. Experimental results show that by integrating the simulation model with simulated annealing or SQG methods, we are able to design a good FMS. Our method can yield *good* results in acceptable computation time. However, the performance of each method is sensitive to how the user defines the algorithm's operating parameters. Many pilot runs have been done in order to obtain *good* operating parameters for each of the algorithms.

In addition, when using system throughput as the objective function, we may design as large a system as possible in order to achieve the highest throughput. It is probably not necessary to have the highest throughput as long as the system can meet the production plan, because it could be too costly and cause a lower utilization of the system. The tradeoff among using different types of performance measures should be carefully analyzed.

SQG was designed for application to problems in which the decision variables are continuous. Among the disadvantages of SQG are the following problems:

1. Convergence to the global optimum is not guaranteed, but rather to a local optimum.

2. The final solution is dependent upon the starting point chosen.

3. The selection of a good step size sequence is difficult.

4. If the decision variables are discrete, it typically does not perform as well.

Ermoliev and Gaivoronski [ERMO 83] suggested that the *best* procedure for application of SQG is an interactive approach. However, in most cases the interactive procedure is impractical due to the time required to perform as the problem increases in complexity.

Simulated annealing was designed for application to deterministic problems in which the decision variables are discrete. Some disadvantages in its application are:

1. The process is very slow to converge.

2. Proofs of convergence have been given for deterministic functions. However, convergence has not been proven for a stochastic function.

3. A good stopping criteria and cooling schedule have not been defined.

In our study, simulated annealing has some advantages over the SQG method. These are:

1. The starting point chosen for simulated annealing has less effect on the final solution.

2. Decision variables involved are discrete. Simulated annealing is designed specifically for this situation.

3. Simulated annealing appears to give the most consistent results in terms of providing the best solution.

4. The SQG method becomes less effective as the complexity of the response surface increases.

Although each of the methods presented has advantages and disadvantages when applied to the FMS design problem, simulated annealing appears to be the best method in terms of consistency of results within reasonable computation time. We conjecture that simulated annealing has the potential to provide *good* solutions of not only deterministic but also stochastic optimization problems, and even though SQG does not perform as well as does simulated annealing, it still appears to be a reasonable technique for analyzing certain manufacturing system optimization problems.

Finally, there still remain unknown domains of application of simulated annealing and SQG to the stochastic optimization problems. Opportunities for further research might be directed in the following areas:

1. Inclusion of experimental design techniques in the SQG methods to improve the calculation of the gradients. Originally the estimation of the gradients are performed one variable at a time.

2. The convergence characteristics of simulated annealing when applied to optimization of stochastic systems.

3. The comparison to other types of optimization methods such as Response Surface Methodologies or Genetic Algorithms.

Additionally, exploring the application of Monte Carlo optimization techniques into other areas is also needed.

BIBLIOGRAPHY

[ARBE 84] Arbel, A. and Seidmann, A. 1984. Performance Evaluation of Flexible Manufacturing Systems. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-14(4):606-617.

[BULG 88] Bulgak, A.A. and Sanders, J.L. 1988. Integrating a Modified Simulated Annealing Algorithm with a Simulation of a Manufacturing System to Optimize Buffer Size in Automatic Assembly Systems. *Proceedings of the 1988 Winter Simulation Conference.* San Diego, California, IEEE.

[BUZA 80] Buzacott, J.A. and Shanthikumar, J.G. 1980. Models for Understanding Flexible Manufacturing Systems. *AIIE Transactions* 12(4):339-350.

[BUZA 86] Buzacott, J.A. and Yao, D.D. 1986. Flexible Manufacturing Systems: A Review of Analytical Models. *Management Science* 32(7):890-905.

[CHEN 85] Cheng, T.C.E. 1985. Simulation of Flexible Manufacturing Systems. *Simulation* 45(6):299-302.

[CHOO 86] Choobineh, F. 1986. Economic Justification of Flexible Manufacturing Systems. *1986 International Computers in Engineering Conference and Exhibition.* Chicago, Illinois.

