INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

ON MERGING SEQUENCING AND SCHEDULING THEORY WITH GENETIC ALGORITHMS TO SOLVE STOCHASTIC JOB SHOPS

A Dissertation

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirement for the

degree of

Doctor of Philosophy

By

IBRAHIM M. AL-HARKAN
Norman, Oklahoma
1997

UMI Number: 9728709

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

ON MERGING SEQUENCING AND SCHEDULING THEORY WITH GENETIC ALGORITHMS TO SOLVE STOCHASTIC JOB SHOPS

A Dissertation APPROVED FOR THE
SCHOOL OF INDUSTRIAL ENGINEERING

BY

Bobbie L Foote

Dr. Bobbie L. Foote

Dr. Adedeji B. Badiru

Dr. John Y. Cheung

Dr. John Hawley

Dr. Simin P. Pulat

Dr. Theodore B. Trafalis

ACKNOWLEDGEMENTS

This research has been made possible with the help of certain individuals and organizations. Therefore, I would like to take this opportunity to acknowledge their assistance.

I would like to express my sincere thanks to my advisor, Dr. Bobbie L. Foote, for his direction, encouragement, patience, and understanding. Not only did he act as my academic guide, but he served as my personal mentor as well. His thinking and wisdom have had a permanent effect on my life. I would like also to thank Dr. Foote's family for the kindness and the care they gave me during these years.

I would like also to thank the other members of my committee, Dr. Simin P. Pulat, Dr. Adedeji B. Badiru, Dr. John Y. Cheung, Dr. Theodore B. Trafalis, and Dr. John Hawley. The completion of this work is due to their help and suggestions.

I am very appreciative of Xuemin Liang; the results of her Master's thesis made my fifth experiment easier. Also, I am very appreciative for her valuable suggestions during our research.

I would to thank King Saud University, Riyadh, Saudi Arabia, which gave me the financial assistance to pursue my graduate studies. Also, I thank the University of Oklahoma for accepting me in its graduate program and for awarding me the Ph. D. degree.

Heartfelt thanks go to my best friend Susan Hancock for sharing this dream with me and for her love, intimacy, and cordiality. I am very appreciative of her continual encouragement and continuous motivation during these years. Her love and honesty have taught me to be more trustful, her self-giving has helped me to understand the power of love, her belief in me has made me more confident, her vulnerability has disciplined me to be more fair, her spontaneity has taught

me to be spontaneous, her teaching and guidance have made me a better communicator, her intellectuality has made me a better listener, and her adventuresomeness has made me a better explorer.

I extend my gratitude to Nancy and Dr. Richard Hancock for their support, help, generosity, and guidance. I extend my appreciation to Nancy Hancock for her professional editing of this work. I am very appreciative of Jennifer Hancock for her companionship and help during these years.

I would like to thank my cousin Ahmed for being my comrade during the last two years.

The final acknowledgment must go to my parents for their love, patience, and understanding.

To Nancy and Dr. Richard Hancock; their inspiration and honor have made an everlasting impression on my life

To Susan Hancock; her love, friendship, and teaching have made me a better person

TABLE OF CONTENTS

CHAPTER

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

ABSTRACT

The standard genetic algorithm has been modified to address the job shop problem by constraining the genes in the chromosomes during the genetic operators implementations to match general theoretical sequencing constraints.

When comparing the deterministic constrained and unconstrained genetic algorithms to minimize makespan, the constrained algorithm improved the average percentage errors by 27.44%. Also, when the deterministic constrained and unconstrained genetic algorithms to minimize total tardiness were compared, the constrained algorithm improved the average percentage errors by 248.77%.

The stochastic job shop problem was solved using two genetic algorithms. The first was a stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using probability Gantt charting. The second was a stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using simulation. In these two algorithms, the fitness function was altered to a utility function defined as follows: Probability {total tardiness of a chromosome $\leq$ target total tardiness}. When comparing the two chromosome evaluation methods, the probability Gantt charting deviated from the true mean for both the makespan and the average flow time by 3% and 1.7% respectively. Also, all averages estimated for both the makespan and the average flow time fall within the 90% confidence interval. Furthermore, using probability Gantt charting reduced the CPU time needed by 554.9% when compared to the CPU time needed by simulation. When the results obtained by the two stochastic constrained genetic algorithms were compared, the second algorithm reduced the actual expected total tardiness, the actual worst case total tardiness, and the risk by 30.3%, 56%, and 18% respectively.

ON MERGING SEQUENCING AND SCHEDULING THEORY WITH GENETIC ALGORITHMS TO SOLVE STOCHASTIC JOB SHOPS

CHAPTER I

THE GENERAL SEQUENCING AND SCHEDULING PROBLEM

Introduction

The problem that motivated this study is as follows: suppose there are a number of jobs to be performed. Each job consists of a given sequence of operations which needs to be performed using a number of machines. All operations for each job must be performed in the order given by the sequence. Each operation demands the use of a particular machine for a given time. Each machine can process only one operation at a time. Therefore, given a cost function by which each sequence can be evaluated, the order of operations on each machine that minimizes the cost function needs to be found.

The problem described above is known as a production sequencing and scheduling problem. Sequencing and scheduling problems occur in different industries and circumstances, even though the description of the problem above suggests a manufacturing industry problem. The following are some examples of different situations which need sequencing or scheduling: 1) parts waiting for processing in a manufacturing plant; 2) aircraft waiting for landing clearance at an airport; 3) computer programs running at a computing center; 4) class scheduling in a school, 5) patients waiting in a Doctor's office; 6) ships to be anchored in a harbor, and 7) Saturday afternoon chores at home.

Definitions

Production sequencing and scheduling is one of the most important activities in production planning and control. Morton and Pentico discussed how important the sequencing and scheduling role is, stating that "it pervades all economic activity" (Morton and Pentico 1993, 5). Pinedo further discussed the importance of the sequencing and scheduling problem:

> ...Sequencing and scheduling are forms of decision-making which play a crucial role in manufacturing as well as in service industries. In the current competitive environment, effective sequencing and scheduling has become a necessity for survival in the marketplace. Companies have to meet shipping dates committed to the customers, as failure to do so may result in a significant loss of good will. They also have to schedule activities in such a way as to use the resources available in an efficient manner. (Pinedo 1995, xiii)

The definition of sequencing among researchers is common. Sequencing is defined as the order in which the jobs (tasks) are processed through the machines (resources). Scheduling was defined by Baker as follows:

> ...Scheduling is the allocation of resources over time to perform a collection of tasks.... Scheduling is a decision-making function: it is the process of determining a schedule.... Scheduling is a body of theory: it is a collection of principles, models, techniques, and logical conclusions that provide insight into the scheduling function. (Baker 1974, 2)

Also, Morton and Pentico defined scheduling as follows:

> ...Scheduling is the process of organizing, choosing, and timing resource usage to carry out all the activities necessary to produce the desired outputs at the desired times, while satisfying a large number of time and relationship constraints among the activities and the resources. (Morton and Pentico 1993, 5)

Therefore, from the above two definitions, scheduling can be defined as a decision-making process that is concerned with the allocation of limited machines (resources) over time to perform a collection of jobs (tasks) in which one or several objectives have to be optimized.

The general definition of the sequencing problem can be stated as follows: there are m machines $\{M_1, M_2, \ldots, M_m\}$ available and n jobs $\{J_1, J_2, \ldots, J_n\}$ to be processed. A subset of these machines is required to complete the processing of each job. The flow pattern (process plan) for some or all jobs may or may not be fixed. Each job should be processed through the machines in a particular order that satisfies the job's technological constraints. The processing of job i on machine j is called an operation denoted by O_{ij}. Associated with each operation is a processing time denoted by P_{ij}, and a setup time denoted by S_{ij}. Also, associated with each job is a weight, w_i, a ready (release or arrival) time, r_i, and a due date, d_i. Finally, each job has an allowance time to be in the shop, a_i.

Thus, the general problem is to generate a sequence that satisfies the following conditions: 1) all jobs are processed; 2) all technological constraints are met for all jobs (feasibility condition), and 3) all criteria that were selected are optimized.

Levels of the Sequencing and Scheduling Problem

Sequencing and scheduling are involved in planning and controlling the decision-making process of manufacturing and service industries in several stages. According to several researchers (Baker 1974; Browne, Harhen, and Shivnan 1988; Muchnik 1992; and Morton and Pentico 1993), sequencing and scheduling exist at several levels of the decision-making process. These levels are as follows:

1) **Long-term planning** which has a horizon of 2 to 5 years. Some examples are: plant layout, plant design, and plant expansion.
2) **Middle-term planning** such as production smoothing and logistics which can be done in a period of 1 to 2 years.
3) **Short-term planning** which is done every 3 or 6 months. Examples include: requirements plan, shop bidding, and due date setting.
4) **Predictive scheduling** which is performed in a range of 2 to 6 weeks. Job shop routing, assembly line balancing, and process batch sizing qualify as predictive.
5) **Reactive scheduling or control** which is performed every day or every three days. A few examples are: hot jobs, down machines, and late material.

Level four is the concern of this research, and therefore, sequencing and scheduling methodologies for only this level will be discussed. Specifically, environments, general assumptions, categories, criteria, decision-making goals, and solution methods for the sequencing and scheduling problems will be explained.

Environments of the Sequencing and Scheduling Problem

According to Conway, Maxwell, and Miller (1967), sequencing and scheduling environments are classified according to four types of information: the jobs and operations to be processed; the number and types of machines that comprise the shop; the disciplines that restrict the manner in which assignment can be made, and the criteria by which a schedule will be evaluated. The sequencing and scheduling environments are as follows:

1) **Single machine shop**: one machine and n jobs to be processed.
2) **Flow shop**: there are m machines in series and jobs can be processed in one of the following ways: a) permutational: jobs are processed by a series of m machines in exactly the same order, or b) non-permutational: jobs are processed by a series of m machines not in the same order.
3) **Job shop**: each job has its flow pattern and a subset of these jobs can visit each machine twice or more often. Multiple entries and exits.
4) **Assembly job shop**: a job shop with jobs that have at least two component items and at least one assembly operation.
5) **Hybrid job shop**: the precedence ordering of the operations of some jobs is the same.
6) **Hybrid assembly job shop**: combines the features of both the assembly and hybrid job shop.
7) **Open shop**: there are m machines and there is no restriction in the routing of each job through the machines. In other words, there is no specified flow pattern for any job.

8) **Closed shop**: it is a job shop; however, all production orders are generated as a result of inventory replenishment decisions. In other words, the production is not affected by the customer order.

Assumptions of the Sequencing and Scheduling Problem

The different sequencing and scheduling problem environments have been solved under several assumptions. These assumptions were used to make the scheduling problem tractable and easier. Some of these assumptions are: 1) the set of the jobs and the set of the machines are known and fixed; 2) all jobs and all machines are available at the same time and are independent; 3) all jobs and machines remain available during an unlimited period; 4) the processing time for each job on all machines is fixed, has a known probability distribution function, and sequence independent; 5) setup times are included in processing times; 6) a basic batch size is fixed for all jobs; 7) all jobs and all machines are equally weighted; 8) no preemption is allowed; 9) a definite due date is assigned to each job; 10) each job is processed by all the machines assigned to it; 11) each machine processes all the jobs assigned to it, and 12) the process plan for each job is known and fixed.

For additional constraints that have been used when solving sequencing and scheduling problems, consult Conway, Maxwell, and Miller (1967), Baker (1974), Rinnooy Kan (1976), Bellman, Esogbue, and Nabeshima (1982), French (1982), and Morton and Pentico (1993).

Categories of the Sequencing and Scheduling Problem

When none, one, or more of the assumptions used is/are relaxed, then the sequencing and scheduling problem is categorized into one of the following categories:

1) **Deterministic sequencing and scheduling problems**: when all elements of the problem, such as the state of the arrival of the jobs to the shop, due-dates of jobs, ordering, processing times and availability of machines, do not include stochastic factors and are

determined in advance.

2) **Static sequencing and scheduling problems**: the same as deterministic problems except that the nature of the job arrival is different. The set of jobs over time does not change, and it is available beforehand.
3) **Dynamic sequencing and scheduling problems**: the set of jobs changes over time and jobs arrive at different times.
4) **Stochastic sequencing and scheduling problems**: at least one of the problem elements includes a stochastic factor.

Criteria of the Sequencing and Scheduling Problem

According to Rinnooy Kan (1976) and French (1982), the criteria for sequencing and scheduling problems are classified according to three measures: completion times; due dates, and inventory and machine utilization. With each of the three measures, the following criteria can be associated, as shown in Table 1.

In the sequencing and scheduling literature, there are other criteria such as a combination of two or more of the above mentioned criteria. Also, there are other criteria in the sequencing and scheduling literature that were not mentioned above. For additional criteria, the reader can refer to Conway, Maxwell, and Miller (1967); Baker (1974); Rinnooy (1976); Bellman, Esogbue, and Nabeshima (1982); French (1982); Morton and Pentico (1993); and Pinedo (1995).

Table 1. Criteria associated with each of the three measures.

Criteria based on completion times	
Completion time of job i C_i	The total completion time $\sum_{i=1}^{n} C_i$.
The total weighted completion time $\sum_{i=1}^{n} w_i C_i$.	The total weighted waiting time $\sum_{i=1}^{n} w_i \sum_{j=1}^{m} W_{ij}$.
Flow time of job i $F_i = C_i - r_i$	Maximum completion time (the schedule time, total production time, or makespan) $C_{max} = \max_{1,\ldots,n} \{C_i\}$.
The total flow time $\sum_{i=1}^{n} F_i$.	The total weighted flow time $\sum_{i=1}^{n} w_i F_i$.
Average flow time $\bar{F}$.	Maximum flow time F_{max}.
Waiting time of job i $W_i = F_i - \sum_{j=1}^{m} P_{ij}$.	The total waiting time $\sum_{i=1}^{n} \sum_{j=1}^{m} W_{ij}$.
Average completion time $\bar{C}$.	Average waiting time $\bar{W}$.
Criteria based on due-dates	
Lateness of job i $L_i = C_i - d_i$.	The total lateness $\sum_{i=1}^{n} L_i$.
The total weighted lateness $\sum_{i=1}^{n} w_i L_i$.	Average lateness $\bar{L}$.
Maximum lateness $L_{max} = \max_{1,\ldots,n} \{L_i\}$.	Tardiness of job i $T_i = \max_{1,\ldots,n} \{0, L_i\}$
Earliness of job i $E_i = \max_{1,\ldots,n} \{0, -L_i\}$	Maximum Earliness $E_{max} = \max_{1,\ldots,n} \{E_i\}$
The total tardiness $\sum_{i=1}^{n} T_i$.	The total weighted tardiness $\sum_{i=1}^{n} w_i T_i$.
Average tardiness $\bar{T}$.	Maximum tardiness $T_{max} = \max_{1,\ldots,n} \{T_i\}$
Number of jobs tardy $N_T = \sum_{i=1}^{n} \delta(T_i)$, $\delta(T_i) = 1$ if $T_i > 0$ and $\delta(T_i) = 0$ if $T_i \leq 0$.	
Criteria based on inventory and machine utilization	
Average number of jobs waiting for machines $\bar{N}_w$.	Average number of unfinished jobs $\bar{N}_u$.
Average number of jobs completed $\bar{N}_c$.	Average number of jobs actually being processed $\bar{N}_p$.
Average number of machines idle $\bar{I}$.	Maximum machine idle time I_{max} .
Average utilization $\bar{U} = \sum_{i=1}^{n} \sum_{j=1}^{m} P_{ij} / m.C_{max}$	

Decision-Making Goals in the Sequencing and Scheduling Problem

According to Baker (1974), there are three common types of decision-making goals in sequencing and scheduling problems: efficient utilization of machines; rapid response to demands, and close conformance to prescribed deadlines. The three common goals can be achieved by associating the criteria mentioned above with each of the three goals as follows:

1) Efficient utilization of machines (resources): minimize C_{max} or $\bar{I}$, or maximize $\bar{N}_p$ or $\bar{U}$.

2) Rapid response to demands: minimize $\sum_{i=1}^{n} C_i$; $\sum_{i=1}^{n} F_i$; $\sum_{i=1}^{n} L_i$; $\sum_{i=1}^{n} \sum_{j=1}^{m} W_{ij}$; $\bar{C}$; $\bar{F}$; $\bar{L}$ ' $\bar{N}_W$, or $\bar{W}$.

3) Close conformance to prescribed deadlines: minimize L_{max}; T_{max}; N_T; $\sum_{i=1}^{n} T_i$; $\bar{T}$, or $\sum_{i=1}^{n} w_i T_i$.

Methods of Solution for the Sequencing and Scheduling Problem

Several methods have been developed to solve and model sequencing and scheduling problems that belong to any of the four categories (deterministic, static, dynamic, and stochastic). These methods of solution can be classified as follows:

1. Efficient optimal methods such as Johnson's algorithm to solve a flow shop problem with two machines and n jobs (Johnson 1954).
2. Enumerative methods (implicit and explicit or complete) such as Brown and Lomnicki's branch and bound algorithm (Brown and Lomnicki 1966).
3. Heuristic methods such as Campbell, Dudek, and Smith's algorithm to solve m machines and n jobs flow shop problems (Campbell, Dudek, and Smith 1970).
4. Mathematical models (Integer Programming) such as Wagner's Form to solve the permutation flow shop problem with n jobs and m machines (Wagner 1959).
5. Heuristic search techniques: simulated annealing, genetic algorithms, tabu Search, and artificial Intelligence.
6. Simulation models.
7. Analytical models (such as Jackson's open queueing network model, Jackson 1957a).

Over the last four decades, a large amount of research has been done in each of the seven classes to model and to solve sequencing and scheduling problems. Most of the research that has been done has been reported by Muth and Thompson (1963), Conway, Maxwell, and Miller

(1967), Moore and Wilson (1967), Elmaghraby (1968), Day and Hottenstein (1970), Baker (1974), Rinnooy Kan (1976), Dannenbring (1977), Lemoine (1977), Panwalkar and Iskander (1977), Graham et al. (1979), Bellman, Esogbue, and Nabeshima (1982), French (1982), Graves (1981), Blackstone, Phillips, and Hogg (1982), Park, Pegden, and Enscore (1984), Forst (1984), Raghavachari (1988), Rodammer and White (1988), Buxey (1989), Cheng and Gupta (1989), Kovalev et al. (1989), Cheng and Sin (1990), Nof, Rajan, and Frederick (1990), Bahouth (1991), Noronha and Sarma (1991), Dudek, Panwalkar, and Smith (1992), Maccarthy and Liu (1993), Kamath (1994), Morton and Pentico (1994), Koulamas, Antony, and Jaen (1994), Szelke and Kerr (1994), Yen and Pinedo (1994), Pinedo (1995), Shirhatti and Kamath (1995), and Hall and Sriskandarajah (1995).

Purpose of the Study

The purpose of this study is not to survey all the work that has been done to model and to solve the sequencing and scheduling problem, but rather to study the related research that has been done to model and to solve the job shop problem. Another major focus of this research is the control of a dynamic stochastic job shop environment. The control of this environment was accomplished by an integrated model that used sequencing and scheduling theory, heuristic search techniques, and dispatching rules. The integrated model consists of a heuristic search technique, the genetic algorithm, that used the available sequencing and scheduling theories and dispatching rules to enhance its search procedures. The results of this integration is the constrained genetic algorithm. The constrained genetic algorithm is the main thrust and the focus of this research.

CHAPTER II

LITERATURE REVIEW

Introduction

In this chapter, the related work that has been done to model and to solve the job shop problem will be reviewed. Specifically, this chapter will first review the dispatching rules and the simulation studies that have been done to investigate the dispatching rules. Next, a description of the best-known heuristic, Shifting Bottleneck, that has been used to solve the job shop problem will be presented. Next, an introduction to genetic algorithms (GAs) will be given. Then, a summary of the GA methodology developed to solve sequencing and scheduling problem will be given. Next, a review of the genetic algorithm applications to sequencing and scheduling problems will be given. Then, the constrained genetic algorithm that was developed by Al-Harkan and Foote (1994, 1996) is introduced. Next, analysis of the results obtained by the constrained genetic algorithm will be given. Finally, the research gaps will be discussed.

Dispatching Rules

Over the last four decades, the job shop problem has been solved using dispatching rules (also called scheduling rules, sequencing rules, decision rules, or priority rules). These dispatching rules are used to determine the priority of each job. The priority of a job is determined as a function of job parameters, machine parameters, or shop characteristics. When the priority of each job is determined, jobs are sorted and then the job with the highest priority is selected to be processed first.

Baker (1974, 216-217) and Morton and Pentico (1993, 373) classified dispatching rules as follows: local, global, static, dynamic, and forecast. **Local rules** are concerned with the local available information. **Global rules** are used to dispatch jobs using all information available on the shop floor. **Static rules** do not change over time, and ignore the status of the job shop floor. **Dynamic rules** are time dependent, and change according to the status of the job shop floor. **Forecast rules** are used to give priority to jobs according to what the job is going to come across in the future, and according to the situation at the local machine.

Several dispatching rules have been reported by many researchers. These reports have been made by Conway, Maxwell, and Miller (1967, 113-129, 219-247), Moore and Wilson (1967), Day and Hottenstein (1970), Jones (1973), Baker (1974, 214-231), Rinnooy (1976, 51-52), Panwalkar and Iskander (1977), Buffa and Miller (1979, 485-535), Blackstone, Phillips, and Hogg (1982), Forst (1984), Sen and Gupta (1984), Dayhoff and Atherton (1986), Sawaqed (1987), Cheng and Gupta (1989), Haupt (1989), Nof, Rajan, and Frederick (1990), Ramasesh (1990), Bahouth (1991), Bhaskaran and Pinedo (1992), Morton and Pentico (1994, 372-378: 389-395), and Pinedo (1995, 143-148). The following are some of the dispatching rules that have been developed, investigated, and implemented by several researchers and practitioners:

1. **SPT or SEPT**: Shortest Processing Time or Shortest Expected Processing Time. The job with the smallest operation processing time is processed first. The SPT rule has several versions.
    - **SRPT**: Total Shortest Remaining Processing Time.
    - **TSPT**: Truncated SPT. The job with the smallest operation processing time is processed first, but if there is a job with an operation waiting time larger than W, that job is processed first, W is arbitrarily chosen.
    - **WSPT**: Weighted Shortest Processing Time. The job with the smallest ratio is processed first. The ratio is computed by dividing the operation processing time of the job by its weight.

- **LWR**: Least Work Remaining in terms of the number of operations.
- **TWORK**: Total Work in terms of processing time.
- **AJF-SPT**: Assembly jobs first with SPT rule. If there are assembly and non-assembly products waiting for a specific machine, then the assembly products are selected first. The SPT rule is used to select one of them.

2. **LPT or LEPT**: Longest Processing Time or Longest Expected Processing Time. The job with the largest operation processing time is processed first. There are other versions of LPT.
    - **TLPT**: Total LPT.
    - **LRPT**: Total Longest Remaining Processing Time.
    - **MWR**: Most Work Remaining in terms of the number of operations.
3. **EDD**: Earliest Due Date. The job with the smallest due date is processed first. There are three versions of EDD rule.
    - **ODD**: Operation Due Date. The operation with the smallest due date is processed first.
    - **MDD**: Modified Due Date. From the set of jobs waiting for a specific machine, jobs are assigned a new due date, and EDD is performed on this set. The new due dates are assigned in one of two ways. In the first, a job with negative slack is assigned a due date that is equal to the current time plus the processing time. In the second, a job with positive slack is assigned its original due date.
    - **MODD**: Modified Operation Due Date. From the set of operations waiting for a specific machine, operations are assigned a new due date, and ODD is performed on this set. This means the new operation due dates are assigned using the two ways used in the MDD, but instead of using EDD, the ODD is used.
4. **JST**: Job Slack Time. The job with minimum slack is processed first. The job slack time is computed as the difference between the job due date, the work remaining, and the current time.

The JST rule has five versions.

- **OST or S/OPN**: Operation Slack Time. The job with the smallest operation slack is processed first. The OST is determined by dividing the JST by the number of job operations remaining.
- **A/OPN**: Allowance over remaining number of operation. The job with the smallest ratio is processed first.
- **S/A**: Slack time over Allowance: The job with the smallest ratio is processed first.
- **WPT+WOST**: Weighted Processing Time plus Weighted Operation Slack Time. The job with the smallest value is processed first.
- **S/RPT**: Slack over Remaining work Time. S/RPT is computed as job slack divided by the remaining work time.

5. **CR**: Critical Ratio. The job with the smallest ratio is processed first. The CR is determined by dividing job's allowance by the remaining work time. The CR has one version.
    - **OCR**: Operation Critical Ratio. The operation with the smallest ratio is processed first. The OCR is determined by dividing operation's allowance by the operation process time.
6. **RANDOM**: Service In Random Order. A job is randomly selected from the set of jobs which are queued at the machine. RANDOM has one version.
    - **Biased-RANDOM**: Service In Biased Random Order. When RANDOM rule is applied, jobs are equally likely to be selected from the set of jobs waiting. However, in the Biased-RANDOM rule, jobs are not equally likely to be selected. The selection process is biased according to one or more of the other dispatching rules such SPT or EDD. To apply the Biased-RANDOM to a set of jobs waiting, a dispatching rule is selected first (say SPT). Then, the set of jobs waiting are sorted according to the dispatching rule selected (i.e., SPT). Next, jobs in the ordered list are assigned selection probabilities which are usually

computed according to geometric distribution. The job in the first position will be given the largest selection probability and the job in the last position will be given the smallest selection probability. By doing so, the jobs early in the ordered list of jobs are more likely to be selected, while jobs late in the ordered list of jobs are less likely to be selected.

7. **FCFS or SORT**: First Come, First Served or Smallest Ready Time. The job which arrives first at the machine will be served first. There is one version of FCFS rule.
    - **FASFS or SRT**: First At Shop, First Served or Smallest Release Time. A job arriving first at the shop is given priority to go first in all machines.
8. **LCFS**: Last Come, First Served. The job which arrives last will be served first.
9. **LFJ**: Least Flexible Job. The job with the least flexibility is processed first.
10. **FOFO**: First Off, First On. The job with the operation that could be completed earliest will be processed first even if this operation is not yet in the queue. In this case, the machine will be idle until the operation arrives.
11. **LAWINQ**: Least Anticipated Work In Next Queue. From the set of jobs waiting for a specific machine, a job will be selected that will encounter the smallest queue at the next machine in its route.
12. **COVERT**: Cost OVER Time. COVERT is a composite rule that puts the job with the largest COVERT ratio in first position. The COVERT ratio is computed by dividing an anticipated tardiness for the associated job and its operation processing time. The COVERT rule has two versions.
    - **ATC**: Apparent Tardiness Cost. The ATC introduces the effect of job weight and it uses a different function to estimate the tardiness associated with each job. ATC gives priority to a job with the largest ATC value.
    - **ATEC**: Apparent Tardiness and Earliness Cost. ATEC is a generalization of both

COVERT and ATC. It includes a different function to account for tardiness and earliness in its computations.

As mentioned earlier, the above dispatching rules are determined according to job parameters, machine parameters, and shop characteristics. The above rules can be classified into four classes. The first class consist of rules that deal with the processing time (that is, rules 1 and 2). Rules 3, 4, and 5 are a class of rules which involve due dates. Class three consists of rules numbering 6, 7, 8, 9, 10, and 11 that involve shop and/or job characteristics. Finally, class four is formed by a combination of the other three classes and is known as rule 12.

Since the job shop problem can be viewed as a network of queues, the effects of the dispatching rules can be tested using queueing network theory. Open queueing network (OQN) theory, developed by Jackson in 1957 (Jackson 1957a, 1963), can only be used to test the effect of the FCFS rule. However, the effects of the other rules are difficult to describe using OQN theory. Therefore, other dispatching rules have been tested using computer simulation models. As a results, simulation modeling of the job shop has been receiving much attention over the last four decades.

In the late fifties, a group of researchers simulated the job shop environment and published the results. This team is considered the pioneers of the field. The group consisted of Jackson (1957b), Nelson and Jackson (1957), and Rowe (1958) from the University of California Los Angeles (UCLA). The work done by the this group has influenced all the investigations done since.

During the sixties, a series of investigations was done to continue studying the effect of dispatching rules. This series of investigations was encouraged by the results obtained by the UCLA group. These attempts were made by Baker and Dzielinski (1960), Conway, Johnson, and Maxwell (1960), Nanot (1963), Carroll (1965), Conway (1965a, 1965b), Gere (1966), and Conway, Maxwell, and Miller (1967, 219-247).

Baker and Dzielinski (1960) at International Business Machine corporation (IBM) simulated a job shop that can have several shop sizes, which ranged from nine to thirty non-identical machines. They investigated the effect of RANDOM, FCFS, and SPT on the total manufacturing time. Two main conclusions obtained: the SPT rule is superior in minimizing the total manufacturing time and the size of the shop is not a significant factor.

Conway, Johnson, and Maxwell (1960) investigated the effect of thirteen dispatching rules on the distribution of the following four performance measures: completion times; lateness; work-in-process, and utilization. In their experiment, an open job shop was simulated which had five machines and three levels of shop load (heavy, medium, and light). They concluded that for all performance measures, the SPT rule was the best among all rules tested.

Nanot (1963) investigated ten dispatching rules using six different shop sizes and over 2.44×10^6 orders. Four of these shop sizes were assumed to have medium loads, and the others had high loads. Four conclusions came out of this study: 1) the SPT rule is the best under all conditions; 2) FCFS and FASFS rules have low standard deviations; 3) FCFS rule achieved a small proportion of jobs tardy if the shop is not heavily loaded; and 4) job shop size is not a significant factor.

Carroll (1965) investigated the effect of the COVERT rule on several job shop configurations. He used six other dispatching rules to investigate the effectiveness of the COVERT rule. He concluded that when the objective function is the mean tardiness the COVERT rule is superior to SPT, FASFS, and TSPT.

Conway (1965a) investigated the effects of twenty dispatching rules on the work-in-process (WIP) by simulating an open job shop that had nine machines and ten thousand jobs were processed. Conway measured the WIP using five measures. These are: number in queue; work remaining in terms of processing time; total work content; work completed, and imminent operation work content. Conway concluded that the SPT dominated all the other rules tested. Conway

(1965b) continued his investigations and used the same job shop configuration in Conway (1965a). In this study, job lateness was used as the performance measure. The effect of the due date tightness was investigated using several methods to estimate the due dates. The estimations of due dates were based on the number of operations (NOP) required, TWORK, constant lead time for all jobs (CON), and random method. For each of these four methods, he compared nine dispatching rules. These rules were RANDOM, FCFS, FASFS, EDD, SPT, LPT, ODD, JST, and WPT+OST. He concluded that the SPT was the best rule among the rules tested, and that the SPT rule was insensitive to due date tightness.

Gere (1966) investigated the effects of eight dispatching rules where the total tardiness was the objective function. Both static and dynamic environments were tested. The job shop simulated had a variety of configurations: 4 to 6 machines, 6 to 60 jobs, and 1 to 16 operations per job. Besides the general assumption mentioned before, Gere assumed no assembly and no labor constraint. He concluded that the non-random dispatching rules (JST, OST, S/A, a modified S/A, SPT, and a combined SPT and S/A) are more significant than random rules (FCFS, and RANDOM). Also, rules that were based on job slack were more effective than the SPT rule.

Conway, Maxwell, and Miller (1967, 219-247) presented several interesting studies. The most interesting was a study that was done by Wayson (1965) to test SPT and FCFS rules with machine flexibility. The average number in queue was used as the performance measure. Wayson concluded that the SPT rule was more sensitive to machine flexibility than FCFS.

Several important conclusions can be obtained from the above series of studies:

1. The SPT rule minimizes the average flow time, average lateness, average number in queue, average tardiness, and percentage of jobs tardy. The SPT is insensitive to due date tightness.
2. COVERT rule is superior in minimizing the mean tardiness when compared to SPT and TSPT.
3. Job slack rules are more effective to minimize the tardiness.

4. The size of the shop is not a significant factor.
5. The FCFS rule achieves a small proportion of jobs tardy if the shop is not heavily loaded.
6. The OST minimized the percentage of jobs tardy and the conditional average tardiness.

The above conclusions have inspired researchers to study the effect of the dispatching rule in more complex and different job shop environments. Also, advancements in computer technology and software that can be used to simulate and study the job shop environment have helped researchers to do more work in this fruitful area. Thirteen studies had been performed during the seventies to investigate more difficult job shop environments. These attempts have been performed by Hottenstein (1970), Putnam et al. (1971), Ashour and Vaswani (1972), Elvers (1973, 1974), Holloway and Nelson (1974), Irastorza and Deane (1974), Eilon, Chowdhury, and Serghiou (1975), Hershauer and Ebert (1975), Berry and Finlay (1976), Eilon and Chowdhury (1976), Nelson, Holloway, and Wong (1977), Hurrion (1978), and Weeks (1979). Some of these studies will be discussed in the following paragraphs.

Hottenstein (1970) studied the process of speeding up the job delivery which is called expediting. One of two reasons can be used to accelerate jobs: 1) the due date of a job has been revised or 2) job slack has become negative. Under normal operating conditions, the SPT rule is used. When jobs belong to the expediting set of jobs, however, the jobs are processed according to either SPTEX or FCFSEX rules. The SPTEX rule gives priority to jobs according to the SPT rule, and the FCFSEX gives priority according to the FCFS rule. Hottenstein simulated hybrid job shops and pure flow shops using six machines. Two types of loads were used. Six performance measures were used: average number of jobs in the system; flow time; percentage of jobs tardy; percentage of early-request jobs shipped late; average tardiness, and average tardiness for early-request jobs. Conclusions of this study can be summarized as follows: SPT and SPTEX rules performed almost the same under all performance measures and conditions, and the FCFSEX had the worst performance. Eilon, Chowdhury, and Serghiou (1975), performed a similar study and in their study, jobs were quickened according to an expediting criterion which was computed as the job slack combined with a control parameter (U). The control parameter was used to regulate the percentage of jobs that can be put in the set of expediting jobs. Then the SPT rule was applied to the jobs which were in the expediting set. The researchers named their general procedures the SPT* rule which was performed by first computing a classification index as follows: $F_i = JST_i - U$. Then, if $F_i \leq 0$, job i is put in the expediting set. Otherwise, job i is put in the normal set.

The effect of due date assignments was studied by Ashour and Vaswani (1972), Elvers (1973), Eilon and Chowdhury (1976), and Weeks (1979). A common conclusion of this series of studies is that dispatching rules that were due-date based performed better when due dates were assigned according to number of operations and work content of a job. In other words, these dispatching rules performed better when due dates were assigned according to expected flow time and job shop congestion. Also, the average tardiness was minimized by S/OPN rule.

Elvers (1974) investigated the effects of sixteen arrival distributions on ten dispatching rules using tardiness as the performance measure. The sixteen arrival distributions were three parameters for each of binomial distribution, bimodal distribution, discrete uniform distribution, left skew distribution, and right skew distribution, plus the Poisson distribution. Some of the dispatching rules used were: FCFS; FASFS; SRPT; SEPT; EDD; OST, and JST. Elvers simulated a job shop with eight machines, and concluded that the dispatching rules are not affected by the arrival distributions in the performance measure tested.

The effect of incorporating queueing waiting time in the calculations of both job slack time and critical ratio was investigated by Berry and Finlay (1976). They used flow time, job lateness, and work-in-process as the performance measures. They estimated the queue time using historical queue waiting times. Berry and Finlay simulated a job shop with ten machines and fifteen products. They concluded that the incorporation of queueing waiting time in the calculations of JST and CR rules did not improve the performance of these rules, which implies no improvement in the shop performance.

Since the early eighties, more and more studies have been published to investigate the effect of dispatching rules in more realistic job shop environments. Some of these studies have been performed by Arumugam and Ramani (1980), Baker and Bertrand (1981, 1982), Miyazaki (1981), Dar-El and Wysk (1982), Muhlemann, Lockett, and Farn (1982), Elvers and Taube (1983), Baker (1984), Ragatz and Mabert (1984), Elvers and Trelevn (1985), Rachamadugu, Raman, and Talbot (1986), Russell, Dar-El, and Taylor (1987), Sawaqed (1987), Vepsalainen and Morton (1987, 1988), Kanet and Christy (1989), Schultz (1989), Anderson and Nyirenda (1990), Karsiti, Cruz, and Mulligan (1992), Kanet and Zhou (1993), Raghu and Rajendran (1993), Rohleder and Scudder (1993a, 1993b), Vig and Dooley (1993), Udo (1993), Bahouth and Foote (1994), and Chang (1994).

Using decision theory, Arumugam and Ramani (1980) compared five dispatching rules to be selected to minimize a combined criterion. This criterion consisted of work-in-process inventory and delivery performance. The five dispatching rules used were: lowest value time; highest value time; customer priority; SPT, and OST. They simulated a job shop with sixty-four machines, ninety-one workers, and nineteen products. Arumugam and Ramani simulated a job shop with various shop loads, and they concluded that the SPT dominated all dispatching rules in all job shop configurations they tested. Kanet and Zhou (1993) used decision theory to developed a dispatching rule which is called MEANP and they tested it against six other dispatching rules. The other dispatching rules used were: SPT; FCFS; ODD; COVERT; ATC, and MODD. They simulated a job shop with a single machine, and they concluded that the MEANP approach was better than all the dispatching rules when both tardiness and flow time were the criteria.

One of the most important elements that affect the performance of dispatching rules are the due date setting rules. The effect of due date setting rules on the dispatching rules have been investigated by several researchers. These attempts have been made by Baker and Bertrand (1981, 1982), Miyazaki (1981), Baker and Kanet (1983), Baker (1984), Ragatz and Mabert (1984), Kanet and Christy (1989), Udo (1993), Vig and Dooley (1993), and Chang (1994). Several conclusions came out of these studies:

1. When job flow time estimates are used to predict due dates, the due dates produced are more robust and accurate to uncontrollable job shop (Miyazaki 1981 and Vig and Dooley 1993).
2. The relative performance of the dispatching rules was affected by the tightness of the due dates (Baker and Bertrand 1981).
3. For practicability, the best due date setting rule is the total work content (TWK) rule which provides the best results for tardiness performance measures (Baker and Bertrand 1981 and 1982, Baker 1984, and Kanet and Christy 1989). According to Kanet and Christy (1989), the TWK reduces work-in-process. TWK = kP, where k is the due date factor and P is the total work required.
4. According to Baker (1984), the second best due date setting rule is the number of operations (NOP) rule which is computed as follows: NOP = km, where k is the due date factor and m is the number of operations required by the job.
5. There is no advantage to using the slack-based dispatching rules over the simple allowance-based rule (Baker 1984).
6. When assigning due dates, both job characteristics and shop status information should be included (Ragatz and Mabert 1984, Udo 1993, Chang 1994).
7. In estimating a job due date, information about machine center congestion and the routing of the job is more useful than knowing general information about the job shop conditions (Ragatz and Mabert 1984).
8. When estimating due dates, the use of more details provides only marginal improvement in the performance of the due date setting rules (Ragatz and Mabert 1984).

9. Due dates that are assigned according to analytical analysis are favorable (Baker and Bertrand 1981).

Dar-El and Wysk (1982) investigated the effect of the release mechanism on six dispatching rules using tardiness as a performance measure. The release of jobs to the shop floor is controlled by delaying jobs according to two actions known as flushed and un-flushed. An un-flushed action was taken when no more jobs were allowed to enter the system. A flushed action is taken when all remaining jobs are completed, and all machines in the job shop are empty. The following were the dispatching rules used: SPT; FCFS; LCFS; EDD; OST, and LAWINQ. The researchers simulated a job shop with four machines which had three types of load (70%, 77%, and 85%). Dar-El and Wysk concluded that the best rules that should be selected to manage a job shop with such behavior were SPT and LAWINQ.

The effect of dispatching rules when incorporating machine breakdowns was investigated by Muhlemann, Lockett, and Farn (1982). They tested twelve dispatching rules using seven performance measures. The twelve dispatching tested were: RANDOM; FCFS; EDD; SPT; LWR; SPT*; S/OPN; ODD; LCFS; CR; OST, and a composite rule which was developed by Farn (1979). The seven performance measures were: lateness; makespan; conditional lateness; percentage of jobs late; average queue time; mean tardiness, and average ration of flow time to process time. Their job shop had twelve machines and processed twelve products. They tested four cases of breakdowns where each had different arrival times and repair times. Also, Muhlemann, Lockett, and Farn included a rescheduling factor in their experiment. This factor was handled by having two sets of jobs waiting for any machine. The first set had the initial jobs, and the second set had the newly arrived jobs. The rescheduling was done in a certain frequency to include the newly arrived jobs in the initial set. From the results obtained, Muhlemann, Lockett, and Farn concluded that, in general, the SPT rule was the best when rescheduling was infrequent. However, the SPT^* and the composite rules were far better than the SPT when rescheduling was performed frequently. The mean tardiness was minimized by JST, SPT, and EDD rules. The CR rule minimized the conditional mean lateness. Frequent rescheduling resulted in better performance for the shop.

Elvers and Taube (1983) studied the effects of efficiencies and inefficiencies of machines and workers on five dispatching rules (SPT, EDD, JST, OST, and FCFS) using percentage of jobs completed on time as a criterion. In other words, they studied the effect of workers' learning and loss of knowledge in terms of the processing times. To represent workers' learning and loss of knowledge using the processing times, the processing times were fluctuated accordingly. In their experiment, Elvers and Taube compared two cases which they called stochastic and deterministic. The stochastic case is with efficiencies and inefficiencies of machines and workers. The deterministic case is without efficiencies and inefficiencies of machines and workers. The study simulated a job shop with eight machines and six types of loads which ranged from 84.5% to 97.9% of capacity. From their results, it is clear that when the job shop is heavily loaded, SPT was superior. However, when the job shop load was under 91.6%, EDD, JST, OST, and FCFS were superior to SPT. Finally, they concluded that the incorporation of efficiencies and inefficiencies in terms of the processing did affect the performance of the dispatching rules in most situations.

Russell, Dar-El, and Taylor (1987) simulated an open job shop to test three alternative formulations of COVERT rule and ten other dispatching rules to test the effect of due date tightness. The ten dispatching rules were: FCFS; EDD; JST; S/OPN; SPT; MDD; MODD; ATC, two versions of TSPT. Eight performance measures were used: average flow time; average tardiness; average conditional tardiness; average lateness; root mean square of tardiness; root mean square of conditional tardiness; percent tardy job, and maximum tardiness. The job shop simulated, as designed by Baker (1984), consisted of four machines which had a 90% utilization level. From their results, it is clear that the SPT rule was superior in minimizing the average flow time, average lateness, and percent of job tardy. The lowest value for average conditional tardiness, root mean square of tardiness, and root mean square of conditional tardiness was achieved by COVERT rule. The MODD was superior in minimizing the average tardiness and TSPT was superior in minimizing maximum tardiness. For loose due dates (20% tardy), MODD was superior in minimizing all performance measures except for the average flow time which was minimized by SPT. The SPT was superior in minimizing the average flow time, the maximum tardiness, and average lateness when due dates were moderate (40% tardy). Also, under tight due dates, COVERT was superior in minimizing the average conditional tardiness, the root mean square tardiness, and the root mean square conditional tardiness. The MODD was superior in minimizing the average tardiness.

Sawaqed (1987) performed a study where he investigated a hybrid assembly job shop with bottleneck machine. He investigated the effect of the position of the bottleneck machine on various performance measures. Sawaqed tried to answer several questions in his study. However, the two most important questions that are related to our study are:

> ...Does the location of bottleneck machines influence the relative performance of dispatching rules? Is it sufficient to manage a job shop by managing its bottleneck machines? (Sawaqed 1987, ix)

To answer these two questions Sawaqed simulated a hybrid assembly job shop with nine machines, nine products, six criteria, and six dispatching rules. The load for non-bottlenecks was 75% and for the bottleneck it was 90%. Out of the nine products, there were four assembly products. The six criteria were: average flow time; average tardiness; average lateness; average staging time; percentage of tardy, and maximum tardiness. Six dispatching rules were used (FASFS, FCFS, SPT, EDD, AJF-SPT, and SRPT). The results of this investigation concluded that the location of the bottleneck machine does not affect the relative performance of the superior dispatching rules. For example, SPT will be superior wherever the bottleneck is.

Next, Sawaqed performed another experiment to investigate the effect of managing the job shop by managing its bottleneck machines. The bottleneck machines were identified by first identifying the average utilization level of all nine machines, then the machine with over 85% utilization level was identified as the bottleneck machine. In his experiment there were three bottleneck machines with a utilization level of 97%, 86%, and 95%. In terms of dispatching rules, Sawaqed developed and used four management policies to schedule jobs on bottleneck and non-bottleneck machines. These policies were: 1) EDD for both; 2) SPT for non-bottlenecks and EDD for bottlenecks; 3) EDD for non-bottlenecks and SPT for bottlenecks, and 4) SPT for both. Then Sawaqed (1987, x) concluded that "the most crucial element in managing a job shop is the management of its bottleneck machines."

Schultz (1989) developed a new rule that combined SPT with tardiness-based rules which was named CEXSPT rule. Schultz tested the CEXSPT and six other dispatching rules by simulating an open job shop that was designed by Russell, Dar-El, and Taylor (1987). The six dispatching rules used were: MODD; COVERT; SPT; ODD; S/OPN, and OCR. Four performance measures were used which were: average flow time; average tardiness; average conditional tardiness, and proportion of job tardy. Schultz concluded that the SPT was superior in minimizing the average flow time, CEXSPT was superior in minimizing average tardiness, and COVERT was superior in minimizing average conditional tardiness. Both MODD and SPT were superior in minimizing the proportion of job tardy.

Vepsalainen and Morton (1987) developed and tested the effect of the ATC rule which considered the influence of multiple machines by using look-ahead parameters. They compared the ATC rule with five dispatching rules using three performance measures. The five dispatching rules were: FCFS; EDD; OST; WSPT, and COVERT. The four performance measures were: the normalized weighted tardiness; percentage of jobs tardy; the work-in-process, and the work-in-system. Vepsalainen and Morton simulated three types of job shops with ten machines and five shop loads (80%, 85%, 90%, 95% and 97%). Vepsalainen and Morton generalized their conclusions for the three types of job shops because of similar patterns. For all utilization and under tight due dates, they ranked the dispatching rules to minimize the weighted tardiness as follows: ATC; COVERT; WSPT; OST; EDD, then FCFS. In all utilization levels and when the due dates are loose, the ATC was ranked first to minimize the weighted tardiness, COVERT was second. When due dates are loose and the utilization is low (<90%), the OST was ranked third, but, with high utilization (>90%), the WSPT rule was ranked third. The ATC rule was the best under all utilization levels and due dates types to minimize the percentage of jobs tardy. When due dates were tight and utilization was low (<85%), COVERT performed better than WSPT, but WSPT was better when the utilization level was higher than 85%. Also, when due dates were loose and utilization was lower than 95%, COVERT performed better than WSPT, and when the utilization was higher than 95%, WSPT performed better. The WIP was minimized by the ATC and EDD rules when the utilization was high (≥ 90%) and the due dates were loose. However, when the utilization was lower than 90%, the WSPT rule was the first to minimize WIP, then the ATC and the EDD rules. In all shop loads and under tight due dates, the EDD was the best rule to minimize the WIP. The EDD and OST rules were the best for WIS under tight due dates and all utilization levels. The EDD rule was superior in all utilization levels when due dates were loose. However, when the utilization was lower than 85%, the ATC rule was ranked second, but when the utilization was higher than 85%, the OST rule was ranked second.

The computations of the ATC and COVERT rules required the computation of the expected waiting time for each operation of each job under consideration. Vepsalainen and Morton (1987) used a unique method to compute the expected waiting time which was a multiplier of the processing time of a specific job under consideration ($W=aP_{ij}$, where a is the multiplier and Pij is the processing time of operation j for job i). Therefore, Vepsalainen and Morton (1988) continued their research and investigated the effect of different estimates of the expected waiting times on

ATC and COVERT. They tested three methods to estimate the waiting time, using the models of the previous study. These three methods are: multiple of processing time (STD); priority-based (PRIO), and lead-time iteration (ITER). They found that an accurate estimate of the waiting time helped the ATC and the COVERT rules to reduce the tardiness. With respect to minimizing tardiness, the ATC/ITER combination was the best minimizer and COVERT/PRIO was the second.

Anderson and Nyirenda (1990) developed two new methods to compute the operation due date. These two methods were: CR+SPT and S/RPT+SPT. Using the CR+SPT, the due date for operation j of job i is computed as follows: $ODD = \max(OCR^{*}P_{ij}, P_{ij})$. Also, the S/RPT+SPT computed an operation due date as follows: $ODD = \max(S/RPT^{*}P_{ij}, P_{ij})$. Anderson and Nyirenda simulated an open job shop with eight machines to compare the performance of these two methods when they were used to computed the operation due date in the MODD rule. Also, they compared the performance of the MODD with four other dispatching rules. These rules were: SPT; CEXSPT, and two versions of COVERT. Four performance measures were used: mean flow time; mean tardiness; proportion of tardy jobs, and conditional mean tardiness. The shop load was kept at 90% utilization level. The results of this study indicated that the SPT rule was superior in minimizing the average flow time in all due dates types, and also superior in minimizing the percentage of jobs tardy when due dates were tight. When the due dates were very tight, the MODD rule was superior in minimizing the mean tardiness. The S/RPT+SPT rule was the best to minimize the mean tardiness when due dates were moderate, and superior in minimizing the percentage of jobs tardy when due dates were loose. The CR+SPT rule was better than S/RPT+SPT rule in minimizing the average tardiness when due dates were loose.

Raghu and Rajendran (1993) developed a new dispatching rule that is sensitive to the machine utilization level, job processing time, and operation due date. Raghu and Rajendran tested their rule against six dispatching rules (SPT, EDD, MOD, ATC, S/RPT+SPT, and CR+SPT).

Four performance measures were used: average flow time; average tardiness; percentage of jobs tardy, and root mean square of average tardiness. They simulated an open job shop with twelve machines and two shop loads (86% and 95%). The results of this study indicated that at 85% utilization level and in all cases of due dates and processing times, RR and SPT rules performed equally and they were the best to minimize the average flow time. However, when the utilization level was 95%, the RR rule was ranked first and SPT was second. For both the average tardiness and root mean square of average tardiness, the RR rule was superior in combinations tested. The S/RPT+SPT and the CR+SPT rules were ranked second to minimize the average tardiness. The EDD was ranked second with respect to root mean square average tardiness. For percentage of jobs tardy, the SPT rule was ranked first, the S/RPT+SPT rule was ranked second, then the RR rule was ranked third.

A similar study to Dar-El and Wysk (1982) was recently performed by Rohleder and Scudder (1993b). In this study, four job release mechanisms were tested to minimize earliness and tardiness simultaneously. The four release rules were immediate release (IR), modified infinite loading (MIL), modified Ow and Morton (MOM), and operation early or tardy release (OETR). The release time in the IR rule was the arrival time of the job. The MIL rule derived its release time by using the attributes of jobs and the shop congestion. The MOM rule obtained its release time by using the job's due date, processing times, and early and tardy costs. The OETR rule used overall information of the job, and produced release times for each operation of each product at all machines. The OETR rule forced machines to have two queue types, which were active and inactive. The active queue kept jobs that had been released, and the inactive queue held jobs that had not yet been released. The active and inactive behavior performed by the OETR rule simulated the construction of a delay schedule and the other three release rules used a non-delay schedule. Four dispatching rules were used: FCFS; EDD; weighted COVERT, and modified ATEC. Rohleder and Scudder simulated an open job shop with six machines with three levels of utilization

(70%, 80%, and 90%). The results of this study indicated that in terms of dispatching rules, the modified ATEC was superior in all cases tested. The OETR rule was the best at all utilization levels and due date types. When utilization was high, the IR was ranked second, but, as utilization levels decreased, MOM competed with IR. In terms of importance between dispatching rules and release rules, they conclude that dispatching rules were effective in reducing early or tardy costs when the utilization was high and due dates were tight. However, release rules were effective when the utilization level was low and the due dates were loose.