[ERMO 83] Ermoliev, Y.M. 1983. Stochastic Quasigradient Methods and Their Application to System Optimization. *Stochastic* 9:1-36.

[ERMO 84] Ermoliev, Y.M. and Gaivoronski, A. 1984. Stochastic Quasigradient Methods and Their Implementation. Working Paper, WP-84-55, IIASA, Laxenburg, Austria.

[EVAN 89] Evans, G.W., and Brown, P.A. 1989. A Multiobjective Approach to the Design of Flexible Manufacturing Systems. *Proceedings of 1989 International Industrial Engineering Conference and Societies' Manufacturing and Productivity Symposium.* Toronto, Ontario, Canada, IIE.

[FUCH 88] Fuchs, J.H. 1988. *The Handbook of Advanced Manufacturing Methods.* Prentice-Hall, Inc., Englewood Cliffs, NJ.

[GEMM 88] Gemmill, D.D. and Sanders, J. L. 1988. Optimization Approaches to the Portfolio Problem. *Ph. D. Thesis.* University of Wisconsin, Madison.

[GLYN 86a] Glynn, P.W. 1986. Stochastic Approximation for Monte Carlo Optimization. *Proceedings of the 1986 Winter Simulation Conference.* Washington, D.C., IEEE.

[GLYN 86b] Glynn, P.W. 1986. Optimization of Stochastic Systems. *Proceedings of the 1986 Winter Simulation Conference.* Washington, D.C., IEEE.

[GOLD 86] Golden, B.L. and Skiscim, C.C. 1986. Using Simulated Annealing to Solve Routing and Location Problems. *Naval Research Logistics Quarterly* 33:261-279.

[HAID 83] Haider, S.W. and Blank, L.T. 1983. A Role of Computer Simulation in the Economic Analysis of Manufacturing Systems. *Proceedings of the 1983 Winter Simulation Conference.* Arlington, Virginia, IEEE.

[HAJE 86] Hajek, B. 1986. Cooling Schedules for Optimal Annealing. *Mathematic of Operations Research* 13(2):311-329.

[HO 79] Ho, Y.C., Eyler, M.A. and Chien, T.T. 1979. A Gradient Technique for General Buffer Storage Design in Production Line. *International Journals of Production Research* 17(6):557-580.

[HUTC 79] Hutchinson, G.K. 1979. Flexible Manufacturing Systems in the United States. *Automation Manufacturing*, 743-749.

[KIRK 83] Kirkpatrick, S., Gelett, C.D. and Vecchi, M.P. 1983. Optimization by Simulated Annealing. *Science* 220(4598):621-630.

[KIRK 84] Kirkpatrick, S. 1984. Optimization by Simulated Annealing: Quantitative Studies. *Journal of Statistical Physics* 34(5):975-986.

[KLAH 81] Klahorst, H.T. 1981. Flexible Manufacturing Systems: Combining Elements to Lower Costs, Add Flexibility. *Industrial Engineering* 13(11): 112-117.

[KUMA 86] Kumar, K.R. and Vannelli, A. 1986. Design of Flexible Production Systems: Capacity Balancing and Subcontracting Strategy. *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications.* Ann Arbor, Michigan, ORSA.

[KUSI 86] Kusiak, A. 1986. Design of Flexible Manufacturing Systems. *CASA/SME Flexible Manufacturing Systems '86 Conference.* Detroit, Michigan, CASA.

[LAAR 92] Laarhoven, P., Arrts, E. and Lenstra, J.K. 1992. Job Shop Scheduling by Simulated Annealing. *Operations Research* 40(1):113-125.

[LAW 91] Law, A.M. and Kelton, W.D. 1991. *Simulation Modeling and Analysis.* McGraw-Hill, Inc., New York.

[LEE 91] Lee, Y.H. and Iwata, K. 1991. Part Ordering through Simulation-Optimization in an FMS. *International Journals of Production Research* 29(7):1309-1323.

[LIU 88] Liu, C.M. and Sanders, J.L. 1988. Stochastic Design Optimization of Asynchronous Flexible Assembly Systems. *Annals of Operations Research* 15:131-154.

[MEKE 87] Meketon, M.S. 1987. Optimization in Simulation: A Survey of Recent Results. *Proceedings of the 1987 Winter Simulation Conference.* Atlanta, Georgia, IEEE.

[MITR 86] Mitra, D., Romeo, F. and Sangiovanni A. 1986. Convergence and Finite-Time Behavior of Simulated Annealing. *Advanced Applied Probability* 18:747-771.

[MORI 91] Morito, S., Takano, T., Mizukawa, H. and Mizoguchi, K. 1991. Design and Analysis of Flexible Manufacturing System with Simulation: Effects of Flexibility on FMS Performance. *Proceedings of the 1991 Winter Simulation Conference.* Phoenix, Arizona, IEEE.

[PEGD 90] Pegden, C.D., Shannon, R.E. and Sadowski, R.P. 1990. *Introducing to Simulation Using SIMAN.* McGraw-Hill, Inc., New York.

[RANK 83] Ranky, P. 1983. *The Design and Operation of Flexible Manufacturing Systems.* North-Holland, Inc., New York.

[SCHR 85] Schriber, T.J. 1985. A GPSS/H Model for a Hypothetical Flexible Manufacturing System. *Annals of Operations Research* 3:171-188.

[SHAN 84] Shannon, R.E. 1984. Artificial Intelligence and Simulation, Key-note Address. *Proceedings of the 1984 Winter Simulation Conference.* Dallas, Texas, IEEE.

[STEC 85] Stecke, K.E. 1985. Design, Planning, Scheduling, and Control Problems of Flexible Manufacturing Systems. *Annals of Operations Research* 3:3-12.

[STEC 86] Stecke, K.E. and Kim, I. 1986. A Flexible Approach to Implement the Short-Term FMS Planning Function. *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications.* Ann Arbor, Michigan, IEEE.

[STOC 91] Stockman, B., Evans, G. and Mollaghasemi, M. 1991. Comparison of Global Search Methods for Design Optimization Using Simulation. *Proceedings of the 1991 Winter Simulation Conference.* Phoenix, Arizona, IEEE.

[SURI 85] Suri, R. 1985. An Overview of Evaluative Models for Flexible Manufacturing System. *Annals of Operations Research* 3:13-21.

[SURI 87] Suri, R. and Leung, Y.T. 1987. Single Run Optimization of a SIMAN Model for Closed Loop Flexible Assembly Systems. *Proceedings of the 1987 Winter Simulation Conference.* Atlanta, Georgia, IEEE.

[TAND 91] Tandiono, E. and Gemmill, D.D. 1991. Stochastic Optimization of Cost of Automatic Assembly System. *M.S. Thesis.* Iowa State University.

[WILH 87] Wilhelm, M.R. and Ward, T.L. 1987. Solving Quadratic Assignment Problems by 'Simulated Annealing'. *IIE Transactions* 19(1):107-119.

[WHIT 85] Whitney, C.K. and Suri, R. 1985. Algorithms for Part and Machine Selection in Flexible Manufacturing Systems. *Annals of Operations Research* 3:239-261.