Bahouth and Foote (1994) developed and implemented three dispatching rules to manage two bottleneck machines in a hybrid assembly job shop with one assembly machine. The three dispatching rules were developed by using Johnson's flow shop algorithm. The three developed rules were:

1. JNP: Johnson No Priority rule. Parts were scheduled or rescheduled in all machines according to the sequence obtained by Johnson's algorithm which was applied on the two bottlenecks.
2. JHP: Johnson Half Priority rule. Parts were scheduled or rescheduled according to JNP, but the first priority was given to a part on which only one operation was performed. If ties occurred among the jobs that were prioritized by JHP, then JNP was used. JHP was only applied before the assembly operation.
3. JFP: Johnson Full Priority rule. Parts were scheduled or rescheduled according to JNP, but the first priority was given to a part on which the maximum number of operations was performed. If ties occurred among the jobs that were prioritized by JFP, then JNP was used. JFP was applied at any machine.

The purpose of this study was not only the development of new dispatching rules but also the investigation of the management of two bottlenecks in a hybrid assembly job shop. Bahouth and Foote simulated a job shop with nine machines where two of them were bottlenecks. The total flow time was used as the performance measure. They studied the effect of five factors. These factors were: the interarrival times; percentage deviation between the assumed process time and the actual process time; the difference in average processing time between the two bottlenecks; the dispatching rules, and location of the bottlenecks. The performance of the three dispatching rules was compared with the performance of a superior rule, which was SPT. For the two bottlenecks, six locations were selected. These locations were: 1) the first two stages; 2) the first stage and the second-to-last stage; 3) the first and last stages; 4) the second and last stages; 5) the last two stages, and 6) the second stage and the second-to-last stage. The results of this study indicated that for the time between job creations, the JNP rule performed better than the other rules. Also, they found that for the difference in the average process time between the two bottlenecks, the SPT rule was superior when the two bottlenecks were located in the first two stages. The JNP rule performed better than the other rules when the two bottlenecks were located in the last two stages. The SPT rule performance deteriorated when there was more than one non-bottleneck machine between the two bottlenecks. Finally, Bahouth and Foote concluded the following:

> ...Flow shop sequencing rules can be applied to manage job shops: When a job shop has two bottleneck machines, a modified version of the Two-Machine Flow Shop Johnson rule can be used... The above results can only be applied to cases when the two bottleneck machines are not on parallel branches of the product structure, and when jobs use the two bottleneck machines in the same sequence. (Bahouth and Foote 1994, 2476)

Shifting Bottleneck Algorithm

Several heuristics have been developed by several researchers to solve the job shop problem. Most of these heuristics have been reported by Conway, Maxwell, and Miller (1967), Baker (1974), Rinnooy (1976), Bellman, Esogbue, and Nabeshima (1982), French (1982), Morton and Pentico (1993), and Pinedo (1995). However, this section will be devoted to the recently developed heuristic which is known as the Shifting Bottleneck (SB) algorithm. The SB algorithm was developed in 1988 by Adams, Balas, and Zawack. Then, in 1993 it was modified by Dauzere-Peres and Lasserre. The SB algorithm was recently extended by Balas, Lenstra, and Vazacopoulos (1995). The SB algorithm is reviewed in this research study for four reasons:

1) It is the only well-known heuristic that simulates the management of bottleneck machines in the job shop environment.
2) It is known to be superior among all heuristics that were used to solve the job shop problems.
3) The SB algorithm and genetic algorithms were combined in several implementations.
4) The results obtained by the SB algorithm has been used as a benchmark to test the performance of several genetic algorithms.

The SB algorithm was developed to solve the general sequencing problem that was defined in chapter I where the makespan was minimized. The idea of the SB algorithm was described by Adams, Balas, and Zawack (1988), who stated:

> ...We sequence the machines one at a time, consecutively. In order to do this, for each machine not yet sequenced we solve to optimality a one-machine scheduling problem that is a relaxation of the original problem, and use the outcome both to rank the machines and to sequence the machine with highest rank. Every time a new machine has been sequenced, we reoptimize the sequence of each previously sequenced machine that is susceptible to improvement by again solving a one-machine problem. (Adams, Balas, and Zawack 1988; 393)

The above description of the SB algorithm can be re-stated as follows. The SB algorithm sequences machines sequentially one at a time. The machines that have not yet been sequenced are ignored, and the machines that have been sequenced have their sequences held fixed. At each step, the SB algorithm determines a bottleneck machine from the set of machines that have not yet been sequenced by performing two steps:

1. Solving a one-machine scheduling problem for each un-sequenced machine.
2. The machine that yielded the maximum makespan is selected to be the bottleneck machine from the set of machines that have not yet been sequenced.

Then, the associated sequence that was obtained by the one-machine scheduling problem is used to sequence the bottleneck machine chosen. Every time a bottleneck machine is sequenced, a re-optimization procedure for the set of machines that have been sequenced is performed. The re-optimization is performed by freeing up and re-sequencing each machine in turn with the sequences on the other machines held fixed.

To test the quality of solutions obtained by the SB algorithm, several small problems for which an optimal solution was known were solved by Adams, Balas, and Zawack. Also, this team solved large problems which had up to 500 operations and ten machines. From the results obtained, they found out that the SB algorithm was able to find the optimal solution in all the problems with ten machines and over 30 jobs. The SB algorithm found in five minutes the optimal solution to a difficult problem that was designed by Fisher and Thompson (1963). In comparison, the optimal solution had only recently been found with extensive effort. The SB algorithm determination of the bottleneck machine was stable and accurate when there were many more jobs than machines, and this situation led the SB algorithm to converge to the optimal solution.

Adams, Balas, and Zawack solved forty problems to compare their algorithm to ten dispatching rules. These dispatching rules were FCFS, late start time (LST), early finish time (EFT), late finish time (LFT), most immediate successor (MIS), first available (FA), SPT, LPT, RANDOM, and JST. For the forty problems, they did not report the results of each dispatching rule. However, for each problem, they reported the best solution obtained by one of the dispatching rules and compared it to the solution obtained by the SB algorithm. From the results reported, the SB algorithm dominated in 38 problems.

Dauzere-Peres and Lasserre (1993) observed one of the weaknesses of the SB algorithm which was "...When the procedure fixes the sequence on a machine, it may thereby create a precedence constraint between some pair of jobs on some unsequenced machine" (Balas, Lenstra, Vazacopoulos 1995, 95). This problem was treated by both Dauzere-Peres and Lasserre (1993) and Balas, Lenstra, Vazacopoulos (1995). Dauzere-Peres and Lasserre (1993) developed a heuristic to solve the one-machine problem with delay precedence constraints. Also, an optimization procedure was developed by Balas, Lenstra, Vazacopoulos (1995) for solving the one-machine scheduling problem with delay precedence constraints. Both studies reported that the improved SB algorithm obtained better results than the original SB algorithm.

Genetic Algorithms

This section is devoted to describing the genetic algorithms which were developed by Holland in 1975. Since genetic algorithms (GAs) are adaptive and flexible, they have attracted several researchers from different fields such as computer sciences and engineering, operations research, business, social science, etc. As Holland puts it "...The last five years have seen the number of researchers studying genetic algorithms increase from dozen to hundreds" (Holland 1992, ix). The theory and the application of the GAs have been reported by several researchers. Some of these reports are by Grefenstette (1985, 1987), Goldberg (1989), Liepins and Hilliard (1989), Schaffer (1989), Belew and Booker (1991), Davis (1991), Rawlins(1991), Holland (1992), Forrest (1993), Lin (1993), Michalewicz (1994), Srinivas and Patnaik (1994), Chambers (1995a, 1995b), Eshelman (1995), Mattfeld (1996), Osman and Kelly (1996), and Gen and Cheng (1997). In these reports, the GAs were shown to be successfully applied to several optimization problems. For example, they have been applied to routing, scheduling, adaptive control, game playing, cognitive modeling, transportation problems, traveling salesman problems, optimal control problems, database query optimization, etc.

The GAs are stochastic search techniques whose search algorithms simulate natural phenomena (biological evolution). The basic idea of the GAs is that the strong tend to adapt and survive while the weak tend to die. One of the strengths of GAs is that they use past information to direct their search with the assumption of improved performance. The formal description of the GA which was provided by Grefenstette is as follows:

> ...A genetic algorithm is an iterative procedure maintaining a population of structures that are candidate solutions to specific domain challenges. During each temporal increment (called a generation), the structures in the current population are rated for their effectiveness as domain solutions, and on the basis of these evaluations, a new population of candidate solutions is formed using specific genetic operators such as reproduction, crossover, and mutation. (Grefenstette 1985)

Holland's original GA is known as the simple GA (SGA). The SGA works with a population of binary chromosomes (chromosomes are also called strings or individuals). Each chromosome is made of genes of 0s and 1s (genes are also called features, charters, alleles or decoders). As mentioned earlier, the GAs have attracted several researchers. The binary representation of population, however, was not suitable for all applications. Hence, the population representations have changed from binary to various representations such as real values, integer values, characters, lists of rules, etc. These changes have been made to simplify the representation of the population of chromosomes to be appropriate for the problem under consideration. Michalewicz elaborated on this subject:

> ...Classical genetic algorithms, which operate on binary strings, require a modification of an original problem into appropriate (suitable for GA) form, this would include mapping between potential solutions and binary representation, taking care of decoders or repair algorithm, etc. This is not usually an easy task. (Michalewicz 1994, 7)

A rich discussion about the unsuitableness and suitableness of binary representation of populations can be found in Michalewicz (1994, 1-10). The general procedures of the GA are as follows:

1. Initialize a population of binary or non-binary chromosomes.
2. Evaluate each chromosome in the population using the fitness function.
3. Select chromosomes to mate (reproduction).
4. Apply genetic operators (crossover and mutation) on chromosome selected.
5. Put chromosomes produced in a temporary population.
6. If the temporary population is full, then go to step 7. Otherwise, go to step 3.
7. Replace the current population with the temporary population.
8. If termination criterion is satisfied, then quit with the best chromosome as the solution for the problem. Otherwise, go to step 2.

In the above steps, there are several elements that need to be explained. The first element is the size of the population and how to generate the initial population. The initial population of chromosomes can be generated randomly or by using some heuristics that are suitable for the problem considered. The determination of the population size is a crucial element in the GAs. Selecting a very small population size increases the risk of prematurely converging to a local optimal. Large population sizes increase the probability of converging to a global optimal, but it will take more time to converge. In most of the GA applications, the population size was maintained at a constant.

The second element of the GAs is the fitness function, which is very important to the GAs process of evolution. The GA without a fitness function is blind because, as mentioned earlier, the GA directs its search using historical data which are the fitness values of each chromosome. The GA will use the fitness value of each chromosome to determine if the chromosome can survive and produce offspring, or die.

The selection of chromosomes to reproduce is the third element of the GA. This is a very important element in the GA because it plays an important role in the convergence of the GA. If the selection process is always biased to only accept the best chromosome, the algorithm will quickly have a population of almost the same chromosomes which will cause the GA to converge to a local optimum. Several selection methods have been employed by several researchers to select among the best performers. Some of these methods are: the proportional selection scheme; the roulette wheel selection; deterministic selection; ranking selection; tournament selection, etc.

In step four, two genetic operators were used. The first operator is crossover, which combines the features of two fittest chromosomes and carries these features to the next generation by forming two offspring. The SGA performs the crossover by selecting two chromosomes and a random crossover position (single-position crossover method), then the corresponding parental segments are swapped to form two new children. Several crossover methods have been developed and applied to binary representation. One of them is the two-position crossover method, which is performed by selecting two crossover positions in two chromosomes and then swapping segments between the two chromosomes. The multi-position crossover method is a natural extension of the two-position crossover. A version of the multi-position crossover method is the segmented crossover method, which varies the number of segments during the implementation of the GAs while the multi-position crossover uses a fixed number of segments. Shuffle crossover was proposed as a crossover method which first shuffles the crossover positions in the two selected chromosomes. Then it exchanges the segments between the crossover positions and finally un-shuffles the chromosomes. The final crossover method proposed is the uniform crossover, a generalization of the one-position and multi-position crossover methods. The uniform crossover method produces two new children by exchanging genes in two chromosomes according to a crossover probability and a random value given to the same gene in both chromosomes. The random value assigned to each gene is uniformly distributed between 0 and 1 and denoted by X_i, $i = 1,\ldots,n$ where n is the number of genes. The uniform crossover is performed as follows: Let P_1 and P_2 be two parents in which each has n genes so that $P_1 = \{P_{11}, P_{12}, P_{13},\ldots, P_{1n}\}$ and $P_2 = \{P_{21}, P_{22}, P_{23},\ldots, P_{2n}\}$. These two parents will produce two children which are denoted by C_1 and C_2. Hence, if the crossover probability is P_c, then the uniform crossover is performed as follows:

$$\text{If } X_i < P_c \text{ then } C_{1i} = P_{1i} \text{ and } C_{2i} = P_{2i} \text{ and If } X_i \geq P_c \text{ then } C_{1i} = P_{2i} \text{ and } C_{2i} = P_{1i}$$

To demonstrate how the uniform crossover method works, assume that there are two chromosomes and each gene is assigned a random value as shown below:

P1: 0110000111, and P2: 0001011111

Assume X_i:= 0.79, 0.83, 0.44, 0.88, 0.11, 0.89, 0.59,0.7, 0.45, and 0.14, for i = 1,...,10.

Assume that the P_c is 0.5. The implementation of the uniform crossover method will result in the following children:

C1: 0011011111, and C2: 0100000111

The second operator is mutation, which alters one or more of the chromosome genes randomly to ensure search diversification, which hopefully will lead the population out of the local optimum. In the SGA approach, the mutation is performed by first selecting a mutation position. Then, if the gene value is 0, it is flipped to 1. If the gene value is 1, then it is changed to 0.

Finally, the last element in the GA procedures is the stopping criterion. Several criteria have been suggested. One of them is that the GA will stop if the maximum number of generations has been reached, or if the population has converged. The convergence of the population has been interpreted by researchers through several measures. One of them is that the GA converges after a chromosome with a certain high fitness value is located. Another one is that the GA converges after all chromosomes have attained a certain degree of homogeneity (that is, all of them have almost the same fitness value).

The following example will demonstrate the above producers of the GA. Assume that there is an initial randomly generated population of four binary chromosomes which each contain ten genes:

Chromosome 1: 0000000111 Chromosome 2: 0001011111

Chromosome 3: 0110101011 Chromosome 4: 1111111011

The fitness function is assumed to be the sum of ones in a chromosome. Therefore, chromosomes 1, 2, 3, and 4 have fitness values of 3, 6, 6, and 9 respectively. If the minimum value is sought, then the best performer is chromosome 1. Assume chromosomes 1 and 2 are arbitrarily selected. To perform crossover, assume that the two-position crossover method is used. The two positions are denoted by "|" below.

Parent 1: 000|0000|111 Parent 2: 011|1001|110

When the two segments of the genes between the two crossover positions in each parent are exchanged, these two parents will produce the following children:

Child 1: 0001001111 Child 2: 0110000110

To perform mutation, selection a mutation position (denoted by "|") in child 2, and the result of the mutation method is given below:

Child 2: 01|1|0000110 results of mutation Child 2: 0100000110

The fitness values for the newly generated chromosomes are 5 and 3 for children 1 and 2 respectively. These two children will replace the two parents that produced them, and the final population will be as follows:

Child 1: 0001001111 Child 2: 0100000110

Chromosome 3: 0110101011 Chromosome 4: 1111111011

At this point, the GA procedure is terminated with child 2 as the best solution obtained for the problem under consideration with an objective function value of 3.

Genetic Algorithms and Sequencing and Scheduling Problems

As mentioned earlier, a binary representation of a population was not suitable for all applications. One of the applications that the binary representation was not suitable for, but can be applied to, is the combinatorial optimization problems. Some of these combinatorial optimization problems are the traveling salesman problem (TSP), the bin packing problem, the job scheduling problem (JSP), the plant layout, etc. Several representations of population have evolved from the applications of the genetic algorithms (GAs) to the TSP. Because of the similarities between TSP and JSP, these representations have been used in JSP. In the following paragraphs, population representations and the associated genetic operators that have been applied to JSP will be discussed. The only representation that will be discussed in detail is the order-based representation because it is used in this study.

Ordinal representation was developed by Grefenstette et al. (1985). It was developed to represent a population in a GA approach that solved a TSP. In the ordinal representation method, all classical crossover methods that were explained earlier can be applied to the ordinal representation method. However, the classical mutation method cannot be applied because there is no gene that can be flipped to either 0 or 1. Therefore, several mutation methods have been developed to handle such population representations and other representations. One of these mutation methods is the simple inversion, which is performed by first selecting two mutation positions in a chromosome. The segment between these two positions is reversed. The second mutation method, called insertion, is where a gene is selected and inserted in a random place. Displacement is the third method, which is performed by selecting a string of genes which is inserted in a random position. Order-based mutation (OBM) is the fourth method, which selects two genes randomly and swaps them. A version of the order-based mutation is position-based mutation (PBM), which selects two genes randomly and then inserts the second gene before the first. Scramble sub-sequence mutation (SSM) is another mutation method, that selects a sub-sequence in a chromosome, and scrambles the genes in the sub-sequence to produce a new chromosome.

The second representation method is an order-based representation (also called permutation ordering representation, path representation, natural representation, or direct representation) where a chromosome is represented by a sequence of jobs. This method has been applied extensively by several researchers. These studies were attempted by Liepins et al. (1987), Cleveland and Smith (1989), Bagchi et al. (1991), Falkenauer and Bouffouix (1991), Syswerda (1991), Stöpller and Bierwirth (1992), Fang, Ross, and Corne (1993), Gupta, Gupta, and Kumar (1993), Neppalli (1993), Vempati, Chen, and Bullington (1993), Gen, Tsumjimura, and Kubota (1994), Sridhar and Rajendran (1994), Bierwirth (1995), Bierwirth, Kopfer, Mattfel, and Rixen (1995), Chen, Vempati, and Aljaber (1995), Croce, Tadei, and Volta (1995), Kobayashi, Ono, and Yamamura (1995), Lee and Choi (1995), Reeves (1995), Rubin and Ragatz (1995), and Mattfeld (1996).

In the order-based representation method, a chromosome is formed as a sequence of jobs, such as: 4-6-9-7-5-3-1-2-8. This chromosome is interpreted as follows: job 4 is sequenced first, job 6 is sequenced second, and likewise until job 2 is sequenced second to last, and then job 8 is sequenced last. Clearly this representation is simple and has a meaningful interpretation. All mutation methods that are applied to the ordinal representation method can be applied to the order based representation method. However, infeasible chromosomes will be generated when the classical crossover methods that was explained in the previous section are performed. The infeasible chromosomes produced by the classical crossover can be demonstrated by the following example. Assume that in the initial population there are two parents which are:

Parent 1: 4-6-9-7-5-3-1-2-8 and Parent 2: 8-2-4-6-9-1-3-5-7

A single-position crossover method is performed on the two parents, where the single-position crossover is denoted by '|' as shown below.

Parent 1: 4-6-9-7-5-3-|1-2-8 and Parent 2: 8-2-4-6-9-1-|3-5-7

The result of the crossover is shown below:

Child 1: 4-6-9-7-5-3-3-5-7 and Child 2: 8-2-4-6-9-1-1-2-8

It is obvious that both of the children represent infeasible sequences because both of them have only six jobs out of the nine jobs, and each has three duplicated jobs. Therefore, to solve this infeasibility problem, several crossover methods that produce feasible chromosomes were proposed by several researchers:

1. Order Crossover (OX) by Davis (1985).
2. Partially Mapped Crossover (PMX) by Goldberg and Lingle (1985).
3. Sub-sequence-Swap crossover (SSX) and Sub-sequence-Chunk Crossover (SCX) by Grefenstette et al. (1985).
4. Cycle Crossover (CX) by Oliver, Smith, and Holland (1987).

5. Edge Recombination Crossover (ERX) by Whitley, Starkweather, and Fuguay (1989).
6. Linear order Crossover (LOX) by Falkenauer and Bouffouix (1991).
7. Order-based Crossover (OBX) and Position-based Crossover (PBX) by Syswerda (1991).
8. Enhanced edge recombination crossover (EERX) by Starkweather et al. (1991).

Only four crossover methods (PMX, LOX, OBX, and PBX) will be explained in this study for two reasons. First, the PMX has been extensively used in the GA implementations and hence is discussed in following paragraphs. Second, the other three crossover methods (LOX, OBX, and PBX) will be discussed later in Chapter III because they are implemented in this study. For detailed explanations of the other crossover methods, the reader can refer to references associated with each method or refer to Michalewicz (1994) and Gen and Cheng (1997).

The PMX was developed by Goldberg and Lingle (1985) to handle the infeasibility problem in a GA approach that was applied to TSP. However, it has been applied by several researchers to solve JSP (Liepins et al. (1987), Cleveland and Smith (1989), Bagchi et al. (1991), Gupta, Gupta, and Kumar (1993), Vempati, Chen, and Bullington (1993), Sridhar and Rajendran (1994), and Chen, Vempati, and Aljaber (1995)).

Given two parents, the PMX first randomly selects two positions which are the same in both parents. Then segments between these two positions are exchanged. The exchanging of the segments will define a series of mappings between genes. The defined mappings will be used to replace genes that are causing infeasibility in the new chromosomes. The following example will show how the PMX works assuming that the following parents are given:

Parent 1: 4-6-9-7-5-3-1-2-8 and Parent 2: 8-2-4-6-9-1-3-5-7

The two cutting positions on the two parents are selected where the two positions are denoted by '|' as shown below:

Parent 1: 4-6-|9-7-5-3|-1-2-8 and Parent 2: 8-2-|4-6-9-1|-3-5-7

The result of the segment swapping is shown below:

Child 1: x-x-|4-6-9-1|-x-x-x and Child 2: x-x-|9-7-5-3|-x-x-x

From the segments swapped, the defined mappings are as follows: 4⇔9, 6⇔7, 9⇔5, and 1⇔3. Therefore, the defined mappings will be used to correct infeasibility. In parent 1, job 4 is mapped as follows: 4⇔9⇔5 (job 4 is replaced with job 5). Job 6 is replaced with job 7. Job 1 is replaced with job 3. Both jobs 2 and 8 are not causing infeasibility, hence, they are not involved. In parent 2, job 3 is replaced with job 1. Job 5 is replaced with job 4, because of the mapping, 5⇔9⇔4. Job 7 is replaced with 6. The result of the PMX is two feasible children given below:

Child 1: 5-7-|4-6-9-1|-3-2-8 and Child 2: 8-2-|9-7-5-3|-1-4-6

The order-based representation can be easily interpreted and applied to single machine and flow shop problems because both the single machine and the flow shop problems are permutation scheduling problems. However, a job shop problem is not a permutation scheduling problem and hence the order-based representation is not easily interpreted and applied to job shop problems. As a result of this difficulty, several variations of the order-based representation have been developed to handle the interpretation problem faced in the job shop implementations. These variations will be discussed in the next section.

A binary representation of the population was applied by Nakano and Yamada (1991). Both the classical crossover and mutation methods were applied. A random key representation method was developed by Bean (1994), and implemented by Norman and Bean (1994). In the random key representation method, all the classical crossover and mutation operators can be applied.

As mentioned earlier, the population representations can be represented by various representations such as integer values. The integer value representation of population was suggested by Dorndorf and Pesch (1995). They proposed two GA applications to use this type of representation. In the first, the chromosomes were formed of genes which represented an integer value which corresponded to a dispatching rule number from a given list of dispatching rules. The integer values in the second application depict a machine number. This means a chromosome was formed of genes each of which represented a machine number from a list of machines that were in the shop. In this representation, all classical crossovers will always produce feasible chromosomes. Also, all mutation methods that are applied to the ordinal representation method can be applied to this representation.

Applications of genetic algorithms to Sequencing and Scheduling Problems

In this section, a listing of most of the genetic algorithm (GA) studies that have been applied to all sequencing and scheduling problems will be given. However, since the focus of this research study is the job shop problem, the GAs that have been applied recently to job shop problems will be discussed in more depth.

The GAs were applied to single machine problems by Liepins et al. (1987), Gupta, Gupta, and Kumar (1993), Lee and Choi (1995), Lee and Kim (1995), and Rubin and Ragatz (1995). Liepins et al. (1987) applied a GA approach to minimize lateness. In their study, they compared the performance of three crossover methods (PMX, greedy weak crossover heuristics, and greedy powerful crossover heuristic). They concluded that PMX dominated both crossover methods. Gupta, Gupta, and Kumar (1993) tried to minimize flow time variance using a GA approach. In their study, they tested the effect of the GA parameters, which were population size, number of generations, problems size, crossover rate, and mutation rate. They found that most of these parameters have significant effects on the GA approach--especially the population size and the number of generations. Only the crossover rate had an insignificant effect. Lee and Choi (1995) applied a GA approach to solve a single machine problem where the total earliness and tardiness penalties was minimized. Lee and Kim (1995) developed a parallel GA to solve a single machine, using a common due date where the weighted sum of the total earliness and total tardiness was minimized. A GA approach to handle sequence dependent set-up time has been applied by Rubin and Ragatz (1995) where the total tardiness was minimized.

Cleveland and Smith (1989) used a GA approach to solve a flow shop problem where the total flow time was minimized. Neppalli (1993) tested the effect of the genetic parameters on the GA approach, using both total flow time and the makespan as performance measures. Neppalli concluded that the application of GAs are problem dependent, and the non-random initial population has a significant effect on the GA convergence. A GA approach was used to minimize the C_{max} in flow shop problems by Stöpller and Bierwirth (1992), Vempati, Chen, and Bullington (1993), Sridhar and Rajendran (1994), Chen, Vempati, and Aljaber (1995), and Reeves (1995). Stöpller and Bierwirth (1992) developed a parallel GA to the solve the flow shop problem. Reeves (1995) compared GA and simulated annealing, and found that when the problem is small, the two are comparable, but as the problem gets bigger, the GA performs better.

Davis (1985) was the first to apply GAs to job shop problems. However, he was not the only one. Several researchers have been attempting to solve the job shop problem using GAs. These attempts were made by Bagchi et al. (1991), Falkenauer and Bouffouix (1991), Nakano and Yamada (1991), Fang, Ross, and Come (1993), Gen, Tsumjimura, and Kubota (1994), Norman and Bean (1994), Bierwirth (1995), Bierwirth, Kopfer, Mattfel, and Rixen (1995), Kobayashi, Ono, and Yamamura (1995), Croce, Tadei, and Volta (1995), Dorndorf and Pesch (1995), and Mattfeld (1996).

Davis (1985) presented a conceptual and instructional study to show how the GA can be applied to job shop. Davis attempted to solve a job shop problem, using an indirect representation of the population which allows the use of Holland's crossover operator. Davis represented a chromosome as a preference list of operations where the chromosome is time dependent and machine controlling. Each machine has a list of these chromosomes, which are activated sequentially as time passes. Davis's representation of each chromosome has four elements. The first element is the activation time of the chromosome. The second element is a preference list of operations, and the third and fourth elements are keys to control the machine, which are 'wait' and 'idle'. However, for reasons that have been reported by several researchers, Davis's work can be summarized by the following statements:

> ...The performance of the Davis-style approach in initial runs on Problem 1 was not particularly notable. Some improvement was observed over time, but the final solution obtained was not as good as that obtained by the standard-GA. (Cleveland and Smith 1989, 167)
>
> ...Davis (1985) uses an intermediate representation which is guaranteed to produce legal schedule when operated upon by genetic recombination operators. However, the example used is not very complicated, and there are no significant results. (Bagchi et al. 1991, 11)
>
> ...Davis (1985) discusses a more indirect encoding that permits the use of the traditional crossover operator. For this encoding, a chromosome consists of a sequence of job preferences combined with times at which these job preferences become active. However, this encoding suffers from inflexibility due to the need to determine an appropriate time scale and appropriate machine idle and waiting time periods. (Norman and Bean 1994, 6)
>
> ...Davis (1985) presented an application of genetic search to a simple job shop scheduling problem. The focus of the paper was on developing a workable representation of the problem. Only a single example problem was presented, with very limited computational experience. (Rubin and Ragatz 1995, 87)

Bagchi et al. (1991) developed and implemented a GA approach to solve a job shop problem. They designed a hypothetical job shop that had three machines and could process three products. The eleven orders produced by the job shop were orders for one of three products with a specific batch size. Each of the three products had several alternative process plans, including three process plans for product one, and two process plans for products two and three. All the process plans had three operations except one. All operations could be processed by two alternative machines except two of them were processed by only one alternative machine.

In their study, Bagchi et al. used three representations of the population which are variants of the order-based representation. The first representation is a simple order-based representation, but the second and third representations are known as problem-specific-based representation. In the first representation, each gene in a chromosome represented the order priority. A chromosome in the second representation was formed by genes that had two elements. The first element of a gene was the order priority, and the second was the process plan assigned to the order. The third representation was the same as the second representation; however, the third representation was more specific than the second representation. In the second element of the gene, the third representation not only assigns a process plan to an order, but also specifies the machines to perform the operations in the process plan assigned. Bagchi et al. compared the three representations using machine utilization as the performance measure and found that the third representation was superior. The major conclusion of their study was:

> ...To enhance the performance of the algorithm and to expand the search space, a chromosome representation which stores problem-specific information is devised. (Bagchi et al. 1991, 10)

Falkenauer and Bouffouix (1991) solved a job shop problem using a GA approach where jobs had different release times. Falkenauer and Bouffouix used an order-based representation version which is known as preference-list-based representation. In this representation a chromosome is formed by several sub-chromosomes. These sub-chromosomes contain genes which represent the preference list for a specific machine. Each gene in the sub-chromosome represents an operation to be performed on that machine. For example, if there are three machines in the job shop, then there will be three sub-chromosomes in a chromosome. Also, if each machine performs five operations, there will be five genes in each sub-chromosome. In their implementation, each chromosome was evaluated, using a simulation model for the problem under consideration. The LOX and PMX were used as the crossover operators and inversion was the mutation operator. Each of these crossover methods was implemented on two chromosomes by crossing the first sub-chromosome of one parent with the first sub-chromosome of the other parent, the second with the second, and likewise until the last with the last.

Falkenauer and Bouffouix performed their experiment using three job shop models which they called small, big, and giant. The small model had 24 operations, the big had 60 operations, and the giant had 250 operations. In their GA approach they maximized the difference of the summation of weighted earliness and the summation of squared tardiness where the earliness was given a weight between 0 and 1. Falkenauer and Bouffouix used a pilot study to determine the GA parameters. From the pilot study, they fixed the following parameters: crossover rate was 0.6; mutation rate was 0.033; the population size was 30, and the number of generations was 100. To evaluate the performance of the GA, they used the following dispatching rules: SPT and JST. Falkenauer and Bouffouix performed ten replicates for each model mentioned above. From their results, they concluded the following: the GA is superior when compared to the dispatching rules, and LOX performed better than PMX.

Nakano and Yamada (1991), as mentioned in the previous section, developed a GA approach to solve job shop problems using binary representation of the population. The classical crossover and mutation operators were applied as they were by Holland. They evaluated their chromosomes using semi-active schedules. In their experiment they solved three well-known problems designed by Fisher and Thompson (1963). From their results, it was clear that their GA approach obtained results comparable to the results obtained by other approaches.

Fang, Ross, and Corne (1993) and Gen, Tsumjimura, and Kubota (1994) implemented GA approaches that utilized a variant of an order-based representation known as operation-based representation. In this representation a chromosome is formed by genes which represent an integer value which corresponds to a job number. In each chromosome, a job's number will be repeated according to its number of operations. Therefore, a chromosome becomes a sequence of operations for all jobs. For example, if there are three machines and three jobs in the job shop and all jobs go through all machines, then there will be 9 genes in a chromosome as follows: 3-1-1-3-2-3-2-1-2, where the first 3 stands for operation 1 of job 3, the first 1 stands for operation 1 of job 1, the second 1 stands for operation 2 of job 1, and likewise until the third 2 stands for operation 3 of job 2. In the chromosome given, each job was repeated three times because each of them had three operations. The given chromosome can be interpreted when the process plan of each job is given. Hence, assume that the process plans for jobs 1, 2, and 3 are as follows: 1-2-3, 1-3-2, and 2-1-3 respectively (where numbers in the process plans indict the machine number). Then, the chromosome above can be interpreted as follows: job 3 is processed first at machine 2, job 1 is processed first at machines 1 and 2, job 3 is processed second at machine 1, job 2 is processed third at machine 1, job 3 is processed first at machine 3, job 2 is processed second at machine 3, job 1 is processed third at machine 3, and job 2 is processed third at machine 2.

Gen, Tsumjimura, and Kubota (1994) implemented their GA approach to solve a job shop problem where the makespan was minimized. In their implementation, each chromosome was evaluated using deterministic Gantt charting. Specifically, for each chromosome, they constructed a semi-active schedule. Gen, Tsumjimura, and Kubota developed their own crossover operator which they named partial schedule exchange crossover (for detailed explanations for the developed crossover operator, the reader can refer to Gen and Cheng 1997). They developed their own crossover method because all the other crossover methods that can be applied to the order-based representation cannot be applied to operation-based representation. The OBM was used as the mutation operator and the elitist method was used as the production method. Dynamic population size was utilized where at the end of each generation the population size was increased by a percent of the summation of mutation and crossover rates. Then, the population size was reduced to the original size, where only the best individuals were selected from the inflated population size. Gen, Tsumjimura, and Kubota solved three well-known benchmarks from Fisher and Thompson (1963). In their experiment, they used the following parameters: crossover rate was 0.4; mutation rate was 0.3; the population size was 60, and the number of generations was 5000. They compared their results to branch and bound approaches and other GAs. From their results, it is clear that they performed better than the other GAs but not better than branch and bound approaches.

Norman and Bean (1994) performed a study in which they developed and implemented a GA approach to solve a job shop problem using a random key representation method. They designed the GA approach to solve a job shop with m machines and n jobs where these jobs arrive at the job shop separately. Also, setup times were sequence dependent, and machine down time and scarce tools were incorporated. The GA approach was applied to the described job shop model to minimize the total tardiness. In the GA implementation, the elitist method, which enforces preserving the best chromosomes, was used in the reproduction process. A variant of the binary tournament was used to select two chromosomes to reproduce. Uniform crossover and immigration mutation were the two genetic operators used. In every generation, the immigration mutation method inserted a new random chromosome. By using the immigration mutation, the study tried to eliminate the effect of the elitist reproduction, which causes premature convergence. In this study, the GA approach terminates if the best solution found has not changed for 15 generations.

Norman and Bean incorporated problem specific data to enhance the performance of the GA approach by using ready times and due dates to prioritize jobs. They stated:

> The scheduling application incorporates problems specific into the random keys encoding to improve the rate of convergence. Recall that for the general random keys encoding the random keys for all the genes are uniform (0,1) variates. The scheduling application contains problem specific data which can be used to bias the random key values of the jobs. If the problem objective is to minimize total tardiness then it is likely that jobs that have early ready and due times will be found early in the optimal sequence. Likewise, jobs with late ready and due times will probably be found late in the optimal sequence. (Norman and Bean 1994, 13)

The enhancement incorporated in their model was performed when the chromosomes were generated. That is, if job 5 has to be before job 2 in the optimal sequence, the uniform random number assigned to job 2 will be biased to be large (for example, the random number for job 2 will be uniformly distributed between 0.8 and 1 instead of being uniformly distributed between 0 and 1). By doing so, job 2 will usually be located in later positions in the sequence. On the other hand, job 5 will be assigned a smaller random number which will often locate it in earlier positions. The example given by Norman and Bean was not a good example to demonstrate the data specific enhancement. In addition, they did not give any explanations of how to handle difficult situations. This enhancement does not incorporate job processing times, which does not make it robust enough. The reason for not being robust enough is that their objective function, total tardiness, is a function of ready times, due dates, and processing times. Also, this enhancement is performed only on the initial population and not during the evolution process. This implies that this enhancement is predictive and not reactive.

Norman and Bean performed an elementary testing by solving three types of data sets. The first consisted of a single machine and 16 jobs. The second set had seven problems which each contained two machines and 350 jobs. Five problems were in the third data set, with each problem having ten machines and 250 jobs. For the first data set, ten replications were performed and the GA approach was able to obtain the optimal solution provided by Kanet and Sridharan (1991). They concluded that the results of all the data sets were encouraging, and claimed that the GA approach was good in solving the job shop problem.

Bierwirth (1995) developed a GA approach (GP-GA) to solve a job shop problem using an operation-based representation where the makespan was minimized. In the GP-GA, each chromosome was evaluated according to an active schedule. As mentioned earlier, all the crossover methods that can be applied to an order-based representation cannot be applied to operation-based representation. Therefore, Bierwirth developed a crossover method which is a generalization of OX (GOX). In the conducted experiment, the following parameters were used: the population size was 100, and two levels of the number of generations were 100 and 150. Ranking selection method was used to select chromosomes to reproduce. Bierwirth solved twelve standard problems which were designed by Fisher and Thompson (1963) and Lawrence (1984). Bierwirth performed a total of 100 replicates for the two problems that were designed by Fisher and Thompson and 25 replicates for the other ten problems that were designed Lawrence (1984).

From the results obtained, Bierwirth reported that the average solutions for all problems were within a percentage of deviation of errors that ranged between 0.7% and 7%. Also, Bierwirth concluded that the GP-GA was a promising approach. Bierwirth, Kopfer, Mattfel, and Rixen (1995) performed a preliminary study in which they extended the GP-GA to solve dynamic job shop problem where jobs had different release times.

Croce, Tadei, and Volta (1995) developed a GA approach to solve a job shop problem using a preference-list-based representation that was developed by Falkenauer and Bouffouix (1991). In their implementation, each chromosome was evaluated using a simulation model for the problem considered. Croce, Tadei, and Volta claimed that schedules produced by the simulation model were only non-delay schedules. Hence, they constructed schedules with look-ahead function to introduce delay. The look-ahead function used by Croce, Tadei, and Volta violated the definition of non-delay schedule to a certain extent so that some of the delay schedules could be incorporated in the final solution. The look-ahead function was accomplished as follows: when a machine finishes processing and becomes available to process the operations waiting for it, an operation with the highest priority will be scheduled to be processed. However, before scheduling this operation, the look-ahead function will first determine the processing time and the completion time of the candidate operation. Then, the look-ahead function will check to see if there is an operation which will arrive before the candidate operation finishes and has higher priority than the candidate operation. If there is an operation that satisfies both conditions, then the machine will stay idle until the new operation arrives. Otherwise, the candidate will be scheduled.

The LOX was the crossover method used by Croce, Tadei, and Volta. The OBM was applied by swapping genes within a sub-chromosome. The steady-state reproduction was the reproduction method used, where at each generation a number of new chromosomes were inserted. Croce, Tadei, and Volta performed a pilot study to determine the GA parameters. From the pilot study, they fixed the following parameters: crossover rate was 1; mutation rate was 0.03; the population size was 300, and ten new chromosomes were inserted at each generation for the reproduction method. Croce, Tadei, and Volta applied the GA approach developed to minimize the makespan using eleven standard problems by performing five runs for each of them. Three of these problems were designed by Fisher and Thompson (1963), and the other eight were designed by Lawrence (1984). The optimal solutions for these problems were provided by Fisher and Thompson (1963), and Lawrence (1984). Croce, Tadei, and Volta obtained the results for the eleven problems and compared the best obtained result for each problem with the best obtained results of three other studies which had solved the same eleven problems. One of these studies which solved the eleven problems by the simulated annealing (SA) algorithm was performed by Laarhoven, Aarts, and Lenstra (1992). The second study was performed by Dell'Amico and Trubian (1993) who solved the eleven problems using the tabu search (TS) approach. The Shifting Bottleneck (SB) algorithm (Adams, Balas, and Zawack 1988) was the third heuristic that also was used to solve the eleven problems.

From the results of this study and the other three studies, it is clear that the tabu search approach was superior. Out of the eleven problems, the TS converges to the optimal solution in ten problems. The SA approach found the optimal solution to 8 problems. The SB and GA found the optimal solutions to 7 and 6 problems respectively.

As mentioned earlier, Dorndorf and Pesch (1995) proposed a GA approach that used an integer value representation of population which was used to solve a job shop problem where the makespan was minimized. Recall from the previous section that they proposed two GAs, which they named P-GA and SB-GA. In the P-GA, each chromosome consisted of n-1 genes where n-1 is the number of operations in the problem under consideration. Each gene was represented by an integer value which corresponded to a dispatching rule number from a list of twelve dispatching rules (SPT, LPT, LRPT, SRPT, RANDOM, FCFS, TWORK, TLPT, MWR, LWR, longest operation successor, and longest operation reaming processing time). This implies that each gene can have an integer value between 1 and 12. In the P-GA, the schedules were constructed using an active schedule algorithm which was developed by Giffler and Thompson (1960). At each iteration of Giffler and Thompson's algorithm a conflict set of operations is formed which can have one or more operations. From the conflicting set of operations, an operation is selected randomly or by using a single dispatching rule. Hence, this selection problem motivated Dorndorf and Pesch to developed their P-GA approach, which was used to solve the conflict in selecting an operation. The selection of an operation was performed by referring to the gene that was associated with this operation, and this gene would prioritize this operation according to the relevant dispatching rule.

In the second application, the developed GA approach (SB-GA) was part of the shifting bottleneck (SB) algorithm. Recall that the SB algorithm sequences machines sequentially, one at a time until all machines are sequenced. It should be clear that the sequence of machine selection affect the quality of solutions obtained. Again, the selection problem motivated Dorndorf and Pesch to develop the SB-GA approach which controlled the machine selection at the first step of the SB algorithm. Each chromosome in the SB-GA approach consisted of m genes where m is the number of machines in the job shop. Each gene represented a machine number which could have any value between 1 and m.

Dorndorf and Pesch used three well-known benchmarks by Fisher and Thompson (1963) to tune their parameters. They used the elitist method in both GA approaches. For the P-GA, they used the following parameters: crossover rate was 0.65; mutation rate was 0.001; inversion rate was 0.7, and the population size was 200. In the SB-GA, mutation and inversion were not implemented, the crossover rate was 0.75, and the population size was 40. Dorndorf and Pesch randomly generated and solved 105 problems by the P-GA and the SB-GA, and then compared the results obtained to the results of four other heuristics. These were: a random selection; dispatching rules, and two versions of the SB algorithm. Also, they solved 40 problems that were designed by Lawrence (1984). Then they concluded that with respect to the makespan, the SB-GA performed better than the SB and the other heuristics. However, in terms of CPU time, the SB performed better than all heuristics. On the other hand, the SB algorithm dominated the P-GA approach in both time and objective function. The improvement gained by using the SB-GA over the SB algorithm was on the average very small. Also, the CPU time needed by SB-GA was increased by a huge percentage in both small and large problems.

Kobayashi, Ono, and Yamamura (1995) implemented a GA approach to solve a job shop problem where chromosomes were represented using the preference-list-based representation. In their implementation, each chromosome was evaluated using an active schedule. The OX and sub-sequence exchange crossover (SXX) were used as the crossover methods and mutation was not applied. Kobayashi, Ono, and Yamamura tuned their GA with two well-known benchmarks which were designed by Fisher and Thompson (1963). From the pilot study, they fixed the following parameters: crossover rate was 1.0, and the population size was 600. Random selection without replacement was used to select chromosomes. In their final experiment, they performed a total of 100 replicates for Fisher and Thompson's problems and they concluded that SXX performed better than OX, and the GA approach developed was promising.

Mattfeld (1996) developed three GA approaches to solve the job shop problem using operation-based representation. In all the GAs developed (GA1, GA2, and GA3), each chromosome was evaluated using a semi-active schedule, then the resultant schedule was re-optimized using a hill climbing algorithm. Also, a proportional selection method was used. Using GA1, they compared three mutation operators, PBM, OBM, and SBM, and concluded that PBM was the best. Also, using GA1, they compared two crossover operators, GOX and a developed version of PBX (called GPX). The conclusion of the second experiment was that the GOX was superior. Also, Mattfeld performed an experiment where the GA1 was compared with pure GA. The pure GA used neither semi-active schedules nor hill climbing algorithm. Then he concluded that GA1 achieved better results than the pure GA in fewer generations. The parameters used in the GA1 implementation were as follows: crossover rate was 0.6; mutation rate was 0.03; the population size was 100; the number of generations was 100, and the number of neighbors was 100. Using the parameters mentioned, Mattfeld solved twelve benchmarks to evaluate the performance of the GA1. Two of these problems were designed by Fisher and Thompson (1963), three of them were designed by Adams, Balas, and Zawack (1988), and the other seven were designed by Lawrence (1984). From the results obtained, Mattfeld reported that the average percentage error of deviation ranged between 1.3% and 4.8%. In the GA2, Mattfeld (1996) introduced structured population GA. Using the same parameters except for the crossover rate was 1, and the number of neighbors was 4, using a population structure of 10x10. In the GA2, Mattfeld used an acceptance criterion to either accept or reject the replacement of a parent by its offspring. The same twelve problems were solved by the GA2 and Mattfeld reported that the average percentage of errors ranged between 0.4% and 1.1%. The GA3 used the same parameters used by GA2, except the crossover and mutation rates were auto-adaptive. When the same twelve problems were solved by the GA3, the percentage of errors ranged between 0.3% and 1%.

Constrained Genetic Algorithm Study

In this section, an introduction will be given to the constrained genetic algorithm (CGA) which was developed by Al-Harkan and Foote (1994, 1996). The CGA was developed to address the single machine total weighted tardiness (TWT) problem which is strongly NP-hard. The proposed CGA approach obtained close to optimal solutions with much less deviation from optimal and much less computational effort than the conventional or unconstrained GA (UGA), which does not exploit the problem structure. This superior performance was achieved by combining sequencing and scheduling theory with the genetic algorithms methodology. Our approach can be called a hybrid GA, since it incorporates local search features in its procedures. However, we offer an additional feature that of constraining the order of certain elements of the chromosomes according to precedence relationships established theoretically. Hence, we called our approach a constrained GA. This section is organized as follows: in the following passage, the study motivation will be presented. Then the UGA and the CGA are introduced, followed by a section that will give tests and comparisons of the algorithms. Then concluding remarks which led to this dissertation topic will be given.

Motivation

This study was motivated by several scheduling problems that are classified as NP-hard problems which can be solved by using implicit enumerative methods which are branch and bound (B&B) and dynamic algorithm (DA). One of these problems is the total weighted tardiness. For large-sized problems, B&B and DA will take a long time to find the optimal solution; also, the time required by the B&B is unpredictable. Hence, these implicit enumeration methods are only efficient when time is not considered a factor. When faced with this reality, a search for a substitution method that is efficient and gives good results was the next alternative. Several methods have been found to solve such NP-hard problems: one of them is the GA approach. Researchers claim that GAs give fairly good and close to optimal solutions faster than the implicit enumeration methods. Wainwright expanded on that where he stated:

> The GAs are a robust search technique that will produce "close" to optimal results in a "reasonable" amount of time.... The GAs should be used when a good fitness function is available; when it is feasible to evaluate each potential solution; when a near-optimal, but not optimal solution is acceptable; and when the state-space is too large for other methods. (Wainwright 1993, 12-13)

Also, Koulamas, Antony, and Jaen elaborated on the robustness of these search techniques:

> OR researchers are increasingly turning towards new solution techniques such as neural networks, genetic algorithms, simulated annealing, and tabu search to solve management science problems. These techniques can be used as heuristics for finding near optimal solutions to a problem, and serve as alternatives to problem specific heuristics.... Typically, these techniques have been successfully applied to NP-hard problems. (Koulamas, Antony, and Jaen 1994, 41)

Knowing that the GA is fast and give fairly good results, the question that raised itself was how we could improve the quality of their solutions The answer to this question was the work performed in this study.

Unconstrained Genetic Algorithm

Before discussing the details of the CGA, an introduction to the unconstrained genetic algorithm (UGA) will be given. The UGA used the general GA procedures mentioned in the previous section. The following paragraphs describe the parameters that were used in the UGA. The UGA parameters were selected according to pilot runs that were done previously. These parameters are: the population size; the number of generations; the generation of the initial population; the selection methods; the reproduction methods (crossover and mutation), and termination criterion. The population size and the number of generations are determined as a function of the problem size (i.e., the number of jobs). The initial population for the UGA was randomly generated. Two selection methods were used in this study. The first method was the elitist method, which enforces preserving the best chromosomes in the reproduction process. Thus, at each generation the elitist method will be used to move a fraction of the population to the next generation. The second was a variant of the binary tournament that was suggested by Norman and Bean (1994). The variant method is performed by first randomly selecting two chromosomes from the population. Then the genetic operators are applied to these two chromosomes. Next, the best of the two produced chromosomes will be selected and allowed to enter the pool of the potential chromosomes for the next generation. The tournament procedures will be repeated until a new generation of chromosomes is produced. The linear order crossover (LOX) and order-based mutation (OBM) were used as the genetic operators. The UGA terminated its procedures when the maximum number of generations had been reached.

Constrained Genetic Algorithm

The section will give a discussion of the proposed constrained genetic algorithm (CGA). In the UGA, a random population of feasible sequences was generated to be used as an initial population. This starting initial population will affect the quality of solutions and the time taken to obtain the solution. This claim was the conclusion of a sensitivity study that will be discussed later. Hence, this step can be improved by using one of the heuristics that solve for the TWT. Three heuristics were used to generate three of the initial sequences. These three heuristics are the SPT, the EDD, and the ATC. Thus, when the CGA was implemented, three chromosomes were generated according to the SPT, the EDD, and the ATC heuristics. The rest of the population was randomly generated to avoid the bias that might be caused by the three heuristics.

As mentioned earlier, the OBM procedures were to select two jobs at random and swap them; however, swapping these two jobs could fail to satisfy standard dominance conditions of the TWT problem. Hence, dominance rules can be used to avoid dominated swapping of jobs, and so better objective values can be obtained. Two theorems can be used as dominance rules for the TWT problem. These are:

Rule 1: For two jobs j and k, if $P_j \leq P_k$, $d_j \leq d_k$, and $W_j \geq W_k$, then there exists an optimal sequence in which job j appears before job k.

Rule 2: If there exists a job k that satisfies $d_k \geq \sum_{j=1}^{n} P_j$, then there exists an optimal sequence in which job k is assigned the last position in the sequence.

The dominance rules were implemented on the children produced by the LOX operator by ordering the set of jobs located in the segment between the crossover positions according to a precedence constraint based on the dominance rule. The motivation behind only ordering the jobs in the crossover block was to avoid the bias that might be caused if the whole chromosome was sorted, which would tend to create a whole set of chromosomes that were similar, tending to localize the search. Further, sorting the whole chromosome is time-consuming. These two conjectures are the conclusions of a sensitivity test study that will be discussed in the following section. The UGA approach was modified to adopt all mentioned improvements, which resulted in the CGA approach. For detailed explanations for both the UGA and the CGA, the reader can refer to Al-Harkan and Foote (1994, 1996).

Algorithm Tests and Comparisons

The UGA and the CGA are stochastic in nature, which makes the theoretical analysis quite difficult, especially with the analysis of convergence. Therefore, the algorithm's behavior can only be determined computationally through a series of experiments which will be presented in this section.

In ordered to evaluate the performance of the UGA and the CGA, a DP approach was adopted to obtain the optimal solution for the problems that were solved in this research study. The adopted DP was proposed by Baker (1974). The three approaches were coded in FORTRAN 90 for a Gateway 2000 (Pentium-90) computer using the Microsoft FORTRAN PowerStation™, professional edition, version 4.0[1]. Microsoft Windows™ 95[2] was chosen to be the operating system. Several experiments were performed to compare the quality of solutions obtained by both the CGA and the UGA with the results obtained by a DP approach (which gives the optimal solution). Experiments conducted have four problem sizes, four problem types, four parameter cases, and three versions for the CGA and the UGA.

The quality of solutions obtained was measured by a percentage of deviation from optimal. The percentage of deviation is defined in terms of the performance measure used in this research, the TWT. The percentage of deviation measure was calculated for each of the two algorithms as

---

[1] Microsoft FORTRAN PowerStation is a trademark of Microsoft Corporation.