APPENDIX Modified Simulated Annealing

```
C
C (* Optimizing the system throughput with Simulated Annealing *)
C
      SUBROUTINE EVENT(I)
C
      GOTO (1,2,3),I
1     CALL NEXT_CONFIGURE
      RETURN
2     CALL ANNEALING
      RETURN
3     CALL EVALUATE
      RETURN
      END
C*
C*.....................................................
C*
      SUBROUTINE NEXT_CONFIGURE
      COMMON/A001/Kth_RUN,PROD_RATE(12)/A003/BATCH_NO,
     +MAX_ITERATION,TEMPERATURE,FACILITY(9),PRE_RESOURCE(9),UPDATE
C
      INTEGER FACILITY,UPDATE,PRE_RESOURCE
      REAL RDOM1
C
      PRINT *,'Kth-RUN # = ',Kth_RUN
      IF (Kth_RUN .EQ. 0) CALL INPUT
      IF (UPDATE .EQ. 1) THEN
         DO 15 INDEX=1,8
            RDOM1=RAND(9)
            IF (RDOM1 .LT. 0.5) THEN
               FACILITY(INDEX)=PRE_RESOURCE(INDEX)-1
            ELSE
```

```
               FACILITY(INDEX)=PRE_RESOURCE(INDEX)+1
             ENDIF
             IF (FACILITY(INDEX) .GE. 10) FACILITY(INDEX)=10
            IF (FACILITY(INDEX) .LE. 1) FACILITY(INDEX)=1
15         CONTINUE
             DO 16 INDEX=1,8
                CALL SETMR(INDEX,FACILITY(INDEX))
                PRINT *,FACILITY(INDEX)
16           CONTINUE
      ENDIF
      RETURN
      END
C*
C*.....................................................
C*
      SUBROUTINE ANNEALING
      COMMON/A001/Kth_RUN,PROD_RATE(12)/A002/NEW_MEAN,OPTI_MEAN,
     +OPTI_Var,NEW_Var/A003/BATCH_NO,MAX_ITERATION,TEMPERATURE,
     +FACILITY(9),PRE_RESOURCE(9),UPDATE
C
      DIMENSION XRESOURCE(8)
      INTEGER SIGNIFICANT,UPDATE,FACILITY,PRE_RESOURCE,BATCH_NO,
     +        EQ1_STAGE,EQ2_STAGE,EQ3_STAGE,XCHANGE,XRESOURCE
      REAL OPTI_MEAN,NEW_MEAN,U,Y,T_i,CHECK_POINT,OPTI_Var,
     +     NEW_Var,TEMPERATURE,XVAR,XMEAN
C
      IF (Kth_RUN .EQ. 0) THEN
         UPDATE=1
         Kth_RUN=Kth_RUN+1
         RETURN
      ENDIF
C
      IF (SIGNIFICANT(OPTI_MEAN,NEW_MEAN,OPTI_Var,NEW_Var) .EQ. 0)
     +   THEN
C
         IF (XCHANGE .EQ. 0) THEN
            DO 25 INDEX=1,8
                  XRESOURCE(INDEX)=PRE_RESOURCE(INDEX)
25          CONTINUE
            XMEAN=OPTI_MEAN
            XVAR=OPTI_VAR
C   (save S1)
```

```
            OPTI_MEAN=NEW_MEAN
            OPTI_VAR=NEW_VAR
            CALL STORING
C   (exchange S2 by S1)
            XCHANGE=1
            RETURN
         ELSE
             IF ((SIGNIFICANT(XMEAN,NEW_MEAN,XVar,NEW_Var) .EQ. 0)
     +           .OR. (NEW_MEAN .GT. XMEAN)) THEN
                CALL OUTPT2(Kth_RUN,OPTI_MEAN)
                DO 30 INDEX=1,8
                      XRESOURCE(INDEX)=PRE_RESOURCE(INDEX)
30              CONTINUE
                XMEAN=OPTI_MEAN
                XVAR=OPTI_VAR
                OPTI_MEAN=NEW_MEAN
                OPTI_VAR=NEW_VAR
                CALL STORING
                EQ2_STAGE=EQ2_STATGE+1
                RETURN
             ELSE
                DO 28 INDEX=1,8
                      FACILITY(INDEX)=PRE_RESOURCE(INDEX)
                      PRE_RESOURCE(INDEX)=XRESOURCE(INDEX)
28              CONTINUE
                NEW_MEAN=OPTI_MEAN
                NEW_VAR=OPTI_VAR
                OPTI_MEAN=XMEAN
                OPTI_VAR=XVAR
                GOTO 100
             ENDIF
         ENDIF
C
C
      ELSE
          IF (XCHANGE .EQ. 1) THEN
             CALL OUTPT2(Kth_RUN,OPTI_MEAN)
             EQ2_STAGE=EQ2_STAGE+1
          ENDIF
      ENDIF
C
C
```

```
100   EQ2_STAGE=EQ2_STATGE+1
      XCHANGE=0
      IF (NEW_MEAN .LE. OPTI_MEAN) THEN
         U=RAND(6)
         Y=ABS(NEW_MEAN-OPTI_MEAN)
         CHECK_POINT=EXP(-(Y/TEMPERATURE))
         PRINT *,'TEMPERATURE=',TEMPERATURE
         PRINT *,'(U,CHECK_POINT) =',U,CHECK_POINT
         IF (U .LT. CHECK_POINT) THEN
C        (* acceptance *)
            OPTI_MEAN=NEW_MEAN
            OPTI_Var=NEW_Var
            CALL STORING
            CALL OUTPT2(Kth_RUN,OPTI_MEAN)
            EQ1_STAGE=0
         ELSE
C        (* rejection *)
            EQ1_STAGE=EQ1_STAGE+1
         ENDIF
      ELSE
C        (* acceptance *)
            OPTI_MEAN=NEW_MEAN
            OPTI_Var=NEW_Var
            CALL STORING
            CALL OUTPT2(Kth_RUN,OPTI_MEAN)
            EQ1_STAGE=0
      ENDIF
      IF ((EQ1_STAGE .GE. 5) .OR. (EQ2_STAGE .GE. 10)) THEN
          CALL INTERVAL(Kth_RUN,OPTI_MEAN,OPTI_Var,BATCH_NO,1)
          Kth_RUN=Kth_RUN+1
          TEMPERATURE = TEMPERATURE * 0.90
          EQ1_STAGE=0
          EQ2_STAGE=0
          UPDATE=1
      ENDIF
      IF (Kth_RUN .EQ. (MAX_ITERATION+1)) THEN
          CALL INTERVAL(Kth_RUN,OPTI_MEAN,OPTI_Var,BATCH_NO,3)
          CLOSE(70)
          CLOSE(71)
          CLOSE(72)
          STOP
      ELSE
```

```
          UPDATE=1
          RETURN
        ENDIF
        END
C*
C*.....................................................
C*
        SUBROUTINE INTERVAL(SRUN,MU,XDEVIATION,BATCH,SIGN)
C
        INTEGER BATCH,SIGN,SRUN
        REAL MU,XDEVIATION,LOW_BOUND,UP_BOUND
C
        UP_BOUND=MU+1.282*SQRT(XDEVIATION)/SQRT(REAL(BATCH))
        LOW_BOUND=MU-1.282*SQRT(XDEVIATION)/SQRT(REAL(BATCH))
C
        IF (SIGN .EQ. 1) THEN
           CALL OUTPT1(SRUN,MU,LOW_BOUND,UP_BOUND)
        ELSEIF (SIGN .EQ. 2) THEN
           CALL OUTPT5(SRUN,MU,LOW_BOUND,UP_BOUND)
        ELSEIF (SIGN .EQ. 3) THEN
           CALL OUTPT4(UP_BOUND,MU,LOW_BOUND)
        ELSE
           CALL OUTPT1(SRUN,MU,LOW_BOUND,UP_BOUND)
           CALL OUTPT5(SRUN,MU,LOW_BOUND,UP_BOUND)
        ENDIF
        END
C*
C*.....................................................
C*
        SUBROUTINE EVALUATE
        COMMON/SIM/D(50),DL(50),S(50),SL(50),X(50),DTNOW,TNOW,
       +           TFIN,J,NUMREP
        COMMON/A001/Kth_RUN,PROD_RATE(12)/A002/NEW_MEAN,OPTI_MEAN,
       +OPTI_Var,NEW_Var/A003/BATCH_NO,MAX_ITERATION,TEMPERATURE,
       +FACILITY(9),PRE_RESOURCE(9),UPDATE
C