[2] Microsoft Windows 95 is a trademark of Microsoft Corporation.

follows:

$$\alpha_i = 100(TWT_i - TWT_{opt}/TWT_{opt})$$

where:

α_i The percentage deviation of the solution obtained by algorithm i from the optimal solution.

TWT_i The total weighted tardiness obtained by algorithm i (i.e., UGA, or CGA).

TWT_{OPT}: The total weighted tardiness obtained by the DP.

As mentioned earlier, the population size for both GAs was determined as a function of the problem size. Five population sizes were tested in the pilot study, which were 2n, 2.5n, 3n, 3.5n, and 4n. The two population sizes which gave good and similar results were 3.5n and 4n. Therefore, both population sizes were used in the final testing of the CGA and the UGA algorithms. The number of generations had three levels that were tested in the pilot study. These levels were: n^2; $n^{2.5}$, and n^3. Both n^2 and $n^{2.5}$ gave similar results and hence they were selected to participate in the final testing of the two GAs. In the elitist selection method, a fraction of the population is enforced in the next generation. The fraction value used in this study was 5%.

The two population sizes and two levels of the number of generations were used to support the hypothesis that the CGA will perform better than the UGA in terms of quality of solutions and computational effort when the population size and the number of generations are smaller. The motivation behind this hypothesis is that it is known in the GA community that increasing the population size and the number of generations should improve the performance of the GA. Therefore, it is desirable to have an algorithm that will achieve close to an optimal solution with much less deviation from optimal and with much less time. Also, it is more challenging for our proposed CGA approach.

The effect of the genetic operators on the performance of the CGA and the UGA was tested by implementing three versions for each of the two algorithms. The first version of the CGA and the UGA was implemented using only the mutation operator (named CGA_OBM and UGA_OBM). In the second version, only the crossover operator was implemented and the algorithms were named CGA_LOX, and UGA_LOX. The crossover and mutation operators were both implemented in the third version and the algorithms were named CGA_OBM_LOX and UGA_OBM_LOX.

To carry out these experimental tests, several groups of random problems were generated according to the following parameters: $n = 18, 20, 22$, or 24. P_j was uniformly distributed between (1, 10) or (1,5), W_j was uniformly distributed between (1,10), and d_j was uniformly distributed between $(P_j, P_j + k\,n)$ for $k = 1$ or 1.5 (k is the due dates factor). The largest problem size that was solved in this study was limited to twenty four jobs because of the computer memory limitations imposed by the DP approach. According to Emmons (1975), this method of generating the due dates provides relatively difficult problems for the algorithm to solve, which is desirable for a testing environment.

Thus, there were four cases for n, two cases for P_j, one case for W_j, and two cases for d_j, for a total of sixteen combinations of parameters. Nine problems were generated for each combination. The random number starting seed has an impact on the behavior of the UGA and the CGA. Therefore, to ensure the same testing environment, a different random number starting seed was used to implement each of the thirty-six instances which were associated with the same problem types and different problem sizes. For the 144 problems, the tests were performed as follows: 1) the process times for the jobs, the weights, and the due date were generated; 2) the computer codes for the CGA_OBM, CGA_LOX, CGA_OBM_LOX, UGA_OBM, UGA_LOX, UGA_OBM_LOX, and DP were implemented; 3) the TWT were computed and recorded; 4) the CPU times were recorded, and 5) the percentages of deviations were calculated.

The results for each combination obtained by the seven approaches were averaged over the nine instances relating to the same problem type, problem size, population size, and the number of generations. The results obtained are reported in Tables A.1 through A.16 in Appendix A. From the sixteen tables, it should be clear that the experiment was designed to have four problem types and four parameter cases. The structure of problem type I has low variability in the processing times (with variance of 1.3) and tight due dates. On the other hand, problem type II has low variability in the processing times but looser due dates. Problem types III and IV were structured to have high variability in the processing times (with variance of 6.75) with tighter due dates in the former, and looser due dates in the latter. The parameters for cases I and II were selected to have small population size with fewer generations in case I than in case II. Cases III and IV have large population size and fewer generations in the former, and more generations in the latter.

From the sixteen tables, it can be seen that the CGA outperforms the UGA in all three versions with respect to both the average percentage deviations and the average CPU time. From the results given in these tables, it is clear that the CGA deviated from the optimal solutions with an average of 9.3% or less while the UGA deviated with 12.81% or less. Also, the average percentage of deviations over all problem types, problem sizes, and parameter cases that were achieved by the CGA_OBM, CGA_LOX, and CGA_OBM_LOX were 2.89%, 0.57%, and 0.04% respectively. These averages achieved by the three versions of the UGA were 3.3%, 0.92%, and 0.05%. From these results, it can be implied that the CGA_OBM, the CGA_LOX, and the CGA_OBM_LOX reduced the percentage deviation by 15.6%, 61.95%, and 25% respectively. The maximum deviations from the optimal solutions were smaller for the CGA in two versions, but not statistically smaller. The percent of optimal solutions found by the CGA were highly significant at an α smaller than 0.000001. The maximum CPU time needed by the CGA and the UGA were 40.83 seconds and 66.96 seconds respectively.

From Tables A.2, A.3, A.4, and A.5, it can be seen that the maximum average deviations from optimal solution achieved by the CGA_OBM was 9.3% in case I, 5.89% in case II, 6.97% in case III, and 7.81% in case IV. The maximum average deviations for the UGA_OBM were 12.81%, 8.76%, 5.78%, and 8.1% in cases I, II, III, and IV respectively. In cases I and III, the CGA_OBM outperformed the UGA_OBM in eleven combinations. In nine combinations, the CGA_OBM performed better than the UGA_OBM in cases II and IV. These results support the hypothesis that the CGA would perform better than the UGA in terms of quality of solutions and computational effort when the population size and the number of generations are smaller. The performance of the UGA_OBM was consistent with the finding in the GA literature. The UGA_OBM performance improved when the number of generations increased, however, no significant improvement occurred when the population size was increased. The behavior of the CGA_OBM in case III was consistent and stable in all problem types. The CGA_OBM behaved by reducing the percentage deviation as the problem size increased. In case I, similar behavior was achieved by the CGA_OBM only for problem type II. The maximum deviations from the optimal solutions were smaller for the CGA_OBM in nine combinations in case I, in eleven combinations in case III, and in eight combinations in case IV. However, this accomplishment is not statistically significant. The percent of optimal solutions found by the CGA_OBM and UGA_OBM were 23.61% and 15.97% in case I, 36.11% and 34.72% in case II, 27.08% and 18.06% in case III, and 44.44% and 31.94% in case IV respectively (see Table A.14). This implies that the CGA_OBM found optimal solutions at statistically significant level.

From Tables A.6 through A.9, it can be seen that the CGA_LOX outperformed the UGA_LOX in fourteen combinations in cases I and II. In cases I and II, the maximum average deviation from optimal solutions obtained by the CGA_LOX was 2.24%, while it was 2.17% for the UGA_LOX. The CGA_LOX and UGA_LOX accomplished their solutions in cases III and IV with maximum average deviations of 0.96% and 2.32% respectively. In cases III and IV, the

CGA_LOX outperformed the UGA_LOX in eleven combinations. Again, these results support the hypothesis that the CGA should perform better than the UGA when the population size and the number of generations are smaller. The behavior of the UGA_LOX was not consistent with past GA literature. The UGA_LOX performed better when the population size was increased, but showed no significant improvement when the number of generations was increased. The maximum deviations from the optimal solutions were smaller for the CGA_LOX in all cases. In cases I and II, the CGA_LOX was smaller in ten combinations and in cases III and IV it was smaller in nine combinations. The percent of optimal solutions found by the CGA_LOX was 42.36% in cases I and II, and 46.53% in cases III and IV. For the UGA_LOX, these percentages were 15.27% in cases I and II, and 27.08% in cases III and IV (see Table A.15). Again, it can be implied from these results that the CGA_LOX found more optimal solutions at a highly statistically significant level ($\alpha = 0.000001$).

Tables A.10 through A.13 present the results obtained by the CGA_OBM_LOX and the UGA_OBM_LOX for the four parameter cases. From the results given in these tables, the CGA_OBM_LOX outperformed the UGA_OBM_LOX in six combinations in cases I and II, in seven combinations in case III, and in five combinations in case IV. These findings do support the hypothesis proposed in this study. The performance of the UGA_OBM_LOX was improved as the population size and the number of generations were increased. The maximum average deviations from the optimal solutions obtained by the CGA_OBM_LOX in all cases ranged between 0% and 0.33% and the range for the UGA_OBM_LOX was between 0% and 0.35%. The maximum deviations from the optimal solutions achieved by the CGA_OBM_LOX were larger in all cases. From Table A.16, it can be seen that the percent of optimal solutions found by the CGA_OBM_LOX was larger than the percent the UGA_OBM_LOX obtained. These percentages for the CGA_OBM_LOX were as follows: 92.36% in case I; 95.83% in case II; 94.44% in case III, and 96.53% in case IV. The percentages for the UGA_OBM_LOX were 87.5%, 88.88%, 89.58%, and 93.06 in cases I, II, III, and IV respectively.

From the above analysis of the results, it is clear that CGA_OBM_LOX performed better than both CGA_OBM and CGA_LOX with respect to the percentage deviation. This shows how strong the CGA is when both operators participated in the evolution process. In addition, the CGA_OBM_LOX was so robust that it diminished the normal behavior of the UGA. When the CGA_LOX and the CGA_OBM are compared, it is clear that the CGA_LOX was better. This implies that the crossover operator contributed to improving the performance of the CGA more than the mutation operator.

From the results given in Tables A.1 through A.13, it is clear that all the three versions of the CGA and the UGA needed less time to obtain near optimal solutions than the time needed by the DP approach in all problem types and parameter cases. The three versions of the CGA demanded less time than the UGAs versions in all problem types and parameter cases. The times needed by the CGA_OBM were 59% lower than the UGA_OBM demanded. Also, the CGA_LOX and CGA_OBM_LOX needed CPU times 40% lower than UGA_LOX and UGA_OBM_LOX needed. This implies that the CGA was the fastest approach in obtaining near optimal solutions in all problem types, problem sizes, parameters cases, and the three versions.

When the CGA was implemented, the starting population was seeded with three chromosomes generated according to three heuristics, which were the SPT, EDD, and ATC. Then the rest of the population was randomly generated. It was claimed earlier that this was done to enhance the performance of the CGA. To verify this claim, a sensitivity test study which involved 504 problems was performed to test the performance of the CGA when using the same random starting population used in the UGA procedures. The conclusion of the test study was that the quality of solutions obtained by the CGA was decreased by an average of 11.47% when the CGA used the same starting population. Further, the CPU time increased by an average of 5.4%. This implies that the performance of the CGA was improved when the initial population was seeded with the three heuristics.

As mentioned earlier, the dominance rules were implemented on the children produced by the LOX operator by sorting the set of jobs that are located in the segment between the crossover positions according to a precedence constraint that is based on the dominance rule. Thus, a sensitivity test study was done which involved 576 problems to test the performance of the CGA when the whole chromosome was sorted when the LOX procedures were performed. The results of this study were compared to the results of the UGA mentioned earlier. The conclusion of the test study was that sorting the whole chromosome doubled the CPU time based on our stopping rules. In addition, a set of chromosomes that were almost identical was formed, which localized the search. Also, sorting the whole chromosome improved the chance of performing better than the UGA by 8.8%.

Conclusion

From the computational results, it was clear that the CGA was better than the UGA in both quality of solutions obtained and the CPU time needed to obtain the close to optimal solutions in all the four cases. The three versions of the CGA reduced the percentages of deviations from optimal by 15.6%, 61.95%, and 25% respectively. Also, they obtained close to optimal solutions with 59% lower CPU time than the three versions of the UGA demanded. When the population size is 4n and the number of generations is $n^{2.5}$, the CGA_OBM_LOX was the best performer in terms of quality and effort in all problem types and sizes.

To sum up, the concept of the constrained search procedure is appealing and the CGA should be extended to solve larger job shop problem. This conclusion led to the topic of this dissertation, which is to extend the concept of the constrained GA procedure to solve large job shop problems.

Research Gaps

In the previous sections, several unexplored issues were mentioned. These unsolved issues not only originated from the literature review but also from the conclusion given in the "Constrained Genetic Algorithm Study" section. Some of the unexplored issues which could be investigated further in this research can be summarized as follows:

1. The constrained genetic algorithm described in the previous section has not been applied to solve job shop problems that have more than one machine.
2. The methods used to evaluate chromosomes are either simulation or deterministic Gantt charting. Both methods are extremes in the sense that deterministic is neither dynamic nor stochastic. Simulation can evaluate the chromosomes stochastically and dynamically, but it is very expensive since it requires large amounts of time. Evaluating the chromosomes using deterministic Gantt charting with stochastic process time has not been applied before.
3. In a job shop environment, machines are normally classified as critical and non-critical machines. The genetic algorithm treats machines equally and blindly. This implies that the genetic algorithm does not manage bottleneck machines. The management of bottlenecks has not been incorporated in the genetic algorithm.
4. Lee, Sikora, Shaw (1993) developed a genetic algorithm to optimize both the lot size for each product and the schedule makespan. Husband and Mill (1991) developed a genetic algorithm to optimize the process plan for each product. Parallel genetic algorithm models were solved simultaneously in which each had its own set of product process plans. Optimizing lot sizes and process plans simultaneously in a genetic algorithm has never been implemented. Also, simultaneous incorporation of lot sizes and process plans into the genetic algorithm representation has never been done.
5. Crossover and mutation operators depend on each other and support each other to improve the

performance of the genetic algorithm. Also, coupling one crossover method with different mutation methods would improve the performance of the genetic algorithm differently. The same behavior is true when one mutation method is coupled with a different crossover method. This conjunction was supported by Syswerda (1991) and Al-Harkan and Foote (1996). Syswerda (1991) tested the performance of a genetic algorithm using the following methods: order-based mutation (OBM); position-based mutation (PBM); order-based crossover (OBX), and position-based crossover (PBX). Syswerda tested the performance of each of the operators individually and when they were coupled. The OBM performed better than PBM when no crossover method was used. Also, PBX performed better than OBX when no mutation operator was used. When PBX was combined OBM, the best performance was achieved by the genetic algorithm. Also, Al-Harkan and Foote (1996) tested the performance of the genetic algorithm when only OBM was used, when only linear order crossover (LOX) was used, and when LOX and OBM were combined. The results showed that when LOX and OBM were combined, the genetic algorithm achieved its best performance. The following crossover and mutation methods were claimed to be good for order-based problems: LOX; PBX; OBX; OBM; SSM, and PBM. Therefore, combining each of these crossover methods with each of the three mutation methods has not been tested when they are implemented in a job shop environment.

6. As mentioned earlier, the binary tournament is performed by first randomly selecting two parents from the population. Then the genetic algorithm operators are applied to these two parents. Next, the best of the two produced offspring will be selected and allowed to enter the pool of the potential chromosomes for the next generation. These procedures will be repeated until a new generation of chromosomes is produced. In the genetic algorithm community this type of binary tournament is known as binary tournament with replacement, which means children will always replace their parents. The other extreme of this binary tournament is to

always select the best among the parents and children. This method is known as binary tournament without replacement. Both extremes have contributed to a major problem that the genetic algorithm has been criticized for, namely premature convergence. The premature convergence is caused by the loss of population diversity where the loss of diversity in the population can cause an increase in the selection pressure. To decrease the selection pressure and increase population diversity in the binary tournament, simulated annealing algorithm had been used by several researchers (Mahfoud and Goldberg (1992) and Chen and Flann (1994)). The binary tournament method utilizes the simulated annealing approach in making the decision whether to accept or reject a produced child according to a probability of acceptance which is the core of the simulated annealing approach. The incorporation of the simulated annealing approach in the genetic algorithm to solve job shop problem has not been applied before.

7. The general priority of the job shop (or individual priorities at each machine) dispatches jobs by either static discipline or dynamic discipline. To explain this statement, consider the following situation. When a machine finishes processing and becomes available to process the operations waiting for it, an operation with the highest priority in the priority list will be scheduled to be processed. However, if this operation has not yet arrived, there will be two actions possible in this situation. The first is to schedule the next operation that is waiting and is next on the priority list. This means that the dynamic discipline rule has been applied. The second action is to make the machine stay idle until the operation with the highest priority arrives. This means that the static discipline has been applied. The combination of both priorities with a control parameter to know when to switch between the two priorities has not been attempted. Also, the combining of these discipline in which the dynamic discipline is applied at the bottleneck machine and the static discipline is applied at the non-bottleneck machine has not been applied before.

CHAPTER III

RESEARCH PROGRAM

Introduction

This research study is an extension of the previous research that was concerned with applying the genetic algorithm (GA) to job shop problems. The main objective of this research is to answer the following questions:

1. Does the constrained genetic algorithm perform better than the unconstrained genetic algorithm when both algorithms are extended to solve dynamic stochastic job shops?
2. What is the impact of the population size on the accuracy of the deterministic constrained genetic algorithm to minimize makespan?
3. What is the impact of nine genetic operator combinations on the performance of the deterministic constrained genetic algorithm to minimize makespan and which of the nine genetic operator combinations would be the best?
4. Is the evaluation of the chromosomes using the probability Gantt charting as effective as simulation evaluation?
5. What is the performance of the stochastic constrained genetic algorithm to minimize total tardiness when lot sizes, process plans, and machine priority lists are optimized simultaneously?
6. What is the potential gain from incorporating the probability distribution function of the processing times in the genetic algorithm?

None of the above questions has been answered in the literature reviewed. Therefore, this study attempts to answer them. Before attempting to answer the six research questions, the GA approach needed to be extended to solve a large job shop model, which is the first part of the research program. Then, after extending the GA approach, a series of experiments was performed to answer the six research questions, which is the second part of the research program.

The organization of this chapter is as follows. In the following section, the structure of the extended genetic algorithm will be discussed. Next, a description of the genetic algorithm computer code logic and organization will be given. Then, an introduction to the deterministic genetic algorithms to minimize makespan will be given, followed by a discussion of the deterministic genetic algorithms to minimize total tardiness. Next, a description of the stochastic genetic algorithms will be presented, followed by a presentation of a dynamic stochastic genetic algorithm. Finally, a brief discussion of the pilot studies performed will be given.

Genetic Algorithm Structure

In this section, several elements and parameters for the GA approach will be discussed. These elements and parameters are: population representation method; schedule building and fitness function evaluation; population size; generation of the initial population; selection methods; crossover and mutation operators, and termination criteria.

Population Representation

In this study, a chromosome was represented by the representation method developed by Falkenauer and Bouffouix (1991). This representation method is known as the preference-list-based representation method. This representation method has been chosen in this study for several reasons:

1. It is the oldest representation method that was used to represent chromosomes in a job shop implementation (Davis (1985)).

2. It has been used more often than the other representation methods in job shop implementations (Davis (1985), Falkenauer and Bouffouix (1991), Croce, Tadei, and Volta (1995), and Kobayashi, Ono, and Yamamura (1995)).
3. It represents the solution space that this study attempted to optimize which is the solution space of the sequence of jobs at each machine.
4. It is the only straightforward representation for a job shop problem to search the solution space for the original attempted job shop problem.
5. It does not require encoding of genes, which is necessary for the other order-based representations.
6. It is the only natural extension of the representation method that was used in the single machine GA model.
7. It works with sub-chromosomes, which means it treats machines individually. This representation gives the GA designer and the scheduler the opportunity to easily differentiate among machines so that the focus can be directed to bottlenecks and non-bottlenecks.

As mentioned earlier, the population in the preference-list-based representation is represented by chromosomes formed by several sub-chromosomes. These sub-chromosomes contain genes which represent the preference list for a specific machine. Each gene in the sub-chromosome represents an operation to be performed on that machine. For example, if there are three machines in the job shop, there will be three sub-chromosomes in a chromosome as shown in Figure 1. Also, if each machine performs five operations, there will be five genes in each sub-chromosome. In Figure 1, two chromosomes are given: the first chromosome is 1-3-4-5-2-3-4-5-2-1-5-4-2-1-3, and the second chromosome is 2-3-5-1-4-4-5-3-2-1-5-2-3-1-4. By referring to chromosome 1 in Figure 1, it can be seen that the preference list of machines 1, 2, and 3 are 1-3-4-5-2, 3-4-5-2-1, and 5-4-2-1-3 respectively. From the preference list of each machine, it should be clear that job 1 is given the first priority at machine 1, job 3 is given the first priority at machine 2, and job 5 is given the first priority at machine 3. Also, jobs 2, 1, and 3 are given the last priority at machines 1, 2, and 3 respectively.

	The preference list of machine 1					The preference list of machine 2					The preference list of machine 3				
	Sub-chromosome 1					Sub-chromosome 2					Sub-chromosome 3				
	↓					↓					↓				
Chromosome 1	1	3	4	5	2	3	4	5	2	1	5	4	2	1	3
Chromosome 2	2	3	5	1	4	4	5	3	2	1	5	2	3	1	4

Figure 1. Chromosome representation.

Schedule Building and Fitness Function Evaluation

Chromosomes are usually evaluated by either simulation or deterministic Gantt charting. For the representation method used in this study, simulation evaluation was applied by Falkenauer and Bouffouix (1991) and Croce, Tadei, and Volta (1995) and the deterministic evaluation was applied by Kobayashi, Ono, and Yamamura (1995). When Falkenauer and Bouffouix evaluated their chromosomes using discrete-event simulation model, they gave full control to the discrete event calendar to construct their schedule types. This means that the simulation model could have produced either an active schedule or a non-delay schedule. Croce, Tadei, and Volta used simulation to evaluate their chromosomes. However, they claimed that schedules produced by the simulation model were only non-delay schedules. Hence, they constructed schedules with a look-ahead function to introduce delay in schedules constructed. By doing so, they were trying to construct both active and non-delay schedules. Kobayashi, Ono, and Yamamura evaluated each chromosome using deterministic Gantt charting, where schedules were constructed using an active schedule heuristic developed by Giffler and Thompson (1960).

These types of schedules used to evaluate chromosomes were defined and discussed by Baker as follows:

> ...The set of all schedules in which no local left-shift can be made is called the set of semiactive schedules.... This set dominates the set of all schedules, which means that it is sufficient to consider only semiactive schedules to optimize any regular measure of performance.... The set of all schedules in which no global left-shift can be made is called the set of active schedules, and is clearly a subset of the set of semiactive schedules..... ...in optimizing any regular

> measure of performance it is sufficient to consider only active schedules.... The number of active schedules still tends to be large, and it is sometimes convenient to focus on an even smaller subset called the nondelay schedules. In a nondelay schedule no machine is kept idle at a time when it could begin processing some operation.... ...the active schedules are generally the smallest dominant set in the job shop problem. The nondelay schedules are smaller in number but are not dominant.... ...the best nondelay schedule can usually be expected to provide a very good solution, if not an optimum. (Baker 1974, 181-187)

Also, Conway, Maxwell, and Miller defined and discussed the same schedule types as follows:

> ...Active schedules, schedules on which it is not possible to perform a left-shift on any operation. A lift-shift of an operation is any decrease in the time at which the operation starts that does not require an increase in the starting-time of any other operation.... ...a nondelay schedule; simply stated, there is no instance in which a job is delayed when the machine that is to process the next operation is available and idle.... Nondelay schedule are by definition a subset of the active schedules, but not a dominating subset in the same sense that the active schedules dominate the semiactive. It is not true that in every problem there is an optimal schedule among the nondelay schedules.... ...when one lacks a procedure for constructing an optimal schedule directly and must resort to a heuristic or sampling approach, it may be more profitable to address the nondelay schedules than active schedules even though one may, in doing so, forfeit the infinitesimal probability that an optimal schedule may be obtained. (Conway, Maxwell, and Miller (1967), 111-112)

To demonstrate the relationships among all type of schedules, a Venn diagram that explains the relationships is given in Figure 2 which is taken from Baker (1974).

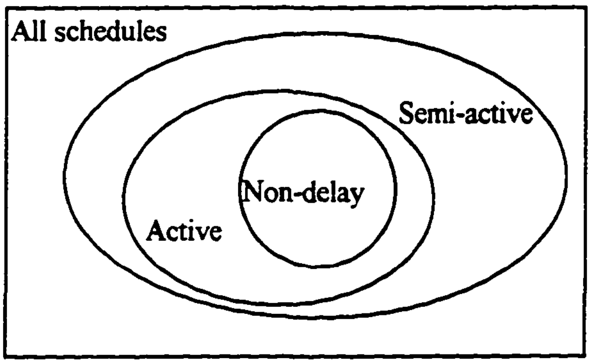
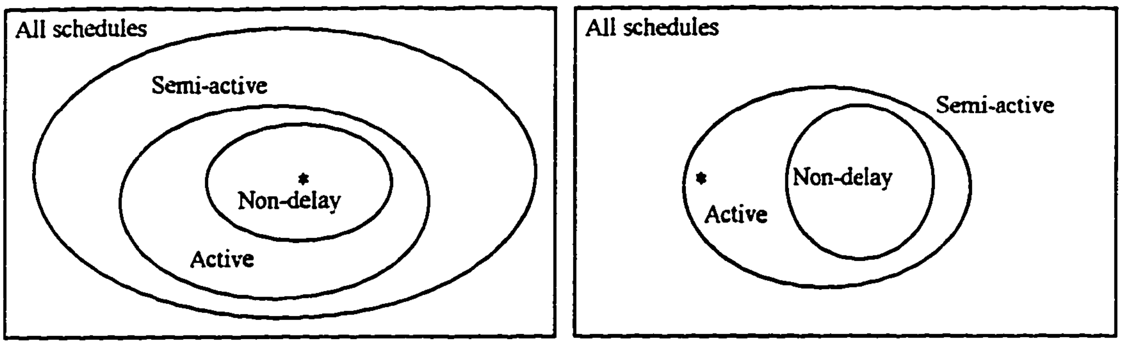


Figure 2. Venn diagram of schedule relationships.

The following two Venn diagrams were constructed to show the location of the optimal solution with respect to the schedule types. Figure 3.a illustrates an optimal schedule that is a non-delay schedule assuming that there is only one global optimal schedule. In Figure 3.b, a unique global optimal schedule is shown to be an active schedule.



(a) A situation in which the unique optimal solution is a non-delay schedule.

(b) A situation in which the unique optimal is an active schedule.

Figure 3. Venn diagram illustrating the optimal solution location with respect to the schedule types.

It should be clear that the location of the unique optimal schedule is not known in advance. Hence, it would seem to make sense to generate and to evaluate chromosomes according to both active and non-delay schedules. By doing so, the probability of finding the unique optimal is increased versus using only one of either schedule types. Therefore, in this study, chromosomes were generated in the initial population according to both active and non-delay schedules (see "Initial Population Generation" section for full details of the generation process). The generation process was as follows. The population was divided into two sub-populations. Then one sub-population was generated according to active schedules and the other sub-population was generated according to non-delay schedules. Also, during the evolution process, chromosomes were evaluated according to their original schedule type generator. This means that if a parent was generated according to an active schedule (or non-delay schedule), then all of its offspring produced throughout the evolution process would be evaluated according to an active schedule (or non-delay schedule). This implies that a child would inherit an original schedule type generator from one of its parents, which makes the original schedule type generator an attribute that was associated with each chromosome throughout the evolution process.

To generate both active and non-delay schedules, two heuristics developed by Giffler and Thompson (1960) were used in this study. For detailed explanations for both heuristics, the reader can refer to Baker (1974, 189-191).

When active and non-delay schedules were generated, the completion time for each job was obtained. These completion times can be used to compute any of the scheduling performance measures that belong to the three common types of decision making goals. Recall from Chapter I that these goals were: efficient utilization of machines; rapid response to demands, and close conformance to prescribed deadlines. In this study, two performance measures were selected to accomplish the first and last goals; these were the makespan and the total tardiness. These two performance measures were used as the fitness function in two separate GA models. Besides these two performance measures, the average flow time and the number of jobs tardy were computed. This means that the four performance measures were computed in every GA model, but one of them was minimized.

The purpose of computing the fitness function for all the GA models was to direct the search of the GA approach. However, the purpose of computing the other three performance measures was to break ties among chromosomes when the selection method was applied. When the makespan was minimized, the other three performance measures were used to break ties in the following order: 1) the total tardiness; 2) the number of jobs tardy, then 3) the average flow time. When the total tardiness was minimized, the order was as follows: 1) sum of the makespan and the average flow time, then 2) the number of jobs tardy.

Population Size

As mentioned earlier in the "Constrained Genetic Algorithm Study" section, the population size (Pop_size) was determined as a function of the problem size (i.e., the number of jobs). The population sizes were 3.5n and 4n, where n is the number of jobs. The same idea has been applied to the extended GA by using the number of machines and the number of jobs to compute the population size. Three population sizes selected were 44+nm, 44+2nm, and 44+4nm, where n is the number of jobs and m is the number of machines (i.e., Pop_size = 44+nm, 44+2nm, or 44+4nm). The reason for adding the constant to the population size will be discussed in the following section.

Initial Population Generation

In the developed constrained genetic algorithm discussed in the "Constrained Genetic Algorithm Study" section, three of the chromosomes in the initial population were generated according to three heuristics: the shortest processing time (SPT); the earliest due date (EDD), and the apparent tardiness cost (ATC). The rest of the population was randomly generated. Seeding the starting initial population with heuristics has two advantages: 1) makes the GA achieve good quality solutions in a shorter amount of time and 2) increases the efficiency of the GA, which also allows us to have a reasonable population size (i.e. smaller population size). Thus, the same concept was applied in the extended GA. The starting initial population for the GA was seeded with forty-four heuristics which were:

1. Earliest due date heuristic was used to generate both an active and a non-delay schedule: EDD(A) and EDD(ND).
2. Operation due date heuristic was used to generate both an active and a non-delay schedule: ODD(A) and ODD(ND).
3. Modified due date heuristic was used to generate both an active and a non-delay schedule: MDD(A) and MDD(ND).
4. Modified operation due date heuristic was used to generate both an active and a non-delay schedule: MODD(A) and MODD(ND).
5. Shortest processing time heuristic was used to generate both an active and a non-delay schedule: SPT(A) and SPT(ND).
6. Total work heuristic was used generate both an active and a non-delay schedule: TWORK(A)

and TWORK(ND).

7. Total shortest remaining processing time heuristic was used to generate both an active and a non-delay schedule: SRPT(A) and SRPT(ND).
8. Least anticipated work in next queue heuristic was used to generate both an active and a non-delay schedule: LAWINQ(A) and LAWINQ(ND).
9. Least work remaining heuristic was used to generated both an active and a non-delay schedule: LWR(A) and LWR(ND).
10. Smallest release time heuristic was used to generate both an active and a non-delay schedule: SRT(A) and SRT(ND).
11. Smallest ready time heuristic was used to generate both an active and a non-delay schedule: SORT(A) and SORT(ND).
12. Job slack time heuristic was used to generate both an active and a non-delay schedule: JST(A) and JST(ND).
13. Operation slack time heuristic was used to generate both an active and a non-delay schedule: OST(A) and OST(ND).
14. Slack over remaining work time heuristic was used to generate both an active and a non-delay schedule: S/RPT(A) and S/RPT(ND).
15. Weighted processing time plus weighted operation slack time heuristic was used to generate both an active and a non-delay schedule: WPT+WOST(A) and WPT+WOST(ND).
16. Critical ratio heuristic was used to generate both an active and a non-delay schedule: CR(A) and CR(ND).
17. Operation critical ratio heuristic was used to generate both an active and a non-delay schedule: OCR(A) and OCR(ND).
18. Cost over time heuristic was used to generate both an active and a non-delay schedule: COVERT(A) and COVERT(ND).

19. Apparent tardiness cost heuristic was used to generate both an active and a non-delay schedule: ATC(A) and ATC(ND).
20. Allowance over remaining number of operation heuristic was used to generate both an active and a non-delay schedule: A/OPN(A) and A/OPN(ND).
21. Total longest remaining processing time heuristic was used to generate both an active and a non-delay schedule: LRPT(A) and LRPT(ND).
22. Most work remaining heuristic was used to generate both an active and a non-delay schedule: MWR(A) and MWR(ND).

Then the rest of the population was generated according to four random heuristics:

1. Service in random order heuristic was used to generate both active and non-delay schedules: RANDOM(A) and RANDOM(ND).
2. Service in biased random order heuristic was used to generate both active and non-delay schedules in which the SPT rule and the EDD rule were equally likely to be selected to bias the probability of selection: Biased-RANDOM(A) and Biased-RANDOM(ND).

As mentioned earlier, two heuristics developed by Giffler and Thompson (1960) were used in this study to generate both active and non-delay schedules. At each iteration of Giffler and Thompson's heuristics a conflict set of operations is formed which can have one or more operations. From the conflicting set of operations, an operation is selected using one of the dispatching heuristics mentioned above. When the dispatching heuristic used does not resolve the conflict uniquely, a tie break rule is needed. In this study, the SPT rule was used to break ties.

Figure 4, shows the distribution of chromosomes in the population. Specifically, it shows the number of chromosomes generated according to active and non-delay schedules. Also, it shows the number of chromosomes generated according to dispatching heuristics using both active and non-delay schedules. In addition, it illustrates the number of chromosomes generated according to random and biased random heuristics using both schedule types.

← Pop_size →

Pop_size: The total number of chromosomes in the population.

← A →	← ND →
A: The total number of chromosomes that are generated according to active schedules = Pop_size/2.	ND: The total number of chromosomes that are generated according to non-delay schedules = Pop_size/2.

← AH →	← AR →	← ABR →	← NDH →	← NDR →	← NDBR →
AH: The total number of chromosomes that are generated according to active heuristics schedules = 22.	AR: The total number of chromosomes that are generated according to active random heuristic schedules = (A-22)/2.	ABR: The total number of chromosomes that are generated according to active biased random heuristic schedules = (A-22)/2.	NDH: The total number of chromosomes that are generated according to non-delay heuristic schedules = 22.	NDR: The total number of chromosomes that are generated according to non-delay random heuristic schedules = (ND-22)/2.	NDBR: The total number of chromosomes that are generated according to non-delay biased random heuristic schedules = (ND-22)/2.

Figure 4. Chromosomes distribution across the population.

From Figure 4, it should be clear that the sum of the parameters AR+ABR+NDR+NDBR is equal to Pop_size-44. Thus, if Pop_size is less than 44, this will cause the starting initial population to be seeded only with some of the forty-four heuristics Hence, to avoid this problem, a constant which is 44 was added to the three population sizes mentioned in the previous section (44+nm, 44+2nm, and 44+4nm). Doing so guaranteed that the starting initial population would be seeded with forty-four heuristics and a number of nm random heuristics, where n is the number of jobs and m is the number of machines.

Selection Methods

There are several selection methods that have been developed and implemented by many researchers. Two selection methods were used in this study. The first is the elitist method, which enforces preserving the best chromosome in the reproduction process. Thus, at each generation, the elitist method was used to move the best chromosome to the next generation. If there was a tie among chromosomes that had the same best solution, then one of them would be selected according to a random mechanism that assigned to each chromosome an equal probability of being selected. The selection probability was computed as 1/k, where k is the number of chromosomes that have the same best solution.

The second selection method is a variant of the binary tournament that was suggested by Norman and Bean (1994). In the binary tournament, the tournament size is two. The variant method is performed by first selecting randomly two parents from the population. Then the genetic operators are applied to these two parents. Next, the best of the two produced children will be selected and allowed to enter the pool of the potential chromosomes for the next generation. These procedures will be repeated until a new generation of chromosomes is produced. The binary tournament was used in this study for two reasons:

1. The binary tournament was used as the selection method in the GA discussed in the "Constrained Genetic Algorithm Study" section.
2. It opens the door for the GA designer to work with a major problem that the GA approach has been criticized for, which is the premature convergence.

The premature convergence is caused by the loss of population diversity, which increases the selection pressure. To decrease the selection pressure and increase population diversity in the binary tournament, simulated annealing (SA) algorithm had been used by several researchers (Mahfoud and Goldberg (1992) and Chen and Flann (1994)). The binary tournament method uses the SA approach in making the decision whether to accept or reject a produced child according to a probability of acceptance which is the core of the SA approach. Thus, in this study, the SA was incorporated in the GA approach when the binary tournament was applied. The SA approach and its parameters will be discussed later in this section.

As mentioned earlier in Chapter II, in the binary tournament there are several possibilities of competitions between two parents and two children. In this study, three binary tournaments were held. The first two tournaments were held between a parent and its child. The third tournament was a competition between the winners of the first two tournaments. Figure 5 shows a flow chart that demonstrates the selection process of parents and their children when the genetic operators are implemented.

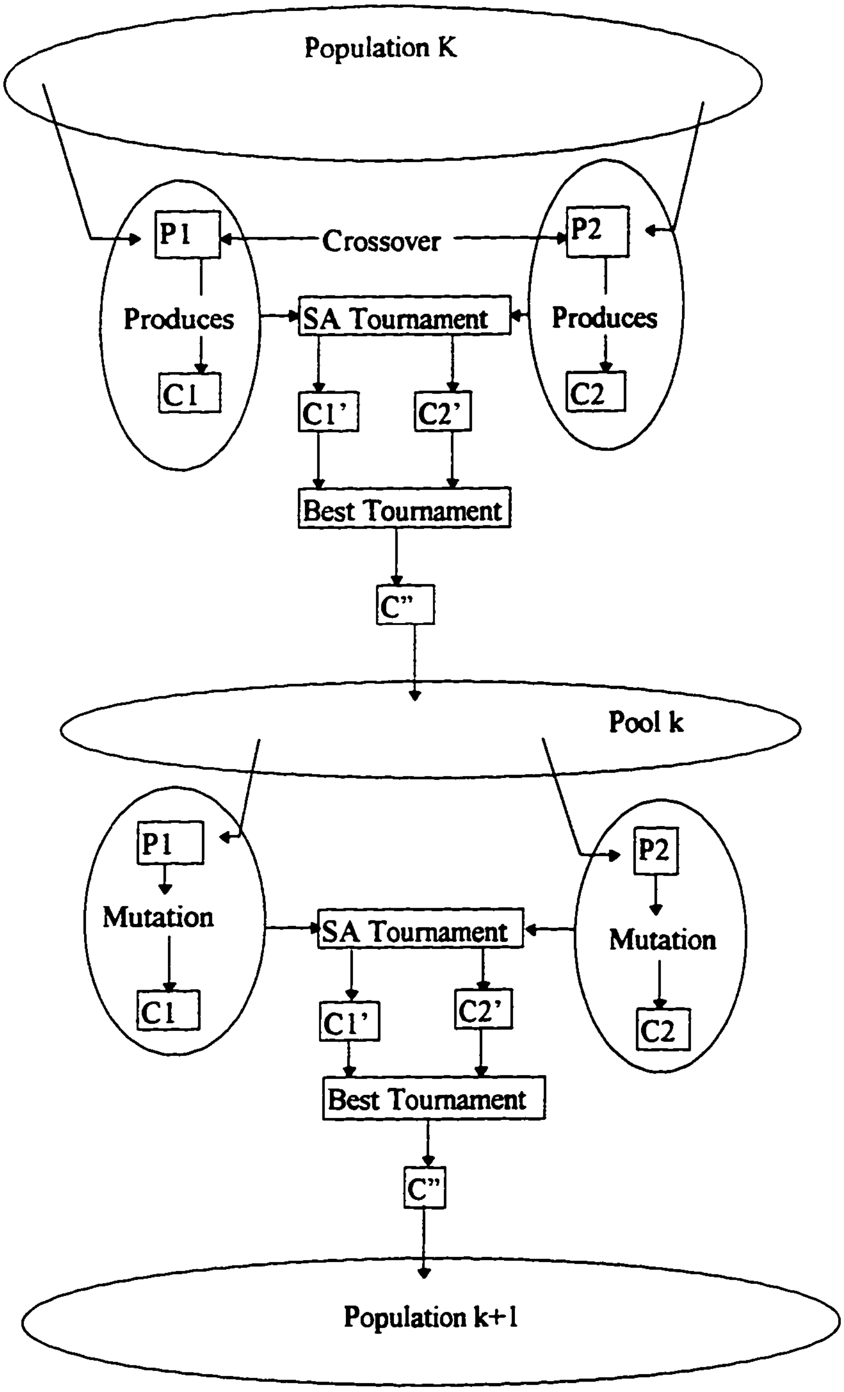


Figure 5. Binary tournament flow chart.

As shown in Figure 5, two parents (P1 & P2) were selected randomly from population k. Then, the crossover operator was applied to these two parents and two children were produced (C1 & C2). Next, two tournaments were held between each parent and its child using the SA approach. This implies that P1 competed against C1 and the result was C1', and P2 competed against C2 and the result was C2'. Then a third tournament was held between the winners of the first two tournaments, in which the better of the two was selected. This means that C1' competed against C2'. Next, the winner, C", entered pool k of the potential chromosomes for the next generation. These procedures would be repeated until a new generation of chromosomes was produced. Then the same steps were implemented using the mutation operator. When the mutation operator was applied, two parents were selected from the temporary pool k. Then the winner of three tournaments is allowed to enter population k+1, which is the population of chromosomes for the next generation.

Since the SA approach was incorporated in the GA approach, a brief introduction to SA will be given. The SA, one of the heuristic search techniques, was developed initially in the study of the cooling and annealing process of hot materials. The SA starts with an initial feasible solution which is randomly generated. Then it creates a neighbor solution of the initial solution by using some kind of perturbation function. Next, the change in the objective function is calculated. Assume that the minimization of the objective function is sought; if a reduction in the objective function is found by the neighbor, the current solution is replaced by the generated neighbor. Otherwise, an acceptance function (P) and a random number (x) are used to either accept or reject the neighbor solution. The acceptance function is computed as follows: $P = e^{(-\Delta f/T)}$ where Δf is the change in the objective function and T is the temperature. The SA compares the P to a random number x (where: x is uniformly distributed between 0 and 1) as follows: if $x < P$, then accept the neighbor solution (i.e., bad solution); otherwise, retain the previous solution. By accepting a bad solution, the SA attempts to avoid entrapment in a local optimum.

The annealing schedule consists of the following parameters: the starting temperature value (T_s); the final temperature value (T_f); a cooling parameter (CP); the number of iterations performed at each temperature (Z_1), and stopping criterion (Z_2). In this study, the starting and the final temperature values were computed using the acceptance function ($P= e^{(-\Delta f/T)}$) as follows:

$$T_s= -\Delta f_s / \ln(P_s) \quad \text{and} \quad T_f= -\Delta f_f / \ln(P_f)$$

Where:

T_s: The starting temperature.

Δf_s : The starting possible maximum difference in the objective function. In this study, Δf_s was estimated as one standard deviation from the average of the objective function of chromosomes in the initial population.

P_s: The starting probability of accepting a bad solution. The P_s was assigned a random value that was uniformly distributed between 0.8 and 0.99.

T_f: The final temperature.

Δf_f: The final possible minimum difference in the objective function. In this study, Δf_f was assigned a value of 1 when the fitness function was the makespan and the total tardiness. Also, Δf_f was equal to 0.001 when the fitness function was a utility function that was associated with each chromosome, which will be explained later.

P_f: The final probability of accepting a bad solution (P_f=0.01).

The temperature was reduced every Z_1 iteration as follows: CP*T, where Z_1= 2(Pop_Size-1)+2(0.4Pop_size). The CP was computed as follows: CP= $\log(T_f/T_S)/\log(Z_2)$, where Z_2 is the number of generations to reach the freezing stage. The Z_2 was uniformly distributed between 75 and 125. All of the above parameters were selected according to pilot runs that have been done, which will be discussed in the "Pilot Investigations" section.

Crossover and Mutation Operators

Starkweather et al. (1991) compared six crossover methods: partially mapped crossover (PMX); order crossover (OX); cycle crossover (CX); enhanced edge recombination crossover (EERX); order-based crossover (OBX), and position-based crossover (PBX). They applied these six methods to a scheduling problem and a traveling salesman problem. Then they concluded that the PMX was the worst for scheduling problems, while it was the second best for the traveling salesman problem. Also, they concluded that for the scheduling problem the following crossover methods performed almost the same: EERX; OX; OBX, and PBX. Another crossover method that has been used in the scheduling literature is the linear order crossover (LOX). Therefore, in this study, three of these crossover methods were selected and used: the LOX; the OBX, and the PBX. These three operators will be explained in the following three sections.

The crossover rate was determined as a function of the population size (Pop_size) as follows: (Pop_size-1)/Pop_size. This implies that the number of chromosomes that were generated according to the crossover operator procedures was Pop_size-1. However, when the tournament selection method was applied, the number of chromosomes that participated in the crossover process was 2(Pop_size-1).

According to Davis (1991), Syswerda (1991), and Michalewicz (1994), three mutation methods are known to perform well in the order-based representation. The first is the order-based mutation (OBM) and the second is the position-based mutation (PBM), a version of the OBM. The scramble sub-sequence mutation (SSM) is the third well-known mutation operator. Therefore, these three mutation operators were selected and used in this study, which will be explained in the following section.

The mutation rate that was used in this study was 0.4. This means that the number of chromosomes produced using the mutation operator was 40% of the Pop_size; also, the number of chromosomes that participated in the mutation process was 2(40%Pop_size).

Linear Order Crossover Operator

The linear order crossover (LOX) operator developed by Falkenauer and Bouffouix (1991) is a version of the order crossover. The LOX is performed by first selecting two sub-chromosomes on two parents, as shown in Figure 6 where sub-chromosome 2 was selected:

	Sub-chromosome 1									Sub-chromosome 2								
Parent 1	1	2	3	4	5	6	7	8	9	4	6	9	7	5	3	1	2	8
Parent 2	9	8	4	5	3	2	1	7	6	8	2	4	6	9	1	3	5	7

Figure 6. Two parents selected to mate.

Then, two cutting positions are selected on the two sub-chromosomes (sub-parents) as shown below (the two cutting positions are denoted by '|'):

Sub-parent 1: 4-6-|9-7-5-3|-1-2-8 and Sub-parent 2: 8-2-|4-6-9-1|-3-5-7

Second, the first sub-child is formed by removing the genes from sub-parent 2 that are located in the segment between the cutting positions in sub-parent 1. This step will form a partial sub-child as follows:

Sub-child 1: 8-2-|4-6-x-1|-x-x-x

Next, slide the empty 'x' positions toward the center of the sub-child so that the crossover segment is filled with empty 'x' positions. This step will result in the following partially formed sub-child:

Sub-child 1: 8-2-|x-x-x-x|-4-6-1

Finally, place the genes in the segment between the two cutting positions in sub-parent 1 in the empty 'x' positions. This step will result in the following sub-child:

Sub-child 1: 8-2-|9-7-5-3|-4-6-1

The same steps can be performed to generate sub-child 2 as follows:

Step 1= sub-child 2: x-x-|x-7-5-3|-x-2-8

Step 2= sub-child 2: 7-5-|x-x-x-x|-3-2-8

Step 3= sub-child 2: 7-5-|4-6-9-1|-3-2-8

The result of the LOX is as follows:

Sub-child 1: 8-2-9-7-5-3-4-6-1 and Sub-child 2: 7-5-4-6-9-1-3-2-8

Order-Based Crossover Operator

The order-based crossover (OBX) operator is also a version of the order crossover. It was developed by Syswerda (1991) to handle the infeasibility problem in a GA approach that was applied to a scheduling problem. Using the same two sub-chromosomes that were used in the previous section, the OBX is performed by first selecting several random positions on two sub-parents as shown below (the positions selected are underlined).

Sub parent 1: 4-$\underline{6}$-9-7-$\underline{5}$-3-$\underline{1}$-$\underline{2}$-8 and Sub-parent 2: 8-$\underline{2}$-4-6-$\underline{9}$-1-$\underline{3}$-$\underline{5}$-7

Next, the first sub-child is formed by removing the genes from sub-parent 1 that are located in the positions selected in sub-parent 2. This step will form a partial sub-child as follows:

Sub-child 1: 4-6-x-7-x-x-1-x-8

Then, the empty 'x' positions are filled with genes selected in sub-parent 2 using the order these genes appear in sub-parent 2 (i.e., 2-9-3-5). This step will result in the following sub-child:

Sub-child 1: 4-6-2-7-9-3-1-5-8

The same steps can be performed to generate sub-child 2 as follows:

Step 1= Sub-child 2: 8-x-4-x-9-x-3-x-7

Step 2= Sub-child 2: 8-6-4-5-9-1-3-2-7

The result of the OBX is as follows:

Sub-child 1: 4-6-2-7-9-3-1-5-8 and Sub-child 2: 8-6-4-5-9-1-3-2-7

Position-Based Crossover Operator

The position-based crossover (PBX) operator is similar to the OBX and was developed by Syswerda (1991). A similar procedures can be applied to perform the PBX; however, the position of genes selected in sub-parent 2 are imposed in sub-child 1 whereas the OBX imposes the order of genes. Using the same two sub-chromosomes that were selected before, the PBX is performed by first selecting several random positions on two sub-parents as shown below (the positions selected are underlined).

Sub-parent 1: 4-6-9-7-5-3-1-2-8 and Sub-parent 2: 8-2-4-6-9-1-3-5-7

Next, Sub-child 1 is formed by copying the genes selected in sub-parent 2. This step will form a partial sub-child as follows:

Sub-child 1: x-2-x-x-9-x-3-5-x

Then, the empty 'x' positions are filled with genes from sub-parent 1, where these genes to be copied from sub-parent 1 are not already in sub-child 1 and they are copied using the order given in sub-parent 1 (i.e., 4-6-7-1-8). This step will result in the following sub-child:

Sub-child 1: 4-2-6-7-9-1-3-5-8

The same steps can be performed to generate sub-child 2 as follows:

Step 1 = Sub-child 2: x-6-x-x-5-x-1-2-x

Step 2 = Sub-child 2: 8-6-4-9-5-3-1-2-7

The result of the PBX is as follows:

Sub-child 1: 4-2-6-7-9-1-3-5-8 and Sub-child 2: 8-6-4-9-5-3-1-2-7

Order-Based Mutation Operator

As mentioned in Chapter II, the order-based mutation (OBM) operator is implemented by selecting two genes randomly and swapping them. In this study, the OBM was performed by first selecting a sub-chromosome on a parent which is given in Figure 7 where sub-chromosome 2 was selected:

	Sub-chromosome 1									Sub-chromosome 2								
Parent	1	2	3	4	5	6	7	8	9	4	6	9	7	5	3	1	2	8

Figure 7. A parent selected to mutate.

Then, two positions were selected on sub-parent 2 as shown below (the positions selected are underlined).

Sub parent: 4-6-9-7-5-3-1-2-8

Next, the sub-child is formed by swapping the genes selected as follows:

Sub-child: 4-5-9-7-6-3-1-2-8

Position-Based Mutation Operator

The position-based mutation (PBM) operator is a version of the OBM operator and it is performed by selecting two genes randomly and then inserting the second gene before the first. Using the same sub-chromosome that was used in the previous section, the PBM is performed by first selecting two positions on a sub-parent as shown below (the positions are underlined):

Sub parent: 4-6-9-7-5-3-1-2-8

In the above selection, gene 6 was selected before gene 5. Next, the sub-child is formed by inserting gene 5 before gene 6 as follows:

Sub-child: 4-5-6-9-7-3-1-2-8

Scramble Sub-Sequence Mutation Operator

The scramble sub-sequence mutation (SSM) operator was developed by Davis (1985). The SSM selects a sub-sequence in a sub-chromosome, and scrambles the genes in the sub-sequence selected. Using the same sub-chromosome that was used in the previous section, the SSM is performed by first selecting a block of genes on a sub-parent as shown below (the two cutting positions for the block selected are denoted by '|'):

Sub-parent: 4-6-|9-7-5-3|-1-2-8

Next, the sub-child is formed by scrambling genes within block as follows:

Sub-child: 4-6-3-5-7-9-1-2-8

Genetic Operators Implementations

As mentioned earlier, the population in this study was represented by chromosomes that were formed by several sub-chromosomes. Hence, to implement crossover and mutation operators to this type of representation, several questions needed to be answered. These questions were:

1. Should these genetic operators be applied to all or some of the sub-chromosomes?
2. How many sub-chromosomes should be selected if it were decided to apply the genetic operators to selected sub-chromosomes?
3. Which sub-chromosomes should be selected if it were decided to apply the genetic operators to selected sub-chromosomes?
4. How many genes should be selected in each sub-chromosome?
5. Which genes should be selected in each sub-chromosome?