        INTEGER COUNTING,BATCH_NO,FACILITY,PRE_RESOURCE,Kth_RUN,
       +        XNUM_1,XNUM_2
        REAL PROD_RATE,TOTAL_RATE,NEW_MEAN,OPTI_MEAN,NEW_Var,
       +     CYC_TIME,XTAL_1,XTAL_2
C
        IF (COUNTING .EQ. 0) THEN
```

```
         XTAL_1 = TAVG(1) * TNUM(1)
         XNUM_1 = TNUM(1)
         COUNTING=COUNTING+1
         RETURN
      ENDIF
C
      XTAL_2 = TAVG(1) * TNUM(1)
      XNUM_2 = TNUM(1)
      PRINT *,'(XNUM_1,X_NUM2)=',XNUM_1,XNUM_2
C
      IF (XNUM_2 .EQ. XNUM_1) THEN
         PROD_RATE(COUNTING) = 0.0
         GOTO 110
      ELSE
         CYC_TIME = (XTAL_2 - XTAL_1)/(XNUM_2 - XNUM_1)
      ENDIF
C
      PROD_RATE(COUNTING)=(1/CYC_TIME)*60
C
110   XTAL_1 = XTAL_2
      XNUM_1 = XNUM_2
      TOTAL_RATE=TOTAL_RATE+PROD_RATE(COUNTING)
C
      IF (COUNTING .EQ. BATCH_NO) THEN
C
         IF (Kth_RUN .EQ. 0) THEN
            OPTI_MEAN=TOTAL_RATE/BATCH_NO
            PRINT *,OPTI_MEAN
            TOTAL_RATE = 0.0
            OPTI_Var=VARIANCE(OPTI_MEAN,BATCH_NO)
            CALL INTERVAL(Kth_RUN,OPTI_MEAN,OPTI_Var,BATCH_NO,4)
            CALL OUTPT2(Kth_RUN,OPTI_MEAN)
         ELSE
            NEW_MEAN=TOTAL_RATE/BATCH_NO
            PRINT *,OPTI_MEAN
            PRINT *,NEW_MEAN
            TOTAL_RATE = 0.0
            NEW_Var=VARIANCE(NEW_MEAN,BATCH_NO)
         ENDIF
C
         XTAL_1 = 0.0
         XTAL_2 = 0.0
```

```
         XNUM_1 = 0
         XNUM_2 = 0
         COUNTING=0
         RETURN
      ENDIF
      COUNTING=COUNTING+1
      RETURN
      END
C*.....................................................
C*
      SUBROUTINE INPUT
      COMMON/A003/BATCH_NO,MAX_ITERATION,TEMPERATURE,
     +FACILITY(9),PRE_RESOURCE(9),UPDATE
C
      INTEGER FACILITY,BATCH_NO,MAX_ITERATION,PRE_RESOURCE
      REAL TEMPERATURE
C
      OPEN(UNIT=70, FILE='AGRATE.A01',STATUS='NEW')
      OPEN(UNIT=71, FILE='ANLOUT.A01',STATUS='NEW')
      WRITE(71,*)'   +#+++B1+++B2+++B3+++B4+++LR+++MC++++RATE+'
C
      PRINT *,'HELLO ! Please enter the required data by order.'
      PRINT *
      PRINT *,'(1) The initial value for setup queue ?'
      READ *,FACILITY(1)
      PRINT *,'(2) The initial value for machine queue ?'
      READ *,FACILITY(2)
      PRINT *,'(3) The initial value for deburing queue ?'
      READ *,FACILITY(3)
      PRINT *,'(4) The initial value for QC queue ?'
      READ *,FACILITY(4)
      PRINT *,'(5) The initial value for Labors ?'
      READ *,FACILITY(5)
      PRINT *,'(6) The initial value for Machine  ?'
      READ *,FACILITY(6)
      PRINT *,'(7) The initial value for Deburing Robot ?'
      READ *,FACILITY(7)
      PRINT *,'(8) The initial value for CMM ?'
      READ *,FACILITY(8)
      PRINT *,'(9) The number of batches ?'
      READ *,BATCH_NO
      PRINT *,'(10) The stop condition (max # of iterations) ?'
```

```
      READ *,MAX_ITERATION
      PRINT *,'(11) Enter the control parameter (floating-point)?'
      READ *,TEMPERATURE
      PRINT *,'Thank you for your cooperation !! '
C
      DO 20 INDEX=1,8
         CALL SETMR(INDEX,FACILITY(INDEX))
         PRE_RESOURCE(INDEX)=FACILITY(INDEX)
20    CONTINUE
      RETURN
      END
C*
C*.............................................................
C*
      SUBROUTINE OUTPT1(SRUN,MEAN_o,LOW,UP)
      REAL MEAN_o,LOW,UP
      INTEGER SRUN
C
      WRITE(70,*) SRUN,LOW,MEAN_o,UP
      END
C*
C*.............................................................
C*
      SUBROUTINE OUTPT5(SRUN,MEAN_o,LOWER,UPPER)
      REAL MEAN_o,LOWER,UPPER
      INTEGER SRUN

      WRITE(73,*) SRUN,LOWER,MEAN_o,UPPER
      END
C*
C*.............................................................
C*
      SUBROUTINE OUTPT2(SRUN,MEAN_o)
      COMMON/A003/BATCH_NO,MAX_ITERATION,TEMPERATURE,
     +FACILITY(9),PRE_RESOURCE(9),UPDATE
C
      INTEGER FACILITY,SRUN,PRE_RESOURCE
      REAL MEAN_o
C
      WRITE(71,50) SRUN,PRE_RESOURCE(1),PRE_RESOURCE(2),
     +PRE_RESOURCE(3),PRE_RESOURCE(4),PRE_RESOURCE(5),
     +PRE_RESOURCE(6),PRE_RESOURCE(7),PRE_RESOURCE(8),MEAN_o
```

```
50    FORMAT(2X,I3,8(1X,I4),2X,F8.4)
      END
C*
C*.....................................................................
C*
      SUBROUTINE OUTPT3(SRUN,MEAN_n)
      COMMON/A003/BATCH_NO,MAX_ITERATION,TEMPERATURE,
     +FACILITY(9),PRE_RESOURCE(9),UPDATE
C
      INTEGER FACILITY,SRUN
      REAL MEAN_n
C
      WRITE(72,60) SRUN,FACILITY(1),FACILITY(2),FACILITY(3),
     +FACILITY(4),FACILITY(5),FACILITY(7),FACILITY(8),
     +FACILITY(8),MEAN_n
60    FORMAT(2X,I3,8(1X,I4),2X,F8.4)
      END
C*
C*.....................................................................
C*
      SUBROUTINE OUTPT4(UBOUND,MEANS,LBOUND)
      REAL UBOUND,MEANS,LBOUND
C
      WRITE(71,*)
      WRITE(71,*)'+++ 95% CONFIDENCE INTERVAL +++'
      WRITE(71,*)'* UPPER-BOUND:',UBOUND
      WRITE(71,*)'* MEAN       :',MEANS
      WRITE(71,*)'* LOW-BOUND  :',LBOUND
      END
C*
C*.....................................................................
C*
      REAL FUNCTION VARIANCE(MEAN,SAMPLE_SIZE)
      COMMON/A001/Kth_RUN,PROD_RATE(12)

      INTEGER SAMPLE_SIZE
      REAL DIFFERENCE,MEAN,PROD_RATE

      DO 70 INDEX=1,SAMPLE_SIZE
            DIFFERENCE=DIFFERENCE+(PROD_RATE(INDEX)-MEAN)**2
70    CONTINUE
      VARIANCE=DIFFERENCE/(SAMPLE_SIZE-1)
```

```
      DIFFERENCE=0.0
      END
C*
C*.....................................................
C*
      INTEGER FUNCTION F_TEST(Var_1,Var_2)
      REAL Var_1,Var_2,RATIO

      IF (Var_1 .GT. Var_2) THEN
         IF (Var_2 .EQ. 0.0) THEN
            RATIO = 9.9
         ELSE
            RATIO=Var_1/Var_2
         ENDIF
      ELSEIF (Var_2 .GT. Var_1) THEN
         IF (Var_1 .EQ. 0.0) THEN
            RATIO = 9.9
         ELSE
            RATIO=Var_2/Var_1
         ENDIF
      ELSE
            RATIO=1.0
      ENDIF

      IF (RATIO .GT. 2.16 ) THEN
         F_TEST=1