In the following paragraphs the answers to the above five questions will be discussed. Some of the above questions were answered by performing several pilot studies.

According to the conclusion of a pilot study that was performed, the problem with applying the genetic operators to all sub-chromosomes is that it was extremely time consuming. Also, it was extremely disturbing for the GA evolution process. This conclusion led to investigating the answer to the second question.

The number of sub-chromosomes that can be selected was randomly determined as a function of the number of machines according to a discrete uniform distribution. For the operators LOX, OBX, PBX, and SSM, the number of sub-chromosomes was uniformly distributed between 1 and 0.5m, where m is the number of machines. Also, the number of sub-chromosomes was uniformly distributed between 1 and 0.69m for OBM and PBM operators.

As mentioned in Chapter II, Sawaqed (1987, x) concluded that "the most crucial element in managing a job shop is the management of its bottleneck machines." This conclusion helped to answer question number three on which sub-chromosomes should be selected (i.e., which machines should be selected). To answer question number three using Sawaqed's conclusion, machines were first classified as bottlenecks and non-bottlenecks according to the total work content of each machine. Then, according to the number of sub-chromosomes determined, sub-chromosomes were selected as follows:

1. If the number of sub-chromosomes = 1, then the top bottleneck was selected.
2. If the number of sub-chromosomes = 2, then the top two bottlenecks were selected.
3. If the number of sub-chromosomes > 2, then the top two bottlenecks were selected and the other m-2 machines were equally randomly selected.

Recall from the pervious sections that the OBM operator is implemented by selecting two genes randomly and swapping them. Also, the PBM operator is implemented by selecting two genes randomly and then inserting the second gene before the first. Therefore, the answer to question number four for these two operators was two genes.

However, the answer to question number four was different for the operators LOX, OBX, PBX, and SSM. This question was answered randomly as a function of the machine load (i.e., the sub-chromosome size) according to a discrete uniform distribution using the following procedures:

1. The maximum number of genes (MG) was determined as follows: MG = 0.7ML where ML is the machine load (i.e., the sub-chromosome size).
2. Then the number of genes (G) was randomly determined according to a discrete uniform distribution with a = 2 and b = MG.

When the number of genes (G) was determined, the genes were randomly defined as a function of the machine load according to a discrete uniform distribution with a=1 and b = ML. For the OBM and the PBM operators, two genes were randomly defined and for the OBX and the PBX operators, a G different genes were randomly defined. The genes in the LOX and SSM operators were defined as follows:

1. The position of the starting gene (PSG) was generated according to a uniform distribution between 1 and ML-G.
2. Then the position of the ending gene was determined as follows: PSG + G-1.

3. Then, when the starting and the ending positions were defined, the genes in this block were selected.

Termination Criteria

Several criteria have been suggested for the GA convergence. Goldberg (1989) and Davis (1991) suggested that the GA converges if all chromosomes have attained a certain degree of homogeneity (that is, all of them have almost the same fitness value). A version of this criterion is that the GA will converge if the percentage of the best solution in the population is greater than or equal to fifty percent. Another convergence criterion is that the GA will converge if the maximum number of generations has been reached or a certain time limit has been reached. Also, the GA will converge after a chromosome with a certain high fitness value is located. Furthermore, the GA will terminate if the best solution has not been changed for a number of generations.

To take advantage of several termination criteria, the termination criterion in this study was a combined criterion. The GA approach was terminated if one of the following conditions was satisfied:

1) The maximum number of generations has been reached.
2) The best solution has not been changed for a number of generations.
3) A certain time limit has been reached.

The first termination condition was used in the GA discussed in the "Constrained Genetic Algorithm Study" section. The maximum number of generations was determined as a function of the problem size, which was n^3, where n is the number of jobs. The same concept was used to determine the minimum number of generations, which was 4nm, but the maximum number of generations was fixed, which was 200 generations. The number of generations required by condition number two was determined as 10% of the number of generations. Ten minutes was the time limit required by condition number three. The GA approach investigated the termination conditions after a threshold of generations, which was determined as 10% of the number of generations.

Genetic Algorithm Computer Program Logic and Organization

In this section the necessary elements for the GA computer program logic and organization are discussed. The general logic of the GA approach follows the general steps for the GA approach. The organization of the GA computer program is illustrated in Figure 8. From Figure 8, it can be seen that besides the MAIN program there are nineteen subroutines, three functions, and the IMSL™ Mathematical and Statistical libraries[3].

The MAIN program begins by calling the CGA_VAR subroutine to define and to initialize all global variables. Then, it calls DATA_FILE subroutine to initialize the data file name. Next, a call is made to JOBS_DATA subroutine to read job data and to do all necessary initialization and computations (e.g. the maximum number of operations, classification of machines, ...etc.). Also, the JOBS_DATA subroutine calls the EXPECTED_WAITING_TIME subroutine to compute the expected waiting time of each operation for each job on each machine. The GA_PARAMETERS subroutine is called next by the MAIN program to initialize and to define the GA parameters. Next, the MAIN program calls the INITIALIZE_POPULATION subroutine to generate the starting initial population. When the initial population is generated, the MAIN program calls the selected crossover subroutine. This means one of the following subroutines is selected: LOX; OBX, or PBX. When the chosen crossover subroutine is performed, the MAIN program makes a call to one of the following mutation subroutines: SSM; OBM, or PBM. Then, if the termination condition is satisfied, the MAIN program makes a call to two subroutines: OUTPUT and DEALLOCATE_ARRAYS. The former subroutine will print the results and the latter subroutine

---

[3] IMSL Mathematical and Statistical libraries is a trade mark of Visual Numerics, Inc.

will re-initialize all variables. If the termination condition is not satisfied, then the procedure is repeated.

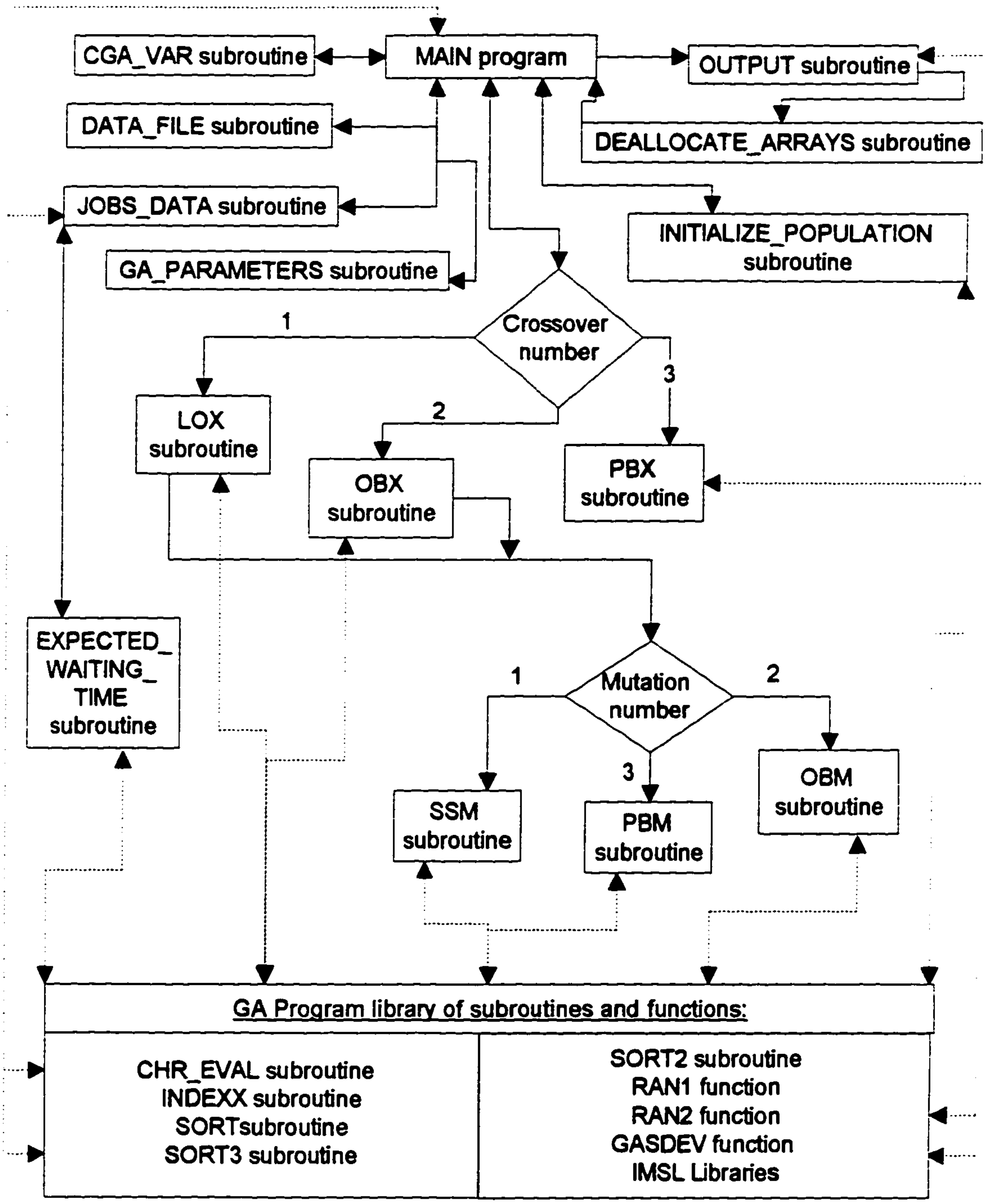


Figure 8. Genetic algorithm computer program organization.

The MAIN program and the following subroutines used the GA program library of subroutines and functions: JOBS_DATA; INITIALIZE_POPULATION; LOX; OBX; PBX;

SSM; OBM; PBM; EXPECTED_WAITING_TIME, and OUTPUT. This library contains the following subroutines and functions: CHR_EVAL; SORT; SORT2; SORT3; INDEXX; RAN1; RAN2; GASDEV, and the IMSL libraries. The CHR_EVAL subroutine is used to evaluate each chromosome. The SORT, the SORT2, the SORT3, and the INDEXX are sorting subroutines. All of these subroutines sort only one array. However, they differ from each other with respect to how many arrays need to be rearranged while sorting the array sought. The RAN1 and RAN2 functions are used to generated uniform distribution values between 0 and 1. The GASDEV function generates random numbers according to standard normal distribution.

Several versions of the GA approach were developed and will be discussed in the following sections. These versions of the GA approach were coded in FORTRAN 90 for a Gateway 2000 computer using the Microsoft FORTRAN PowerStation™, professional edition, version 4.0[4]. The Gateway 2000 computer has a 90MHZ Pentium CPU, 40MB of RAM, and 1 GB IDE hard drive running Microsoft Windows™ 95[5]. The Microsoft FORTRAN PowerStation package was used because it is the only FORTRAN 90 development system for Microsoft Windows 95. Also, the Microsoft FORTRAN PowerStation was chosen because it allows a complete interface to the IMSL Mathematical and Statistical libraries, which contains 1000 classic mathematical and statistical functions. The Microsoft Windows 95 was chosen to be the operating system because it provides a 32-bit operating system with flat memory model, which made it easy to program and faster to execute.

The computer codes for the GA versions will not be included in this dissertation. The reason for not including the computer codes is space limitation. For a full listing of the computer codes, the reader can refer to Al-Harkan and Foote (1997).

---

[4] Microsoft FORTRAN PowerStation is a trademark of Microsoft Corporation.

[5] Microsoft Windows 95 is a trade mark of Microsoft Corporation.

Deterministic Genetic Algorithm to Minimize Makespan: CGA_Cmax and UGA_Cmax

This section will discuss the deterministic constrained genetic algorithm to minimize makespan (CGA_Cmax) and the deterministic unconstrained genetic algorithm to minimize makespan (UGA_Cmax). The CGA_Cmax and the UGA_Cmax used the same elements and parameters discussed in the "Genetic Algorithm Structure" section. The only difference between the CGA_Cmax and the UGA_Cmax is that the CGA_Cmax used dominance rules when performing the crossover and the mutation operators. The following figure demonstrates the structure of both the CGA_Cmax and UGA_Cmax. In Figure 9, the structure of the CGA_Cmax and UGA_Cmax is given.

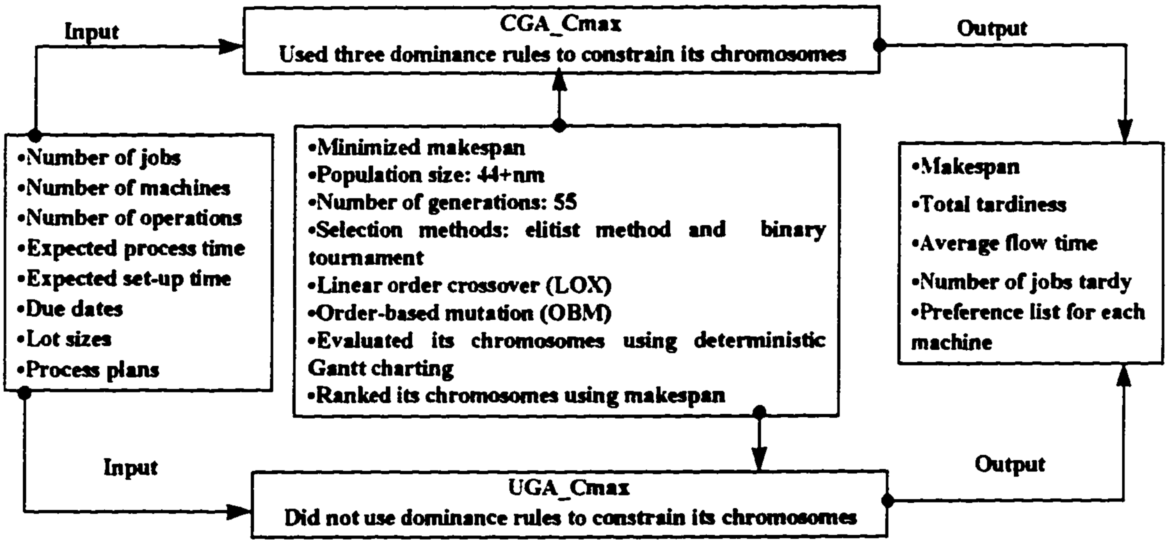


Figure 9. CGA_Cmax and UGA_Cmax structure.

The dominance rules used in the genetic operators should be selected to minimize the objective function sought in the CGA model, which is the makespan in the CGA_Cmax. This implies that the dominance rules are objective function dependent. Before listing the dominance rules used in the CGA_Cmax model, a brief description of how these rules were selected will be given in the following paragraph.

Chang, Sueyoshi, and Sullivan (1996) ranked forty-two dispatching rules by using data envelopment analysis in a job shop environment. The ranking for the dispatching rules was first accomplished by associating the dispatching rules with several performance measures. Then the dispatching rules were ranked according to each performance measure. Thus, in this study, six dispatching rules were selected to be used as the dominance rules. These six rules were ranked by Chang, Sueyoshi, and Sullivan among the first fifteen dispatching rules to minimize the makespan. A pilot study was performed to compare the performance of these six dispatching rules.

According to the conclusions of the pilot study, the following theorem and two dispatching rules were selected to be used in the CGA_Cmax model as the dominance rules:

Dominance rule 1 (theorem 1): For two jobs i and k, if $p_{ij} \le p_{kj}$, and $d_{ij} \le d_{kj}$ then there exists an optimal sequence in which job i appears before job k, where p_{ij} and p_{kj} are the expected processing times for jobs i and k on machine j. Also, d_{ij} and d_{kj} are the expected due dates of jobs i and k at machine j. In this study, d_{ij} was computed as follows: $d_{ij}= d_i - a_i$, where d_i is the original due date and a_i is the total remaining work for job i. Also, a_i was computed as follows: $a_i = w_i + p_i$, where p_i is the expected remaining processing time for job i, and w_i is the expected remaining waiting time for job i. The w_i was computed as follows: $w_i =\Sigma\ w_{ij}$, where w_{ij} is the waiting of job i at machine j. The w_{ij} was computed using two methods. The first, multiple of processing time, was computed as follows: $w_{ij} = bp_{ij}$, where b was assigned a random value that was uniformly distributed between 1 and 2. The second is an iterative method, which consisted of 5 simulation runs. In every simulation run, the waiting time of job i at machine j was computed as follows: $w_{ij}^{k} = (1-a)w_{ij}^{k-1} + aq_{ij}^{k}$, where k is the simulation run number (k=1,…,5), a is a smoothing parameter which was 0.95, $w_{ij}^{0}=$ bp_{ij}, and q_{ij}^{k} was the actual waiting time during the k^{th} simulation. The first method was used in the deterministic genetic algorithms, while the second method was used in the stochastic genetic algorithms.

Dominance rule 2 (dispatching rule 1): Select a job with the smallest ratio of the processing time to the total work remaining: $P_{ij}/(w_i + p_i)$, where p_{ij} is the expected processing time of job i on machine j, p_i is the expected remaining processing time for job i, and w_i is the expected remaining waiting time for job i, which was computed as explained above.

Dominance rule 3: (dispatching rule 2): Select a job with the largest total remaining processing time: $(w_i + p_i)$, where and p_i is the expected remaining processing time for job i, and w_i is the expected remaining waiting time for job i, which was computed as explained above.

The CGA_Cmax utilized these dominance rules exclusively in the way they are ordered above. This implies that when the CGA_Cmax was implemented the dominance rules were applied by first investigating the satisfaction of dominance rule one. If rule one was satisfied, then none of the other rules would be investigated. Otherwise, dominance rule two would be investigated. Finally, dominance rule three would be investigated if dominance rule two was not satisfied.

Recall from the previous sections that the OBM operator is implemented by selecting two jobs randomly and swapping them. Hence, for the OBM operator the dominance rules were applied to avoid swapping of jobs that satisfy the dominance rules. Also, the PBM operator is implemented by selecting two jobs randomly and then inserting the second job before the first. Thus, if inserting the second job before the first violates one of the dominance rules, then the second job will not be inserted before the first job.

During the implementation of the dominance rules to the OBM and the PBM operators, a cycling problem can exist. This cycling problem occurs when several attempts have been made to perform mutation without finding any pair of jobs that do not satisfy the dominance rules. This situation happened at the later stages of the CGA evolution process. This cycling problem was handling by restricting the number of cycles to a fixed number of cycles as follows. A sub-chromosome on any chromosome would be attempted for mutation the number of the machine load times. This means that if there are four machines and each with a load of 4, then the maximum number of mutation attempts would be equal to 16. Also, a selected chromosome would be attempted for mutation 114 times.

Recall that the LOX, the OBX, the PBX, and the SSM operators are implemented by first selecting several random jobs on either one or two sub-parents. Then one or two children are produced. The dominance rules were implemented on the children produced by ordering the set of jobs selected according to a precedence constraint based on the dominance rules.

Deterministic Genetic Algorithm to Minimize Total Tardiness: CGA_TT and UGA_TT

The deterministic constrained genetic algorithm to minimize total tardiness (CGA_TT) and the deterministic unconstrained genetic algorithm to minimize total tardiness (UGA_TT) are discussed in this section. The CGA_TT and the UGA_TT used the same elements and parameters discussed in the "Genetic Algorithm Structure" section. Again, the difference between the CGA_TT and the UGA_TT is that the genetic operators in the CGA_TT produced children that were altered not only by the operator's procedures but also by the dominance rules, while no alteration was performed in the UGA_TT. The CGA_TT used only one dominance rule when performing the crossover and the mutation operators. The dominance rule used by the CGA_TT is dominance rule number one mentioned in the previous section. In Figure 10, the structure of both the CGA_TT and UGA_TT is given.

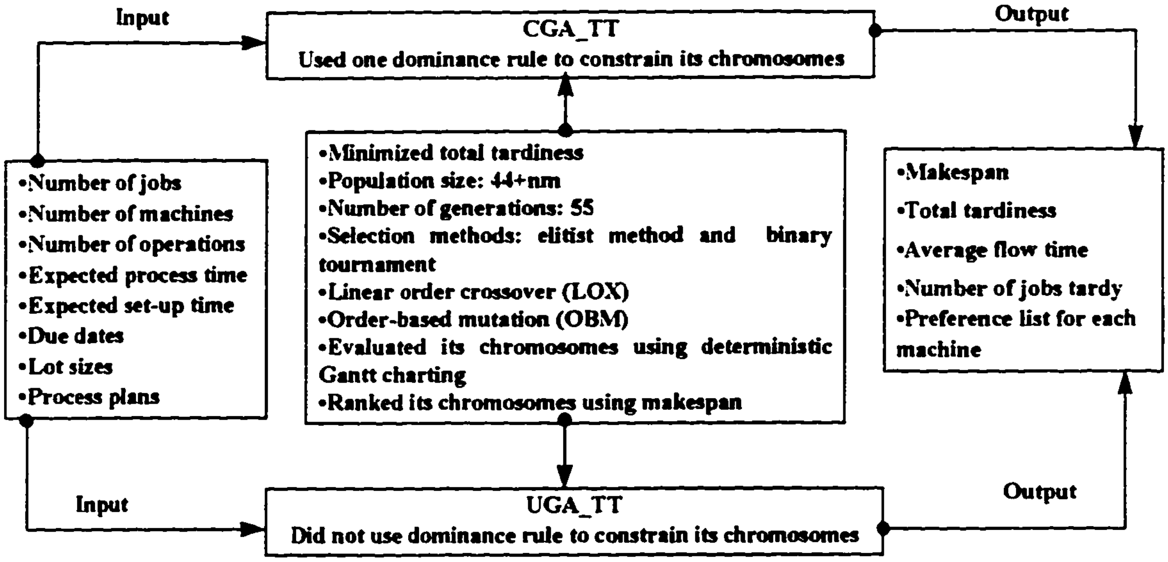


Figure 10. CGA_TT and UGA_TT structure.

Stochastic Genetic Algorithm

It was mentioned in Chapter II that the methods used to evaluate chromosomes are either simulation or deterministic Gantt charting. Both methods are extremes, in the sense that deterministic charting does not account for uncertainty and simulation, while accounting for uncertainty, is expensive in terms of time to get accurate estimates. Therefore, an evaluation method between the two extremes was proposed in this study and will be described in this section. This method is called probability Gantt charting. Also, in this section four models will be described: stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using probability Gantt charting (CGA_WSPT); stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using simulation (CGA_SIM); stochastic unconstrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using probability Gantt charting (UGA_WSPT), and stochastic unconstrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using simulation (UGA_SIM). The CGA_WSPT and the UGA_WSPT models used probability Gantt charting to evaluate their chromosomes and the CGA_SIM and the UGA_SIM models evaluated their chromosomes using simulation.

The organization of this section is as follows. A description of the probability Gantt charting is given first, followed by a description of the CGA_WSPT and the UGA_WSPT. Then, a utility function approach used to rank chromosomes will be presented. Finally, the CGA_SIM and the UGA_SIM models will be discussed.

Chromosome Evaluation Method: Probability Gantt Charting

In the proposal of this dissertation, the probability Gantt charting was included. However, Liang (1996) developed the methodology and the computer code required to implement the probability Gantt charting and also performed several experiments that are beneficial to this research. Since some of the conclusions of Liang's research will be given later in this section, only a brief introduction to this evaluation method will be given. For detailed explanations of the evaluation method and experiments performed, the reader can refer to Liang (1996).

To evaluate chromosomes, all the required performance measures are computed using job completion times. The completion times are obtained using one of three evaluation methods mentioned in the previous section. When deterministic Gantt charting is used, the completion times of jobs are computed according to the expected value of both the processing times and set-up times, which are deterministic values. On the other hand, when simulation evaluation is used, the completion times of jobs are computed according to stochastic processing times and stochastic set-up times. When probability Gantt charting is used, the completion times of jobs are computed using deterministic Gantt charting with uncertainty in the process time, which makes it stochastic evaluation method.

The basic concept of the probability Gantt charting is that it estimates the completion time of a job based on the probability that a job will be out of the machine. In other words, the probability of the job being out will be determined first, and from that the completion time of a job will be determined. To demonstrate the probability Gantt charting computations, assume that there is a job that has a processing time which is normally distributed with the following parameters: μ = 10 and σ = 2. Using these parameters, the normal distribution and the cumulative distribution functions can be constructed. From the cumulative distribution function, the completion time of the job can be computed using a 90% probability that the job will be completed (i.e., Pr = 0.9, where Pr is the probability selected). The density function and the cumulative distribution function of the process time are given below.

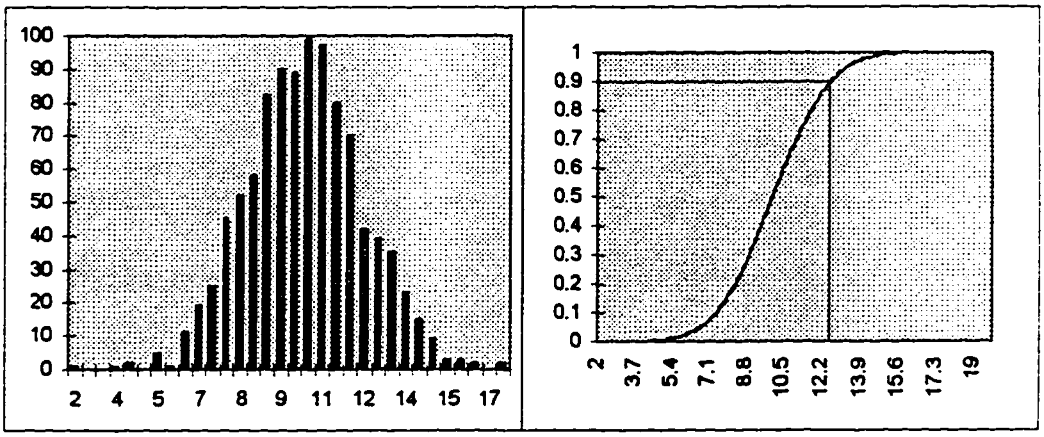


Figure 11. The density function and the cumulative distribution function of the process time.

From Figure 11, it can be seen that when the probability of 90% was specified, the time to complete a job can be found at 12.564. Mathematically, this completion time can be computed by using the following formula: time out (TO) = $\mu + Z\sigma$, where Z is determined according to the probability value from the standard normal table (Z=1.282 for 0.9). Thus, TO=10+1.282x2 = 12.564.

The following example will explain the concept of the probability Gantt charting, and demonstrates the difference between standard deterministic and probability Gantt charting. Assume that in a job shop there are four machines which process three products. The process plan and the lot size for each product are give in Table 2. Also, the process times and set-up times for each machine are normally distributed with the parameters given in Table 3:

Table 2. Three products' data.

Product number	Order Size	Routing (machine number)
1	15	1-4
2	10	2-3-4
3	20	1-2-3-4

Table 3. Four machines' data.

Machine number	Mean of the processing time (hours)	Variance of the processing time	Mean of the set-up time (hours)	Variance of the Set-up time
1	0.1	0.00083	0.75	0.021
2	0.2	0.00083	1.25	0.021
3	0.15	0.00083	1.5	0.083
4	0.075	0.0021	0.875	0.01

Assume that the priority list for the three machines is the same and is as follows: product 3 has the first priority; product 2 has the second priority, then product 1 has lowest priority. Assume that the three products are available at time zero and the dynamic priority is applied. To construct the standard deterministic Gantt chart for this example, the completion time (C_{ij}) for product i on machine j has to be computed. The general formula to compute the processing times is as follows:

$$C_{ij} = \max(r_i, C_{k,j}) + P_{ij}\, Q_i + S_{ij}$$

Where:

r_i: Ready time of product i.

S_{ij}: Set-up time of product i on machine j.

P_{ij}: Process time of product i on machine j.

Q_i: The lot size of product i.

$C_{k,j}$: Completion time of the product k on machine j, where job k proceeded job i.

In the standard deterministic Gantt charting, the expected mean values are used. For normal distribution, this means that the standard deterministic Gantt charting is implemented with a probability of 50% that the product will be completed at each machine. This is similar to saying that the probability Gantt charting is implemented with a Pr value equal to 0.5. Also, for exponential distribution, this means that the probability Gantt charting is implemented with a Pr value equal to 0.632.

The general formula to compute the completion times using the probability Gantt charting is as follows:

$$C_{ij} = \max(r_i, C_{i-1}) + \{P_{ij} Q_i + S_{ij}\} + Z\sqrt{Q_i * V[P_{ij}] + V[S_{ij}]}$$

Where:

$V[P_{ij}]$: The variance of the processing time of product i on machine j.

$V[S_{ij}]$: The variance of the set-up time of product i on machine j.

The Pr value for the probability Gantt charting was selected to be 0.9. Using the above two formulas, the completion times for the three products can be computed and are given in the following table:

Table 4. Products' completion times.

Product number	Standard Gantt chart	Probability Gantt chart
1	7.00	7.55
2	8.63	9.4
3	15.38	15.43

From Table 4, the effect of the variability caused by the variance on the completion times can be seen, which we hoped to find more accurate than the deterministic computations.

Stochastic Genetic Algorithm to Minimize Total Tardiness and to Evaluate Chromosomes using Probability Gantt Charting: CGA_WSPT and UGA_WSPT

This section will discuss the stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using probability Gantt charting (CGA_WSPT) and the stochastic unconstrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using probability Gantt charting (UGA_WSPT).

To implement the proposed evaluation method, the CGA_TT model was extended to evaluate its chromosomes according to the described evaluation method--the probability Gantt charting. The result of this extension was a CGA with stochastic process time (CGA_WSPT) model. The CGA_WSPT model was an extension of the CGA_TT, which implies that it attempted to minimize the total tardiness. Also, it used the same elements and parameters used by the CGA_TT. The dominance rule used by the CGA_TT was also used by the CGA_WSPT.

When the dominance rule was taken out of the CGA_WSPT model, the result was a new model which was the UGA_WSPT. This means that the UGA_WSPT is identical to the CGA_WSPT, except the UGA_WSPT does not incorporate the dominance rule when it applied the genetic operators. Figure 12 illustrates the structure of the CGA_WSPT and the UGA_WSPT.

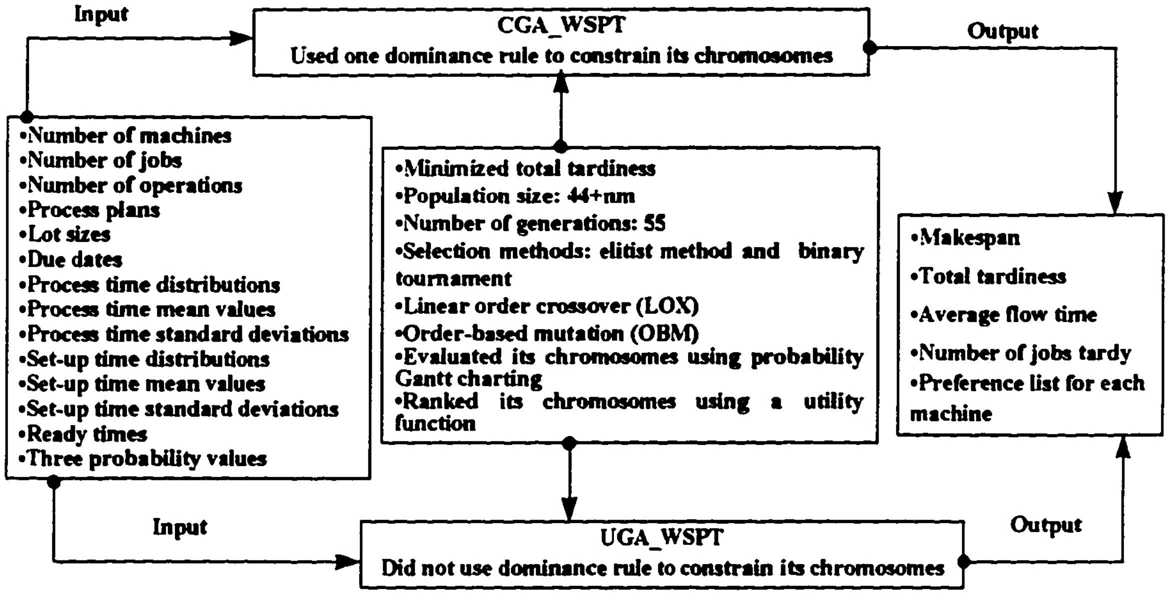


Figure 12. CGA_WSPT and UGA_WSPT structure.

Chromosome Ranking Method: Utility Function Approach

It should be clear that when the probability Gantt charting is implemented with different levels of probability, each probability level would have a different result. It should be obvious that the values assigned to each level are probability distribution dependent. Also, it should be clear that there is an infinite number of probability levels. Thus, there is a need to narrow the range of the probability levels. This goal was attempted by the research that was done by Liang (1996). Liang narrowed the number of probability levels to only three for specific probability distributions.

Therefore, when the CGA_WSPT and UGA_WSPT were implemented, three probability levels were used to evaluate each chromosome. This implies that each chromosome would have three total tardiness values computed from three sets of completion times. Given that each chromosome has three total tardiness values, it is difficult to rank chromosomes. Hence, to rank chromosomes according to a single fitness function value, a decision should be made which of the three total tardiness values should be associated with each chromosome. To solve this difficulty, a utility function constructed using the three total tardiness values is associated with each chromosome. For each chromosome, the utility function is assumed to be the cumulative distribution function of the normal distribution. To compute the utility function for each chromosome, the average and the standard deviation of the three total tardiness values are computed first. Then a target value for the total tardiness is determined and used when computing the utility function value. The utility function for each chromosome is compute as follows:

$$U = \text{Probability}\{\text{Total tardiness of the chromosome} \leq \text{Target total tardiness}\} = \{TT - \overline{TT} / \sigma_{TT}\}$$

Where:

U: The chromosome utility function value.

TT: A target value for the total tardiness.

$\overline{TT}$: The average of the three total tardiness values.

σ_{TT}: the standard deviation of the three total tardiness values .

Using the utility function computed for each chromosome, chromosomes can be ranked according to the utility function in descending order. This implies that the chromosome with the largest utility function value is preferred.

When the CGA_WSPT and UGA_WSPT were implemented, the target value was determined as the minimum average total tardiness obtained among the chromosomes generated in the initial population. Mathematically, this target value was computed as follows: $TT = \min\{\overline{TT}_{Ci} \mid i=1,\ldots,Pop\_size\}$, where: $\overline{TT}_{Ci}$: the average of the three total tardiness values for chromosome i.

The following example illustrates how beneficial the utility function is. Assume that there are two chromosomes with the following averages and standard deviations for the three total tardiness values associated with each chromosome: $\overline{TT}_{C1} = 137; \sigma_{TT_{C1}} = 40$; $\overline{TT}_{C2} = 130$, and $\sigma_{TT_{C2}} = 60$. Assume that target value is 150; then the utility function values for chromosomes one and two are 0.626 and 0.629 respectively. Clearly, chromosome two has a higher utility value and hence it is selected even though it has higher variability.

Stochastic Genetic Algorithm to Minimize Total Tardiness and to Evaluate Chromosomes using Simulation: CGA_SIM and UGA_SIM

The stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using simulation (CGA_SIM) and the stochastic unconstrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using simulation (UGA_SIM) are discussed in this section.

As mentioned earlier in the "Stochastic Genetic Algorithm" section, the CGA_SIM model evaluated its chromosomes using simulation. The CGA_SIM model was extended from the CGA_TT model, which means that the CGA_SIM is identical to the CGA_WSPT except they are different in the chromosome evaluation method used. The CGA_SIM used the utility function approach mentioned earlier to rank its chromosomes.

In the CGA_SIM each chromosome should be evaluated several times to reach a certain confidence level for the results obtained. By doing so, the results obtained by the simulation would not be far away from the true mean. Hence, to determine the number of evaluations for each chromosome (i.e., the number of replications for the simulation), the following sequential procedure was used, which was proposed by Law and Kelton (1991):

1. Make n_0 replication of the simulation and set $n=n_0$, where $n_0 \geq 2$.

2. Compute $\overline{TT}$ and $\delta(n,\alpha)$ from $TT_1, TT_2, \ldots, TT_n$.

3. If $\delta(n,\alpha)/|\overline{TT}| > \gamma'$, then replace n by b+1, make an additional replication of the simulation, and go to step 1. Otherwise, use $\overline{TT}$ as the point estimate for the true mean of the total tardiness (μ) and stop.

Where:

n: The number of replications that has been performed.

$\delta(n,\alpha)$: Confidence-interval half length, where $\delta(n,\alpha) = t_{n-1,\ 1-\alpha/2}\sqrt{S^2 / n}$, where S^2 is the variance, and $t_{n-1,\ 1-\alpha/2}$ is the upper critical point for the t distribution with n-1 degree of freedom.

γ': The adjusted relative error, where $\gamma'=\gamma/(1+\gamma)$, where γ is the relative error and $\gamma= |\overline{TT} - \mu|/|\mu|$.

From the above sequential procedure, the following confidence interval can be obtained:

$$[\overline{TT} - \delta(n,\alpha), \overline{TT} + \delta(n,\alpha)]$$

This confidence interval is an approximate $100(1-\alpha)$ percent confidence interval for μ with the desired relative error.

In this research, the above parameters were assigned the following values: $n_0 = 11$; $\gamma=0.1$; $\gamma'=0.09$, and $\alpha=0.1$. Using the parameter values implied that 90% confidence intervals were constructed in this research study with a relative error of 0.1

When the dominance rule was taken out of the CGA_SIM model, the result was a new model which was the UGA_SIM. This means that the UGA_SIM is identical to the CGA_SIM, except the UGA_SIM does not incorporate the dominance rule when it applied the genetic operators. The structure of the CGA_SIM and UGA_SIM is given in Figure 13.

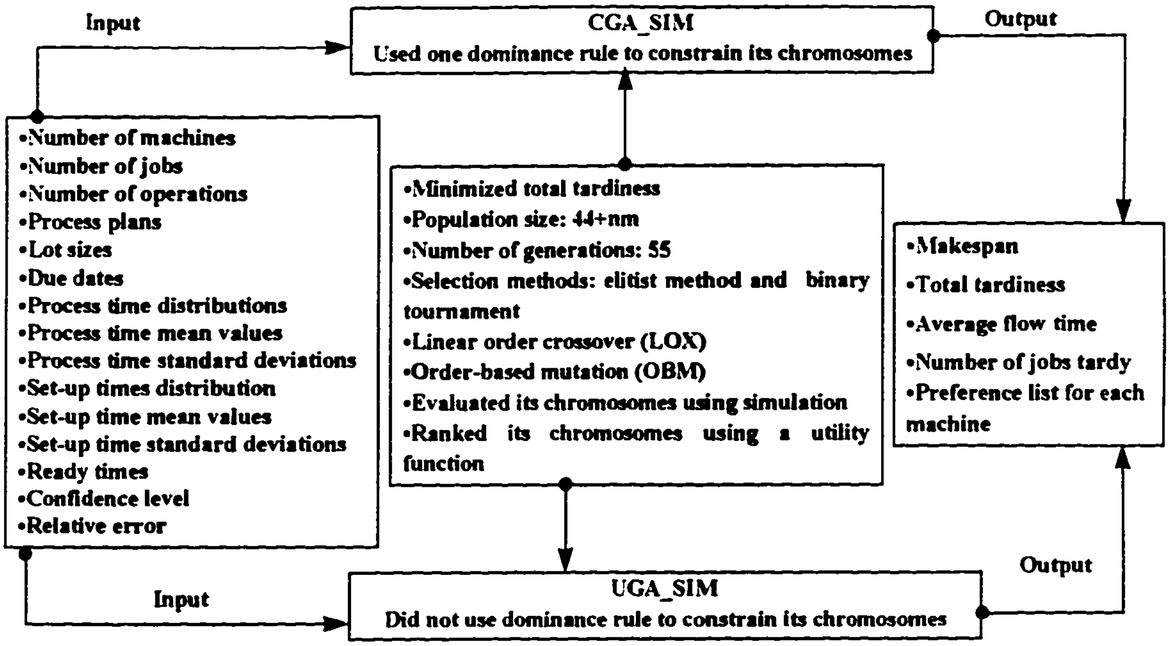


Figure 13. CGA_SIM and UGA_SIM structure.

Dynamic Stochastic Constrained Genetic Algorithm to Minimize Total Tardiness and to Evaluate Chromosomes using Probability Gantt Charting: CGA_APP

This section presents the final constrained genetic algorithm, constrained genetic algorithm with alternative process plan (CGA_APP), which was intended to be the integrated production model that controls a dynamic stochastic job shop environment. This model is an extension of the CGA_WSPT model, which attempted to minimize the total tardiness. Also, it used the same elements and parameters used by the CGA_WSPT. The dominance rule that was used by the CGA_WSPT was also used by the CGA_APP. In addition to attempting to minimize the total tardiness, the CGA_APP attempted to optimize simultaneously the lot sizes and the process plans for the products involved in the production plan. Specifically, the CGA_APP can handle products that each have a set of top alternative process plans and from which the lot size for each product can be optimized.

The CGA_APP attempted to optimize product lot sizes and process plans as follows. For each product there is a set of process plans from which a set of top alternative process plans can be selected. Then, for the products in the production plan, several sets of process plans can be formed randomly from the set of top alternative process plans for each product. A set of products process plans can be formed randomly as follows. For each product, the process plan number can be uniformly selected between 1 and the maximum number of top alternative process plans. Using the order size (O_i) and selected process plan for each product, the lot sizes (Q_i) for all products can be optimized. The optimized lot size for each product could be less than or equal to the order size. When the lot size for a product is less than the order size, then this product should be re-produced to satisfy its order size. This means that the maximum number of re-productions for product i is O_i/Q_i. At this stage of the optimization process, the lot sizes have been optimized with respect to the products' process plans selected.

The above procedures can be repeated for several iterations (say k) in which in every iteration a set of products' process plans is selected and then the products' lot sizes are optimized. At the final iteration, there will be k sets of products' process plans in which each set has its optimized lot sizes. For each set of the k sets of products' process plans and their optimized lot sizes, the total tardiness can be minimized using the same procedures used in the CGA_WSPT. Figure 14 demonstrates the selection of the top alternative process plans for each product and the formation of the k sets of products' process plans and lot sizes.

The lot sizes for a product can all be released at time zero or can be released according to a releasing mechanism. In this study, the releasing mechanism consisted of dispatching rules that attempted to minimize the total tardiness. Hence, these dispatching rules were used to release product lot sizes considering the minimization of the total tardiness. The releasing mechanism is the last component that completed the design of the CGA_APP. Figure 15 shows the components required for the CGA_APP design and development.

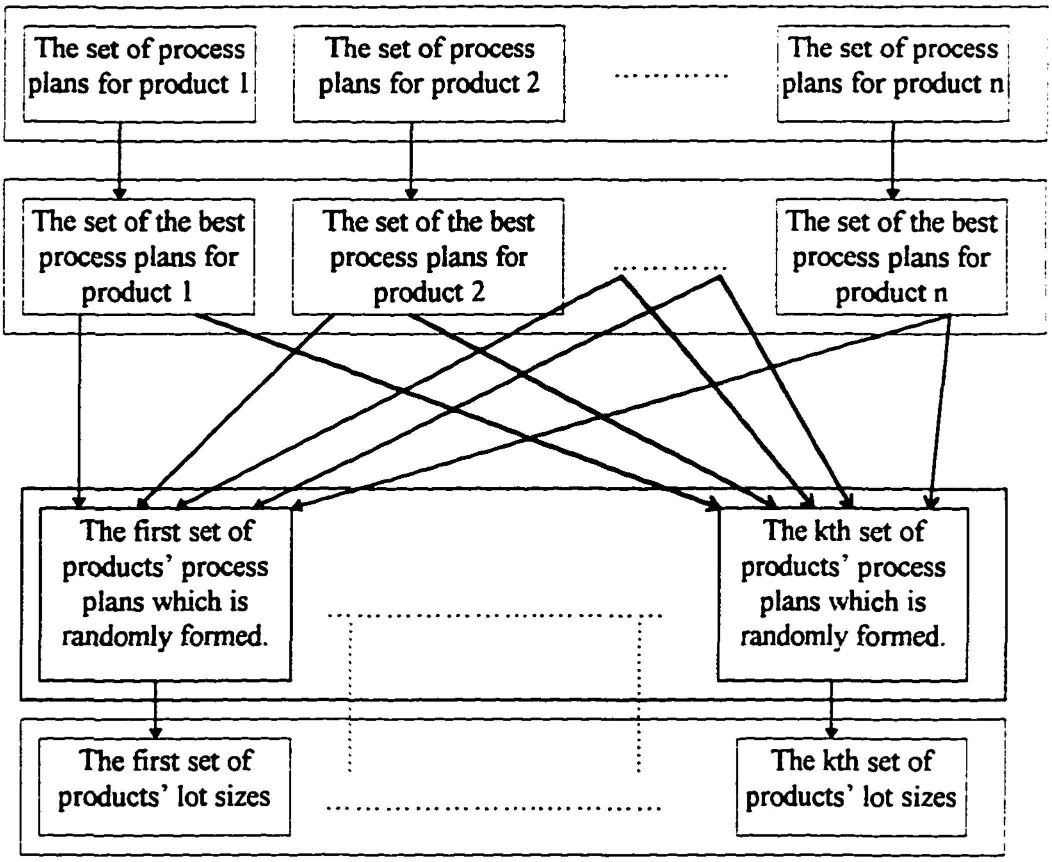


Figure 14. Process plans selection and lot sizes optimization.

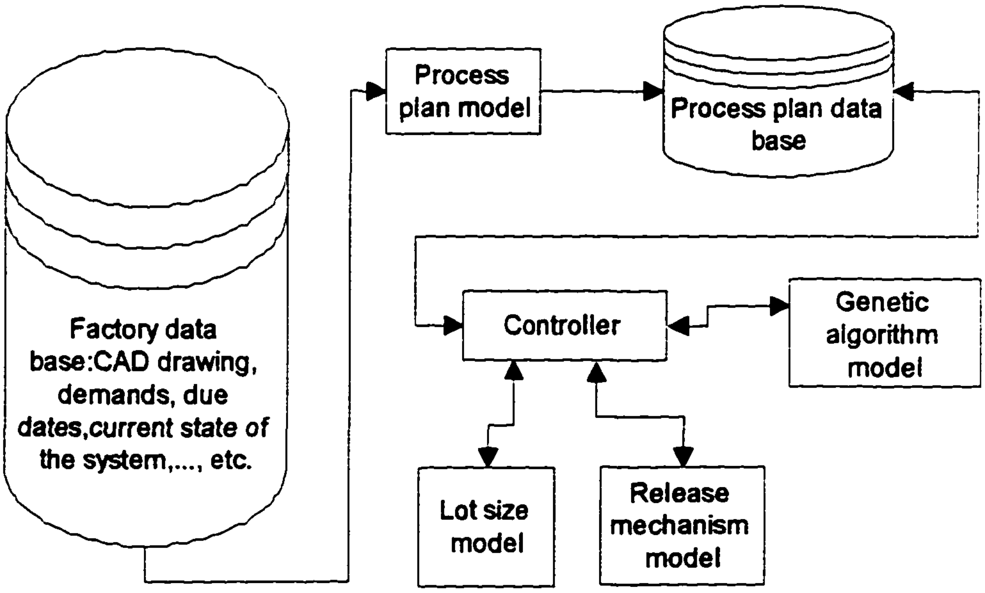


Figure 15. The CGA_APP model components.

This completes the description of the integrated production model, CGA_APP, that controls a dynamic stochastic job shop environment. It is dynamic because at different time in the production horizon different sets of products are produced. Also, the lot sizes for a product can be released at different time. It is a stochastic model because stochastic process times are used in the CGA_APP model. Figure 16 demonstrates the structure of the CGA_APP.

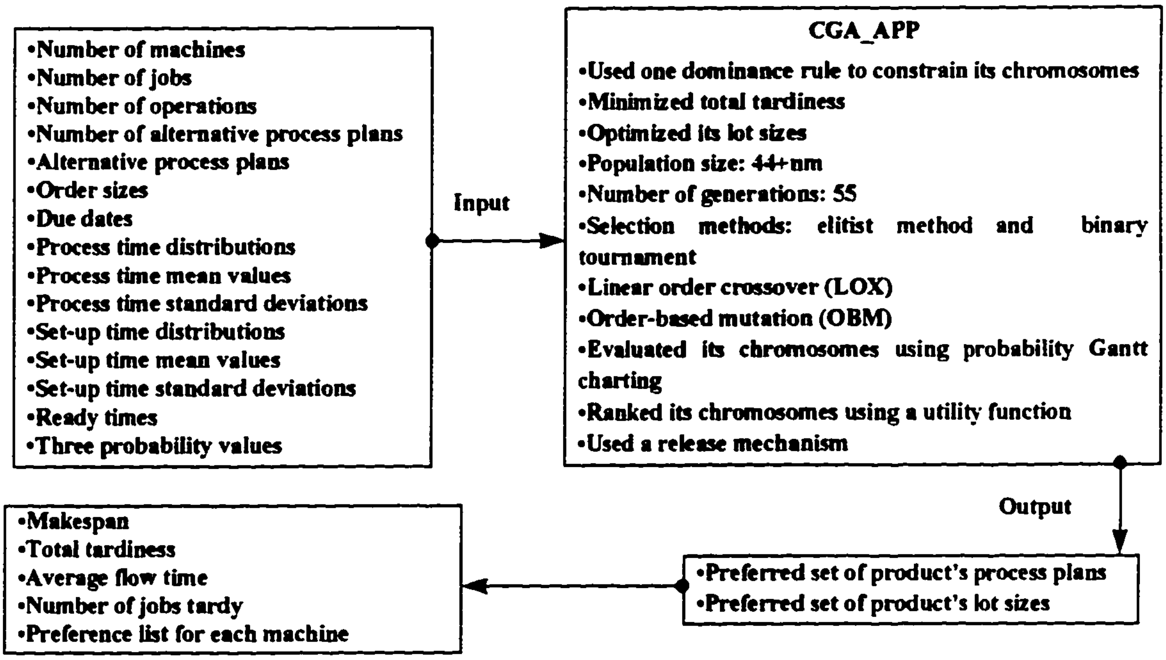


Figure 16. CGA_APP structure.

It should be clear from Figure 14 that to implement the CGA_APP model, the population of chromosomes must be divided into k sub-populations. Each of these sub-populations consists of chromosomes generated according to products that belong to one of the k sets of product process plans. These sub-populations must be constructed for two reasons. First, the selected products' process plans and their optimized lot sizes for one of the k sets cannot be mixed with one of the other k sets. The second reason is to simplify the genetic operators' implementations. Figure 17 shows the structure of the population of chromosomes that contains k sub-populations.

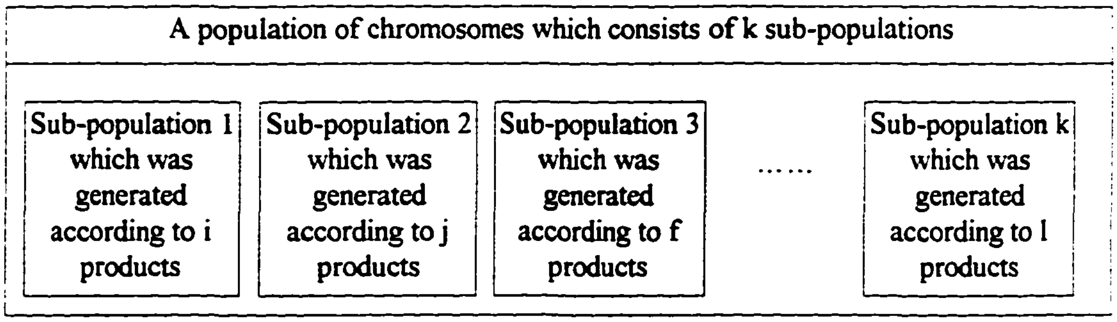


Figure 17. A population of chromosomes which consists of k sub-populations.

From Figure 17, it should be clear that the lengths of chromosomes in each sub-population are different because each sub-population was generated according to different numbers of products. For example, sub-population 1 was generated according to i products while sub-population k was generated according to l products where $i \neq l$. Having chromosomes with different lengths in each sub-population has made the implementation of the genetic operators very difficult. Also, it complicated the computer program coding and execution.

Several complications came up during the attempts to implement the CGA_APP which caused the incompletion of this model. One of these complications occurred in the attempts to integrate the lot size model in the CGA_APP. Other complications came up when attempts were made to implement the genetic operators because of the different chromosome lengths in each sub-population. However, the implementation of the CGA_APP model will be demonstrated in experiment VI on which each component of the CGA_APP model was implemented separately.

Pilot Investigations

In this section, a list of some of the pilot investigations that have been performed in this study will be given. These pilot investigations were performed to tune some of the parameters in all of the GA models. In these pilot investigations, three well-known benchmarks were used, which were designed by Fisher and Thompson (1963). The list of the pilot investigations is as follows:

1. Recall from the "Schedule Building and Fitness Function Evaluation" section that when the total tardiness was minimized, the other three performance measures were used to break ties in the

following order: 1) sum of the makespan and the average flow time, then 2) the number of jobs tardy. A pilot study was performed to investigate breaking ties not only by the preceding order but also in the following order: 1) the makespan; 2) the average flow time, then 3) the number of jobs tardy. The results obtained by using both orders to break ties showed that the first order was better.

2. Recall from the "Selection Methods" section that in the binary tournament two parents were selected and two children were produced. However, only one of them entered the pool of the potential chromosomes for the next generation. Another way of handling this tournament is to select two parents and allow the two children produced to enter the pool. Hence, these two ways were investigated in a pilot study and from the results obtained the first way of handling the tournament was better.
3. A pilot study was performed to investigate the performance of the CGA with and without the SA approach. From the results obtained, it was clear that incorporating the SA in the CGA was beneficial.
4. In the annealing schedule, when the starting temperature value was computed (T_s), a starting probability (P_s) of accepting a bad solution was uniformly distributed between 0.8 and 0.99. Also, when the cooling parameter was computed, the number of generations (Z_2) to reach the freezing stage was uniformly distributed between 75 and 125. A pilot study was performed to compare the performance of the CGA when the following combinations of P_s and Z_2 were used: (0.99, 125); (0.95, 125); (0.9, 125); (0.85, 125); (0.8, 125); (0.99, 75); (0.95, 75); (0.9, 75); (0.85, 75); (0.8, 75), and (P_s is uniformly distributed between 0.99 and 0.8, and Z_2 is uniformly distributed between 125 and 75). The results showed that the last combination was the best.
5. In dominance rule 1, the d_{ij}, the expected due date of job i on machine j, needed to be determined. A pilot study was performed to compare two methods to compute the d_{ij}. These methods were: $d_{ij} = d_i - a_i$, where d_i is the original due date and a_i is total remaining work for job

i, and $d_{ij} = d_{ij-1} + a_i$, where d_{ij-1} is the expected due date of job i on machine j-1 and a_i is the total time that job i has spent so far in the shop floor. The results showed that the first method was better.

6. When the CGA_WSPT and the CGA_SIM were implemented, chromosomes were ranked according to their utility function value. However, chromosomes could be ranked according to the average total tardiness associated with each chromosome. A pilot study was performed to compare the performance of the CGA_WSPT when both methods were used to rank chromosomes. The result showed that ranking chromosomes according to the utility function value was better.
7. When both the CGA_WSPT and the CGA_SIM were implemented, a target value for the total tardiness was determined as the minimum average total tardiness obtained among chromosomes generated in the initial population. However, the target value can be determined by simulating the job shop in which the SPT could be used to dispatch operations. In a pilot study, these two ways of determining the target were compared. The results showed that the first method was better.

CHAPTER IV

EXPERIMENTS AND EXPERIMENTAL RESULTS

Introduction

In this chapter, seven experiments will be discussed. Experiment I was conducted to investigate the effect of the genetic operator combinations on the performance of the deterministic constrained genetic algorithm to minimize makespan (CGA_Cmax). In experiment II, the impact of the population size on the performance of the CGA_Cmax was investigated. Experiment III compared the performance of the deterministic constrained genetic algorithm to minimize makespan (CGA_Cmax) with the deterministic unconstrained genetic algorithm to minimize makespan (UGA_Cmax). Also, the performance of the deterministic constrained genetic algorithm to minimize total tardiness (CGA_TT) and the deterministic unconstrained genetic algorithm to minimize total tardiness (UGA_TT) were evaluated in experiment IV. Experiment V investigated which of the chromosome evaluation methods was better. The effect of lot sizing and alternative process plans was investigated in experiment VI. Experiment VII investigated the potential gain from incorporating the probability distribution function of the processing times in the genetic algorithm.

For the seven experiments, the computer package STATGRAPHICS™ version 5[6] was used to perform the required ANOVA procedure and Tukey's range test and ranking procedures. The significance level used to test the significance of the factors included in each experiment was

---

[6] STATGRAPHICS is a trademark of Statistical Graphics Corporation.

0.05.

Nine well-known benchmarks were used in the seven experiments. Three of these problems were designed by Fisher and Thompson (1963) and the other six were designed by Lawrence (1984). In Tables 5 and 6, the nine problems are described. In these tables, the problem number and name are given in columns 1 and 2. In columns 3 and 4, the problem size in terms of the number of jobs and machines is given. Also, the problem size with respect to the number of operations is given in column 5. The optimal solution of the problem is given in column 6. From these tables, it should be clear that all problems solved are rectangular size.

Table 5. Benchmarks proposed by Fisher and Thompson.

Problem no.	Problem name	No. of jobs	No. of machines	No. of operation	Optimal solution (makespan)
1	FT06	6	6	36	55
2	FT10	10	10	100	930
3	FT20	20	5	100	1165

Table 6. Benchmarks proposed by Lawrence.

Problem no.	Problem name	No. of jobs	No. of machines	No. of operations	Optimal solution (makespan)
4	LA21	15	10	150	1046
5	LA25	15	10	150	977
6	LA27	20	10	200	1235
7	LA29	20	10	200	1153
8	LA38	15	15	225	1196
9	LA40	15	15	225	1222

The above benchmarks were selected for several reasons. First, they have been used by several researchers to test their GA approaches. They are known to be difficult problems. The optimal solution with respect to the makespan for each of these problems is known, which is good for purposes of comparisons.

The above nine problems were designed to be solved for the makespan performance measure, which does not require the due dates in its computation. Hence, for the first three experiments, jobs were given a common due date which is the optimal makespan of the problem considered. However, for experiments IV, V, and VII, the due dates were computed according the results obtained from experiment III in which jobs were given due date based on flow time estimates. The computational for these due dates will be discussed later in this chapter. In experiment VI, the due dates were computed according to the total work content (TWK) rule.

To have a fair compression for all the GA models when they solved the nine problems, the number of generations was set to 55. Also, the population size was set to 44+4nm.

Experiment I: The Effect of Genetic Operator Combinations

In this section, a description of experiment I is given. This experiment was performed to investigate the impact of the genetic operator combinations on the performance of the CGA_Cmax. Also, this experiment was performed to determine which of the nine operator combinations would be the best for the CGA and UGA versions. The nine operator combinations that were tested in this experiment were as follows:

1. Linear order crossover and scramble sub-sequence mutation (LS).
2. Linear order crossover and order-based mutation (LO).
3. Linear order crossover and position-based mutation (LP).
4. Order-based crossover and scramble sub-sequence mutation (OS).
5. Order-based crossover and order-based mutation (OO).
6. Order-based crossover and position-based mutation (OP).
7. Position-based crossover and scramble sub-sequence mutation (PS).
8. Position-based crossover and order-based mutation (PO).
9. Position-based crossover and position-based mutation (PP).

In this experiment five problems were solved: FT06; FT10; FT20; LA25, and LA29. Ten replicates were made for the first three problems, only five for the last two problems. This means that there were three problems with ten replicates, two problems with five replicates, and nine operator combinations, a total of 360 problems. Using the CGA_Cmax model, the 360 problems were solved.

The results obtained for each problem and for each combination are reported in Tables B.1 through B.45 in Appendix B. These results were summarized and are given in Tables 7 through 11. In these tables, the first column is the combination number. The second column lists four statistics: the average value; the standard deviation value; the maximum value, and the minimum value. In column three, the number of alternatives of the best solution at the end of the evolution process is given. The CPU time needed by the CGA_Cmax model is given in column four. In columns 5, 6, 7, and 8, the following performance measures are given: the makespan; the number of jobs tardy; the average flow time, and the total tardiness. The percentage of error is given in column 9. The percentage of error was calculated as follows:

$$\alpha = 100((Cmax - Cmax_{opt})/Cmax_{opt})$$

Where:

α: The percentage deviation of the solution obtained by the CGA_Cmax from the optimal solution.

Cmax: The makespan obtained by the CGA_Cmax.

$Cmax_{OPT}$: The optimal makespan.

This experiment was designed to have two-factor factorial design. The first factor was the genetic operator combinations and the second factor was the problem number. There were nine levels for the first factor and five levels for the second factor. A two-way ANOVA procedure was conducted on the results obtained. The results showed a significant level of 0.02 for the first factor, which means that the genetic operator combinations are different. To further investigate the significance of these operator combinations, one-way ANOVA Tukey's range test procedures were performed. The results of the range test ranked the combinations as follows: LO; PS; LS; OO; PP; LP; OS; PO, then OP. Also, the results of Tukey's test showed that the LO combination is the only combination that is significantly different from the other eight combinations. Also, there was no significant difference among the other eight combinations.

The above conclusion can be supported by the percentage errors given in the nine tables. It should be clear that when ranking each combination according to the percentage errors for each problem, the LO and LS combinations are the best performers. Also, the LO and LS tied in the first three positions over the five problems. This implies that the LOX method is the best among the crossover methods. Then the question is which mutation method should be selected: OBM or SSM? To answer this question, the mutation methods were ranked according the percentage errors for each problem. Then it was clear that the OBM method is the best performer.

Also, comparing the average percentage errors obtained when using the LO combination with the average percentage errors obtained when using the other eight combinations, the LO combination improved the average percentage errors by approximately 10%.

This implies that the LO combination is the best among the nine combinations. Thus, the LO combination was the only genetic operator combination that was used in experiments II, III, IV, V, VI, and VII.

Table 7. Experiment I: Summary of results obtained for problem FT6.

Case no.	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
I (LS)	Average	160.400	55.085	55.400	0.300	51.150	0.400	0.727
	Std	35.747	2.768	0.699	0.483	0.669	0.699	1.271
	Maximum	188.000	61.900	57.000	1.000	52.000	2.000	3.636
	Minimum	103.000	52.400	55.000	0.000	49.500	0.000	0.000
II (LO)	Average	138.300	66.626	55.700	0.400	50.800	0.700	1.273
	Std	42.620	5.570	0.949	0.516	0.919	0.949	1.725
	Maximum	188.000	72.830	57.000	1.000	51.500	2.000	3.636
	Minimum	93.000	56.470	55.000	0.000	49.500	0.000	0.000
III (LP)	Average	161.600	66.136	55.800	0.400	50.600	0.800	1.455
	Std	41.097	5.901	1.033	0.516	0.966	1.033	1.878
	Maximum	188.000	73.270	57.000	1.000	51.500	2.000	3.636
	Minimum	97.000	58.820	55.000	0.000	49.500	0.000	0.000
IV (OS)	Average	163.100	59.413	56.000	0.500	50.717	1.000	1.818
	Std	38.974	3.255	1.054	0.527	1.179	1.054	1.917
	Maximum	188.000	65.360	57.000	1.000	52.500	2.000	3.636
	Minimum	98.000	55.370	55.000	0.000	49.500	0.000	0.000
V (OO)	Average	143.600	68.351	56.200	0.600	50.650	1.200	2.182
	Std	46.705	10.541	1.033	0.516	1.055	1.033	1.878
	Maximum	188.000	95.520	57.000	1.000	52.000	2.000	3.636
	Minimum	84.000	60.470	55.000	0.000	49.500	0.000	0.000
VI (OP)	Average	152.800	75.002	56.600	0.800	50.983	1.600	2.909
	Std	54.760	26.687	0.843	0.422	1.355	0.843	1.533
	Maximum	188.000	144.620	57.000	1.000	52.500	2.000	3.636
	Minimum	65.000	54.930	55.000	0.000	49.500	0.000	0.000
VII (PS)	Average	177.800	52.883	56.000	0.500	50.467	1.000	1.818
	Std	20.509	2.403	1.054	0.527	0.996	1.054	1.917
	Maximum	188.000	57.390	57.000	1.000	51.500	2.000	3.636
	Minimum	132.000	50.260	55.000	0.000	49.500	0.000	0.000
VIII (PO)	Average	162.000	60.242	55.600	0.300	50.700	0.600	1.091
	Std	29.885	5.371	0.966	0.483	0.827	0.966	1.757
	Maximum	188.000	68.660	57.000	1.000	51.500	2.000	3.636
	Minimum	123.000	51.030	55.000	0.000	49.500	0.000	0.000
IX (PP)	Average	167.000	59.189	55.400	0.200	50.883	0.400	0.727
	Std	32.090	3.763	0.843	0.422	0.774	0.843	1.533
	Maximum	188.000	62.950	57.000	1.000	51.500	2.000	3.636
	Minimum	115.000	53.120	55.000	0.000	49.500	0.000	0.000

Table 8. Experiment I: Summary of results obtained for problem FT10.

Case no.	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
I (LS)	Average	385.300	547.856	965.900	3.900	872.950	88.100	3.860
	Std	134.750	20.305	7.062	0.738	30.260	19.376	0.759
	Maximum	444.000	577.260	984.000	5.000	895.100	110.000	5.806
	Minimum	8.000	515.590	957.000	3.000	814.900	53.000	2.903
II (LO)	Average	441.500	567.703	964.400	3.800	872.900	91.600	3.699
	Std	7.561	20.004	2.271	0.422	33.016	13.664	0.244
	Maximum	444.000	598.360	968.000	4.000	895.100	99.000	4.086
	Minimum	420.000	537.940	960.000	3.000	819.000	53.000	3.226
III (LP)	Average	428.600	559.394	966.000	3.800	879.940	90.900	3.871
	Std	34.056	15.330	3.887	0.422	23.704	22.762	0.418
	Maximum	444.000	584.130	976.000	4.000	895.100	131.000	4.946
	Minimum	342.000	539.970	964.000	3.000	839.300	53.000	3.656
IV (OS)	Average	390.000	558.976	965.200	3.800	878.700	86.500	3.785
	Std	68.082	12.577	1.932	0.422	24.064	19.260	0.208
	Maximum	444.000	579.350	968.000	4.000	895.100	96.000	4.086
	Minimum	261.000	542.280	964.000	3.000	836.200	50.000	3.656
V (OO)	Average	386.100	568.248	964.200	3.800	875.590	84.200	3.677
	Std	54.106	20.085	2.394	0.632	29.671	19.832	0.257
	Maximum	444.000	606.380	968.000	5.000	895.100	99.000	4.086
	Minimum	306.000	542.120	960.000	3.000	821.600	53.000	3.226
VI (OP)	Average	423.400	557.428	966.900	3.400	858.420	82.000	3.968
	Std	43.030	15.374	5.896	0.516	35.130	31.319	0.634
	Maximum	444.000	578.750	978.000	4.000	911.000	137.000	5.161
	Minimum	335.000	534.810	960.000	3.000	821.600	50.000	3.226
VII (PS)	Average	428.800	570.093	963.600	3.700	872.460	79.600	3.613
	Std	31.333	38.080	3.502	0.483	29.653	21.438	0.377
	Maximum	444.000	645.870	968.000	4.000	895.100	96.000	4.086
	Minimum	365.000	541.840	956.000	3.000	821.600	50.000	2.796
VIII (PO)	Average	420.500	566.929	965.200	4.200	875.330	100.300	3.785
	Std	42.019	22.596	1.687	1.033	26.012	25.395	0.181
	Maximum	444.000	618.460	968.000	6.000	895.100	140.000	4.086
	Minimum	322.000	534.150	964.000	3.000	825.000	50.000	3.656
IX (PP)	Average	424.800	570.017	964.900	4.000	887.250	98.800	3.753
	Std	38.415	14.665	5.131	0.471	23.360	22.987	0.552
	Maximum	444.000	594.130	979.000	5.000	900.100	156.000	5.269
	Minimum	351.000	550.570	960.000	3.000	821.600	61.000	3.226

Table 9. Experiment I: Summary of results obtained for problem FT20.

Case no.	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
I (LS)	Average	314.000	1660.036	1183.300	1.900	884.970	29.900	1.571
	Std	162.086	65.077	5.056	0.316	21.927	11.628	0.434
	Maximum	444.000	1784.370	1193.000	2.000	921.500	52.000	2.403
	Minimum	27.000	1576.250	1178.000	1.000	857.550	17.000	1.116
II (LO)	Average	390.700	1778.911	1188.400	1.800	892.510	34.500	2.009
	Std	137.300	207.658	9.348	0.632	23.939	13.517	0.802
	Maximum	444.000	2318.180	1203.000	3.000	927.800	63.000	3.262
	Minimum	7.000	1594.220	1178.000	1.000	860.400	17.000	1.116
III (LP)	Average	365.400	1666.184	1188.800	1.800	875.000	34.400	2.043
	Std	144.608	81.613	9.378	0.422	24.840	14.089	0.805
	Maximum	444.000	1778.820	1210.000	2.000	908.200	57.000	3.863
	Minimum	38.000	1531.110	1178.000	1.000	829.950	17.000	1.116
IV (OS)	Average	211.000	1725.299	1189.400	2.100	912.355	45.400	2.094
	Std	156.323	74.819	9.395	0.738	29.691	24.167	0.806
	Maximum	444.000	1856.700	1203.000	3.000	951.650	74.000	3.262
	Minimum	1.000	1621.510	1178.000	1.000	856.150	13.000	1.116
V (OO)	Average	271.400	1753.798	1186.000	2.000	897.870	32.400	1.803
	Std	199.856	73.158	5.774	0.000	26.328	11.047	0.496
	Maximum	444.000	1851.700	1194.000	2.000	936.850	50.000	2.489
	Minimum	7.000	1646.440	1178.000	2.000	850.050	18.000	1.116
VI (OP)	Average	260.700	1721.600	1191.500	1.900	890.080	38.400	2.275
	Std	175.011	57.815	7.721	0.568	11.986	12.677	0.663
	Maximum	444.000	1821.000	1203.000	3.000	907.500	66.000	3.262
	Minimum	1.000	1598.660	1180.000	1.000	872.400	24.000	1.288
VII (PS)	Average	259.400	1710.346	1185.900	1.700	893.285	30.100	1.794
	Std	146.821	31.754	7.355	0.483	23.561	13.212	0.631
	Maximum	444.000	1755.250	1197.000	2.000	924.000	59.000	2.747
	Minimum	23.000	1666.940	1178.000	1.000	857.750	13.000	1.116
VIII (PO)	Average	288.600	1746.597	1190.000	2.100	901.675	43.300	2.146
	Std	206.618	97.499	8.000	0.876	12.371	20.039	0.687
	Maximum	444.000	1916.350	1203.000	4.000	917.750	88.000	3.262
	Minimum	1.000	1617.940	1178.000	1.000	873.850	13.000	1.116
IX (PP)	Average	267.800	1792.102	1185.100	1.900	886.995	29.300	1.725
	Std	182.850	199.060	8.198	0.568	14.878	11.156	0.704
	Maximum	444.000	2329.770	1203.000	3.000	919.950	46.000	3.262
	Minimum	4.000	1662.980	1178.000	1.000	868.250	17.000	1.116

Table 10. Experiment I: Summary of results obtained for problem LA25.

Case no.	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
I (LS)	Average	271.800	2006.494	1005.000	4.800	922.680	98.600	2.866
	Std	272.705	131.896	5.788	1.483	13.150	45.357	0.592
	Maximum	644.000	2153.030	1015.000	7.000	933.667	174.000	3.889
	Minimum	3.000	1877.080	1000.000	3.000	902.333	51.000	2.354
II (LO)	Average	266.000	1817.002	1003.800	5.000	932.307	85.800	2.743
	Std	343.891	64.436	1.789	0.000	1.403	1.095	0.183
	Maximum	643.000	1906.630	1007.000	5.000	933.667	87.000	3.071
	Minimum	1.000	1748.390	1003.000	5.000	930.867	84.000	2.661
III (LP)	Average	294.800	1742.576	1004.800	5.000	927.987	91.800	2.845
	Std	325.136	62.373	2.049	0.707	15.292	21.970	0.210
	Maximum	644.000	1797.550	1007.000	6.000	938.333	127.000	3.071
	Minimum	1.000	1668.800	1003.000	4.000	901.000	67.000	2.661
IV (OS)	Average	338.800	1985.214	1006.600	4.800	928.533	92.200	3.030
	Std	187.033	63.702	5.899	0.837	6.108	29.132	0.604
	Maximum	643.000	2068.600	1014.000	6.000	936.667	129.000	3.787
	Minimum	179.000	1893.550	1002.000	4.000	921.000	63.000	2.559
V (OO)	Average	375.000	1833.044	1006.600	6.000	936.280	112.000	3.030
	Std	232.911	84.890	2.881	0.707	6.146	17.335	0.295
	Maximum	644.000	1930.250	1010.000	7.000	941.133	139.000	3.378
	Minimum	8.000	1699.290	1002.000	5.000	925.667	94.000	2.559
VI (OP)	Average	447.000	1841.620	1008.600	5.600	932.400	105.800	3.234
	Std	222.841	51.613	4.393	0.894	4.111	29.372	0.450
	Maximum	644.000	1890.320	1014.000	7.000	938.733	147.000	3.787
	Minimum	80.000	1763.930	1003.000	5.000	928.867	84.000	2.661
VII (PS)	Average	183.400	1915.624	1009.000	5.600	933.720	122.400	3.275
	Std	160.082	40.173	5.292	0.894	7.269	41.932	0.542
	Maximum	328.000	1972.700	1017.000	7.000	944.933	176.000	4.094
	Minimum	3.000	1867.680	1003.000	5.000	925.667	84.000	2.661
VIII (PO)	Average	238.600	1832.422	1006.000	5.000	926.800	85.600	2.968
	Std	270.961	30.101	3.606	0.707	10.913	5.857	0.369
	Maximum	644.000	1860.490	1011.000	6.000	937.733	92.000	3.480
	Minimum	40.000	1781.780	1002.000	4.000	909.867	78.000	2.559
IX (PP)	Average	405.200	1788.748	1013.600	5.200	926.747	130.400	3.746
	Std	240.339	48.102	6.542	0.837	12.330	34.122	0.670
	Maximum	644.000	1852.140	1022.000	6.000	939.267	165.000	4.606
	Minimum	1.000	1738.500	1007.000	4.000	906.133	84.000	3.071

Table 11. Experiment I: Summary of results obtained for problem LA29.

Case no.	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
I (LS)	Average	143.400	5472.996	1214.400	9.000	1111.970	319.600	5.325
	Std	186.428	148.490	6.731	1.414	9.935	124.102	0.584
	Maximum	364.000	5663.700	1220.000	11.000	1128.350	464.000	5.811
	Minimum	1.000	5321.510	1203.000	8.000	1104.050	215.000	4.337
II (LO)	Average	116.600	4953.486	1210.400	7.400	1095.820	271.200	4.978
	Std	227.027	108.287	14.605	1.517	22.985	53.719	1.267
	Maximum	522.000	5110.820	1229.000	10.000	1131.200	320.000	6.592
	Minimum	1.000	4810.480	1191.000	6.000	1067.550	186.000	3.296
III (LP)	Average	221.000	5185.452	1212.600	7.600	1108.370	264.800	5.169
	Std	350.582	466.029	8.142	1.949	33.474	116.160	0.706
	Maximum	834.000	5754.160	1224.000	10.000	1141.150	420.000	6.158
	Minimum	4.000	4729.910	1205.000	5.000	1071.650	164.000	4.510
IV (OS)	Average	175.000	5416.996	1223.200	8.800	1101.140	398.000	6.089
	Std	374.047	130.518	4.266	2.387	29.825	98.247	0.370
	Maximum	844.000	5586.090	1229.000	12.000	1125.550	519.000	6.592
	Minimum	1.000	5288.890	1219.000	6.000	1049.500	272.000	5.724
V (OO)	Average	360.800	5170.630	1212.600	7.800	1099.740	300.600	5.169
	Std	344.646	190.425	11.216	1.924	17.718	113.315	0.973
	Maximum	844.000	5439.430	1226.000	10.000	1120.900	431.000	6.331
	Minimum	22.000	5014.970	1200.000	5.000	1084.150	162.000	4.076
VI (OP)	Average	548.400	5223.834	1222.000	8.200	1118.650	393.600	5.984
	Std	341.753	231.483	10.840	1.924	17.204	127.902	0.940
	Maximum	843.000	5613.220	1235.000	10.000	1135.850	488.000	7.112
	Minimum	9.000	5038.870	1206.000	5.000	1094.100	190.000	4.597
VII (PS)	Average	3.200	5577.368	1212.400	8.800	1102.910	312.000	5.152
	Std	3.347	136.944	3.362	1.304	19.003	76.834	0.292
	Maximum	9.000	5820.070	1216.000	10.000	1129.900	380.000	5.464
	Minimum	1.000	5494.590	1207.000	7.000	1084.450	194.000	4.683
VIII (PO)	Average	310.200	4942.396	1218.200	8.600	1113.750	343.400	5.655
	Std	356.028	186.802	7.328	1.817	12.105	87.999	0.636
	Maximum	844.000	5148.560	1227.000	11.000	1129.450	445.000	6.418
	Minimum	3.000	4705.850	1209.000	7.000	1097.950	222.000	4.857
IX (PP)	Average	178.000	5027.176	1213.400	8.400	1103.180	318.400	5.239
	Std	199.541	357.278	2.608	1.517	5.141	57.639	0.226
	Maximum	506.000	5648.310	1217.000	10.000	1108.750	389.000	5.551
	Minimum	16.000	4749.790	1210.000	6.000	1096.100	236.000	4.944

Experiment II: The Effect of Population Size

In this experiment the impact of the population size on the performance of the CGA_Cmax model was investigated. It was mentioned in Chapter III that three population sizes were selected to be tested: 44+nm; 44+2nm, and 44+4nm. The same five problems solved in experiment I were used in this experiment. Thus, there were three population sizes, ten replicates for three problems, and five replicates for two problems, a total of 120 problems. The CGA_Cmax model was used to solve the 120 problems.

The results obtained for each problem and for each population size are reported in Tables C.1 through C.15 in Appendix C. These results were summarized and are given in Tables 12 through 16. These tables have the same design described in the previous section except for the first column. In column 1, the population size number is given. The formula used to compute the percentage of error in the previous section was used in this experiment.

This experiment was designed to have two-factor factorial design. The first factor was the population size and the second factor was the problem number. There were three levels for the first factor and five levels for the second factor. A two-way ANOVA procedure was conducted on the results obtained. The results showed a significant level of 0.0001 for the first factor, which means that the population sizes are significantly different. To further investigate the significance of these population sizes, one-way ANOVA Tukey's range test procedures were performed. The results of the range test ranked the population sizes as follows: 44+4nm; 44+2nm, then 44+nm. Also, the results of Tukey's range test grouped the following population sizes: 44+4nm and 44+2nm. This implies that these two population sizes are not significantly different; however, they are significantly different from 44+nm.

The above analysis of the results suggests that the performance of the CGA_Cmax was the same when the following population sizes were used: 44+4nm and 44+2nm. Also, it recommends that those population sizes were better than 44+nm. However, from Tables 12 through 16 it can be seen that when the population size was increased from 44+nm to 44+2nm, the makespan was improved by approximately 0.5%. Also, increasing the population size from 44+nm to 44+4nm improved the makespan by approximately 0.81%. In addition, the percentages of increase in the CPU times ranged between 48.1% and 82.267% when the population size was increased from 44+nm to 44+2nm. Also, when the population size was increased from 44+nm to 44+4nm, the range of the percentages of increase in the CPU time was between 136.1% and 243.48%. With these marginal improvements in the makespan and the huge increase in the CPU times, the following conclusion is given. It is sufficient to state that when the population size was 44+nm, acceptable results were obtained with both reasonable CPU times and good quality solutions. Therefore, this population size, 44+nm, was the only population size that was used in experiments III, IV, V, VI, and VII.

Table 12. Experiment II: Summary of results obtained for problem FT6.

Population size	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
44+nm	Average	62.600	28.226	55.700	0.700	51.767	1.300	1.273
	Std	21.737	4.984	1.059	1.252	0.763	2.791	1.926
	Maximum	80.000	40.810	58.000	4.000	53.667	9.000	5.455
	Minimum	30.000	22.350	55.000	0.000	51.000	0.000	0.000
44+2nm	Average	83.200	41.788	55.700	0.500	51.050	0.800	1.273
	Std	31.650	9.025	0.949	0.707	1.039	1.135	1.725
	Maximum	116.000	64.100	57.000	2.000	53.000	3.000	3.636
	Minimum	13.000	32.460	55.000	0.000	49.500	0.000	0.000
44+4nm	Average	138.300	66.626	55.700	0.400	50.800	0.700	1.273
	Std	42.620	5.570	0.949	0.516	0.919	0.949	1.725
	Maximum	188.000	72.830	57.000	1.000	51.500	2.000	3.636
	Minimum	93.000	56.470	55.000	0.000	49.500	0.000	0.000

Population size displacement	Percentage of increase in CPU time	Percentage of improvement in makespan
From 44+nm to 44+2nm.	48.048	0.000
From 44+nm to 44+4nm.	136.045	0.000
From 44+2nm to 44+4nm.	59.438	0.000

Table 13. Experiment II: Summary of results obtained for problem FT10.

Population size	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
44+nm	Average	135.600	178.846	976.100	4.200	871.650	126.700	4.957
	Std	11.759	7.365	5.607	0.789	27.260	38.592	0.603
	Maximum	144.000	193.330	987.000	5.000	900.800	185.000	6.129
	Minimum	114.000	167.960	968.000	3.000	825.000	50.000	4.086
44+2nm	Average	239.600	325.977	970.000	3.800	872.200	113.100	4.301
	Std	13.914	17.352	7.860	0.632	33.477	41.908	0.845
	Maximum	244.000	362.560	985.000	5.000	912.200	192.000	5.914
	Minimum	200.000	298.350	960.000	3.000	825.400	53.000	3.226
44+4nm	Average	441.500	567.703	964.400	3.800	872.900	91.600	3.699
	Std	7.561	20.004	2.271	0.422	33.016	13.664	0.244
	Maximum	444.000	598.360	968.000	4.000	895.100	99.000	4.086
	Minimum	420.000	537.940	960.000	3.000	819.000	53.000	3.226

Population size displacement	Percentage of increase in CPU time	Percentage of improvement in makespan
From 44+nm to 44+2nm.	82.267	0.625
From 44+nm to 44+4nm.	217.426	1.199
From 44+2nm to 44+4nm.	74.154	0.577

Table 14. Experiment II: Summary of results obtained for problem FT20.

Population size	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
44+nm	Average	113.900	550.079	1194.000	2.300	893.955	50.900	2.489
	Std	50.382	34.981	7.024	0.675	23.838	22.903	0.603
	Maximum	144.000	622.190	1204.000	4.000	929.600	98.000	3.348
	Minimum	2.000	512.790	1182.000	2.000	858.200	23.000	1.459
44+2nm	Average	179.500	940.457	1186.700	1.800	895.785	31.000	1.863
	Std	87.423	42.228	6.395	0.422	19.382	9.955	0.549
	Maximum	244.000	1013.920	1193.000	2.000	924.150	49.000	2.403
	Minimum	12.000	881.780	1178.000	1.000	869.850	19.000	1.116
44+4nm	Average	390.700	1778.911	1188.400	1.800	892.510	34.500	2.009
	Std	137.300	207.658	9.348	0.632	23.939	13.517	0.802
	Maximum	444.000	2318.180	1203.000	3.000	927.800	63.000	3.262
	Minimum	7.000	1594.220	1178.000	1.000	860.400	17.000	1.116

Population size displacement	Percentage of increase in CPU time	Percentage of improvement in makespan
From 44+nm to 44+2nm.	70.968	0.611
From 44+nm to 44+4nm.	223.392	0.469
From 44+2nm to 44+4nm.	89.154	-0.143

Table 15. Experiment II: Summary of results obtained for problem LA25.

Population size	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
44+nm	Average	104.600	554.848	1016.400	5.400	932.947	137.200	4.033
	Std	95.508	36.611	12.502	1.140	7.142	23.983	1.280
	Maximum	194.000	615.660	1032.000	7.000	940.667	164.000	5.629
	Minimum	3.000	517.950	1007.000	4.000	925.667	105.000	3.071
44+2nm	Average	240.600	985.494	1009.400	4.800	924.600	96.200	3.316
	Std	68.744	42.255	4.615	1.095	13.260	21.347	0.472
	Maximum	343.000	1036.280	1015.000	6.000	940.667	131.000	3.889
	Minimum	176.000	941.750	1003.000	3.000	905.267	78.000	2.661
44+4nm	Average	266.000	1817.002	1003.800	5.000	932.307	85.800	2.743
	Std	343.891	64.436	1.789	0.000	1.403	1.095	0.183
	Maximum	643.000	1906.630	1007.000	5.000	933.667	87.000	3.071
	Minimum	1.000	1748.390	1003.000	5.000	930.867	84.000	2.661

Population size displacement	Percentage of increase in CPU time	Percentage of improvement in makespan
From 44+nm to 44+2nm.	77.615	0.689
From 44+nm to 44+4nm.	227.477	1.240
From 44+2nm to 44+4nm.	84.375	0.555

Table 16. Experiment II: Summary of results obtained for problem LA29.

Population size	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
44+nm	Average	144.200	1442.146	1224.400	10.000	1125.750	409.200	6.193
	Std	129.010	53.474	8.081	1.000	28.681	137.583	0.701
	Maximum	244.000	1482.600	1233.000	11.000	1154.850	601.000	6.938
	Minimum	1.000	1351.390	1214.000	9.000	1080.050	256.000	5.291
44+2nm	Average	202.000	2583.258	1217.000	8.000	1110.120	314.200	5.551
	Std	159.465	83.443	11.853	2.828	34.936	99.746	1.028
	Maximum	442.000	2690.860	1231.000	12.000	1147.250	460.000	6.765
	Minimum	35.000	2470.220	1200.000	6.000	1058.350	218.000	4.076
44+4nm	Average	116.600	4953.486	1210.400	7.400	1095.820	271.200	4.978
	Std	227.027	108.287	14.605	1.517	22.985	53.719	1.267
	Maximum	522.000	5110.820	1229.000	10.000	1131.200	320.000	6.592
	Minimum	1.000	4810.480	1191.000	6.000	1067.550	186.000	3.296

Population size displacement	Percentage of increase in CPU time	Percentage of improvement in makespan
From 44+nm to 44+2nm.	79.126	0.604
From 44+nm to 44+4nm.	243.480	1.143
From 44+2nm to 44+4nm.	91.753	0.542

Experiment III: Comparison of CGA_Cmax and UGA_Cmax

The performance of the CGA_Cmax and the UGA_Cmax were compared in this experiment. The nine problems discussed earlier in this chapter were solved in this experiment. The CGA_Cmax and the UGA_Cmax were used to solve the nine problems using five replicates. Thus, each of these models solved a total of 45 problems.

The results obtained for each problem by both models are reported in Tables D.1 through D.18 in Appendix D. These results were summarized and are given in Tables 17 through 25. These tables have the same design described in the "Experiment I" section except for the first column, which instead of listing the combination number lists the model type. The percentage of error in this experiment was calculated as follows:

$$\alpha_i = 100((Cmax_i - Cmax_{opt})/Cmax_{opt})$$

Where:

α_i: The percentage deviation of the solution obtained by algorithm i from the optimal solution.

$Cmax_i$: The makespan obtained by the algorithm i (i.e., CGA_Cmax or UGA_Cmax).

$Cmax_{OPT}$: The optimal makespan.

This experiment was designed to have two-factor factorial design. The first factor was the model type and the second factor was the problem number. There were two levels for the first factor and nine levels for the second factor. A two-way ANOVA procedure was conducted on the results obtained. The results showed a significant level of 0.00000012 for the first factor, which means that the two models are significantly different. To further investigate the significance of these two models, one-way ANOVA Tukey's range test procedures were performed. The results of the range test showed that they are significantly different and ranked the CGA_Cmax as better than the UGA_Cmax.

The conclusion from the Tukey's range test can be supported by the percentage errors obtained for each problem by both models. From the results obtained, it can be seen that the average percentage errors over the nine problems for the CGA_Cmax ranged between 1.091% and 6.672% and the range for the UGA was between 2.5% and 8.077%. Also, when the average percentage errors over the nine problems obtained by both models were compared, the CGA_Cmax improved the average percentage errors by approximately 27.44%. From these results, it should be clear that the CGA_Cmax performed better than the UGA_Cmax.

The adaptation curves of both the CGA_Cmax and UGA_Cmax are given in Figure 18 for the LA25 problem with respect to the best makespan obtained in each generation. From Figure 18, it can be seen that the CGA achieved better results in fewer generations, which supports the hypothesis that the CGA should perform better than the UGA using fewer generations.

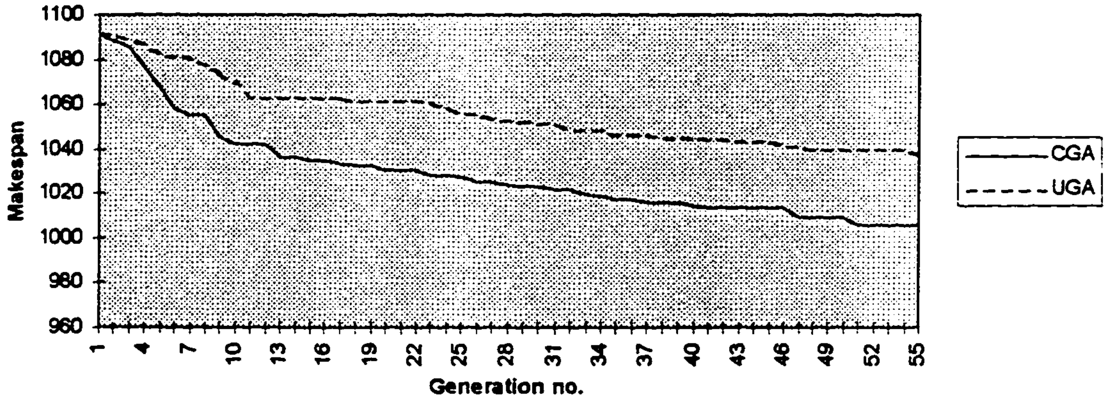


Figure 18. The adaptation curves for the CGA_Cmax and the UGA_Cmax for the LA25 problem.

From the nine tables, it can be seen that the averages of the CPU times needed by the UGA_Cmax were lower than the CPU times needed by the CGA_Cmax except for in one problem. The average CPU time needed by the CGA_Cmax over the nine problems was 805.39 seconds, while the average CPU time required by the UGA_Cmax was 767.37 seconds. This means that the UGA reduced the CPU time by only 4.7%, which is an insignificant reduction.

Table 17. Experiment III: Summary of results obtained for problem FT6.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_Cmax	Average	59.200	24.736	55.600	0.800	51.833	1.800	1.091
	Std	23.669	2.216	1.342	1.789	1.048	4.025	2.439
	Maximum	80.000	27.190	58.000	4.000	53.667	9.000	5.455
	Minimum	21.000	22.570	55.000	0.000	51.000	0.000	0.000
UGA_Cmax	Average	73.000	20.926	57.000	1.000	49.600	2.000	3.636
	Std	6.285	0.498	0.000	0.000	0.091	0.000	0.000
	Maximum	78.000	21.640	57.000	1.000	49.667	2.000	3.636
	Minimum	63.000	20.330	57.000	1.000	49.500	2.000	3.636

Table 18. Experiment III: Summary of results obtained for problem FT10.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_Cmax	Average	85.600	175.432	971.200	4.000	866.180	118.200	4.430
	Std	69.762	6.490	7.155	1.000	38.032	45.483	0.769
	Maximum	144.000	180.980	976.000	5.000	895.600	184.000	4.946
	Minimum	6.000	167.900	960.000	3.000	823.700	61.000	3.226
UGA_Cmax	Average	132.200	171.458	970.000	4.200	853.100	118.000	4.301
	Std	20.789	9.941	6.285	1.643	45.494	57.000	0.676
	Maximum	144.000	188.670	976.000	7.000	934.000	178.000	4.946
	Minimum	96.000	163.130	960.000	3.000	825.400	53.000	3.226

Table 19. Experiment III: Summary of results obtained for problem FT20.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_Cmax	Average	89.400	527.978	1194.200	2.600	890.560	53.600	2.506
	Std	68.948	12.993	7.294	1.140	24.969	22.131	0.626
	Maximum	144.000	537.500	1200.000	4.000	930.350	73.000	3.004
	Minimum	3.000	506.090	1182.000	1.000	861.700	17.000	1.459
UGA_Cmax	Average	112.800	492.738	1194.200	2.000	806.090	52.000	2.506
	Std	59.403	26.103	13.864	0.000	32.207	28.258	1.190
	Maximum	144.000	529.870	1210.000	2.000	856.700	82.000	3.863
	Minimum	7.000	468.680	1180.000	2.000	768.550	19.000	1.288

Table 20. Experiment III: Summary of results obtained for problem LA21.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_Cmax	Average	166.000	578.586	1102.000	4.200	973.413	152.400	5.354
	Std	53.670	16.172	6.000	0.447	10.702	36.562	0.574
	Maximum	191.000	600.110	1112.000	5.000	988.200	216.000	6.310
	Minimum	70.000	558.320	1097.000	4.000	960.200	127.000	4.876
UGA_Cmax	Average	38.600	582.616	1100.200	4.200	963.040	175.600	5.182
	Std	38.914	21.990	6.058	0.447	9.866	28.919	0.579
	Maximum	84.000	609.890	1109.000	5.000	975.067	206.000	6.023
	Minimum	1.000	555.030	1094.000	4.000	951.400	144.000	4.589

Table 21. Experiment III: Summary of results obtained for problem LA25.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_Cmax	Average	80.000	597.598	1005.600	5.200	929.600	102.200	2.927
	Std	69.653	14.152	3.975	0.447	2.277	29.794	0.407
	Maximum	188.000	616.100	1012.000	6.000	933.667	155.000	3.582
	Minimum	22.000	577.050	1003.000	5.000	928.467	84.000	2.661
UGA_Cmax	Average	79.800	540.908	1037.600	4.800	880.000	179.800	6.203
	Std	85.692	26.877	8.905	1.095	28.661	47.267	0.911
	Maximum	194.000	586.170	1051.000	6.000	926.333	234.000	7.574
	Minimum	1.000	519.710	1029.000	3.000	849.933	127.000	5.322

Table 22. Experiment III: Summary of results obtained for problem LA27.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_Cmax	Average	49.400	1607.736	1291.400	5.400	1147.420	191.800	4.567
	Std	52.276	69.983	10.188	2.074	15.639	104.428	0.825
	Maximum	129.000	1708.180	1305.000	8.000	1166.500	368.000	5.668
	Minimum	2.000	1530.610	1278.000	3.000	1129.700	89.000	3.482
UGA_Cmax	Average	70.400	1483.432	1305.400	5.800	1142.540	256.400	5.700
	Std	101.530	39.216	7.603	0.447	14.746	39.450	0.616
	Maximum	244.000	1536.430	1314.000	6.000	1167.150	303.000	6.397
	Minimum	1.000	1440.920	1296.000	5.000	1128.050	196.000	4.939

Table 23. Experiment III: Summary of results obtained for problem LA29.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_Cmax	Average	52.400	1547.418	1217.400	9.800	1120.280	395.800	5.585
	Std	104.923	44.259	6.387	2.168	26.240	123.435	0.554
	Maximum	240.000	1607.780	1224.000	12.000	1151.450	576.000	6.158
	Minimum	3.000	1508.420	1208.000	7.000	1079.850	262.000	4.770
UGA_Cmax	Average	124.200	1473.432	1236.000	9.400	1098.230	464.600	7.199
	Std	118.122	65.485	10.320	0.894	17.542	113.997	0.895
	Maximum	243.000	1540.770	1250.000	10.000	1119.000	636.000	8.413
	Minimum	1.000	1371.490	1227.000	8.000	1071.600	341.000	6.418

Table 24. Experiment III: Summary of results obtained for problem LA38.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_Cmax	Average	69.400	1104.334	1275.800	5.400	1146.973	231.000	6.672
	Std	114.921	47.548	13.864	0.894	7.187	74.887	1.159
	Maximum	269.000	1154.540	1300.000	7.000	1159.333	364.000	8.696
	Minimum	1.000	1059.460	1268.000	5.000	1140.533	190.000	6.020
UGA_Cmax	Average	102.400	1079.124	1292.600	5.400	1148.760	333.800	8.077
	Std	128.282	28.378	7.470	0.548	11.940	78.085	0.625
	Maximum	267.000	1110.540	1303.000	6.000	1160.133	449.000	8.946
	Minimum	7.000	1049.630	1282.000	5.000	1131.200	259.000	7.191

Table 25. Experiment III: Summary of results obtained for problem LA40.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_Cmax	Average	222.000	1084.702	1277.000	4.800	1159.267	168.000	4.501
	Std	87.247	35.200	2.236	0.447	10.845	42.497	0.183
	Maximum	269.000	1120.320	1278.000	5.000	1176.067	226.000	4.583
	Minimum	67.000	1028.090	1273.000	4.000	1148.000	135.000	4.173
UGA_Cmax	Average	249.800	1061.718	1285.000	5.000	1140.640	202.800	5.156
	Std	32.813	42.759	6.928	0.707	13.622	48.515	0.567
	Maximum	269.000	1126.900	1294.000	6.000	1160.667	272.000	5.892
	Minimum	192.000	1009.250	1278.000	4.000	1125.933	139.000	4.583

The following table compares the results of the best solution obtained by the CGA_Cmax for the nine problems with the results of other GA approaches. These approaches were selected because they are the only approaches that solved the problems that were solved in this research.

Table 26. Comparison of the CGA_Cmax with other GA approaches.

Problem	Optimal	CGA	N&Y	P_GA	SB_GA	GP_GA	CTV	GTK	GA3
FT06	55	55	55	55	55	55	55	55	55
FT10	930	960	965	960	938	936	946	962	930
FT20	1165	1182	1215	1249	1178	1181	1178	1175	1165
LA21	1046	1097	-	1139	1074	-	1097	-	1047
LA25	977	1003	-	1014	1008	-	-	-	977
LA27	1235	1278	-	1378	1272	1269	-	-	1236
LA29	1153	1208	-	1336	1204	1233	-	-	1180
LA38	1196	1268	-	1296	1251	1251	-	-	1201
LA40	1222	1273	-	1321	1274	1252	-	-	1228

N&Y: A GA approach by Nakano and Yamada (1991).
P_GA: A GA approach by Dorndorf and Pesch (1995).
SB_GA: A GA approach by Dorndorf and Pesch (1995).
GP_GA: A GA approach by Bierwirth (1995).
CTV: A GA approach by Croce, Tadei, and Volta (1995).
GTK: A GA approach by Gen, Tsumjimura, and Kubota (1994).
GA3: A GA approach by Mattfeld (1996).

It should be clear from Table 26 that only four GA approaches solved all the nine problems that were solved in this research study. However, this does not make this comparison insignificant. All the above GA approaches solved the three problems designed by Fisher and Thompson (1963) and four of them solved the problems designed by Lawrence (1984). From Table 26, it is clear that the CGA obtained reasonable results given the fact that all of the other GA approaches were tuned to obtained the best for most of the problems solved. Also, all of these GA approaches were implemented with larger number of replicates, larger number of generations, and larger population sizes (refer to Chapter II for details of these approaches). For example, the N&Y approach was implemented using a population size of 1000 and the number of generations was 150. Also, the number of generations in the GTK approach was 5000. The best solutions for both the GP_GA and GA3 were obtained after 100 replications and 30 replications respectively.

Table 27 compares the results of the best solution obtained by the CGA_Cmax for the nine problems with the results of other approaches. In Table 27, the fourth column presents the results obtained by the shifting bottleneck algorithm. Columns 5 and 6 give the results for two tabu search approaches. In columns 7 and 8, the results obtained by two simulated annealing algorithms are given. The last column presents the results of a hybrid approach, which is a combination of a simulated annealing algorithm and the shifting bottleneck algorithm. Again, from Table 27, it is clear that the CGA obtained reasonable results given the fact that it was not tuned to compete with other approaches.

Table 27. Comparison of the CGA_Cmax with other approaches.

Problem	Optimal	CGA	ABZ	DT	NS	LAL	YRN	YN
FT06	55	55	55	55	55	55	-	-
FT10	930	960	930	935	930	930	-	930
FT20	1165	1182	1178	1165	1165	1165	-	1165
LA21	1046	1097	1084	1048	1055	1063	1050	1046
LA25	977	1003	1017	979	988	992	985	977
LA27	1235	1278	1291	1242	1259	1269	1262	1235
LA29	1153	1208	1239	1182	1164	1218	1188	1154
LA38	1196	1268	1255	1203	1209	1215	1209	1198
LA40	1222	1273	1269	1233	1234	1234	1235	1228

ABZ: The Shifting Bottleneck (SB) algorithm by Adams, Balas, and Zawack (1988).
DT: A tabu search (TA) approach by Dell'Amico and Trubian (1993).
NS: A tabu search (TA) approach by Nowicki and Smutnicki (1996).
LAL: A simulated annealing (SA) algorithm by Laarhoven, Aarts, and Lenstra (1992).
YRN: A simulated annealing (SA) algorithm by Yamada, Rosen, and Nakano (1994).
YN: A hybrid approach (SA plus SB) by Yamada and Nakano (1996).

Experiment IV: Comparison of CGA_TT and UGA_TT

In this experiment the performance of the CGA_TT and the UGA_TT models were compared. The same nine problems that were solved in experiment III were used in this experiment. As mentioned earlier in this chapter, the due dates for the problems solved in experiments IV, V, and VII were computed using the results obtained in experiment III. The due dates in this experiment were computed using the procedures given below. These procedures produced very tight due dates and consequently very difficult problems, which is good for the purpose of comparisons. These procedures are as follows:

1. From the results given in the tables in Appendix D, the best replicate among the five replicates for each problem was selected first. For example, from Table D.1 the best replicate is replicate number three.
2. Then, for each problem, several completion times for each job were computed using the number of alternatives of the best solution obtained for the replicate selected. For example, from Table D.1 the number of alternatives associated with the third replicate is 80. This means that there were 80 completion times for each job for the FT06 problem.
3. Next, for each problem, the average and the standard deviation for each job's completion were

computed: $\overline{C_i}$ & σ_{C_i}. Hence, for the 80 alternative sequences, the average and the standard deviation for each job's completion were computed.

4. Then, for each problem, the maximum average completion time among all jobs was determined: $\overline{C}_{max}$.
5. Finally, for each problem, the due date for each job was computed as follows: $d_i = \overline{C_i}$ - max[($\overline{C}_{max}$ -optimal solution), σ_{C_i}].

Thus, the above procedures which are based on job flow time estimates were used to compute the due dates for jobs in the nine problems. Then the nine problems were solved by both the CGA_TT and the UGA_TT, using five replicates. To compare the results of the CGA_TT and the UGA_TT, each of the nine problems was solved by sampling from active and non-delay schedules developed using dispatching heuristics. The sample size was 1000 schedules in which 500 schedules were sampled from active schedules and the rest were sampled from non-delay schedules. The dispatching heuristics used to generate the initial population of chromosomes were used to sample from both schedule types. Then, the result of the dispatching heuristic that obtained the best total tardiness was compared with the results obtained by both the CGA_TT and the UGA_TT.