      ELSE
         F_TEST=0
      ENDIF
      END
C*
C*.....................................................
C*
      INTEGER FUNCTION SIGNIFICANT(MEAN1,MEAN2,Var1,Var2)
C     (if significant = 1, then significantly different)
      INTEGER F_TEST
      REAL Var1,Var2,SEPERATE_VAR,POOL_VAR,SIGMA,CRITICAL_VALUE,
     +     MEAN1,MEAN2

      IF ( F_TEST(Var1,Var2) .EQ. 1) THEN
         SEPERATE_VAR=(Var1+Var2)/10
         SIGMA=SQRT(SEPERATE_VAR)
```

```
      ELSE
         POOL_VAR=(Var1+Var2)*18/20
         SIGMA=SQRT(POOL_VAR)*SQRT(0.2)
      ENDIF

      CRITICAL_VALUE=1.282*SIGMA
      T_TEST=ABS(MEAN2-MEAN1)

      IF (T_TEST .GT. CRITICAL_VALUE) THEN
         SIGNIFICANT=1
      ELSE
         SIGNIFICANT=0
      ENDIF
      END
C*
C*.............................................................
C*
      REAL FUNCTION RAND(ISTRM)
      INTEGER B2E15,B2E16,HI15,HI31,ISTRM,IZGET,IZSET,LOW15,LOWPRD,
     +        MODLUS,MULT1,MULT2,OVFLOW,ZI,ZRNG(50)
      INTEGER IRANDG,RANDST
      SAVE ZRNG

      DATA MULT1,MULT2/24112,26143/
      DATA B2E15,B2E16,MODLUS/32768,65536,2147483647/
      DATA ZRNG/1973272912,281629770,20006270,1280689831,
     +          2096730329,1933576050,913566091,246780520,
     +          1363774876,604901985,1511192140,1259851944,
     +          824064364,150493284,242708531,75253171,1964472944,
     +          1202299975,233217322,1911216000,726370533,
     +          403498145,993232223,1103205531,762430696,
     +          1922803170,1385516923,76271663,413682397,726466604,
     +          336157058,1432650381,1120462904,595778810,877722890
     +         ,1046574445,68911991,2088367019,748545416,622401386,
     +          2122378830,640690903,1774806513,2132545692,
     +          2079249579,78130110,852776735,1187867272,1351423507
     +         ,1645973084/

      ZI     = ZRNG(ISTRM)
      HI15   = ZI / B2E16
      LOWPRD = (ZI - HI15 * B2E16) * MULT1
      LOW15  = LOWPRD/B2E16
```

```
      HI31   = HI15 * MULT1 + LOW15
      OVFLOW = HI31 / B2E15
      ZI     = (((LOWPRD - LOW15 * B2E16) - MODLUS) +
     +          (HI31 - OVFLOW * B2E15) * B2E16) + OVFLOW
      IF (ZI .LT. 0) ZI = ZI + MODLUS
      HI15   = ZI/B2E16
      LOWPRD = (ZI-HI15 * B2E16) * MULT2
      LOW15  = LOWPRD/B2E16
      HI31   = HI15*MULT2+LOW15
      OVFLOW = HI31/B2E15
      ZI     = (((LOWPRD-LOW15*B2E16)-MODLUS) +
     +          (HI31 - OVFLOW * B2E15) * B2E16) + OVFLOW
      IF (ZI .LT. 0) ZI = ZI + MODLUS
      ZRNG(ISTRM) = ZI
      RAND = (2 * (ZI / 256) + 1) / 16777216.0
      RETURN
      END

C*
c*.............................................................
C*
      SUBROUTINE STORING
      COMMON/A003/BATCH_NO,MAX_ITERATION,TEMPERATURE,
     +FACILITY(9),PRE_RESOURCE(9),UPDATE
      INTEGER INDEX,FACILITY,PRE_RESOURCE

      DO 80 INDEX=1,8
         PRE_RESOURCE(INDEX)=FACILITY(INDEX)
80    CONTINUE
      END
```