The results obtained for each problem by both models are reported in Tables E.1 through E.18 in Appendix E. These results were summarized and are given in Tables 28 through 36. These tables have the same design described in the "Experiment I" section except for the first column and the last row. The first column gives the model type instead of the combination number. The last row in each table represents the results obtained by the dispatching heuristic. In this row, column 2 gives the dispatching heuristic name that obtained the minimum total tardiness among the 1000 solutions sampled. In this experiment the percentage of error was computed as follows:

$$\alpha_i = 100((TT_i - TT_{Best})/TT_{Best})$$

Where:

α_i: The percentage deviation of the solution obtained by algorithm i from the best solution.

TT_i: The total tardiness obtained by the algorithm i (i.e., CGA_TT, UGA_TT, or dispatching heuristic).

TT_{Best}: The best obtained total tardiness.

The two-factor factorial design used in experiment III was used in this experiment. A two-way ANOVA procedure was conducted on the results obtained. The results showed a significance level of 0.000000005 for the first factor, which means that the CGA_TT and the UGA_TT are significantly different. To further investigate the significance of these two models, one-way ANOVA Tukey's range test procedures were performed. The results of the range test ranked the CGA_TT before the UGA_TT.

It can be seen from the nine tables that the CGA_TT and UGA_TT outperform the dispatching heuristics in all problems. The CGA_TT and UGA_TT improved the average percentage errors over the best heuristic by 1388.11% and 326.68% respectively.

The Tukey's range test results can be supported by the results obtained for the percentage errors given in the nine tables. From these tables, the percentage errors for the CGA_TT ranged between 0% and 22.38%, while the range for the UGA_TT was between 8.63% and 165.19%. When the average percentage errors over the nine problems obtained by both models were compared, the CGA_TT improved the average percentage errors by approximately 248.77%. From these results it is obvious that the CGA_TT outperformed the UGA_TT.

The adaptation curves of both the CGA_TT and UGA_TT are given in Figure 19 for the FT20 problem with respect to the best total tardiness obtained in each generation. It should be clear that the CGA_TT achieved better results in fewer generations, as shown in Figure 19.

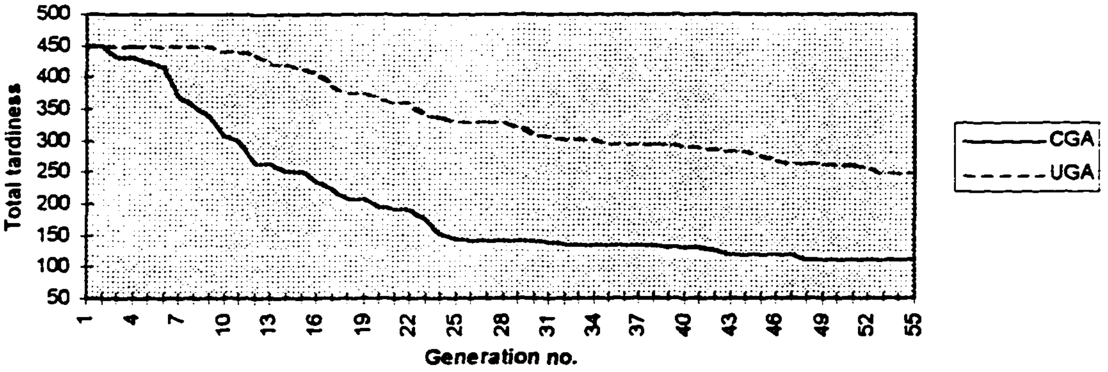


Figure 19. The adaptation curves for the CGA_TT and the UGA_TT for the FT20 problem.

In the nine tables, it can be seen that the averages for the CPU times needed by the CGA_TT were lower than the CPU times needed by the UGA_TT. From the results given in Table 27 through 35, the average CPU time over the nine problems for both the CGA_TT and the UGA_TT can be computed. The average for the CGA_TT was 754.16 seconds and the average for the UGA_TT was 835.12 seconds. This means that the CGA_TT reduced the CPU time by approximately 11%. The results support the hypothesis that the CGA approach obtained better solutions with much less computational effort than the UGA approach.

Table 28. Experiment IV: Summary of results obtained for problem FT6.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_TT	Average	63.800	22.126	55.000	4.000	51.000	5.750	0.000
	Std	18.539	0.602	0.000	0.000	0.000	0.000	0.000
	Maximum	79.000	23.010	55.000	4.000	51.000	5.750	0.000
	Minimum	34.000	21.540	55.000	4.000	51.000	5.750	0.000
UGA_TT	Average	73.600	23.332	56.800	4.000	50.233	7.436	29.32
	Std	14.311	0.959	1.643	0.000	0.435	1.054	18.33
	Maximum	80.000	24.880	58.000	4.000	51.000	8.130	41.39
	Minimum	48.000	22.300	55.000	4.000	50.000	5.750	0.00
Best dispatching heuristic	JST(A) and OST(A)		2.250	57.000	6.000	52.833	15.56	170.61

Table 29. Experiment IV: Summary of results obtained for problem FT10.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_TT	Average	144.000	164.700	1007.800	7.400	813.160	308.832	13.092
	Std	0.000	2.672	55.283	2.510	8.431	33.909	12.417
	Maximum	144.000	168.020	1094.000	10.000	821.600	346.000	26.703
	Minimum	144.000	160.770	960.000	5.000	802.000	273.080	0.000
UGA_TT	Average	142.800	182.662	977.200	8.000	816.760	296.648	8.630
	Std	1.789	4.684	23.552	2.739	6.628	32.272	11.818
	Maximum	144.000	188.400	1003.000	10.000	821.600	332.000	21.576
	Minimum	140.000	176.760	960.000	5.000	809.400	273.080	0.000
Best dispatching heuristic	Biased-RANDOM(ND)		9.170	1148.000	8.000	848.8	638	133.63

Table 30. Experiment IV: Summary of results obtained for problem FT20.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_TT	Average	128.400	425.014	1184.400	4.000	752.390	111.716	18.923
	Std	18.982	16.726	3.286	0.707	2.346	15.195	16.175
	Maximum	144.000	446.440	1188.000	5.000	754.350	123.900	31.893
	Minimum	101.000	406.390	1182.000	3.000	748.450	93.940	0.000
UGA_TT	Average	23.600	514.826	1215.000	8.000	759.040	249.120	165.191
	Std	29.594	34.750	12.000	2.000	6.177	25.273	26.903
	Maximum	73.000	573.640	1228.000	11.000	765.400	279.010	197.009
	Minimum	1.000	484.380	1201.000	6.000	751.250	214.970	128.838
Best dispatching heuristic	A/OPN(ND)		17.630	1228.000	8.000	776.1	449.58	378.58

Table 31. Experiment IV: Summary of results obtained for problem LA21.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_TT	Average	89.400	576.068	1163.800	5.800	933.067	652.588	15.002
	Std	89.941	16.013	51.163	1.789	16.413	56.832	10.015
	Maximum	192.000	595.880	1205.000	8.000	951.933	725.060	27.773
	Minimum	1.000	551.950	1093.000	4.000	914.267	567.460	0.000
UGA_TT	Average	83.400	648.760	1129.000	6.400	927.120	667.680	17.661
	Std	101.520	36.416	28.618	1.517	20.451	32.767	5.774
	Maximum	194.000	705.140	1177.000	8.000	946.600	707.660	24.707
	Minimum	1.000	613.470	1102.000	5.000	898.067	627.060	10.503
Best dispatching heuristic	Biased-RANDOM(ND)		18.560	1205.000	10.000	967.067	1115.86	96.64

Table 32. Experiment IV: Summary of results obtained for problem LA25.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_TT	Average	65.600	565.832	1055.800	5.600	865.720	335.594	22.381
	Std	77.861	25.583	7.662	1.673	10.925	38.491	14.037
	Maximum	194.000	586.660	1068.000	8.000	880.200	378.690	38.097
	Minimum	2.000	530.910	1047.000	4.000	851.467	274.220	0.000
UGA_TT	Average	109.800	607.036	1127.600	4.000	865.987	379.218	38.290
	Std	86.511	16.768	119.427	0.707	11.050	46.006	16.777
	Maximum	194.000	636.540	1333.000	5.000	878.667	423.190	54.325
	Minimum	10.000	595.990	1055.000	3.000	853.067	323.240	17.876
Best dispatching heuristic	ATC(ND)		19.340	1376.000	3.000	905.133	743.89	171.27

Table 33. Experiment IV: Summary of results obtained for problem LA27.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_TT	Average	55.800	1462.720	1398.600	11.400	1081.720	630.292	14.368
	Std	104.488	27.305	142.476	2.408	9.847	45.475	8.252
	Maximum	242.000	1506.170	1570.000	14.000	1094.250	663.110	20.323
	Minimum	1.000	1435.150	1279.000	8.000	1068.250	551.110	0.000
UGA_TT	Average	31.600	1627.498	1503.600	11.600	1094.800	818.848	48.582
	Std	44.236	73.902	85.670	1.342	18.019	117.042	21.238
	Maximum	108.000	1741.690	1600.000	13.000	1118.950	1015.140	84.199
	Minimum	1.000	1538.800	1375.000	10.000	1079.450	729.960	32.453
Best dispatching heuristic	OCR(ND)		32.520	1363.000	17.000	1158.7	1574.11	185.63

Table 34. Experiment IV: Summary of results obtained for problem LA29.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_TT	Average	9.200	1426.282	1510.800	7.000	1028.170	761.802	7.447
	Std	12.418	60.230	102.984	2.345	35.547	43.464	6.130
	Maximum	29.000	1518.800	1592.000	11.000	1080.250	814.000	14.810
	Minimum	1.000	1371.270	1353.000	5.000	980.000	709.000	0.000
UGA_TT	Average	51.600	1575.920	1573.600	8.000	1038.790	957.870	35.102
	Std	107.565	72.651	57.440	1.225	11.339	52.570	7.415
	Maximum	244.000	1660.060	1647.000	9.000	1052.350	1030.670	45.370
	Minimum	1.000	1489.250	1489.000	6.000	1022.950	909.000	28.209
Best dispatching heuristic	ATC(ND)		30.920	1508.000	9.000	1102.750	1681.670	137.19

Table 35. Experiment IV: Summary of results obtained for problem LA38.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_TT	Average	153.800	1070.728	1370.400	10.800	1108.480	796.484	14.903
	Std	133.915	30.650	34.782	1.304	10.002	64.001	9.233
	Maximum	269.000	1110.590	1429.000	12.000	1117.133	858.540	23.855
	Minimum	3.000	1034.410	1339.000	9.000	1092.667	693.180	0.000
UGA_TT	Average	112.000	1140.860	1358.600	11.600	1113.813	776.488	12.018
	Std	125.427	20.400	49.501	1.140	9.123	17.232	2.486
	Maximum	269.000	1173.270	1439.000	13.000	1120.867	795.280	14.729
	Minimum	1.000	1118.560	1313.000	10.000	1099.867	757.180	9.233
Best dispatching heuristic	Biased-RANDOM(ND)		28.340	1523.000	12.000	1139.867	1510.28	117.88

Table 36. Experiment IV: Summary of results obtained for problem LA40.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
CGA_TT	Average	156.800	1074.002	1316.000	11.000	1107.053	367.810	0.000
	Std	110.692	33.850	0.000	0.000	0.119	0.000	0.000
	Maximum	260.000	1114.660	1316.000	11.000	1107.267	367.810	0.000
	Minimum	1.000	1033.530	1316.000	11.000	1107.000	367.810	0.000
UGA_TT	Average	175.200	1195.202	1347.400	9.800	1112.613	424.104	15.305
	Std	109.177	36.736	43.322	1.643	7.954	77.193	20.987
	Maximum	269.000	1238.680	1402.000	11.000	1124.733	514.340	39.839
	Minimum	14.000	1142.180	1316.000	8.000	1107.000	367.810	0.000
Best dispatching heuristic	Biased-RANDOM(ND)		27.080	1359.000	11.000	1152.6	1058.15	187.69

Experiment V: Comparison of CGA_WSPT and CGA_SIM

The purpose of experiment V was to perform the following four comparisons:

1. Compare the CGA_WSPT with the CGA_SIM.
2. Compare the CGA_SIM with the UGA_SIM.
3. Compare the CGA_WSPT with the UGA_WSPT.
4. Compare the UGA_WSPT with the UGA_SIM.

According to the results obtained in experiments III and IV, the CGA versions outperformed the UGA versions. Thus, according to this result, it is sufficient to state that there is no need to compare the UGA versions with the CGA versions in this experiment. Therefore, in this experiment only the CGA_WSPT and the CGA_SIM were compared.

In this experiment, seven of the nine problems discussed earlier were solved. These seven problems were FT06, FT10, FT20, LA21, LA25, LA38, and LA40. For each of the seven problems, a normal distribution was associated with the processing times of each operation. This implies that the process times in the original problem were used as the mean values for the normal distribution. The standard deviation for each of these process times was uniformly distributed between $0.05P_{ij}$ and $0.25P_{ij}$, where P_{ij} is the processing time of job i for operation j. The same due dates that were used in experiment IV were used in this experiment. As mentioned in the previous chapter, when the CGA_WSPT was implemented, three probability levels were used to evaluate each chromosome. Liang (1996) concluded that three Pr values resulted in good estimates for the true mean when the normal distribution was used to generate process times. The errors in these estimates were less than 10%. The three Pr values were 0.5, 0.54, and 0.58. Hence, these three Pr values were used in this study. The CGA_WSPT and the CGA_SIM were used to solve the seven problems using five replicates for each problem.

The results obtained for each problem by both models are reported in Tables F.1 through

F.14 in Appendix F. These results were summarized and are given in Tables 37 through 43. These seven tables were designed to use the same statistics presented in the earlier tables. However, the absolute percentage of error was computed for the four performance measures used in this study. Hence, the absolute percentage errors for these four performance measures are given in the seven tables. Also, 90% confidence intervals on all the four performance were constructed and are given in these tables. The four statistics were computed for the number of replicates made by the simulation to evaluate a chromosome, and are also given in the seven tables. The average number of replicates made by the simulation to evaluate a chromosome is also given in the seven tables. In this experiment, the absolute percentage of error was computed as follows:

$$\alpha = 100(|X_i - X_{SIM}|/X_{SIM})$$

Where:

α: The percentage deviation of the solution obtained by the CGA_WSPT from the simulation solution.

X_i: The value of the performance measure i obtained by the CGA_WSPT (i.e., the makespan, the number of jobs tardy, the average flow time, or the total tardiness).

X_{SIM}: The performance measure value obtained by the simulation.

This experiment was designed to have two-factor factorial design. The first factor was the model type and the second factor was the problem number. There were two levels for the first factor and seven levels for the second factor. A two-way ANOVA procedure was conducted on the results obtained for each of the four performance measures: the makespan; the number of jobs tardy; the average flow time, and the total tardiness. With respect to the makespan, the results showed no significant difference exists between the CGA_WSPT and the CGA_SIM. This implies that the CGA_WSPT and the CGA_SIM are not different. In terms of the last three performance measures, the results showed the following significant levels: 0.00048; 0.0029, and 0.00023. This implies that the CGA_WSPT and the CGA_SIM are significantly different with respect to these three performance measures.

From the results, it can be seen that the percentages of errors for both the makespan and the average flow time ranged between 0.777% and 5.245%, and 0.768% and 3.21% respectively. This implies that the probability Gantt charting evaluation method was a good estimator for the makespan and the average flow time. This observation can be confirmed by observing that all averages estimated by the probability Gantt charting fall within the 90% confidence interval. However, the probability Gantt charting was not a good estimator for the other two performance measures.

When the CPU times needed by both the CGA_WSPT and CGA_SIM were compared, the CGA_WSPT reduced the CPU time by approximately 554.9%.

Table 37. Experiment V: Summary of results obtained for problem FT6.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Number of replicates
CGA_WSPT	Average	18.600	61.802	59.319	4.200	52.176	14.918	
	Std	16.456	0.761	1.041	0.447	0.346	0.373	
	Maximum	45.000	62.510	61.181	5.000	52.331	15.085	
	Minimum	3.000	60.640	58.854	4.000	51.558	14.250	
Percentage of error			Average	3.299	40.000	3.210	25.976	
			Std	1.678	14.907	1.691	4.791	
			Maximum	4.093	66.667	4.291	31.003	
			Minimum	0.301	33.333	0.430	17.998	
CGA_SIM	Average	17.800	1606.998	61.343	3.000	50.564	20.202	67.711
	Std	20.092	189.080	0.051	0.000	0.863	1.009	24.616
	Maximum	52.000	1939.150	61.366	3.000	52.107	20.653	239.000
	Minimum	1.000	1493.150	61.251	3.000	50.178	18.396	11.000
90% Confidence interval			Lower	55.766	2.727	45.967	18.365	
			Upper	66.920	3.273	55.161	22.038	

Table 38. Experiment V: Summary of results obtained for problem FT10.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Number of replicates
CGA_WSPT	Average	136.600	454.090	1057.736	6.000	818.483	495.225	
	Std	4.775	11.033	19.602	0.000	7.250	37.746	
	Maximum	143.000	469.120	1079.332	6.000	827.185	533.409	
	Minimum	131.000	445.670	1036.885	6.000	812.432	452.772	
Percentage of error			Average	3.313	0.000	0.872	21.420	
			Std	1.873	0.000	0.746	6.808	
			Maximum	6.261	0.000	1.746	31.380	
			Minimum	1.487	0.000	0.006	14.730	
CGA_SIM	Average	26.800	3274.452	1087.658	6.000	825.072	631.139	61.125
	Std	27.923	87.465	20.609	0.000	2.888	26.712	21.053
	Maximum	74.000	3404.400	1106.850	6.000	829.347	659.827	239.000
	Minimum	1.000	3172.660	1063.514	6.000	821.901	593.052	11.000
90% Confidence interval			Lower	988.780	5.455	750.065	573.763	
			Upper	1186.536	6.545	900.079	688.515	

Table 39. Experiment V: Summary of results obtained for problem FT20.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Number of replicates
CGA_WSPT	Average	108.200	1274.786	1248.988	7.600	768.406	454.982	
	Std	56.060	47.744	11.838	1.140	7.861	38.425	
	Maximum	141.000	1322.930	1259.168	9.000	775.512	518.120	
	Minimum	9.000	1196.830	1232.552	6.000	758.724	426.703	
Percentage of error			Average	0.777	16.667	0.902	15.897	
			Std	0.668	15.235	0.708	9.506	
			Maximum	1.513	40.000	1.771	23.561	
			Minimum	0.130	0.000	0.166	0.176	
CGA_SIM	Average	30.600	19119.236	1243.105	9.200	771.683	542.307	42.008
	Std	62.843	1612.549	4.488	0.837	2.455	16.109	12.617
	Maximum	143.000	20558.940	1250.980	10.000	774.001	558.224	82.000
	Minimum	1.000	16459.800	1240.339	8.000	767.601	519.031	11.000
90% Confidence interval			Lower	1130.096	8.364	701.530	493.006	
			Upper	1356.115	10.036	841.836	591.608	

Table 40. Experiment V: Summary of results obtained for problem LA21.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Number of replicates
CGA_WSPT	Average	185.800	1644.372	1216.819	6.800	952.550	1061.577	
	Std	5.586	56.025	23.462	1.924	22.982	207.853	
	Maximum	192.000	1728.840	1246.686	9.000	991.736	1363.652	
	Minimum	178.000	1574.880	1183.361	4.000	933.082	888.119	
Percentage of error			Average	3.300	21.944	2.604	19.355	
			Std	2.572	18.488	1.064	12.153	
			Maximum	7.156	50.000	3.910	30.049	
			Minimum	0.820	0.000	1.017	0.157	
CGA_SIM	Average	34.000	7044.898	1254.801	8.200	969.015	1231.324	12.613
	Std	69.907	260.971	35.964	0.447	6.087	61.549	3.129
	Maximum	159.000	7356.060	1319.063	9.000	976.830	1336.242	37.000
	Minimum	1.000	6752.100	1236.545	8.000	960.080	1184.059	11.000
90% Confidence interval			Lower	1140.729	7.455	880.923	1119.385	
			Upper	1368.874	8.945	1057.107	1343.262	

Table 41. Experiment V: Summary of results obtained for problem LA25.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Number of replicates
CGA_WSPT	Average	177.400	1578.162	1224.555	4.000	876.795	652.753	
	Std	6.189	50.997	93.271	1.581	3.024	75.331	
	Maximum	184.000	1636.720	1314.645	6.000	880.297	725.607	
	Minimum	169.000	1518.090	1126.100	2.000	873.222	537.390	
Percentage of error			Average	5.245	38.571	2.583	21.308	
			Std	2.632	20.231	1.110	14.178	
			Maximum	8.328	66.667	3.632	39.887	
			Minimum	2.483	14.286	1.173	4.995	
CGA_SIM	Average	57.800	8261.398	1216.762	6.400	900.109	838.325	15.261
	Std	70.262	419.902	41.099	0.548	7.504	67.117	5.733
	Maximum	173.000	8801.590	1282.798	7.000	906.983	915.192	46.000
	Minimum	7.000	7818.970	1171.990	6.000	890.747	763.758	11.000
90% Confidence interval			Lower	1106.147	5.818	818.281	762.114	
			Upper	1327.376	6.982	981.937	914.536	

Table 42. Experiment V: Summary of results obtained for problem LA38.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Number of replicates
CGA_WSPT	Average	255.600	3121.756	1441.245	9.800	1162.299	1618.590	
	Std	6.986	107.016	44.144	1.304	14.029	125.180	
	Maximum	261.000	3244.340	1498.820	12.000	1177.306	1766.202	
	Minimum	247.000	2974.930	1381.867	9.000	1145.100	1496.685	
Percentage of error			Average	3.221	17.273	1.054	9.870	
			Std	3.175	7.971	0.953	9.128	
			Maximum	8.608	25.000	2.401	22.759	
			Minimum	0.406	9.091	0.065	1.836	
CGA_SIM	Average	32.800	15528.898	1410.554	11.400	1154.950	1474.151	13.968
	Std	60.330	241.702	18.686	0.548	4.075	41.464	3.972
	Maximum	140.000	15870.940	1429.715	12.000	1158.757	1520.735	40.000
	Minimum	1.000	15193.050	1380.032	11.000	1149.703	1417.470	11.000
90% Confidence interval			Lower	1282.322	10.364	1049.955	1340.137	
			Upper	1538.787	12.436	1259.946	1608.164	

Table 43. Experiment V: Summary of results obtained for problem LA40.

Approach	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Number of replicates
CGA_WSPT	Average	249.200	3054.176	1401.987	10.800	1162.790	1191.720	
	Std	11.692	55.406	30.732	1.304	9.717	118.600	
	Maximum	263.000	3109.280	1444.590	12.000	1176.202	1367.004	
	Minimum	231.000	2972.570	1360.974	9.000	1149.836	1081.458	
Percentage of error			Average	2.069	12.788	0.768	12.118	
			Std	1.558	8.267	0.592	5.335	
			Maximum	4.174	20.000	1.590	18.903	
			Minimum	0.464	0.000	0.144	5.488	
CGA_SIM	Average	74.800	18436.866	1414.053	11.000	1167.786	1292.927	15.586
	Std	98.014	1842.928	14.217	0.707	2.895	40.349	4.935
	Maximum	237.000	20720.260	1433.487	12.000	1171.639	1333.530	41.000
	Minimum	1.000	16575.970	1398.818	10.000	1163.909	1227.459	11.000
90% Confidence interval			Lower	1285.503	10.000	1061.624	1175.388	
			Upper	1542.603	12.000	1273.949	1410.465	

Experiment VI: The Effect of Lot Sizing and Alternative Process Plans

In experiment VI, the effect of lot sizing and alternative process plans on the performance of the CGA_WSPT approach was investigated. In this experiment four problems were solved: FT06; FT10; FT20, and LA21. For problem FT06, the products' order sizes were uniformly distributed between 50 and 200, while for the other three problems the order sizes were uniformly distributed between 3 and 12. For each of the four problems, a normal distribution was associated with the processing times of each operation. This implies that the process times in the original problem were used as the mean values for the normal distribution. The standard deviation for each of these process times was uniformly distributed between $0.05P_{ij}$ and $0.25P_{ij}$, where P_{ij} is the processing time of job i for operation j. For each of the four problems, the set-up time for each operation was normally distributed with the parameters given in Table 44:

Table 44. Set-up time parameters.

Problem name	Mean of the set-up time	Standard deviation of the Set-up time
FT06	μ was uniformly distributed between 1 & 10	0.1μ
FT10	μ was uniformly distributed between 2 & 99	0.1μ
FT20	μ was uniformly distributed between 2 & 99	0.1μ
LA21	μ was uniformly distributed between 7 & 99	0.1μ

As mentioned earlier in this chapter, the due dates for the problems solved in this experiment were computed using the total work content (TWK) rule. In this study, the TWK was computed as follows: TWK=kP, where k is the due date factor (k=1.5) and P is the total work required. The P was computed as follows: $P=\Sigma\, Q_iP_{ij} + \Sigma S_{ij}$, where S_{ij} is the set-up time of product i on machine j, P_{ij} is the process time of product i on machine j, and Q_i is the lot size of product i.

The same three Pr values that were used in experiment V were used in this experiment to generate both process times and set-up times. Two alternative process plans and two lot sizing methods were associated with each product in the four problems. The first process plan associated with each product was the original process plan given in the problem under consideration. The second set of products' process plans was formed by reducing the load on the first three bottleneck machines. In the first lot sizing method, the lot size equals the order size, while in the second method the lot size was less than the order size. In the second lot sizing method, the lot sizes were computed according to a lot sizing policy that was proposed by Sawaqed (1987). Sawaqed's policy was to divided the order size to two or three lot sizes. In this study, a lot size was computed by first dividing the order size by three. Then, if the resultant lot size was not an integer number, then the order size was divided by two. Finally, if none of the division procedures produced an integer lot size, the lot size was kept equal to the order size.

From the proceeding paragraph, four cases were constructed as follows:

- Case I: Process plan number one and the lot size equal the order size (PP1 and $O = Q$).
- Case II: Process plan number one and the lot size less than the order size (PP1 & $O < Q$).
- Case III: Process plan number two and the lot size equal the order size (PP2 and $Q = O$).
- Case IV: Process plan number two and the lot size less than the order size (PP2 and $Q < O$).

This means that there were two cases for alternative process plans, two cases for lot sizes, and four problems with five replicates, a total of 80 problems. Using the CGA_WSPT model, the 80 problems were solved.

The results obtained for each problem and for each case are reported in Tables G.1 through G.14 in Appendix G. These results were summarized and are given in Tables 45 through 48. These tables have the same design described in the "Experiment I" section except for the first column. The first column gives the case number instead of the combination number. In this experiment the percentage of error was computed as follows:

$$\alpha_i = 100((TT_i - TT_{Best})/TT_{Best})$$

Where:

α_i: The percentage deviation of the solution obtained for case i from the best solution.

TT_i: The total tardiness obtained for case i (i.e., case I, case II, case III, or case IV).

TT_{Best}: The best obtained total tardiness.

This experiment was designed to have two-factor factorial design. The first factor was the number of alternative process plans and the second factor was the number of methods used to compute the lot size. There were two levels for both factors. A two-way ANOVA procedure was conducted on the results obtained for each of the four performance measures: the makespan; the number of jobs tardy; the average flow time, and the total tardiness. With respect to the makespan, the number of jobs tardy, and the total tardiness, the results showed no significant difference exists between the four cases. In terms of the average flow time, the results showed a significant level of 0.04122 for the second factor. This implies that the two lot sizing methods are significantly different with respect to the average flow time.

From the results, it can be seen that the percentages of errors for case IV were the smallest in all four problems except for in one problem. Also, the percentages of errors for case II were the second smallest in all four problems except for in one problem. When the order size was divided into several lot sizes, the makespan was reduced by approximately 92.31%, the number of jobs tardy was reduced by approximately 564.93%, the average flow time was reduced by approximately 855.78%, and the total tardiness was reduced by approximately 18254.2%.

To summarize, the results in this experiment showed that the potential for improving production criteria is much greater by adjusting lot size plans than by using alternative process plans. Also, this result showed that the choice of alternative process plan must include other criteria besides reducing maximum utilization.

Regarding the CPU times needed, when the order size was divided into several lot sizes, the CPU time was increased by approximately 683.4%.

Table 45. Experiment VI: Summary of results obtained for problem FT6.

Case number	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
I	Average	75.600	59.166	4687.313	2.000	3426.271	1041.690	104208.980
PP1 and O = Q	Std	5.595	3.898	18.813	0.000	18.813	37.626	3741.229
	Maximum	80.000	64.320	4720.967	2.000	3459.925	1108.997	110899.700
	Minimum	69.000	53.830	4678.900	2.000	3417.858	1024.863	102486.300
II	Average	110.000	263.470	4925.309	0.000	2896.768	0.000	0.00
PP1 and O < Q	Std	0.000	5.742	167.488	0.000	28.737	0.000	0.00
	Maximum	110.000	269.410	5000.212	0.000	2942.759	0.000	0.00
	Minimum	110.000	254.640	4625.698	0.000	2871.666	0.000	0.00
III	Average	78.400	57.290	4870.012	3.000	3435.844	241.243	24124.300
PP2 and O = Q	Std	2.191	2.864	0.000	0.000	0.000	0.000	0.000
	Maximum	80.000	61.020	4870.012	3.000	3435.844	241.243	24124.300
	Minimum	76.000	54.430	4870.012	3.000	3435.844	241.243	24124.300
IV	Average	110.000	274.686	4352.367	0.000	2766.900	0.000	0.00
PP2 and O < Q	Std	0.000	3.420	206.273	0.000	144.921	0.000	0.00
	Maximum	110.000	279.080	4647.518	0.000	2917.759	0.000	0.00
	Minimum	110.000	271.670	4148.081	0.000	2525.526	0.000	0.00

Table 46. Experiment VI: Summary of results obtained for problem FT10.

Case number	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
I	Average	140.200	473.538	10357.151	3.000	5272.446	903.299	693.386
PP1 and O = Q	Std	6.340	46.648	63.705	1.000	43.712	354.242	925.934
	Maximum	144.000	548.490	10392.997	4.333	5338.428	1514.745	2266.487
	Minimum	129.000	437.650	10243.996	2.000	5232.494	657.410	59.143
II	Average	230.000	2988.780	7573.123	1.133	4712.381	661.923	588.31
PP1 and O < Q	Std	2.345	37.142	134.055	0.298	53.807	82.262	895.85
	Maximum	232.000	3032.050	7769.071	1.667	4770.877	742.555	2130.83
	Minimum	227.000	2949.280	7456.205	1.000	4628.617	542.068	31.22
III	Average	86.200	463.746	10077.931	3.333	5300.858	1373.808	1571.386
PP2 and O = Q	Std	63.684	20.380	248.525	0.781	51.415	448.527	2351.485
	Maximum	140.000	495.760	10333.062	4.333	5368.325	1758.459	5626.940
	Minimum	2.000	442.640	9672.101	2.667	5241.074	641.122	55.200
IV	Average	229.600	3106.336	7983.432	1.000	4912.624	276.055	0.00
PP2 and O < Q	Std	1.140	120.555	69.195	0.235	78.325	204.665	0.00
	Maximum	231.000	3278.610	8055.037	1.333	5004.182	497.344	0.00
	Minimum	228.000	2989.650	7895.034	0.667	4801.138	27.780	0.00

Table 47. Experiment VI: Summary of results obtained for problem FT20.

Case number	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
I	Average	33.600	1307.622	12151.596	17.200	6397.647	58112.103	2.465
PP1 and O = Q	Std	62.244	102.346	138.949	0.447	59.775	1220.126	2.367
	Maximum	144.000	1425.430	12251.797	18.000	6473.464	59635.769	5.300
	Minimum	1.000	1179.850	11918.124	17.000	6312.958	56343.504	0.000
II	Average	41.200	9372.584	14175.241	25.133	5744.579	84224.419	48.54
PP1 and O < Q	Std	73.690	155.487	292.566	1.016	37.482	1528.299	4.57
	Maximum	171.000	9604.880	14516.519	26.333	5799.659	86133.535	54.89
	Minimum	1.000	9237.750	13961.862	24.000	5694.326	81870.470	44.12
III	Average	84.400	1076.474	12798.005	18.600	6410.588	57910.827	2.073
PP2 and O = Q	Std	71.231	27.074	517.900	0.925	124.906	2424.734	3.128
	Maximum	143.000	1111.750	13683.754	19.667	6544.912	60467.992	7.042
	Minimum	3.000	1044.570	12333.526	17.333	6288.605	55488.338	0.000
IV	Average	51.800	7842.678	14797.549	25.867	5855.952	89392.008	57.64
PP2 and O < Q	Std	87.044	224.881	40.980	1.070	61.500	1844.720	4.76
	Maximum	203.000	8126.720	14822.514	27.000	5911.186	90764.624	63.22
	Minimum	1.000	7586.360	14728.141	24.333	5759.240	86169.473	52.94

Table 48. Experiment VI: Summary of results obtained for problem LA21.

Case number	Statistics	No. of alternatives	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness	Percentage of error
I PP1 and O = Q	Average	173.800	1539.868	11374.222	12.067	7295.548	13911.976	104.555
	Std	33.192	71.405	351.060	1.038	79.194	633.044	30.654
	Maximum	194.000	1656.050	11782.555	13.000	7404.054	15040.765	156.984
	Minimum	115.000	1475.130	10972.215	10.333	7189.398	13557.195	78.881
II PP1 and O < Q	Average	254.600	12961.880	10437.941	10.067	6350.741	8060.584	18.04
	Std	72.765	497.170	101.731	0.983	73.331	397.842	12.34
	Maximum	324.000	13447.410	10583.701	11.667	6438.975	8628.599	34.26
	Minimum	158.000	12297.160	10315.424	9.333	6271.694	7552.052	4.25
III PP2 and O = Q	Average	189.600	1530.566	11743.799	12.067	7194.929	11701.466	71.905
	Std	5.683	44.547	321.191	0.641	65.820	947.054	26.124
	Maximum	194.000	1594.820	12164.899	13.000	7255.654	12609.367	110.177
	Minimum	180.000	1474.370	11354.351	11.333	7100.836	10391.181	37.107
IV PP2 and O < Q	Average	247.600	14174.654	10791.680	9.067	6233.415	6878.220	0.00
	Std	40.352	484.722	208.710	1.479	111.926	659.249	0.00
	Maximum	282.000	14538.830	11055.534	10.667	6344.725	7578.889	0.00
	Minimum	182.000	13361.940	10494.373	6.667	6063.088	5852.796	0.00

Experiment VII: Analysis of Advantage Gained by Explicitly Incorporating the Probability Distribution Function of the Processing Times in the Genetic Algorithm

The purpose of this experiment was to investigate the potential gain from incorporating the probability distribution function of the processing times in the genetic algorithm.

In this experiment, seven of the nine problems discussed earlier were solved. These seven problems were FT06, FT10, FT20, LA21, LA25, LA38, and LA40. For each of the seven problems, a normal distribution was associated with the processing times of each operation. This implies that the process times in the original problem were used as the mean values for the normal distribution. The standard deviation for each of these process times was uniformly distributed between $0.05P_{ij}$ and $0.25P_{ij}$, where P_{ij} is the processing time of job i for operation j. The same due dates that were used in experiment IV were used in this experiment.

To perform the required analysis for the seven problems, the final best solutions obtained by the CGA_TT, the CGA_WSPT, and the CGA_SIM were simulated using a simulation model that was developed for this experiment. In the simulation model developed, each chromosome was simulated several times to reach a certain confidence level for the results obtained. The number of replications for simulating each chromosome was determined using the same sequential procedure that was mentioned in the previous chapter. The simulation model was coded in FORTRAN 90 for a Gateway 2000 computer using the Microsoft FORTRAN PowerStation™, professional edition, version 4.0. For a full listing of the computer code, the reader can refer to Al-Harkan and Foote (1997).

Thus there were seven problems, five replicates for each problem, a chromosome to be simulated obtained by the CGA_TT, a chromosome to be simulated obtained by the CGA_WSPT, a chromosome to be simulated obtained by the CGA_SIM, a total of 105 problems. Using the simulation model developed, the 105 problems were solved.

The results obtained for each problem by simulating the final best solution of the three approaches are reported in Tables H.1 through H.21 in Appendix H. These results were summarized and are given in Tables 49 through 55. These seven tables were designed to use the same statistics presented in the earlier tables. However, the absolute percentage of error was computed for the four performance measures used in this study. Hence, the absolute percentage errors for these four performance measures are given in the seven tables. Also, 90% confidence intervals on all the four performance measures were constructed and are given in these tables. The four statistics were computed for the number of replicates made by the simulation to evaluate a chromosome, and are also given in the seven tables. In this experiment, the absolute percentage of error was computed as follows:

$$\alpha = 100(|X_i - X_{SIM}|/X_{SIM})$$

Where:

α: The percentage deviation of the solution obtained by simulating the final best solution obtained by the CGA_WSPT or the CGA_TT from the CGA_SIM.

X_i: The value of the performance measure i obtained by simulating the final best solution obtained by the CGA_WSPT or the CGA_TT (i.e., the makespan, the number of jobs tardy, the average flow time, or the total tardiness).

X_{SIM}: The performance measure value obtained by simulating the final best solution obtained by the CGA_SIM.

From the results, it can be seen that there is a tremendous gain in modifying the genetic algorithm to incorporate the normal probability distribution function of the processing times.

When the results of the CGA_WSPT and the CGA_SIM were compared, the CGA_SIM reduced the actual expected total tardiness by approximately 30.3%, the CGA_SIM reduced the actual worst case total tardiness by approximately 56%, and the CGA_SIM reduced the risk by approximately 18%.

When the CGA_TT and the CGA_SIM were compared, the CGA_SIM reduced the actual expected total tardiness by approximately 28.7%, the CGA_SIM reduced the actual worst case total tardiness by approximately 52%, and the CGA_SIM reduced the risk by approximately 16.4%.

From these results, it can be stated the CGA_SIM performed better than both the CGA_TT and the CGA_WSPT.

Table 49. Experiment VII: Summary of results obtained by simulating the final best solution obtained by the three approaches for problem FT6.

Approach	Statistics	Number of replicates	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness
CGA_WSPT Probability Gantt charting evaluation	Average	124.800	0.286	63.691	3.800	53.732	27.320
	Std	45.801	0.093	2.108	0.447	0.766	3.290
	Maximum	182.000	0.390	65.780	4.000	54.351	30.806
	Minimum	75.000	0.160	60.831	3.000	52.560	23.253
Percentage of error			Average	4.101	26.667	6.294	35.045
			Std	2.943	14.907	2.618	12.487
			Maximum	7.193	33.333	8.316	49.160
			Minimum	0.686	0.000	2.424	20.118
CGA_TT Deterministic Gantt charting evaluation	Average	125.400	0.298	62.510	4.400	55.694	35.222
	Std	32.323	0.086	1.396	0.548	1.465	8.146
	Maximum	158.000	0.390	64.768	5.000	57.991	48.005
	Minimum	72.000	0.160	61.004	4.000	54.057	26.248
Percentage of error			Average	2.138	46.667	10.171	74.833
			Std	2.002	18.257	3.432	41.171
			Maximum	5.544	66.667	15.571	132.436
			Minimum	0.590	33.333	6.945	27.090
CGA_SIM Simulation evaluation	Average	64.600	0.132	61.343	3.000	50.564	20.202
	Std	14.758	0.049	0.051	0.000	0.863	1.009
	Maximum	91.000	0.220	61.366	3.000	52.107	20.653
	Minimum	58.000	0.110	61.251	3.000	50.178	18.396
90% Confidence interval			Lower	55.766	2.727	45.967	18.365
			Upper	66.920	3.273	55.161	22.038

Table 50. Experiment VII: Summary of results obtained by simulating the final best solution obtained by the three approaches for problem FT10.

Approach	Statistics	Number of replicates	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness
CGA_WSPT Probability Gantt charting evaluation	Average	33.200	0.262	1098.985	7.000	849.983	799.299
	Std	10.208	0.073	16.310	0.000	5.038	36.111
	Maximum	46.000	0.330	1119.780	7.000	855.383	835.797
	Minimum	21.000	0.160	1076.879	7.000	842.864	749.833
Percentage of error			Average	1.515	16.667	3.019	26.838
			Std	1.176	0.000	0.408	8.265
			Maximum	3.100	16.667	3.600	40.931
			Minimum	0.188	16.667	2.551	19.038
CGA_TT Deterministic Gantt charting evaluation	Average	44.200	0.328	1091.643	7.200	857.181	800.007
	Std	18.431	0.165	25.888	0.447	7.949	70.466
	Maximum	68.000	0.500	1114.563	8.000	865.518	911.314
	Minimum	26.000	0.160	1053.114	7.000	845.635	727.646
Percentage of error			Average	2.301	20.000	3.891	27.194
			Std	1.646	7.454	0.818	15.712
			Maximum	4.355	33.333	4.828	53.665
			Minimum	0.815	16.667	2.757	11.497
CGA_SIM Simulation evaluation	Average	11.000	0.076	1087.658	6.000	825.072	631.139
	Std	0.000	0.031	20.609	0.000	2.888	26.712
	Maximum	11.000	0.110	1106.850	6.000	829.347	659.827
	Minimum	11.000	0.050	1063.514	6.000	821.901	593.052
90% Confidence interval			Lower	988.780	5.455	750.065	573.763
			Upper	1186.536	6.545	900.079	688.515

Table 51. Experiment VII: Summary of results obtained by simulating the final best solution obtained by the three approaches for problem FT20.

Approach	Statistics	Number of replicates	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness
CGA_WSPT Probability Gantt charting evaluation	Average	50.800	1.044	1276.787	10.200	786.842	859.353
	Std	10.183	0.210	20.202	1.095	12.750	165.146
	Maximum	63.000	1.320	1307.754	11.000	799.228	1072.514
	Minimum	41.000	0.830	1257.596	9.000	773.072	678.294
Percentage of error			Average	2.710	15.944	1.966	59.226
			Std	1.648	14.396	1.776	35.049
			Maximum	5.430	37.500	4.120	106.638
			Minimum	1.215	0.000	0.183	21.776
CGA_TT Deterministic Gantt charting evaluation	Average	54.800	1.132	1263.585	9.600	780.482	734.830
	Std	6.419	0.108	20.865	0.894	7.226	94.742
	Maximum	62.000	1.210	1296.153	11.000	793.170	890.768
	Minimum	48.000	0.990	1245.166	9.000	776.307	650.280
Percentage of error			Average	1.837	9.500	1.143	35.929
			Std	1.677	16.240	1.238	21.327
			Maximum	4.495	37.500	3.331	71.621
			Minimum	0.330	0.000	0.336	20.768
CGA_SIM Simulation evaluation	Average	42.800	0.868	1243.105	9.200	771.683	542.307
	Std	8.526	0.149	4.488	0.837	2.455	16.109
	Maximum	49.000	0.980	1250.980	10.000	774.001	558.224
	Minimum	28.000	0.610	1240.339	8.000	767.601	519.031
90% Confidence interval			Lower	1130.096	8.364	701.530	493.006
			Upper	1356.115	10.036	841.836	591.608

Table 52. Experiment VII: Summary of results obtained by simulating the final best solution obtained by the three approaches for problem LA21.

Approach	Statistics	Number of replicates	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness
CGA_WSPT Probability Gantt charting evaluation	Average	13.200	0.264	1269.729	9.200	995.896	1564.320
	Std	3.194	0.060	29.029	0.447	5.550	96.679
	Maximum	18.000	0.330	1308.552	10.000	1002.529	1697.678
	Minimum	11.000	0.220	1236.718	9.000	991.294	1479.511
Percentage of error			Average	3.346	12.500	2.775	27.418
			Std	2.373	8.839	0.524	11.775
			Maximum	5.823	25.000	3.383	42.442
			Minimum	0.087	0.000	2.145	10.722
CGA_TT Deterministic Gantt charting evaluation	Average	14.000	0.274	1258.205	9.200	987.633	1350.801
	Std	6.164	0.121	21.823	1.095	22.467	203.631
	Maximum	25.000	0.490	1279.249	11.000	1013.927	1511.519
	Minimum	11.000	0.220	1226.564	8.000	965.896	1007.412
Percentage of error			Average	2.489	12.500	1.915	15.945
			Std	1.613	15.309	1.805	8.510
			Maximum	4.584	37.500	3.973	24.654
			Minimum	0.807	0.000	0.403	2.449
CGA_SIM Simulation evaluation	Average	11.000	0.220	1254.801	8.200	969.015	1231.324
	Std	0.000	0.000	35.964	0.447	6.087	61.549
	Maximum	11.000	0.220	1319.063	9.000	976.830	1336.242
	Minimum	11.000	0.220	1236.545	8.000	960.080	1184.059
90% Confidence interval			Lower	1140.729	7.455	880.923	1119.385
			Upper	1368.874	8.945	1057.107	1343.262

Table 53. Experiment VII: Summary of results obtained by simulating the final best solution obtained by the three approaches for problem LA25.

Approach	Statistics	Number of replicates	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness
CGA_WSPT Probability Gantt charting evaluation	Average	23.800	0.440	1250.859	6.400	907.021	979.590
	Std	5.630	0.085	72.738	1.140	13.754	102.844
	Maximum	30.000	0.550	1330.665	8.000	921.266	1132.230
	Minimum	17.000	0.330	1172.620	5.000	884.953	852.010
Percentage of error			Average	4.387	6.190	1.610	17.640
			Std	2.829	8.518	1.222	17.632
			Maximum	7.232	16.667	3.426	45.239
			Minimum	0.201	0.000	0.229	1.534
CGA_TT Deterministic Gantt charting evaluation	Average	27.800	0.538	1149.020	8.200	918.963	977.310
	Std	5.495	0.093	17.091	1.095	17.286	79.116
	Maximum	36.000	0.660	1179.467	9.000	937.180	1089.149
	Minimum	22.000	0.440	1139.830	7.000	894.271	868.435
Percentage of error			Average	5.714	29.048	2.095	16.665
			Std	3.756	21.652	1.775	5.169
			Maximum	11.077	50.000	3.966	22.771
			Minimum	0.638	0.000	0.103	9.432
CGA_SIM Simulation evaluation	Average	11.000	0.208	1216.762	6.400	900.109	838.325
	Std	0.000	0.027	41.099	0.548	7.504	67.117
	Maximum	11.000	0.220	1282.798	7.000	906.983	915.192
	Minimum	11.000	0.160	1171.990	6.000	890.747	763.758
90% Confidence interval			Lower	1106.147	5.818	818.281	762.114
			Upper	1327.376	6.982	981.937	914.536

Table 54. Experiment VII: Summary of results obtained by simulating the final best solution obtained by the three approaches for problem LA38.

Approach	Statistics	Number of replicates	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness
CGA_WSPT Probability Gantt charting evaluation	Average	13.800	0.352	1446.181	11.000	1178.740	1898.406
	Std	3.347	0.065	30.857	0.707	15.399	194.124
	Maximum	19.000	0.440	1499.826	12.000	1198.998	2161.140
	Minimum	11.000	0.270	1424.848	10.000	1157.978	1639.654
Percentage of error			Average	2.048	6.818	1.277	19.122
			Std	2.215	3.826	1.416	19.303
			Maximum	5.259	9.091	3.707	51.590
			Minimum	0.195	0.000	0.244	1.386
CGA_TT Deterministic Gantt charting evaluation	Average	15.800	0.416	1453.685	11.800	1178.978	1834.070
	Std	5.891	0.139	47.176	0.447	8.151	69.997
	Maximum	26.000	0.660	1534.480	12.000	1189.127	1905.281
	Minimum	11.000	0.330	1419.917	11.000	1168.841	1735.621
Percentage of error			Average	3.133	0.000	1.259	14.982
			Std	3.485	0.000	1.130	13.210
			Maximum	8.765	0.000	2.853	33.643
			Minimum	0.063	0.000	0.202	2.345
CGA_SIM Simulation evaluation	Average	16.000	0.418	1417.334	11.800	1166.521	1609.575
	Std	6.928	0.195	24.518	0.447	12.295	170.283
	Maximum	25.000	0.660	1438.303	12.000	1181.363	1838.386
	Minimum	11.000	0.270	1377.840	11.000	1151.618	1425.649
90% Confidence interval			Lower	1288.486	10.727	1060.474	1463.250
			Upper	1546.183	12.873	1272.569	1755.900

Table 55. Experiment VII: Summary of results obtained by simulating the final best solution obtained by the three approaches for problem LA40.

Approach	Statistics	Number of replicates	CPU time (Sec.)	Makespan (Cmax)	Number Tardy	Average flow time	Total Tardiness
CGA_WSPT	Average	18.400	0.484	1445.592	11.800	1196.738	1677.761
Probability	Std	2.302	0.046	38.805	1.095	18.970	238.560
Gantt charting	Maximum	22.000	0.550	1487.657	13.000	1215.632	1900.967
evaluation	Minimum	16.000	0.440	1399.145	10.000	1171.435	1357.204
Percentage of error			Average	2.684	9.455	2.317	26.792
			Std	2.203	9.569	1.628	20.122
			Maximum	5.851	20.000	4.235	54.870
			Minimum	0.129	0.000	0.306	2.531
CGA_TT	Average	25.400	0.682	1401.923	12.000	1188.951	1540.398
Deterministic	Std	5.079	0.128	8.182	0.000	6.619	61.356
Gantt charting	Maximum	32.000	0.820	1409.616	12.000	1196.604	1599.825
evaluation	Minimum	19.000	0.500	1391.050	12.000	1179.505	1440.542
Percentage of error			Average	1.577	7.636	1.529	15.317
			Std	0.565	8.272	0.915	9.758
			Maximum	1.917	20.000	2.809	25.320
			Minimum	0.574	0.000	0.381	3.454
CGA_SIM	Average	13.800	0.374	1421.166	11.200	1171.091	1341.504
Simulation	Std	3.899	0.118	13.578	0.837	7.168	95.437
evaluation	Maximum	19.000	0.500	1434.649	12.000	1181.550	1475.860
	Minimum	11.000	0.270	1401.565	10.000	1163.909	1227.459
90% Confidence interval			Lower	1291.969	10.182	1064.629	1219.549
			Upper	1550.363	12.218	1277.554	1463.459

CHAPTER V

SUMMARY, CONCLUSIONS, CONTRIBUTIONS, AND RECOMMENDATIONS

Introduction

The first purpose of this chapter is to summarize the goal of this study and to discuss the methodology developed in this research. Next, it gives the findings and the contributions of this research. Then, it offers a list of recommendations for future research.

Summary

Over the last four decades, the control of the job shop problem has been studied using several solution methods, including enumerative methods, heuristic methods, mathematical models, heuristic search techniques, simulation models, and queueing network models. This research study is an extension of the previous research that was concerned with applying one of the heuristic search techniques, the genetic algorithm (GA), to the job shop problem.

The purpose of this study is to solve a dynamic stochastic job shop environment by an integrated model that consists of a constrained genetic algorithm which merges dispatching rules, heuristics, and the available sequencing and scheduling theory with the genetic algorithm to enhance its search procedures.

From the research gaps, the following questions emerged, which this study attempts to answer: 1) Does the constrained genetic algorithm perform better than the unconstrained genetic algorithm when both algorithms are extended to solve dynamic stochastic job shops? 2) What is the impact of the population size on the accuracy of the deterministic constrained genetic algorithm to minimize makespan? 3) What is the impact of nine genetic operator combinations on the performance of the deterministic constrained genetic algorithm to minimize makespan and which of the nine genetic operator combinations would be the best? 4) Is the evaluation of the chromosomes using the probability Gantt charting as effective as simulation evaluation? 5) What is the performance of the stochastic constrained genetic algorithm to minimize total tardiness when lot sizes, process plans, and machine priority lists are optimized simultaneously? 6) What is the potential gain from incorporating the probability distribution function of the processing times in the genetic algorithm?

The research program consisted of two parts. The first was the development of the elements and the parameters related to the genetic algorithm and its variations. The second was the implementation of seven major experiments intended to answer the research questions addressed.

Several elements and parameters for the genetic algorithm were designed, including population representation method, schedule building and fitness function evaluation, population size, generation of the initial population, selection methods, crossover and mutation operators, and termination criteria.

The population of chromosomes was represented using the preference-list-based representation method. Chromosomes were generated in the initial population according to both active and non-delay schedules. Also, during the evolution process, chromosomes were evaluated according to their original schedule type generator, which could be either an active schedule or a non-delay schedule.

For each chromosome, four performance measures were computed; however, one of them was minimized. These performance measures were the makespan, the total tardiness, the average flow time, and the number of jobs tardy. Three of these performance measures were used to break ties among chromosomes when the selection method was applied.

Three population sizes were used: 44+nm; 44+2nm, and 44+4nm, where n is the number of jobs and m is the number of machines. The starting initial population was seeded with forty-four heuristics. Then the rest of the population was generated according to four random heuristics.

Two selection methods were used: the first was the elitist method and the second was the binary tournament. In the binary tournament, the simulated annealing approach was used to make the decision whether to accept or reject a produced child.

Three crossover operators were used: linear order crossover (LOX); order-based crossover (OBX), and position-based crossover (PBX). The following mutation operators were used: order-based mutation (OBM); position-based mutation (PBM), and scramble sub-sequence mutation (SSM).

Three termination criteria were used sequentially. These criteria were: the maximum number of generations had been reached; the best solution had not been changed for a number of generations, and a certain time limit had been reached.

Using the elements and the parameters discussed above, nine genetic algorithms were designed and developed. These algorithms were: a deterministic constrained genetic algorithm to minimize makespan (CGA_Cmax); a deterministic constrained genetic algorithm to minimize total tardiness (CGA_TT); a stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using probability Gantt charting (CGA_WSPT); a stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using simulation (CGA_SIM); a deterministic unconstrained genetic algorithm to minimize makespan (UGA_Cmax); a deterministic unconstrained genetic algorithm to minimize total tardiness (UGA_TT); a stochastic unconstrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using probability Gantt charting (UGA_WSPT); a stochastic unconstrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using simulation (UGA_SIM), and a dynamic stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using probability Gantt charting (CGA_APP).

The CGA_Cmax and the UGA_Cmax models attempted to minimize the makespan. The CGA_TT, the UGA_TT, the CGA_WSPT, the CGA_SIM, the UGA_WSPT, the UGA_SIM, and the CGA_APP models attempted to minimize the total tardiness. In addition to attempting to minimize the total tardiness, the CGA_APP attempted to optimize simultaneously the lot sizes and the process plans for the products involved in the production plan. Specifically, the CGA_APP can handle products that each have a set of top alternative process plans and from which the lot size for each product can be optimized.

In the five constrained genetic algorithms, the genetic operators produced children that were altered not only by the operator's procedures but also by the dominance rules, while no alteration was performed in the four unconstrained genetic algorithms. By performing this alteration to the children produced we offer an additional feature, that of constraining the order of certain elements of the chromosomes according to precedence relationships established theoretically. Hence, we called our approach a constrained genetic algorithm.

The input to the CGA_Cmax, the UGA_Cmax, the CGA_TT, and the UGA_TT included the number of machines, the number of jobs, the number of operations, process plans, lot sizes, due dates, expected process times, expected set-up times, and ready times.

The input to the CGA_WSPT, the UGA_WSPT, the CGA_SIM, and the UGA_SIM consisted of the following: the number of machines; the number of jobs; the number of operations; process plans; lot sizes; due dates; distribution of process times; expected process times; standard deviation of process times; distribution of set-up times; expected set-up times; standard deviation of set-up times, and ready times. In addition to the previous input, the CGA_WSPT and the UGA_WSPT read three probability values to be used in evaluating chromosomes. Also, the CGA_SIM and the UGA_SIM read a specified confidence level and a desired relative error to be used in evaluating chromosomes.

The CGA_APP input consisted of the number of machines, the number of jobs, the number of alternative process plans, alternative process plans, the number of operations, order sizes, due dates, distribution of process times, expected process times, standard deviation of process times, distribution of set-up times, expected set-up times, standard deviation of set-up times, ready times, and three probability values to be used in evaluating chromosomes.

The CGA_Cmax, the UGA_Cmax, the CGA_TT, and the UGA_TT evaluated their chromosomes using the deterministic Gantt charting. The CGA_WSPT and the UGA_WSPT used the probability Gantt charting to evaluate their chromosomes and the CGA_SIM and the UGA_SIM evaluated their chromosomes using simulation.

The CGA_Cmax, the UGA_Cmax, the CGA_TT, and the UGA_TT ranked their chromosomes using their fitness functions (i.e., either makespan or total tardiness). The CGA_WSPT, the UGA_WSPT, the CGA_SIM, and the UGA_SIM ranked their chromosomes using the utility function approach values.

The output from the nine genetic algorithms were as follows: the preference list for each machine; the makespan; the total tardiness; the average flow time, and the number of jobs tardy. In addition to these outputs, the CGA_APP produced a preferred set of product's process plans and a preferred set of product's lot sizes.

The computer programs for the nine genetic algorithms consisted of a main program, nineteen subroutines, three functions, and the IMSL mathematical and statistical libraries. The computer programs for the nine genetic algorithms were coded in FORTRAN 90 for a GATEWAY 2000 (Pentium-90) computer using the Microsoft FORTRAN PowerStation™, professional edition, version 4.0.

As mentioned earlier, seven major experiments were performed in which each experiment was intended to answer one of the research questions addressed. Experiment I was conducted to investigate the effect of the genetic operator combinations on the performance of the deterministic constrained genetic algorithm to minimize makespan (CGA_Cmax). In experiment II, the impact of the population size on the performance of the CGA_Cmax was investigated. Experiment III compared the performance of the deterministic constrained genetic algorithm to minimize makespan (CGA_Cmax) with the deterministic unconstrained genetic algorithm to minimize makespan (UGA_Cmax). Also, the performance of the deterministic constrained genetic algorithm to minimize total tardiness (CGA_TT) and the deterministic unconstrained genetic algorithm to minimize total tardiness (UGA_TT) were evaluated in experiment IV. Experiment V investigated which of the chromosome evaluation methods was better. In experiment VI, the effect of lot sizing and alternative process plans on the performance of the stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using probability Gantt charting (CGA_WSPT) was investigated. Experiment VII investigated the potential gain from incorporating the probability distribution function of the processing times in the genetic algorithm.

Nine well-known benchmarks used in the seven experiments are known to be difficult problems. Three of these problems were designed by Fisher and Thompson (1963) and the other six were designed by Lawrence (1984). In these nine problems, the number of operations ranged between 36 and 225 operations.

For the first three experiments, jobs were given a common due date which is the optimal makespan of the problem considered. However, for experiments IV, V, and VII, the due dates were computed according to flow time estimates. In experiment VI, the due dates were computed according to the total work content (TWK) rule.

For the seven experiments, the number of generations was set to 55. Also, the population size was set to 44+4nm in experiments I and II and the population size was set to 44+nm in the other five experiments. The linear order crossover (LOX) and the order-based mutation were used as the genetic operators in the last six experiments.

For the seven experiments, the computer package STATGRAPHICS™ version 5 was used to perform the required analysis of variance procedures and Tukey's range test and ranking procedures. The significance level used to test the significance of the factors included in each experiment was 0.05. Also, the percentage errors computed and CPU time recorded were used to analyze the results obtained for the seven experiments.

Conclusions

According to the results given in Chapter IV, the following conclusions which correspond to the research questions are given:

Research question 1: Does the constrained genetic algorithm perform better than the unconstrained genetic algorithm when both algorithms are extended to solve dynamic stochastic job shops?

When comparing the average percentage errors over the nine problems obtained by the deterministic constrained genetic algorithm to minimize makespan (CGA_Cmax) and the deterministic unconstrained genetic algorithm to minimize makespan (UGA_Cmax), the CGA_Cmax improved the average percentage errors by approximately 27.44%. This means the CGA_Cmax model outperformed the UGA_Cmax model.

Also, when the average percentage errors over the nine problems obtained by the deterministic constrained genetic algorithm to minimize total tardiness (CGA_TT) and the deterministic unconstrained genetic algorithm to minimize total tardiness (UGA_TT) were compared, the CGA_TT improved the average percentage errors by approximately 248.77%. This implies that the CGA_TT model performed better than the UGA_TT model.

Research question 2: What is the impact of the population size on the accuracy of the deterministic constrained genetic algorithm to minimize makespan?

The results showed at a significance level of 0.0001 that increasing the population size did significantly improve the performance measures.

According to the percentage errors computed, when the population size was increased from 44+nm to 44+2nm, the makespan was improved by approximately 0.5%. Also, increasing the population size from 44+nm to 44+4nm improved the makespan by approximately 0.81%.

Regarding the CPU times recorded, when the population size was increased from 44+nm to 44+2nm, the CPU time was increased by approximately 71.6%. Also, when the population size was increased from 44+nm to 44+4nm, the CPU time was increased by approximately 209.7%.

With these marginal improvements in the makespan and the huge increase in the CPU times, the following conclusions are given. The constrained genetic algorithm was able to obtain good quality solutions with a smaller population size and much less computational effort. From this conclusion we can state that the constrained genetic algorithm was not significantly affected by the population size, which shows how robust the constrained genetic algorithm is.

Research question 3: What is the impact of nine genetic operator combinations on the performance of the deterministic constrained genetic algorithm to minimize makespan and which of the nine genetic operator combinations would be the best?

At a significance level of 0.02, the results showed that the genetic operator combinations significantly affect the performance measures. Hence, the performance of the constrained genetic algorithm was influenced by the genetic operator combinations. According to Tukey's range test and ranking procedures, the LO combination (the linear order crossover (LOX) and the order-based mutation (OBM)) was the best genetic operator combination for the constrained genetic algorithm. Also, comparing the average percentage errors obtained when using the LO combination with the average percentage errors obtained when using the other eight combinations, the LO combination improved the average percentage errors by approximately 10%.

Research question 4: Is the evaluation of the chromosomes using the probability Gantt charting as effective as simulation evaluation?

According to the ANOVA results and regarding the makespan objective, the results showed no significant difference exists between probability Gantt charting and simulation in terms of finding an optimal solution.

The stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using probability Gantt charting (CGA_WSPT) deviated from the true mean for both the makespan and the average flow time by 3.032% and 1.713% respectively. Also, all averages estimated for both the makespan and the average flow time when using the probability Gantt charting fall within the 90% confidence interval.

When the CPU times needed by both the stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using probability Gantt charting (CGA_WSPT) and the stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using simulation (CGA_SIM) were compared, the CGA_WSPT reduced the CPU time by approximately 554.9%.

To sum up, the evaluation of chromosomes using the probability Gantt charting was more effective than simulation when the performance measures are the makespan and the average flow time, but it was not as effective as simulation when the performance measures are the number of jobs tardy and the total tardiness.

Research question 5: What is the performance of the stochastic constrained genetic algorithm to minimize total tardiness when lot sizes, process plans, and machine priority lists are optimized simultaneously?

When the order size was divided into several lot sizes, the makespan was reduced by approximately 92.31%, the number of jobs tardy was reduced by approximately 564.93%, the average flow time was reduced by approximately 855.78%, and the total tardiness was reduced by approximately 18254.2%. Regarding the CPU times recorded, when the order size was divided into several lot sizes, the CPU time was increased by approximately 683.4%.

To sum up, the preliminary experiment showed that the potential for improving production criteria is much greater by adjusting lot size plans than by using alternative process plans. Also, this result showed that the choice of alternative process plan must include other criteria besides reducing maximum utilization.

Research question 6: What is the potential gain from incorporating the probability distribution function of the processing times in the genetic algorithm?

From the results, it can be seen that there is a tremendous gain in modifying the genetic algorithm to incorporate the normal probability distribution function of the processing times.

When the results obtained by both the stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using probability Gantt charting (CGA_WSPT) and the stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using simulation (CGA_SIM) were compared, the CGA_SIM reduced the actual expected total tardiness by approximately 30.3%, the CGA_SIM reduced the actual worst case total tardiness by approximately 56%, and the CGA_SIM reduced the risk by approximately 18%.

When the deterministic constrained genetic algorithm to minimize total tardiness (CGA_TT) and the stochastic constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using simulation (CGA_SIM) were compared, the CGA_SIM reduced the actual expected total tardiness by approximately 28.7%, the CGA_SIM reduced the actual worst case total tardiness by approximately 52%, and the CGA_SIM reduced the risk by approximately 16.4%. From these results, it can be concluded that the CGA_SIM performed better than both the CGA_TT and the CGA_WSPT.

Contributions

This research has contributed to the literature of both genetic algorithms and sequencing and scheduling. These contributions can be summarized as follows:

1. This study developed and implemented a unique and a robust genetic algorithm to solve job shop problems.
2. This study showed that the performance of the genetic algorithm was enhanced when problem specific theoretical results were incorporated. This enhancement incorporated in the genetic algorithm was performed not only when the chromosomes were generated but also during the evolution process. This implies that this enhancement was both predictive and reactive.
3. This study proposed and tested a fast evaluation method for chromosomes using probability Gantt charting which accounts for random variation.
4. This study showed that the management of bottleneck machines can be incorporated in the genetic algorithm. This incorporation was accomplished by first using the preference-list-based representation method, which works with sub-chromosomes, which means it treats machines individually. Second, the first two sub-chromosomes were selected as the top two bottlenecks when implementing the genetic operators.
5. This study designed and developed the required components to implement a genetic algorithm that optimizes lot sizes, process plans, and machine priority lists simultaneously.
6. This study was the first to incorporate the simulated annealing algorithm in the genetic algorithm to solve job shop problems.
7. This study is the first job shop sequencing algorithm to use a risk-based utility function to rank chromosomes.
8. This study structured the population of chromosomes so that both active and non-delay schedules were used to generate and to evaluate chromosomes.

Recommendations for Further Research

In this section, a list of recommendations for further research is given, as follows:

1. Recall from Chapter III that the integration of the components for the dynamic stochastic

constrained genetic algorithm to minimize total tardiness and to evaluate chromosomes using probability Gantt charting (CGA_APP) was incomplete because of the complications mentioned. Hence, this is a fruitful area for further research. Recall that this model was attempted to answer research question 5, which asks what the performance of the constrained genetic algorithm is when lot sizes, process plans, and machine priority lists are optimized simultaneously.

2. There is a need to lower the CPU time needed by the genetic algorithms, which could be accomplished by improving the selection method or using a different selection method.
3. The evaluation of chromosomes using the probability Gantt charting was shown to be an effective method. However, an improvement is needed to make the probability Gantt charting a better estimator for total tardiness by tuning it with several probability values. This implies that the probability Gantt charting is both problem and probability values dependent.
4. In this study, genetic algorithms were developed in which jobs were not allowed to make re-visits to machines. Hence, further research could be done to investigate the issue of job re-visits to machines.
5. In experiment VI, the lot sizes were computed according to a policy that was proposed by Sawaqed (1987). However, there could be an even greater improvement in production criteria due to other lot sizing methods. Thus, a lot sizing method, common cycle time, which was proposed by Foote (1993) can be used to further investigate the lot sizing effect.
6. The following two questions are research gaps that were not attempted: 1) Do multiple criteria affect the performance of the constrained genetic algorithm? 2) What is the impact on the job shop performance measures of combining static and dynamic disciplines?

REFERENCES

Adams, J., Balas, E., and Zawack, D. 1988. "The Shifting Bottleneck Procedure for Job Shop Scheduling." Management Science 34(3), pp. 391-401.

Al-Harkan, I. and Foote, B. L. May-1994. "Genetic Algorithms and Simulated Annealing Algorithm: Unobserved Potential." Working Paper, School of Industrial Engineering, University of Oklahoma, Norman.

Al-Harkan, I. and Foote, B. L. May-1996. "On Merging Optimality Conditions with Genetic Algorithms: An Experimental Investigation." Working Paper, School of Industrial Engineering, University of Oklahoma, Norman.

Al-Harkan, I. and Foote, B. L. May-1997. "The Constrained and the Unconstrained Genetic Algorithms Computer Codes for Job Shop Scheduling Problems." Technical Report, School of Industrial Engineering, University of Oklahoma, Norman.

Anderson, E, J, and Nyirenda, J. C. 1990. "Two New Rules to Minimize Tardiness in a Job Shop." International Journal of Production Research 28(12), pp. 2277-2292.

Arumugam, V. and Ramani, S. 1980. "Evaluation of Value Time Sequencing Rules in a Real World Job-Shop." Journal of Operational Research Society 31, pp. 895-904.

Ashour, S. and Vaswani, S. D. 1972. "A GASP Simulation Study of Job-Shop Scheduling." Simulation 18(1), pp. 1-10.

Bagchi, S., Uckun, S., Miyabe, Y., and Kawamura, K. 1991. "Exploring Problem-Specific Recombination Operators for Job Shop Scheduling." In Belew, R. and Booker, L. Editors Proceedings of the Fourth International Conference on Genetic Algorithms. Los Altos, CA, Morgan Kaufmann Publishers, pp. 10-17.

Bahouth, S. B. 1991. Experimental Investigation On The Johnson Rule for Sequencing Jobs in a Two-Bottleneck Job Shop. Ph.D. Dissertation, The University of Oklahoma.

Bahouth, S. B. and Foote, B. L. 1994. "Managing a Two-Bottleneck Job Shop with a Two-Machine Flow Shop Algorithm." International Journal of Production Research 32(10), pp. 2463-2477.

Baker, C. T. and Dzielinski, B. P. 1960. "Simulation of a Simplified Job Shop." Management Science 6(3), pp. 311-323.

Baker, K. R. 1974. Introduction to Sequencing and Scheduling. New York, John Wiley and Sons.

Baker, K. R. 1984. "Sequencing Rules and Due-Date Assignments in a Job Shop." Management Science 30(9), pp. 1093-1104.

Baker, K. R. and Bertrand, J. W. M. 1981. "An Investigation of Due-Date Assignment Rules with Constrained Tightness." Journal of Operations Management 1(3), pp. 109-120.

Baker, K. R. and Bertrand, J. W. M. 1982. "A Dynamic Priority Rule for Sequencing Against Due-Dates." Journal of Operations Management 3(1), pp. 37-42.

Baker, K. R. and Kanet, J. J. 1983. "Job Shop Scheduling with Modified Due Dates." Journal of Operations Management 4(1), pp. 11-22.

Balas, E., Lenstra, J. K., and Vazacopoulos, A. 1995. "The One-Machine Problem With Delayed Precedence Constraints and Its Use in Job Shop Scheduling." Management Science 41(1), pp. 94-109.

Bean, J. C. 1994. "Genetic Algorithms and Random Keys for Sequencing and Optimization." ORSA Journal on Computing 6(2), pp. 154-160.

Belew, R. and Booker, L. Editors, 1991. Proceedings of the Fourth International Conference on Genetic Algorithms. San Mateo, CA, Morgan Kaufmann Publishers, Inc.

Bellman, R., Esogbue, A. O., and Nabeshima, I. 1982. Mathematical Aspects of Scheduling and Applications. New York, Pergamon Press.

Berry, W. L. and Finlay, R. A. 1976. "Critical Ratio Scheduling with Queue Waiting Time Information: An Experimental Analysis." AIIE Transactions 8(2), pp. 161-168.

Bhaskaran, K. and Pinedo, M. 1992. "Dispatching." In Salvendy, G. ed. Handbook of Industrial Engineering New York, John Wiley & Sons, Incorporated, pp. 2182-2198.

Bierwirth, C. 1995. "A Generalized Permutation Approach to Job Shop Scheduling With Genetic Algorithms." OR Spektrum 17, pp. 87-92.

Bierwirth, C., Kopfer, H., Mattfel, D., and Rixen, I. 1995. "Genetic Algorithm Based Scheduling in a Dynamic Manufacturing Environment." In DeSilva, C. Editor. Proceedings of the Second IEEE Conference on Evolutionary Computation. Perth, IEEE Press, pp. 67-73.

Blackstone, J. H., Jr., Phillips, D. T., and Hogg, G. L. 1982. "A State-of-the-art Survey of Dispatching Rules for Manufacturing Job Shop Operations." International Journal of Production Research 20(1), pp. 27-45.

Brown, A. P. and Lomnicki, Z. A. 1966. "Some Application of the "Branch-and-Bound" Algorithm to Machine Scheduling Problem." Operational Research Quarterly 17(2), pp. 173-186.

Browne, J., Harhen, J. and Shivnan, J. 1988. Production Management Systems. New York, Addison-Wesley Publishing Company.

Buffa, E. S. and Miller, J. G. 1968. Production-Inventory Systems: Planning and Control. Third Edition, Homewood, Illinois, Richard D. IRWIN, Incorporated

Buxey, G. 1989. "Production Scheduling: Practice and Theory." European Journal of Operational Research 39, pp. 17-31.

Campbell, H. G., Dudek, R. A., and Smith, M. L. 1970. "A Heuristic Algorithm for the n Job, m Machine Sequencing Problem." Management Science 16(10), pp. B630-B637.

Carroll, D. C. 1965. Heuristic Sequencing of Single and Multiple Component Jobs. Ph.D. Dissertation, Sloan School of Management, MIT.

Chambers, L. Editor, 1995a. Practical Handbook of Genetic Algorithms Applications. Volume I, New York, CRC Press.

Chambers, L. Editor, 1995b. Practical Handbook of Genetic Algorithms Applications. Volume II, New York, CRC Press.

Chang, Feng-Chang R. 1994. "A Study of Factor Affecting Due-Date Predictability in a Simulation Dynamic Job Shop." Journal of Manufacturing Systems 13(6), pp. 393-400.

Chang, Y., Sueyoshi, T., and Sullivan, R. 1996. "Ranking Dispatching Rules by Data Envelopment Analysis in a Job Shop Environment." IIE Transactions 28, pp. 631-642

Chen, Chuen-Lung, Vempati, V. S., and Aljaber, N. 1995. "An Application of Genetic Algorithms for Flow Shop Problems." European Journal of Operational Research 80, pp. 389-396.

Chen, H. and Flann, N. 1994. "Parallel Simulated Annealing and Genetic Algorithms: A Space of Hybrid Methods." in Davidor, Y., Schwefel, H., and Männer, R. Editors Parallel Problem Solving from Nature-PPSN III. Berlin Heidelberg, Springer-Verlag, pp. 428-438.

Cheng, T. C. E. and Gupta, M. C. 1989. "Survey of Scheduling Research Involving Due Date Determination Decisions." European Journal of Operational Research 38, pp. 156-166.

Cheng, T. C. E. and Sin, C. C. S. 1990. "A State-of-the-art Review of Parallel-Machine Scheduling Research." European Journal of Operational Research 47, pp. 271-292.

Cleveland, G. and Smith, S. F. 1989. "Using Genetic Algorithms to Schedule Flow Shop Releases." In Schaffer, J. Editor Proceedings of the Third International Conference on Genetic Algorithms. Los Altos, CA, Morgan Kaufmann Publishers, pp. 160-169.

Conway, R. W. 1965a. "Priority Dispatching and Work-In-Process Inventory in a Job Shop." Journal of Industrial Engineering 16(2), pp. 123-130.

Conway, R. W. 1965b. "Priority Dispatching and Job Lateness In a Job Shop." Journal of Industrial Engineering 16(4), pp. 228-237.

Conway, R. W., Johnson, B. M., and Maxwell, W. L. 1960. "An Experimental Investigation of Priority Dispatching." Journal of Industrial Engineering 11(3), pp. 221-229.

Conway, R. W., Maxwell, W. L., and Miller, L. W. 1967. Theory of Scheduling. Massachusetts, Addison-Wesley Publishing Company.

Croce, F. D., Tadei, R., and Volta, G. 1995. "A Genetic Algorithm for the Job Shop Problem." Computers and Operations Research 22(1), pp. 15-24.

Dannenbring, D. G. 1977. "An Evaluation of Flow Shop Sequencing Heuristics." Management Science 23(11), pp. 1174-1182.

Dar-El, E. M. and Wysk, R. A. 1982. "Job Shop Scheduling-- A Systematic Approach." Journal of Manufacturing Systems 1(1), pp. 77-88.

Dauzere-Peres, S. and Lasserre, J. B. 1993. "A Modified Shifting Bottleneck Procedure for Job-Shop Scheduling." International Journal of Production Research 31(4), pp. 923-932.

Davis, L. 1985. "Job Shop Scheduling With Genetic Algorithms." In Grefenstette, J. J. Editor, Proceedings of the First International Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates, pp. 136-140.

Davis, L. ed. 1991. Handbook of Genetic Algorithms. New York, Van Nostrand Reinhold.

Day, J. E. and Hottenstein, M. P. 1970. "Review of Sequencing Research." Naval Research Logistic Quarterly 17(11), pp. 11-39.

Dayhoff, J. E. and Atherton, R. W. 1986. "Signature Analysis of Dispatching Schemes in Wafer Fabrication." IEEE Transactions on Components, Hybrid and Manufacturing Technology 9, pp. 498-507.

Dell'Amico, M. and Trubian, M. 1993. "Applying Tabu Search to the Job Shop Scheduling Problem." Annals of Operations Research 41, pp. 231-252.

Dorndorf, U. and Pesch, E. 1995. "Evolution Based Learning in a Job Shop Scheduling Environment." Computers and Operations Research 22(1), pp. 25-40.

Dudek, R. A., Panwalkar, S. S., and Smith, M. L. 1992. "The Lessons of Flowshop Scheduling Research." Operations Research 40(1), pp. 7-13.

Eilon, S. and Chowdhury, I. G. 1976. "Due Dates in Job Shop Scheduling." International Journal of Production Research 14(2), pp. 223-237.

Eilon, S., Chowdhury, I. G., Serghiou, S. S. 1975. "Experiments With the SI^{X} rule in Job-Shop Scheduling." Simulation 24(2), pp. 45-48.

Elmaghraby, Salah E. 1968. "The Machine Sequencing Problem-Review and Extensions." Naval Research Logistics Quarterly 15(2), pp. 205-232.

Elvers, D. A. 1973. "Job Shop Dispatching Rules Using Various Delivery Date Setting Criteria." Production Inventory Management 14(4), pp. 62-70.

Elvers, D. A. 1974. "The Sensitivity of the Relative Effectiveness of Job Shop Dispatching Rules with Respect to Various Arrival Distributions" AIIE Transactions 6(1), pp. 41-49.

Elvers, D. A. and Taube, L. R. 1983. "Time Completion for Various Dispatching Rules in Job Shops." OMEGA: The International Journal of Management Science 11(1), pp. 81-89.

Elvers, D. A. and Treleven, M. D. 1985. "Job-shop vs. Hybrid Flow-Shop Routing in a Dual Resource Constrained System." Decision Sciences 16, pp. 213-222.

Emmons, H. 1975. "One Machine Sequencing to Minimize Mean Flow Time with Minimum Number Tardy." Naval Research Logistic Quarterly 22(3), pp. 585-592.

Eshelman, L. J. Editor, 1995. Proceedings of the Sixth International Conference on Genetic Algorithms. San Francisco, CA, Morgan Kaufmann Publishers.

Falkenauer, E. and Bouffouix, S. 1991. "A Genetic Algorithm for Job Shop." In Proceedings of the 1991 IEEE International Conference on Robotics and Automation, pp. 824-829.

Fang, H., Ross, P., and Corne, D. 1993. "A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems." In Forrest, S. Editor, Proceedings of the Fifth International Conference on Genetic Algorithms. San Mateo, CA, Morgan Kaufmann Publishers, pp. 375-382.

Farn, C. K. 1979. The Dynamic Aspects of Some Scheduling Problems. Master Thesis Manchester University.

Fisher, H. and Thompson, G. L. 1963. "Probabilistic Learning Combination of Local Job-Shop Scheduling Rules." In Muth, J. F. and Thompson, G. L. Editors, Industrial Scheduling. New Jersey, Prentice-Hall, Inc, pp. 225-251.

Foote, B. L. Fall-1993. "Lot Size Selection." Research Notes, School of Industrial Engineering, University of Oklahoma, Norman.

Forrest, S. Editor, 1993. Proceedings of the Fifth International Conference on Genetic Algorithms. San Mateo, CA, Morgan Kaufmann Publishers.

Forst, F. G. 1984. "A review of the Static, Stochastic Job Sequencing Literature." Opsearch 21(3), pp. 127-144.

French, S. 1982. Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop. New York, John Wiley and Sons.

Gen, M. and Cheng, R. 1997. Genetic Algorithms and Engineering Design. New York, John Wiley & Sons, Inc.

Gen, M., Tsumjimura, Y., and Kubota, E. 1994."Solving Job-Shop Scheduling Problems By Genetic Algorithm." In Gen, M. and Kobayashi, T. Editors. Proceedings of the 16th International Conference on Computers and Industrial Engineering. Ashikaga, Japan, pp. 1577-1582.

Gere, W. S. 1966. "Heuristic in Job Shop Scheduling." Management Science 13(3), pp. 167-190.

Giffler, B. and Thompson, G. L. 1960. "Algorithm for Solving Production Scheduling Problems." Operations Research 8(4), pp. 487-503.

Goldberg, D. E. 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Reading, Mass., Addison-Wesley.

Goldberg, D. E. and Lingle R. 1985. "Alleles, Loci, and the Traveling Salesman Problem." In Grefenstette, J. J. Editor, Proceedings of the First International Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates, pp. 154-159.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. 1979. "Optimisation and Approximation in Determining Sequencing and Scheduling: A Survey." Annals of Discrete Mathematics 5, pp. 287-326.

Graves, S. C. 1981. "A Review of Production Scheduling." Operations Research 29(4), pp. 646-675.

Grefenstette, J. Editor, 1987. Proceedings of the Second International Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates.

Grefenstette, J. J. Editor, 1985. Proceedings of the First International Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates.

Grefenstette, J. J., Gopal, R., Rosmaita, B., and Gucht, D. V. 1985. "Genetic Algorithms for the Traveling Salesman Problem." In Grefenstette, J. J. Editor, Proceedings of the First International Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates, pp. 160-168.

Gupta, M., Gupta, Y., and Kumar, A. 1993. "Minimizing Flow Time Variance in a Single Machine System Using Genetic Algorithms." European Journal of Operational Research 70, pp. 289-303.

Hall, N. and Sriskandarajah, C. 1995. "A Survey of Machine Scheduling Problems With Blocking and No-Wait in Process." Operations Research 43, pp. 510-525.

Haupt, R. 1989. "A Survey of Priority Rule-Based Scheduling." OR Spektrum 11, pp. 3-16.

Hershauer, J. C. and Ebert, R. J. 1975. "Search and Simulation Selection of a Job-Shop Sequencing Rule." Management Science 21(7), pp. 833-843.

Holland, J. H. 1992. Adaptation in Natural and Artificial Systems. MIT Press.

Holloway, C. A. and Nelson, R. T. 1974. "Job-Shop Scheduling With Due-Dates and Variable Processing Times." Management Science 20(9), pp. 1264-1275.

Hottenstein, M. P. 1970. "Expedition in Job-Order Control Systems: A Simulation Study." AIIE Transactions 2(1), pp. 46-54.

Hurrion, R. D. 1978. "An Investigation of Visual Interactive Simulation Methods Using the Job-Shop Scheduling Problem." Journal of Operational Research Society 29(11), pp. 1085-1093.

Husband, P. and Mill, F. 1991. "Simulated Co-Evolution as the Mechanism for Emergent Planning and Scheduling." In Belew, R. and Booker, L. Editors, Proceedings of the Fourth International Conference on Genetic Algorithms. San Mateo, CA, Morgan Kaufmann Publishers, Inc, pp. 264-270.

Irastorza, J. C. and Deane, R. H. 1974. "A Loading and Balancing Methodology for Job Shop Control." AIIE Transactions 6(4), pp. 302-307.

Jackson, J. R. 1957a. "Networks of Waiting Lines" Operations Research 5, pp. 518-521.

Jackson, J. R. 1957b. "Simulation Research On Job Shop Production." Naval Research Logistic Quarterly 4(4), pp. 287-295.

Jackson, J. R. 1963. "Jobshop-Like Queueing Systems" Management Science 10(1), pp. 131-142.

Johnson, S. M. 1954. "Optimal Two- and Three-Stage Production Schedules With Setup Times Included." Naval Research Logistic Quarterly 1(1), pp. 61-68.

Jones, C. H. 1973. "An Economic Evaluation of Job Shop Dispatching Rules." Management Science 20(3), pp. 293-307.

Kamath, M. 1994. "Recent Development in Modeling and Performance Analysis Tools for Manufacturing Systems." In Joshi, S. B. and Smith, J. S., eds. Computer Control of Flexible Manufacturing Systems: Research and Development. New York, Chapman and Hall, pp. 231-263.

Kanet, J. J. and Christy, D. P. 1989. "Manufacturing Systems with Forbidden Early Shipments: Implications for Setting Manufacturing Lead Times." International Journal of Production Research 27(5), pp. 783-792.

Kanet, J. J. and Sridharan, V. 1991. "PROGENITOR: A Genetic Algorithm for Production Scheduling." Wirtschafts Informatik, 33, pp. 332-336.

Kanet, J. J. and Zhou, Z. 1993. "A Decision Theory Approach to Priority Dispatching for Job Shop Scheduling." Production and Operations Management 2(1), pp. 2-14.

Karsiti, M. N., and Cruz, J. B., and Mulligan, J. H. 1992. "Simulation Studies of Multilevel Dynamic Job Shop Scheduling Heuristic Dispatching Rules." Journal of Manufacturing Systems 11(5), pp. 346-358.

Kobayashi, S., Ono, I., and Yamamura, M. 1995. "An Efficient Genetic Algorithm for Job Shop Scheduling Problems." In Eshelman, L. J. Editor, Proceedings of the Sixth International Conference on Genetic Algorithms. San Francisco, CA, Morgan Kaufmann Publishers, pp. 506-511.

Koulamas, C., Antony, S., and Jaen, R. 1994. "A Survey of Simulated Annealing Applications to Operations: Research Problems." OMEGA: The International Journal of Management Science 22(1), pp. 41-56.

Kovalev, M. Y., Shafransku, Y. M., Strusfvich, V. A., Tanafv, V. S., and Tuzikov, A. V. 1989. "Approximation Scheduling Algorithms: A Survey." Optimization 20, pp. 859-878.

Laarhoven, P. J. M., Aarts, E. H. L., and Lenstra, J. K. 1992. "Job Shop Scheduling By Simulated Annealing." Operations Research 40, pp. 113-125.

Law, A. and Kelton, W. 1991. Simulation Modeling and Analysis. New York, McGraw-Hill, Inc.

Lawrence, S. 1984. Supplement to "Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques." GSIA, Carnegie-Mellon University.

Lee, C. and Choi, J. 1995. "A Genetic Algorithm For Job Sequencing Problems With Distinct Due Dates and General Early-Tardy Penalty Weights." Computers and Operations Research 22(8), pp. 857-869.

Lee, C. and Kim, S. 1995. "Parallel Genetic Algorithms for the Earliness-Tardiness Job Scheduling Problem with General Penalty Weights." Computer and Industrial Engineering 28(2), pp. 231-243.

Lee, I., Sikora, R., and Shaw, M. 1993. "Joint Lot Sizing and Sequencing with Genetic Algorithms for Scheduling: Evolving the Chromosome Structure." In Forrest, S. Editor, Proceedings of the Fifth International Conference on Genetic Algorithms. San Mateo, CA, Morgan Kaufmann Publishers, pp. 383-389.

Lemoine, A. 1977. "State-of-the-art: Networks of Queues- A Survey of Equilibrium Analysis." Management Science 24(4), pp. 464-481.

Liang, X. 1996. A Methodology for Quick Estimates of Total Tardiness and Number of Tardy Jobs for Stochastic Job Shops. Master Thesis, The University of Oklahoma.

Liepins, G. E. and Hilliard, M. R. 1989. "Genetic Algorithms: Foundations and Applications." Annals of Operations Research 21, pp. 31-58.

Liepins, G. E., Hilliard, M. R., Palmer, M., Morrow, M. 1987. "Greedy Genetics." In Grefenstette, J. J. Editor, Proceedings of the Second International Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates, pp. 90-99.

Lin, Jin-Ling 1993. An Analysis of Genetic Algorithm Behavior for Combinatorial Optimization Problems. Ph.D. Dissertation, The University of Oklahoma.

Maccarthy, B. L. and Liu, J. 1993. "Addressing the Gap in Scheduling Research: A review of Optimization and Heuristic Methods in Production Scheduling." International Journal of Production Research 31(1), pp. 59-79.

Mahfoud, S. W. and Goldberg, D. E. April 1992. "Parallel Recombinative Simulated Annealing: a Genetic Algorithm." Illinois Genetic Algorithms Laboratory (IlliGAL) Report 92002, Department of General Engineering, University of Illinois at Urbana-Champaign, 117 Transportation Building, 104 South Mathews Avenue, Urbana, IL 61801.

Mattfeld, D. 1996. Evolutionary Search and the Job Shop: Investigations on Genetic Algorithms for Production Scheduling. New York, Springer-Verlag.

Michalewicz, Z. 1994. Genetic Algorithms + Data Structure = Evolution Programs. 2d ed. New York, Springer-Verlag.

Miyazaki, S. 1981. "Combined Scheduling System for Reducing Job Tardiness in a Job Shop." International Journal of Production Research 19(2), pp. 201-211.

Moore, J. M. and Wilson, R. C. 1967. "A Review of Simulation Research in Job Shop Scheduling." Production and Inventory Management 8(1), pp. 1-10.

Morton, T. E. and Pentico, D. W. 1993. Heuristic Scheduling Systems. New York, John Wiley and Sons.

Muchnik, M. 1992. Complete Guide to Plant Operations Management. New Jersey, Prentice Hall.

Muhlemann, A. P., Lockett, A. O., Farn, C. K. 1982. "Job Shop Scheduling Heuristics and Frequency of Scheduling." International Journal of Production Research 20(2), pp. 227-241.

Muth, J. F. and Thompson, G. L. 1963. Industrial Scheduling. New Jersey, Prentice-Hall, Inc.

Nakano, R. and Yamada, T. 1991. "Conventional Genetic Algorithm for Job Shop Problems." In Belew, R. and Booker, L. Editors. Proceedings of the Fourth International Conference on Genetic Algorithms. Los Altos, CA, Morgan Kaufmann Publishers, pp. 474-479.

Nanot, Y. R. 1963. An Experimental Investigation and Comparative Evaluation of Priority Disciplines in Job Shop-Like Queueing Networks. Ph.D. Dissertation, University of California Los Angeles.

Nelson, R. T. and Jackson, J. R. 1957. "SWAC Computations For Some m X n Scheduling Problems." Journal of Association for Computing Machinery 4(4), pp. 438-441.

Nelson, R. T., Holloway, C. A., and Wong, R. M. L. 1977. "Centralized Scheduling and Priority Implementation Heuristics for a Dynamic Job Shop Model." AIIE Transactions 9(1), pp. 95-102.

Neppalli, V. R. 1994. Optimized Genetic Algorithms Approach to Solve Flow Shop Scheduling Problem. Master Thesis, Mississippi State University.

Nof, S. Y., Rajan, V. N., and Frederick. S. W. 1990. "Knowledge-Based, Dynamic, Real-Time Scheduling and Rescheduling: A Review and Some Annotated References." Research Memorandum No. 89-16, School of Industrial Engineering, Purdue University, West Lafayette.

Norman, B. A. and Bean, J. C. 1994. "Random Keys Genetic Algorithm for Job Shop Scheduling." Technical Report 94-5, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor.

Noronha, S. J., and Sarma, V. V. S. 1991. "Knowledge-Based Approaches for Scheduling Problems: A Survey." IEEE Transactions on Knowledge and Data Engineering 3(2), pp. 160-171.

Nowicki, E. and Smutnicki, C. 1996. "A Fast Taboo Search Algorithm for the Job Shop Problem." Management Science 42(6), pp. 797-813.

Oliver, I. M., Smith. D. J., and Holland, J. R. C. 1987. "A Study of Permutation Crossover Operators on the Traveling Salesman Problem." In Grefenstette, J. J. Editor, Proceedings of the Second International Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates, pp. 224-230.

Osman, Ibrahim and Kelly, J. Editors, 1996. Meta-Heuristics: Theory and Applications. Boston, Kluwer Academic Publishers.

Panwalkar, S. S. and Iskander, W. 1977. "A Survey of Scheduling Rules." Operations Research 25(1), pp. 45-61.

Park, Y. B., Pegden, C. D., and Enscore, E. E. 1984. "A Survey and Evaluation of Static Flowshop Scheduling Heuristics." International Journal of Production Research 22(1), pp. 127-141.

Pinedo, M. 1995. Scheduling: Theory, Algorithms, and Systems. New York, Prentice Hall.

Putnam, A. O., Everdall, R. Dorman, G. H., Cronan, R. R., and Lindgren, L. H. 1971. "Updating Critical Ratio and Slack Time Priority Scheduling Rules." Production And Inventory Management 12(4), pp. 51-72.

Rachamadugu, R. V., Raman, N., and Talbot, F. B. 1986. "Real-Time Scheduling of an Automated Manufacturing Center." National Bureau Standards Special Publication 724, pp. 293-315.

Ragatz, G. L. and Mabert, V. A. 1984. "A Simulation Analysis of Due Date Assignment Rules." Journal of Operations Management 5(1), pp. 27-39.

Raghavachari, M. 1988. "Scheduling Problems with Non-Regular Penalty Functions--A Review." Opsearch 25(3), pp. 144-164.

Raghu, T. S. and Rajendran, C. 1993. "An Efficient Dynamic Dispatching Rule for Scheduling in a Job Shop." International Journal of Production Economics 32, pp. 301-313.

Ramasesh, R. 1990. "Dynamic Job Shop Scheduling: A Survey of Simulation Research." OMEGA: The International Journal of Management Science 18(1), pp. 43-57.

Rawlins, G. J. E. Editor, 1991. Foundations of Genetic Algorithms. San Mateo, California, Morgan Kaufmann Publishers.

Reeves, C. 1995. "A Genetic Algorithm for Flowshop Sequencing." Computer and Operations Research 22(1), pp. 5-13.

Rinnooy Kan, A. H. G. 1976. Machine Scheduling Problems: Classification, Complexity, and Computations. Martinus Nijhoff, The Hague.

Rodammer, F. A. and White, K. P. 1988. "A Recent Survey of Production Scheduling." IEEE Transactions on Systems, Man and Cybernetics 18(6), pp. 841-851.

Rohleder, T. R. and Scudder, G. 1993a. "Comparing Performance Measure in Dynamic Job Shops: Economics vs. Time." International Journal of Production Economics 32, pp. 169-183.

Rohleder, T. R. and Scudder, G. 1993b. "A Comparison of Order-Release and Dispatch Rules for the Dynamic Weighted Early/Tardy Problem." Production and Operations Management 2(3), pp. 221-238.

Rowe, A. J. 1958. Sequential Decision Rules in Production Scheduling. Ph.D. Dissertation, University of California Los Angeles.

Rubin, P. and Ragatz, G. L. 1995. "Scheduling in a Sequence Dependent Setup Environment With Genetic Search." Computers and Operations Research 22(1), pp. 85-99.

Russell, Dar-El, and Taylor 1987."A Comparative Analysis of the COVERT Job Sequencing Rule Using Various Shop Performance Measures." International Journal of Production Research 25(10), pp. 1523-1540.

Sawaqed, N. M. 1987. Experimental Investigations In a Hybrid Assembly Job Shop. Ph.D. Dissertation, The University of Oklahoma.

Schaffer, J. Editor 1989. Proceedings of the Third International Conference on Genetic Algorithms. San Mateo, CA, Morgan Kaufmann Publishers, Inc.

Schultz, C. R., 1989. "An Expediting Heuristic for the Shortest Processing Time Dispatching Rule." International Journal of Production Research 27(1), pp. 31-41.

Sen, T. and Gupta, S. K. 1984. "A State-of-Art Survey of Static Scheduling Research Involving Due Dates." OMEGA: The International Journal of Management Science 12(1), pp. 63-76.

Sridhar, H. and Rajendran, C. 1994. "A Genetic Algorithm for Family and Job Scheduling in A Flowline-Based Manufacturing Cell." Computers and Industrial Engineering 27(1-4), pp. 469-472.

Srinivas, M. and Patnaik, L. M. 1994. "Genetic Algorithms: A Survey." Computer 27(1), pp. 17-26.

Starkweather, T., McDaniel, S. Mathias, K., Whitley, C., and Whitley, D. 1991. "A Comparison of Genetic Sequencing Operators." In Belew, R. and Booker, L. Editors, Proceedings of the Fourth International Conference on Genetic Algorithms. San Mateo, CA, Morgan Kaufmann Publishers, Inc. pp. 69-76.

Stöpller, S. and Bierwirth, C. 1992. "The Application of a Parallel Genetic Algorithm to the n/m/P/Cmax Flowshop Problem." In Fandel, G., Gulledge, T., and Jones, A. Editors. New Directions For Operations Research in Manufacturing. Berlin Heidelberg, Springer Verlag, pp. 161-175.

Syswerda, G. 1991. "Scheduling Optimization Using Genetic Algorithms." In Davis, L. Editor, Handbook of Genetic Algorithms. New York, Van Nostrand Reinhold, pp. 332-349.

Szelke, E. and Kerr, R. M. 1994. "Knowledge-Based Reactive Scheduling." Production Planning and Control 5(2), pp. 124-145.

Udo, Godwin 1993. "An Investigation of Due-Dates Assignment Using Workload Information of a Dynamic Shop." International Journal of Production Economics 29, pp. 89-101.

Vempati, V. S., Chen, C., and Bullington, S. 1993. "An Effective Heuristic for Flow Shop Problems with Total Flow Time as Criterion." Computer and Industrial Engineering 25(1-4), pp. 219-222.

Vepsalainen, A. P. J. and Morton, T. E. 1988. "Improving Local Priority Rules With Global Leadtime Estimates." Journal of Manufacturing and Operations Management 1, pp. 102-118.

Vepsalainen, A. P. J. and Morton, T. E. 1989. "Priority Rules for Job Shops with Weighted Tardiness Cost." Management Science 33(8), pp. 1035-1047.

Vig, M. M. and Dooley, K. J. 1993. "Mixing Static and Dynamic Flowtime Estimates for Due-Date Assignment." Journal of Operations Management 11, pp. 67-79.

Wagner, H. M. 1959. "An Integer Programming Model for Machine Scheduling." Naval Research Logistic Quarterly 6, pp. 131-140.

Wainwright, R. L. 1993. "Introduction to Genetic Algorithms: Theory and Applications." Tutorial Session, In The Seventh Oklahoma Symposium on Artificial Intelligence. Oklahoma State University, Stillwater, November 18-19.

Wayson, R. D. 1965. "The Effects of Alternative Machines on Two Priority Dispatching Disciplines in the General Job Shops." Master Thesis, Cornell University

Weeks, J. K. 1979. "A Simulation Study of Predictable Due-Dates." Management Science 25(4), pp. 363-373.

Whitley, D., Starkweather, T., and Fuguay D. 1989. "Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator." In Schaffer, J. Editor Proceedings of the Third International Conference on Genetic Algorithms. Los Altos, CA, Morgan Kaufmann Publishers, pp. 133-140.

Yamada, T. and Nakano, R. 1996. "Job-Shop Scheduling by Simulated Annealing Combined with Deterministic Local Search." In Osman, Ibrahim and Kelly, J. Editors, Meta-Heuristics: Theory and Applications. Boston, Kluwer Academic Publishers, pp. 237-248.

Yamada, T., Rosen, E., and Nakano, R. 1994. "A Simulated Annealing Approach to Job Shop Scheduling Using Critical Block Transition Operators." In The 1994 IEEE International Conference on Neural Networks. Piscataway, IEEE, Inc., pp. 4687-4692.

Yen, P. C. and Pinedo, M. L. 1994. "Scheduling Systems: A Survey." Technical Report, Department of Industrial Engineering and Operations Research, Columbia University, New York.

APPENDIX A

EXPERIMENTAL RESULTS OF SINGLE MACHINE MODELS:

Table A.1. Summary of the CPU time results for the dynamic programming approach.

Comb. No.	Problem Size (n)	Problem type	No. of problems solved	CPU time needed by DP
1	18	I[a]	9	9.65
2	18	II[b]	9	9.56
3	18	III[c]	9	9.63
4	18	IV[d]	9	15.09
5	20	I	9	45.05
6	20	II	9	46.08
7	20	III	9	45.27
8	20	IV	9	82.28
9	22	I	9	408.79
10	22	II	9	427.58
11	22	III	9	395.65
12	22	IV	9	547.03
13	24	I	9	2332.93
14	24	II	9	2340.68
15	24	III	9	2378.22
16	24	IV	9	3600.96

Table A.2. Summary of algorithm results (case I: Population size=3.5n & no. of generations =n^2).

				CGA_OBM				UGA_OBM			
Comb. No.	Problem Size (n)	Problem type	No. of problems solved	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)
1	18	I[a]	9	3.49	9.64	2.00	1.31	3.87	19.97	1.00	3.05
2	18	II[b]	9	9.30	29.11	1.00	1.30	4.44	18.36	3.00	3.07
3	18	III[c]	9	1.57	13.36	5.00	1.31	0.91	5.40	4.00	3.01
4	18	IV[d]	9	1.82	8.24	5.00	1.31	2.05	15.33	3.00	3.03
5	20	I	9	1.13	6.33	2.00	1.91	4.47	12.82	1.00	4.56
6	20	II	9	4.19	10.86	1.00	1.90	12.81	45.33	1.00	4.61
7	20	III	9	3.33	9.29	1.00	1.89	3.81	13.74	1.00	4.55
8	20	IV	9	4.06	7.52	0.00	1.90	4.14	11.58	1.00	4.56
9	22	I	9	4.63	19.55	1.00	2.69	3.01	8.52	2.00	6.58
10	22	II	9	3.51	19.31	0.00	2.70	5.19	10.37	0.00	6.66
11	22	III	9	2.57	7.76	2.00	2.68	2.91	10.81	2.00	6.54
12	22	IV	9	2.68	8.28	2.00	2.69	2.13	5.69	1.00	6.57
13	24	I	9	1.63	5.86	2.00	3.69	2.35	7.73	1.00	9.16
14	24	II	9	2.18	5.65	3.00	3.69	4.16	10.17	0.00	9.24
15	24	III	9	1.22	4.93	4.00	3.65	1.16	4.24	1.00	9.07
16	24	IV	9	2.27	16.35	3.00	3.66	2.41	8.63	1.00	9.14

[a] Type I: P_i was uniformly distributed between 1 & 5, and d_i was uniformly distributed between P_j and $P_j + n$.

[b] Type II: P_i was uniformly distributed between 1 & 5, and d_i was uniformly distributed between P_j & $P_j + 1.5n$.

[c] Type III: P_i was uniformly distributed between 1 &10, and d_i was uniformly distributed between P_j & $P_j + n$.

[d] Type IV: P_i was uniformly distributed between 1 & 10, and d_i was uniformly distributed between P_j & $P_j + 1.5n$.

Table A.3. Summary of algorithm results (case II: Population size=3.5n & no. of generations =$n^{2.5}$).

				CGA_OBM				UGA_OBM			
Comb. No.	Problem Size (n)	Problem type	No. of problems solved	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)
1	18	I^a	9	3.97	13.27	3.00	5.50	5.58	11.28	2.00	12.82
2	18	II^b	9	5.55	28.07	3.00	5.52	8.76	25.99	2.00	12.90
3	18	III^c	9	0.98	5.34	6.00	5.51	4.08	9.72	2.00	12.70
4	18	IV^d	9	2.00	12.39	4.00	5.52	5.48	20.48	1.00	12.78
5	20	I	9	0.43	1.73	6.00	8.48	3.03	21.54	4.00	20.25
6	20	II	9	5.89	14.09	2.00	8.52	8.43	60.00	4.00	20.47
7	20	III	9	4.06	11.40	3.00	8.47	0.94	4.06	4.00	20.11
8	20	IV	9	2.06	11.92	4.00	8.47	1.30	7.61	2.00	20.21
9	22	I	9	2.84	20.00	3.00	12.60	2.97	12.92	5.00	30.66
10	22	II	9	5.37	30.49	3.00	12.64	6.39	25.81	2.00	30.98
11	22	III	9	3.28	15.13	1.00	12.58	1.42	5.56	3.00	30.44
12	22	IV	9	3.33	13.03	0.00	12.57	1.96	7.73	4.00	30.59
13	24	I	9	3.07	8.35	2.00	17.94	2.16	10.89	3.00	44.55
14	24	II	9	1.94	7.76	5.00	17.98	3.69	17.46	3.00	44.88
15	24	III	9	1.52	6.82	5.00	17.85	1.17	6.15	5.00	44.17
16	24	IV	9	1.95	7.08	2.00	17.85	1.68	8.24	4.00	44.44

Table A.4. Summary of algorithm results (case III: Population size=4n & no. of generations =n^2).

				CGA_OBM				UGA_OBM			
Comb. No.	Problem Size (n)	Problem type	No. of problems solved	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)
1	18	I^a	9	5.94	22.02	2.00	1.49	2.70	4.25	0.00	3.51
2	18	II^b	9	3.96	10.80	0.00	1.49	2.41	7.31	3.00	3.53
3	18	III^c	9	2.70	12.42	3.00	1.49	1.46	8.44	3.00	3.47
4	18	IV^d	9	6.97	19.78	2.00	1.49	1.72	11.61	2.00	3.50
5	20	I	9	2.57	11.89	3.00	2.18	2.57	14.76	2.00	5.23
6	20	II	9	3.28	12.27	3.00	2.18	5.78	28.90	1.00	5.27
7	20	III	9	1.10	4.66	2.00	2.16	1.92	12.81	3.00	5.19
8	20	IV	9	4.30	9.04	0.00	2.17	3.08	19.31	1.00	5.20
9	22	I	9	1.26	8.16	4.00	3.06	4.57	9.48	1.00	7.53
10	22	II	9	2.69	10.84	2.00	3.07	5.97	13.56	1.00	7.60
11	22	III	9	1.77	10.42	2.00	3.06	2.16	6.56	0.00	7.50
12	22	IV	9	2.65	11.21	1.00	3.05	4.04	11.40	0.00	7.53
13	24	I	9	0.44	2.05	4.00	4.23	2.20	12.04	3.00	10.55
14	24	II	9	1.43	6.43	4.00	4.25	4.07	16.58	2.00	10.63
15	24	III	9	0.53	2.10	4.00	4.22	1.07	3.04	3.00	10.45
16	24	IV	9	1.28	4.14	3.00	4.21	1.33	4.56	1.00	10.50

[a] Type I: P_i was uniformly distributed between 1 & 5, and d_i was uniformly distributed between P_j & P_j + n.

[b] Type II: P_i was uniformly distributed between 1 & 5, and d_i was uniformly distributed between P_j & P_j + 1.5n.

[c] Type III: P_i was uniformly distributed between 1 &10, and d_i was uniformly distributed between P_j & P_j + n.

[d] Type IV: P_i was uniformly distributed between 1 &10, and d_i was uniformly distributed between P_j & P_j + 1.5n.

Table A.5. Summary of algorithm results (case IV: Population size=4n & no. of generations =$n^{2.5}$).

				CGA_OBM				UGA_OBM			
Comb. No.	Problem Size (n)	Problem type	No. of problems solved	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)
1	18	I[a]	9	0.19	0.71	5.00	6.39	4.92	14.19	1.00	14.72
2	18	II[b]	9	3.26	16.75	5.00	6.40	7.56	28.28	2.00	14.82
3	18	III[c]	9	2.63	11.98	5.00	6.38	3.17	7.20	1.00	14.60
4	18	IV[d]	9	5.31	22.97	2.00	6.38	3.89	10.38	2.00	14.67
5	20	I	9	0.90	4.02	6.00	9.73	2.13	8.46	2.00	23.05
6	20	II	9	7.81	39.12	6.00	9.80	8.10	40.00	2.00	23.32
7	20	III	9	1.33	5.38	5.00	9.72	0.95	3.27	3.00	22.95
8	20	IV	9	2.84	14.21	3.00	9.72	1.83	5.28	3.00	23.03
9	22	I	9	4.73	24.04	4.00	14.46	1.94	12.79	4.00	34.95
10	22	II	9	2.15	4.56	2.00	14.52	2.98	15.18	2.00	35.28
11	22	III	9	1.25	8.53	3.00	14.42	2.39	12.41	2.00	34.70
12	22	IV	9	1.56	10.99	4.00	14.46	3.06	17.86	3.00	34.85
13	24	I	9	2.85	17.11	4.00	20.80	2.00	9.04	5.00	51.10
14	24	II	9	2.75	7.32	4.00	20.86	4.41	20.65	4.00	51.48
15	24	III	9	2.15	5.40	1.00	20.69	0.81	3.81	5.00	50.67
16	24	IV	9	1.73	6.54	5.00	20.71	1.01	4.35	5.00	50.95

Table A.6. Summary of algorithm results (case I: Population size=3.5n & no. of generations =n^2).

				CGA_LOX				UGA_LOX			
Comb. No.	Problem Size (n)	Problem type	No. of problems solved	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)
1	18	I[a]	9	1.08	3.96	2.00	1.90	1.47	6.20	0.00	3.14
2	18	II[b]	9	1.62	6.79	3.00	1.88	2.01	6.79	0.00	3.15
3	18	III[c]	9	0.29	1.20	4.00	1.89	0.61	1.64	1.00	3.14
4	18	IV[d]	9	0.17	0.90	5.00	1.88	0.14	0.50	4.00	3.14
5	20	I	9	0.31	2.01	5.00	2.90	0.99	2.67	2.00	4.85
6	20	II	9	1.83	8.00	4.00	2.91	2.17	6.36	1.00	4.87
7	20	III	9	0.13	0.43	5.00	2.90	0.76	3.98	2.00	4.86
8	20	IV	9	0.38	1.98	4.00	2.90	0.55	1.98	3.00	4.85
9	22	I	9	0.61	1.94	4.00	4.37	0.75	2.46	2.00	7.24
10	22	II	9	0.44	1.22	3.00	4.38	1.53	4.17	1.00	7.26
11	22	III	9	0.12	0.43	5.00	4.32	0.42	1.21	0.00	7.24
12	22	IV	9	0.72	1.93	1.00	4.33	0.97	2.51	2.00	7.24
13	24	I	9	0.38	1.75	5.00	6.21	0.72	2.19	2.00	10.31
14	24	II	9	2.24	10.00	1.00	6.19	1.77	3.87	1.00	10.33
15	24	III	9	0.14	1.01	6.00	6.17	0.44	0.95	1.00	10.29
16	24	IV	9	0.22	1.22	4.00	6.15	0.61	1.69	0.00	10.32

[a] Type I: P_i was uniformly distributed between 1 & 5, and d_i was uniformly distributed between P_j & P_j + n.

[b] Type II: P_i was uniformly distributed between 1 & 5, and d_i was uniformly distributed between P_j & P_j + 1.5n.

[c] Type III: P_i was uniformly distributed between 1 &10, and d_i was uniformly distributed between P_j & P_j + n.

[d] Type IV: P_i was uniformly distributed between 1 &10, and d_i was uniformly distributed between P_j & P_j + 1.5n.

Table A.7. Summary of algorithm results (case II: Population size=3.5n & no. of generations =$n^{2.5}$).

				CGA_LOX				UGA_LOX			
Comb. No.	Problem Size (n)	Problem type	No. of problems solved	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)
1	18	I[a]	9	1.08	3.96	2.00	7.93	1.47	6.20	0.00	13.24
2	18	II[b]	9	1.62	6.79	3.00	7.93	2.01	6.79	0.00	13.25
3	18	III[c]	9	0.29	1.20	4.00	7.89	0.61	1.64	1.00	13.21
4	18	IV[d]	9	0.17	0.90	5.00	7.89	0.14	0.50	4.00	13.23
5	20	I	9	0.31	2.01	5.00	12.90	0.99	2.67	2.00	21.56
6	20	II	9	1.83	8.00	4.00	12.90	2.17	6.36	1.00	21.62
7	20	III	9	0.13	0.43	5.00	12.85	0.76	3.98	2.00	21.55
8	20	IV	9	0.38	1.98	4.00	12.87	0.55	1.98	3.00	21.57
9	22	I	9	0.61	1.94	4.00	20.37	0.75	2.46	2.00	33.81
10	22	II	9	0.44	1.22	3.00	20.34	1.53	4.17	1.00	33.83
11	22	III	9	0.12	0.43	5.00	20.09	0.42	1.21	0.00	33.75
12	22	IV	9	0.72	1.93	1.00	20.13	0.97	2.51	2.00	33.78
13	24	I	9	0.38	1.75	5.00	30.25	0.72	2.19	2.00	50.26
14	24	II	9	2.24	10.00	1.00	30.23	1.77	3.87	1.00	50.36
15	24	III	9	0.14	1.01	6.00	30.16	0.44	0.95	1.00	50.16
16	24	IV	9	0.22	1.22	4.00	29.94	0.61	1.69	0.00	50.21

Table A.8. Summary of algorithm results (case III: Population size=4n & no. of generations =n^2).

				CGA_LOX				UGA_LOX			
Comb. No.	Problem Size (n)	Problem type	No. of problems solved	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)
1	18	I[a]	9	0.73	2.84	4.00	2.17	0.44	1.99	4.00	3.61
2	18	II[b]	9	0.87	2.49	3.00	2.18	1.27	9.15	3.00	3.64
3	18	III[c]	9	0.15	0.88	6.00	2.15	0.18	1.20	6.00	3.63
4	18	IV[d]	9	0.19	0.90	5.00	2.16	1.11	7.16	3.00	3.63
5	20	I	9	0.45	2.01	4.00	3.37	0.40	2.01	3.00	5.56
6	20	II	9	0.41	1.70	6.00	3.34	1.47	8.00	4.00	5.56
7	20	III	9	0.02	0.16	8.00	3.32	0.86	4.82	3.00	5.54
8	20	IV	9	0.37	2.11	4.00	3.32	0.37	0.92	2.00	5.56
9	22	I	9	0.66	2.18	5.00	4.98	0.99	1.94	1.00	8.26
10	22	II	9	0.96	5.49	4.00	4.97	0.90	3.66	1.00	8.26
11	22	III	9	0.19	0.68	5.00	4.92	0.55	2.07	4.00	8.25
12	22	IV	9	0.62	1.38	1.00	4.92	0.41	1.04	1.00	8.26
13	24	I	9	0.48	1.30	3.00	7.15	1.31	3.29	0.00	11.87
14	24	II	9	0.88	1.68	1.00	7.13	2.32	6.04	1.00	11.86
15	24	III	9	0.07	0.37	6.00	7.13	0.50	1.53	1.00	11.84
16	24	IV	9	0.30	0.83	2.00	7.07	0.30	0.81	2.00	11.85

[a] Type I: P_i was uniformly distributed between 1 & 5, and d_i was uniformly distributed between P_j & P_j + n.

[b] Type II: P_i was uniformly distributed between 1 & 5, and d_i was uniformly distributed between P_j & P_j + 1.5n.

[c] Type III: P_i was uniformly distributed between 1 &10, and d_i was uniformly distributed between P_j & P_j + n.

[d] Type IV: P_i was uniformly distributed between 1 &10, and d_i was uniformly distributed between P_j & P_j + 1.5n.

Table A.9. Summary of algorithm results (case IV: Population size=4n & no. of generations =$n^{2.5}$).

				CGA_LOX				UGA_LOX			
Comb. No.	Problem Size (n)	Problem type	No. of problems solved	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)
1	18	I[a]	9	0.73	2.84	4.00	9.14	0.44	1.99	4.00	15.23
2	18	II[b]	9	0.87	2.49	3.00	9.17	1.27	9.15	3.00	15.25
3	18	III[c]	9	0.15	0.88	6.00	9.12	0.18	1.20	6.00	15.22
4	18	IV[d]	9	0.19	0.90	5.00	9.13	1.11	7.16	3.00	15.24
5	20	I	9	0.45	2.01	4.00	14.80	0.40	2.01	3.00	24.60
6	20	II	9	0.41	1.70	6.00	14.76	1.47	8.00	4.00	24.66
7	20	III	9	0.02	0.16	8.00	14.71	0.86	4.82	3.00	24.58
8	20	IV	9	0.37	2.11	4.00	14.74	0.37	0.92	2.00	24.64
9	22	I	9	0.66	2.18	5.00	23.20	0.99	1.94	1.00	38.47
10	22	II	9	0.96	5.49	4.00	23.15	0.90	3.66	1.00	38.52
11	22	III	9	0.19	0.68	5.00	22.88	0.55	2.07	4.00	38.39
12	22	IV	9	0.62	1.38	1.00	22.96	0.41	1.04	1.00	38.43
13	24	I	9	0.48	1.30	3.00	34.81	1.31	3.29	0.00	57.78
14	24	II	9	0.88	1.68	1.00	34.75	2.32	6.04	1.00	57.85
15	24	III	9	0.07	0.37	6.00	34.70	0.50	1.53	1.00	57.71
16	24	IV	9	0.30	0.83	2.00	34.48	0.30	0.81	2.00	57.77

Table A.10. Summary of algorithm results (case I: Population size=3.5n & no. of generations =n^2).

				CGA_OBM_LOX				UGA_OBM_LOX			
Comb. No.	Problem Size (n)	Problem type	No. of problems solved	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)
1	18	I[a]	9	0.00	0.00	9.00	2.32	0.13	1.16	8.00	3.73
2	18	II[b]	9	0.03	0.30	8.00	2.30	0.00	0.00	9.00	3.72
3	18	III[c]	9	0.00	0.00	9.00	2.28	0.00	0.00	9.00	3.72
4	18	IV[d]	9	0.00	0.00	9.00	2.28	0.01	0.09	8.00	3.71
5	20	I	9	0.04	0.35	8.00	3.49	0.02	0.17	8.00	5.70
6	20	II	9	0.00	0.00	9.00	3.50	0.00	0.00	9.00	5.72
7	20	III	9	0.00	0.00	9.00	3.48	0.00	0.00	9.00	5.69
8	20	IV	9	0.03	0.25	8.00	3.48	0.03	0.25	8.00	5.70
9	22	I	9	0.02	0.20	8.00	5.13	0.00	0.00	9.00	8.41
10	22	II	9	0.33	2.44	6.00	5.14	0.35	2.44	5.00	8.41
11	22	III	9	0.05	0.43	8.00	5.09	0.07	0.43	7.00	8.39
12	22	IV	9	0.00	0.00	9.00	5.08	0.05	0.39	7.00	8.40
13	24	I	9	0.00	0.00	9.00	7.27	0.00	0.00	9.00	11.94
14	24	II	9	0.21	0.99	7.00	7.26	0.19	0.99	6.00	11.95
15	24	III	9	0.04	0.34	8.00	7.26	0.04	0.34	7.00	11.92
16	24	IV	9	0.00	0.00	9.00	7.20	0.02	0.20	8.00	11.94

[a] Type I: P_i was uniformly distributed between 1 & 5, and d_i was uniformly distributed between P_j & $P_j + n$.

[b] Type II: P_i was uniformly distributed between 1 & 5, and d_i was uniformly distributed between P_j & $P_j + 1.5n$.

[c] Type III: P_i was uniformly distributed between 1 &10, and d_i was uniformly distributed between P_j & $P_j + n$.

[d] Type IV: P_i was uniformly distributed between 1 &10, and d_i was uniformly distributed between P_j & $P_j + 1.5n$.

Table A.11. Summary of algorithm results (case II: Population size=3.5n & no. of generations =$n^{2.5}$).

Comb. No.	Problem Size (n)	Problem type	No. of problems solved	CGA_OBM_LOX				UGA_OBM_LOX			
				Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)
1	18	I[a]	9	0.00	0.00	9.00	9.64	0.00	0.00	9.00	15.67
2	18	II[b]	9	0.00	0.00	9.00	9.64	0.00	0.00	9.00	15.68
3	18	III[c]	9	0.00	0.00	9.00	9.60	0.00	0.00	9.00	15.62
4	18	IV[d]	9	0.00	0.00	9.00	9.60	0.00	0.00	9.00	15.63
5	20	I	9	0.00	0.00	9.00	15.53	0.01	0.09	8.00	25.36
6	20	II	9	0.00	0.00	9.00	15.52	0.00	0.00	9.00	25.44
7	20	III	9	0.00	0.00	9.00	15.49	0.02	0.10	7.00	25.34
8	20	IV	9	0.03	0.25	8.00	15.43	0.13	0.92	7.00	25.36
9	22	I	9	0.00	0.00	9.00	23.96	0.00	0.00	9.00	39.25
10	22	II	9	0.28	2.44	7.00	23.95	0.35	2.44	5.00	39.32
11	22	III	9	0.05	0.43	8.00	23.72	0.03	0.20	7.00	39.19
12	22	IV	9	0.00	0.00	9.00	23.73	0.04	0.39	8.00	39.22
13	24	I	9	0.00	0.00	9.00	35.47	0.00	0.00	9.00	58.26
14	24	II	9	0.21	0.99	7.00	35.43	0.13	0.99	7.00	58.34
15	24	III	9	0.00	0.00	9.00	35.39	0.00	0.00	9.00	58.17
16	24	IV	9	0.00	0.00	9.00	35.15	0.03	0.20	7.00	58.21

Table A.12. Summary of algorithm results (case III: Population size=4n & no. of generations =n^2).

Comb. No.	Problem Size (n)	Problem type	No. of problems solved	CGA_OBM_LOX				UGA_OBM_LOX			
				Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)
1	18	I[a]	9	0.00	0.00	9.00	2.63	0.08	0.71	8.00	4.27
2	18	II[b]	9	0.00	0.00	9.00	2.63	0.02	0.14	8.00	4.27
3	18	III[c]	9	0.00	0.00	9.00	2.64	0.00	0.00	9.00	4.25
4	18	IV[d]	9	0.00	0.00	9.00	2.61	0.00	0.00	9.00	4.26
5	20	I	9	0.04	0.35	8.00	4.02	0.00	0.00	9.00	6.51
6	20	II	9	0.00	0.00	9.00	3.99	0.00	0.00	9.00	6.53
7	20	III	9	0.00	0.00	9.00	3.98	0.02	0.16	8.00	6.49
8	20	IV	9	0.03	0.25	8.00	3.97	0.02	0.19	8.00	6.50
9	22	I	9	0.00	0.00	9.00	5.88	0.00	0.00	9.00	9.62
10	22	II	9	0.33	2.44	6.00	5.88	0.35	2.44	5.00	9.64
11	22	III	9	0.00	0.00	9.00	5.84	0.14	0.61	6.00	9.59
12	22	IV	9	0.04	0.39	8.00	5.85	0.05	0.39	7.00	9.62
13	24	I	9	0.00	0.00	9.00	8.41	0.00	0.00	9.00	13.72
14	24	II	9	0.21	0.99	7.00	8.38	0.02	0.15	8.00	13.76
15	24	III	9	0.00	0.00	9.00	8.36	0.00	0.00	9.00	13.71
16	24	IV	9	0.00	0.00	9.00	8.31	0.02	0.20	8.00	13.71

[a] Type I: P_i was uniformly distributed between 1 & 5, and d_i was uniformly distributed between P_j & $P_j + n$.

[b] Type II: P_i was uniformly distributed between 1 & 5, and d_i was uniformly distributed between P_j & $P_j + 1.5n$.

[c] Type III: P_i was uniformly distributed between 1 &10, and d_i was uniformly distributed between P_j & $P_j + n$.

[d] Type IV: P_i was uniformly distributed between 1 &10, and d_i was uniformly distributed between P_j & $P_j + 1.5n$.

Table A.13. Summary of algorithm results (case IV: Population size=4n & no. of generations =$n^{2.5}$).

				CGA_OBM_LOX				UGA_OBM_LOX			
Comb. No.	Problem Size (n)	Problem type	No. of problems solved	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)	Average %age error	Maximum %age error	No. of Optimal found	CPU Time needed (in seconds)
1	18	I^a	9	0.00	0.00	9.00	11.04	0.00	0.00	9.00	17.92
2	18	II^b	9	0.00	0.00	9.00	11.06	0.00	0.00	9.00	17.96
3	18	III^c	9	0.00	0.00	9.00	11.00	0.00	0.00	9.00	17.90
4	18	IV^d	9	0.00	0.00	9.00	11.01	0.00	0.00	9.00	17.94
5	20	I	9	0.00	0.00	9.00	17.71	0.00	0.00	9.00	28.87
6	20	II	9	0.00	0.00	9.00	17.70	0.00	0.00	9.00	28.96
7	20	III	9	0.00	0.00	9.00	17.66	0.00	0.00	9.00	28.86
8	20	IV	9	0.03	0.25	8.00	17.64	0.00	0.00	9.00	28.89
9	22	I	9	0.00	0.00	9.00	27.41	0.00	0.00	9.00	44.85
10	22	II	9	0.33	2.44	6.00	27.42	0.35	2.44	5.00	44.92
11	22	III	9	0.00	0.00	9.00	27.15	0.05	0.43	8.00	44.77
12	22	IV	9	0.00	0.00	9.00	27.19	0.04	0.39	8.00	44.83
13	24	I	9	0.00	0.00	9.00	40.91	0.00	0.00	9.00	66.96
14	24	II	9	0.11	0.99	8.00	40.83	0.23	1.96	7.00	67.06
15	24	III	9	0.00	0.00	9.00	40.81	0.00	0.00	9.00	66.89
16	24	IV	9	0.00	0.00	9.00	40.51	0.03	0.20	7.00	66.92

Table A.14. Summary of CGA_OBM and UGA_OBM algorithms results.

	Percentages of optimal solutions found		The no. of times the maximum deviation from the optimal was smaller for the CGA_OBM
Case No.	CGA_OBM	UGA_OBM	
I^a	23.61	15.97	9 out of 16
II^b	36.111	34.722	7 out of 16
III^c	27.083	18.056	11 out of 16
IV^d	44.444	31.944	8 out of 16

Table A.15. Summary of CGA_LOX and UGA_LOX algorithms results.

	Percentages of optimal solutions found		The no. of times the maximum deviation from the optimal was smaller for the CGA_LOX
Case No.	CGA_LOX	UGA_LOX	
I^a	42.361	15.278	10 out of 16
II^b	42.361	15.278	10 out of 16
III^c	46.528	27.083	9 out of 16
IV^d	46.528	27.083	9 out of 16

Table A.16. Summary of CGA_OBM_LOX and UGA_OBM_LOX algorithms results.

	Percentages of optimal solutions found		The no. of times the maximum deviation from the optimal was smaller for the CGA_OBM_LOX
Case No.	CGA_OBM_LOX	UGA_OBM_LOX	
I^a	92.361	87.5	4 out of 16
II^b	95.833	88.889	5 out of 16
III^c	94.444	89.583	5 out of 16
IV^d	96.528	93.056	4 out of 16

[a] Type I: P_i was uniformly distributed between 1 & 5, and d_i was uniformly distributed between P_j & $P_j + n$.
[b] Type II: P_i was uniformly distributed between 1 & 5, and d_i was uniformly distributed between P_j & $P_j + 1.5n$.
[c] Type III: P_i was uniformly distributed between 1 &10, and d_i was uniformly distributed between P_j & $P_j + n$.
[d] Type IV: P_i was uniformly distributed between 1 &10, and d_i was uniformly distributed between P_j & $P_j + 1.5n$.

APPENDIX B

RESULTS OF EXPERIMENT I

Table B.1. Summary of results obtained for problem FT6 and case I (LS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	187	55.15	55	0	51.5	0	OST(A)	0.000
6x6	133	55.53	55	0	51	0	LRPT(A)	0.000
6x6	109	52.51	55	0	52	0	COVERT(AS)	0.000
6x6	188	54.82	55	0	51.5	0	CR(A)	0.000
6x6	103	53.99	55	0	51	0	Biased-RANDOM(A)	0.000
6x6	137	53.33	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	188	61.9	55	0	51	0	JST(A)	0.000
6x6	188	52.4	56	1	51.5	1	RANDOM(A)	1.818
6x6	183	54.27	56	1	51.5	1	OST(A)	1.818
6x6	188	56.95	55	0	51	0	S/RPT(A)	0.000

Table B.2. Summary of results obtained for problem FT6 and case II (LO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	187	71.12	55	0	51.5	0	CR(A)	0.000
6x6	116	72.12	55	0	51.5	0	LRPT(A)	0.000
6x6	117	59.82	57	1	49.5	2	RANDOM(ND)	3.636
6x6	188	64.49	55	0	51	0	LRPT(A)	0.000
6x6	93	56.47	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	111	72.83	55	0	51.5	0	LRPT(A)	0.000
6x6	102	70.42	56	1	51.5	1	RANDOM(A)	1.818
6x6	96	66.79	55	0	51	0	RANDOM(A)	0.000
6x6	185	62.83	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	188	69.37	55	0	51.5	0	OST(A)	0.000

Table B.3. Summary of results obtained for problem FT6 and case III (LP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	112	71.62	55	0	51.5	0	RANDOM(A)	0.000
6x6	188	73.27	55	0	51	0	LRPT(A)	0.000
6x6	187	59.04	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	185	58.82	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	187	59.49	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	187	60.53	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	97	68.22	55	0	51.5	0	RANDOM(A)	0.000
6x6	98	69.53	55	0	51	0	RANDOM(A)	0.000
6x6	187	69.98	55	0	51.5	0	LRPT(A)	0.000
6x6	188	70.86	55	0	51.5	0	LRPT(A)	0.000

Table B.4. Summary of results obtained for problem FT6 and case IV (OS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	122	59.59	55	0	50.833	0	RANDOM(A)	0.000
6x6	187	57.84	57	1	49.5	2	COVERT(ND)	3.636
6x6	188	56.47	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	184	62.23	55	0	51	0	Biased-RANDOM(A)	0.000
6x6	98	58.05	55	0	50.833	0	Biased-RANDOM(A)	0.000
6x6	187	65.36	55	0	52	0	RANDOM(A)	0.000
6x6	188	55.37	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	188	55.97	57	1	49.5	2	RANDOM(ND)	3.636
6x6	187	60.91	55	0	52	0	RANDOM(A)	0.000
6x6	102	62.34	57	1	52.5	2	S/RPT(A)	3.636

Table B.5. Summary of results obtained for problem FT6 and case V (OO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	104	74.26	55	0	51	0	RANDOM(A)	0.000
6x6	186	95.52	57	1	52	2	OST(A)	3.636
6x6	84	61.57	55	0	51	0	RANDOM(A)	0.000
6x6	100	66.35	55	0	51	0	Biased-RANDOM(A)	0.000
6x6	113	63.99	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	187	60.59	57	1	49.5	2	SRPT(ND)	3.636
6x6	98	68.44	57	1	52	2	S/RPT(A)	3.636
6x6	188	60.47	57	1	49.5	2	RANDOM(ND)	3.636
6x6	188	62.56	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	188	69.76	55	0	51.5	0	Biased-RANDOM(A)	0.000

Table B.6. Summary of results obtained for problem FT6 and case VI (OP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	187	67.23	55	0	50.833	0	RANDOM(A)	0.000
6x6	65	66.79	55	0	52	0	Biased-RANDOM(A)	0.000
6x6	187	54.93	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	187	80.74	57	1	52.5	2	LRPT(A)	3.636
6x6	185	74.92	57	1	52	2	LRPT(A)	3.636
6x6	186	56.63	57	1	49.5	2	SRPT(ND)	3.636
6x6	82	144.62	57	1	52.5	2	CR(A)	3.636
6x6	187	61.35	57	1	49.5	2	RANDOM(ND)	3.636
6x6	188	56.41	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	74	86.4	57	1	52	2	LRPT(A)	3.636

Table B.7. Summary of results obtained for problem FT6 and case VII (PS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	186	55.25	55	0	51	0	RANDOM(A)	0.000
6x6	186	55.75	55	0	51.5	0	LRPT(A)	0.000
6x6	188	51.36	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	188	57.39	55	0	51.5	0	Biased-RANDOM(A)	0.000
6x6	132	52.51	55	0	51.5	0	JST(A)	0.000
6x6	188	51.58	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	187	50.26	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	188	52.4	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	147	51.74	55	0	51.5	0	RANDOM(A)	0.000
6x6	188	50.59	57	1	49.667	2	LAWINQ(ND)	3.636

Table B.8. Summary of results obtained for problem FT6 and case VIII (PO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	185	63.16	55	0	51	0	JST(A)	0.000
6x6	126	60.53	55	0	50.833	0	S/RPT(A)	0.000
6x6	187	53.99	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	186	58.66	55	0	51	0	LRPT(A)	0.000
6x6	176	51.03	57	1	49.5	2	RANDOM(ND)	3.636
6x6	125	68.66	55	0	51.5	0	S/RPT(A)	0.000
6x6	137	63.38	55	0	51.5	0	S/RPT(A)	0.000
6x6	187	58.49	57	1	49.667	2	RANDOM(ND)	3.636
6x6	123	66.3	55	0	51.5	0	CR(A)	0.000
6x6	188	58.22	55	0	51	0	Biased-RANDOM(A)	0.000

Table B.9. Summary of results obtained for problem FT6 and case IX (PP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	121	61.57	55	0	51.5	0	OST(A)	0.000
6x6	185	62.46	55	0	51	0	Biased-RANDOM(A)	0.000
6x6	187	53.12	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	187	62.67	55	0	51.5	0	Biased-RANDOM(A)	0.000
6x6	188	56.29	55	0	50.833	0	RANDOM(A)	0.000
6x6	115	60.48	55	0	51.5	0	LRPT(A)	0.000
6x6	188	61.13	55	0	51	0	JST(A)	0.000
6x6	186	62.95	55	0	51.5	0	Biased-RANDOM(A)	0.000
6x6	126	57.67	55	0	51	0	LRPT(A)	0.000
6x6	187	53.55	57	1	49.5	2	RANDOM(ND)	3.636

Table B.10. Summary of results obtained for problem FT10 and case I (LS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	377	533.05	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	8	515.59	957	5	875.4	86	RANDOM(ND)	2.903
10x10	444	563.43	964	4	895.1	96	RANDOM(ND)	3.656
10x10	443	546.29	984	3	814.9	110	RANDOM(ND)	5.806
10x10	444	561.01	968	3	839.3	53	RANDOM(ND)	4.086
10x10	405	543.71	968	3	839.3	53	RANDOM(ND)	4.086
10x10	444	566.01	964	4	895.1	96	RANDOM(ND)	3.656
10x10	402	577.26	964	4	895.1	96	RANDOM(ND)	3.656
10x10	444	519.6	962	5	885.1	99	Biased-RANDOM(ND)	3.441
10x10	442	552.61	964	4	895.1	96	Biased-RANDOM(ND)	3.656

Table B.11. Summary of results obtained for problem FT10 and case II (LO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	444	568.36	964	4	895.1	96	RANDOM(ND)	3.656
10x10	444	551.67	968	3	839.3	53	WSPT+WOST(ND)	4.086
10x10	444	588.64	964	4	895.1	96	OCR(ND)	3.656
10x10	444	549.42	964	4	887.9	96	RANDOM(ND)	3.656
10x10	420	537.94	960	4	819	92	RANDOM(ND)	3.226
10x10	444	598.36	968	3	819.4	99	WSPT+WOST(ND)	4.086
10x10	443	590.06	964	4	887.9	96	RANDOM(ND)	3.656
10x10	444	575.18	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	444	554.97	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	444	562.43	964	4	895.1	96	RANDOM(ND)	3.656

Table B.12. Summary of results obtained for problem FT10 and case III (LP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	444	572	964	4	895.1	96	LRPT(ND)	3.656
10x10	444	551.29	968	3	839.3	53	JST(ND)	4.086
10x10	444	550.13	968	3	839.3	53	ODD(ND)	4.086
10x10	444	550.58	964	4	895.1	96	RANDOM(ND)	3.656
10x10	444	584.13	976	4	862.3	131	Biased-RANDOM(ND)	4.946
10x10	342	569.79	964	4	887.9	96	WSPT+WOST(ND)	3.656
10x10	444	576.72	964	4	895.1	96	JST(ND)	3.656
10x10	442	539.97	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	443	558.04	964	4	895.1	96	WSPT+WOST(ND)	3.656
10x10	395	541.29	964	4	895.1	96	Biased-RANDOM(ND)	3.656

Table B.13. Summary of results obtained for problem FT10 and case IV (OS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	377	542.28	964	4	895.1	96	RANDOM(ND)	3.656
10x10	444	562.17	964	4	892.5	96	Biased-RANDOM(ND)	3.656
10x10	444	544.47	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	289	559.36	964	4	895.1	96	MODD(ND)	3.656
10x10	444	558.2	964	4	887.9	96	RANDOM(ND)	3.656
10x10	444	546.73	964	4	887.9	96	RANDOM(ND)	3.656
10x10	381	579.35	968	3	836.2	50	ATC(ND)	4.086
10x10	442	557.71	964	4	895.1	96	RANDOM(ND)	3.656
10x10	261	577.65	968	4	865.9	93	RANDOM(ND)	4.086
10x10	374	561.84	968	3	836.2	50	RANDOM(ND)	4.086

Table B.14. Summary of results obtained for problem FT10 and case V (OO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	344	606.38	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	444	578.25	962	5	885.1	99	Biased-RANDOM(ND)	3.441
10x10	444	568.75	968	3	839.3	53	ATC(ND)	4.086
10x10	444	558.26	964	4	895.1	96	RANDOM(ND)	3.656
10x10	327	545.24	964	4	895.1	96	RANDOM(ND)	3.656
10x10	380	590.39	968	3	839.3	53	RANDOM(ND)	4.086
10x10	353	557.94	964	4	895.1	96	ATC(ND)	3.656
10x10	444	558.43	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	375	576.72	964	4	895.1	96	WSPT+WOST(ND)	3.656
10x10	306	542.12	960	3	821.6	61	RANDOM(ND)	3.226

Table B.15. Summary of results obtained for problem FT10 and case VI (OP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	443	578.75	964	4	895.1	96	RANDOM(ND)	3.656
10x10	444	569.36	978	4	911	137	RANDOM(ND)	5.161
10x10	444	534.81	968	3	836.2	50	ATC(ND)	4.086
10x10	444	552.55	964	4	892.5	96	RANDOM(ND)	3.656
10x10	444	548.49	968	3	839.3	53	RANDOM(ND)	4.086
10x10	444	577.93	975	3	827.4	120	Biased-RANDOM(ND)	4.839
10x10	349	557.71	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	444	566.72	968	3	836.2	50	SPT(ND)	4.086
10x10	335	539.15	960	3	821.6	61	Biased-RANDOM(ND)	3.226
10x10	443	548.81	960	3	829.8	61	SPT(ND)	3.226

Table B.16. Summary of results obtained for problem FT10 and case VII (PS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	365	630.44	964	4	895.1	96	RANDOM(ND)	3.656
10x10	442	549.03	960	3	821.6	61	WSPT+WOST(ND)	3.226
10x10	443	551.23	956	4	869.8	59	Biased-RANDOM(ND)	2.796
10x10	444	542.28	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	444	541.84	964	4	887.9	96	RANDOM(ND)	3.656
10x10	374	576.93	968	3	836.2	50	WSPT+WOST(ND)	4.086
10x10	444	547.06	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	444	645.87	964	4	895.1	96	RANDOM(ND)	3.656
10x10	444	572.82	964	4	892.5	96	RANDOM(ND)	3.656
10x10	444	543.43	968	3	836.2	50	Biased-RANDOM(ND)	4.086

Table B.17. Summary of results obtained for problem FT10 and case VIII (PO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	396	571.61	964	4	895.1	96	RANDOM(ND)	3.656
10x10	379	586.66	968	3	825	97	ATC(ND)	4.086
10x10	444	534.15	968	3	836.2	50	Biased-RANDOM(ND)	4.086
10x10	444	558.98	964	4	895.1	96	RANDOM(ND)	3.656
10x10	444	552.93	964	4	887.9	96	COVERT(ND)	3.656
10x10	444	557.77	966	6	867.5	140	WSPT+WOST(ND)	3.871
10x10	444	556.94	966	6	867.5	140	Biased-RANDOM(ND)	3.871
10x10	444	618.46	964	4	895.1	96	WSPT+WOST(ND)	3.656
10x10	322	563.75	964	4	895.1	96	MODD(ND)	3.656
10x10	444	568.04	964	4	888.8	96	RANDOM(ND)	3.656

Table B.18. Summary of results obtained for problem FT10 and case IX (PP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	444	553.2	964	4	895.1	96	SPT(ND)	3.656
10x10	443	563.38	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	438	550.57	962	5	885.1	99	RANDOM(ND)	3.441
10x10	444	563.04	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	444	573.86	964	4	895.1	96	OST(ND)	3.656
10x10	351	593.25	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	443	572.38	964	4	895.1	96	RANDOM(ND)	3.656
10x10	444	572.27	964	4	895.1	96	WSPT+WOST(ND)	3.656
10x10	444	564.09	979	4	900.1	156	Biased-RANDOM(ND)	5.269
10x10	353	594.13	960	3	821.6	61	RANDOM(ND)	3.226

Table B.19. Summary of results obtained for problem FT20 and case I (LS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	35	1657.76	1185	2	868.75	37	SPT(ND)	1.717
20x5	27	1626.23	1182	2	861.65	31	OCR(ND)	1.459
20x5	444	1660.62	1182	2	912.55	22	RANDOM(ND)	1.459
20x5	366	1584.27	1193	2	872.15	45	SRT(ND)	2.403
20x5	444	1613.93	1178	2	881.4	26	SPT(ND)	1.116
20x5	297	1727.63	1182	2	901.9	29	SPT(ND)	1.459
20x5	266	1576.25	1182	1	857.55	17	SPT(ND)	1.459
20x5	443	1664.14	1178	2	875.95	18	Biased-RANDOM(ND)	1.116
20x5	444	1705.16	1180	2	921.5	22	SRT(ND)	1.288
20x5	374	1784.37	1191	2	896.3	52	SPT(ND)	2.232

Table B.20. Summary of results obtained for problem FT20 and case II (LO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	442	1658.58	1182	2	886.35	34	EDD(ND)	1.459
20x5	7	1739.66	1178	3	911.3	33	LWR(ND)	1.116
20x5	443	1594.22	1197	2	881.8	41	SRT(ND)	2.747
20x5	360	1642.11	1182	1	860.4	17	Biased-RANDOM(ND)	1.459
20x5	444	1911.9	1182	2	905.4	34	SPT(ND)	1.459
20x5	444	1723.18	1182	1	867.65	17	Biased-RANDOM(ND)	1.459
20x5	444	1742.95	1203	2	917.3	43	SPT(ND)	3.262
20x5	444	2318.18	1185	2	903.75	38	Biased-RANDOM(ND)	1.717
20x5	440	1755.69	1203	2	927.8	63	SPT(ND)	3.262
20x5	439	1702.64	1190	1	863.35	25	Biased-RANDOM(ND)	2.146

Table B.21. Summary of results obtained for problem FT20 and case III (LP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	171	1778.82	1184	2	877.15	29	Biased-RANDOM(ND)	1.631
20x5	444	1578.12	1190	2	873.05	48	MODD(ND)	2.146
20x5	444	1765.31	1210	2	902.05	57	SPT(ND)	3.863
20x5	444	1658.47	1194	2	908.2	42	SPT(ND)	2.489
20x5	444	1680.39	1182	2	877.45	32	SPT(ND)	1.459
20x5	444	1531.11	1182	1	829.95	17	Biased-RANDOM(ND)	1.459
20x5	441	1633.38	1193	2	898	44	SPT(ND)	2.403
20x5	341	1706.76	1178	2	871.55	18	Biased-RANDOM(ND)	1.116
20x5	443	1729.72	1193	2	871.05	40	RANDOM(ND)	2.403
20x5	38	1599.76	1182	1	841.55	17	MDD(ND)	1.459

Table B.22. Summary of results obtained for problem FT20 and case IV (OS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	5	1709.23	1180	1	909.3	15	Biased-RANDOM(ND)	1.288
20x5	243	1729.98	1185	2	928.5	39	Biased-RANDOM(ND)	1.717
20x5	319	1726.2	1184	2	898.8	37	MDD(ND)	1.631
20x5	235	1687.21	1202	3	912.95	73	Biased-RANDOM(ND)	3.176
20x5	234	1635.25	1182	2	856.15	24	Biased-RANDOM(ND)	1.459
20x5	343	1621.51	1203	2	937.4	74	SPT(ND)	3.262
20x5	1	1812.54	1178	1	872.55	13	EDD(ND)	1.116
20x5	4	1689.18	1196	3	951.65	69	Biased-RANDOM(ND)	2.661
20x5	444	1785.19	1186	2	922.3	41	SPT(ND)	1.803
20x5	282	1856.7	1198	3	933.95	69	EDD(ND)	2.833

Table B.23. Summary of results obtained for problem FT20 and case V (OO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	75	1851.7	1180	2	879.45	25	Biased-RANDOM(ND)	1.288
20x5	444	1745.7	1188	2	871.05	38	SPT(ND)	1.974
20x5	444	1705.21	1193	2	894.25	42	SPT(ND)	2.403
20x5	77	1646.44	1182	2	913.9	19	TWORK(ND)	1.459
20x5	21	1781.9	1194	2	927.05	50	EDD(ND)	2.489
20x5	444	1797.44	1178	2	897	18	Biased-RANDOM(ND)	1.116
20x5	444	1849.73	1185	2	913.55	31	EDD(ND)	1.717
20x5	315	1648.42	1185	2	936.85	37	Biased-RANDOM(ND)	1.717
20x5	443	1784.2	1193	2	850.05	42	SPT(ND)	2.403
20x5	7	1727.24	1182	2	895.55	22	Biased-RANDOM(ND)	1.459

Table B.24. Summary of results obtained for problem FT20 and case VI (OP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	444	1723.39	1180	2	890.3	26	Biased-RANDOM(ND)	1.288
20x5	372	1716.64	1203	2	907.5	66	SPT(ND)	3.262
20x5	1	1752.29	1194	2	872.4	41	SPT(ND)	2.489
20x5	114	1680.33	1184	2	902.3	30	Biased-RANDOM(ND)	1.631
20x5	272	1598.66	1191	2	876.1	44	SPT(ND)	2.232
20x5	4	1726.2	1182	2	886.75	24	SPT(ND)	1.459
20x5	195	1733.67	1190	3	889.15	43	TWORK(ND)	2.146
20x5	444	1821	1193	1	896.95	28	SPT(ND)	2.403
20x5	444	1700	1198	2	901.35	47	SPT(ND)	2.833
20x5	317	1763.82	1200	1	878	35	SPT(ND)	3.004

Table B.25. Summary of results obtained for problem FT20 and case VII (PS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	318	1703.84	1185	2	906.4	31	COVERT(ND)	1.717
20x5	350	1746.9	1178	1	862.15	13	SPT(ND)	1.116
20x5	23	1747.51	1182	2	920	33	SPT(ND)	1.459
20x5	257	1725.15	1184	1	871.25	19	EDD(ND)	1.631
20x5	186	1755.25	1178	2	882.55	21	SPT(ND)	1.116
20x5	219	1701.26	1197	1	900.3	32	MDD(ND)	2.747
20x5	315	1687.64	1194	2	924	42	SRT(ND)	2.489
20x5	444	1695.88	1182	2	857.75	21	Biased-RANDOM(ND)	1.459
20x5	39	1673.09	1182	2	899.45	30	Biased-RANDOM(ND)	1.459
20x5	443	1666.94	1197	2	909	59	SPT(ND)	2.747

Table B.26. Summary of results obtained for problem FT20 and case VIII (PO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	7	1617.94	1178	2	917.75	21	LWR(ND)	1.116
20x5	444	1638.37	1203	1	910	38	EDD(ND)	3.262
20x5	444	1716.8	1194	2	896.2	48	Biased-RANDOM(ND)	2.489
20x5	444	1758.66	1190	2	899.85	35	LWR(ND)	2.146
20x5	1	1759.37	1197	4	906.85	88	COVERT(ND)	2.747
20x5	1	1661.17	1194	2	909.6	46	COVERT(ND)	2.489
20x5	442	1815.07	1184	3	906.6	48	SPT(ND)	1.631
20x5	444	1866.37	1178	1	873.85	13	Biased-RANDOM(ND)	1.116
20x5	419	1715.87	1191	2	890.9	50	SPT(ND)	2.232
20x5	240	1916.35	1191	2	905.15	46	SPT(ND)	2.232

Table B.27. Summary of results obtained for problem FT20 and case IX (PP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	4	1740.69	1178	2	889.25	18	LWR(ND)	1.116
20x5	17	1745.43	1193	3	868.25	46	SPT(ND)	2.403
20x5	444	1748.06	1178	2	876.5	25	SPT(ND)	1.116
20x5	443	1662.98	1181	2	892.15	18	SRT(ND)	1.373
20x5	442	1686.32	1178	2	876.6	21	SPT(ND)	1.116
20x5	338	2329.77	1203	1	895.55	38	SPT(ND)	3.262
20x5	282	1885.32	1191	2	883.85	45	SPT(ND)	2.232
20x5	350	1670.83	1182	2	894	34	Biased-RANDOM(ND)	1.459
20x5	23	1744.87	1185	2	919.95	31	Biased-RANDOM(ND)	1.717
20x5	335	1706.75	1182	1	873.85	17	Biased-RANDOM(ND)	1.459

Table B.28. Summary of results obtained for problem LA25 and case I (LS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	8	2153.03	1015	7	916.8	174	RANDOM(ND)	3.889
15x10	3	1877.08	1000	3	902.333	51	JST(ND)	2.354
15x10	644	1882.4	1003	5	928.533	93	Biased-RANDOM(ND)	2.661
15x10	306	1988.41	1004	4	932.067	88	WSPT+WOST(ND)	2.764
15x10	398	2131.55	1003	5	933.667	87	Biased-RANDOM(ND)	2.661

Table B.29. Summary of results obtained for problem LA25 and case II (LO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	11	1906.63	1003	5	933.667	86	RANDOM(ND)	2.661
15x10	642	1760.2	1007	5	932.467	84	Biased-RANDOM(ND)	3.071
15x10	1	1835.39	1003	5	930.867	86	RANDOM(ND)	2.661
15x10	643	1834.4	1003	5	930.867	87	Biased-RANDOM(ND)	2.661
15x10	33	1748.39	1003	5	933.667	86	Biased-RANDOM(ND)	2.661

Table B.30. Summary of results obtained for problem LA25 and case III (LP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	644	1797.55	1007	5	932.467	84	WSPT+WOST(ND)	3.071
15x10	644	1668.8	1004	4	901	67	OST(ND)	2.764
15x10	166	1778.37	1003	5	935.8	88	Biased-RANDOM(ND)	2.661
15x10	1	1680.83	1003	5	932.333	93	JST(ND)	2.661
15x10	19	1787.33	1007	6	938.333	127	Biased-RANDOM(ND)	3.071

Table B.31. Summary of results obtained for problem LA25 and case IV (OS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	197	2068.6	1012	4	925.467	112	Biased-RANDOM(ND)	3.582
15x10	367	1893.55	1002	5	936.667	64	RANDOM(ND)	2.559
15x10	308	2011.76	1014	6	921	129	Biased-RANDOM(ND)	3.787
15x10	179	1974.57	1002	4	927.133	63	Biased-RANDOM(ND)	2.559
15x10	643	1977.59	1003	5	932.4	93	ATC(ND)	2.661

Table B.32. Summary of results obtained for problem LA25 and case V (OO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	435	1869.61	1007	6	937.333	104	Biased-RANDOM(ND)	3.071
15x10	8	1824.41	1002	7	941.133	118	RANDOM(ND)	2.559
15x10	447	1930.25	1010	6	939.8	94	RANDOM(ND)	3.378
15x10	341	1841.66	1007	6	937.467	139	Biased-RANDOM(ND)	3.071
15x10	644	1699.29	1007	5	925.667	105	Biased-RANDOM(ND)	3.071

Table B.33. Summary of results obtained for problem LA25 and case VI (OP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	460	1879.17	1007	5	938.733	84	Biased-RANDOM(ND)	3.071
15x10	446	1890.32	1014	7	934.2	147	Biased-RANDOM(ND)	3.787
15x10	644	1817.32	1003	5	930.867	87	RANDOM(ND)	2.661
15x10	80	1763.93	1007	5	928.867	84	Biased-RANDOM(ND)	3.071
15x10	605	1857.36	1012	6	929.333	127	MODD(ND)	3.582

Table B.34. Summary of results obtained for problem LA25 and case VII (PS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	252	1899.05	1007	5	925.667	105	JST(ND)	3.071
15x10	315	1867.68	1007	5	930.733	84	Biased-RANDOM(ND)	3.071
15x10	3	1972.7	1003	5	936.067	89	RANDOM(ND)	2.661
15x10	328	1936.62	1017	6	944.933	176	Biased-RANDOM(ND)	4.094
15x10	19	1902.07	1011	7	931.2	158	Biased-RANDOM(ND)	3.480

Table B.35. Summary of results obtained for problem LA25 and case VIII (PO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	49	1842.8	1003	5	927.467	91	RANDOM(ND)	2.661
15x10	66	1831.82	1002	6	937.733	92	RANDOM(ND)	2.559
15x10	394	1845.22	1007	5	934.733	84	Biased-RANDOM(ND)	3.071
15x10	644	1781.78	1011	4	909.867	83	Biased-RANDOM(ND)	3.480
15x10	40	1860.49	1007	5	924.2	78	RANDOM(ND)	3.071

Table B.36. Summary of results obtained for problem LA25 and case IX (PP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	448	1807.11	1007	6	928.2	113	Biased-RANDOM(ND)	3.071
15x10	644	1803.64	1007	5	931.133	84	Biased-RANDOM(ND)	3.071
15x10	492	1742.35	1017	6	939.267	165	MWR(ND)	4.094
15x10	1	1738.5	1015	5	906.133	128	Biased-RANDOM(ND)	3.889
15x10	441	1852.14	1022	4	929	162	Biased-RANDOM(ND)	4.606

Table B.37. Summary of results obtained for problem LA29 and case I (LS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	21	5663.7	1203	8	1114	233	LRPT(ND)	4.337
20x10	364	5550.55	1218	11	1128.35	446	WSPT+WOST(ND)	5.637
20x10	330	5504.19	1214	8	1105.2	215	RANDOM(ND)	5.291
20x10	1	5325.03	1217	8	1108.25	240	MWR(ND)	5.551
20x10	1	5321.51	1220	10	1104.05	464	JST(ND)	5.811

Table B.38. Summary of results obtained for problem LA29 and case II (LO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	37	4972.35	1202	10	1131.2	282	JST(ND)	4.250
20x10	1	4916.98	1191	6	1067.55	186	ODD(ND)	3.296
20x10	8	4810.48	1218	7	1093.05	311	RANDOM(ND)	5.637
20x10	522	5110.82	1229	7	1088.9	320	LRPT(ND)	6.592
20x10	15	4956.8	1212	7	1098.4	257	ATC(ND)	5.117

Table B.39. Summary of results obtained for problem LA29 and case III (LP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	4	5567.36	1205	7	1120	220	RANDOM(ND)	4.510
20x10	834	5754.16	1217	9	1141.15	354	LRPT(ND)	5.551
20x10	190	5124.17	1205	7	1071.65	164	S/RPT(ND)	4.510
20x10	63	4751.66	1224	10	1135.3	420	LRPT(ND)	6.158
20x10	14	4729.91	1212	5	1073.75	166	JST(ND)	5.117

Table B.40. Summary of results obtained for problem LA29 and case IV (OS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	19	5506.83	1219	6	1106.2	272	CR(ND)	5.724
20x10	1	5294.54	1225	7	1049.5	366	Biased-RANDOM(ND)	6.245
20x10	844	5408.63	1224	10	1125.55	473	LRPT(ND)	6.158
20x10	3	5288.89	1229	12	1108.75	519	WSPT+WOST(ND)	6.592
20x10	8	5586.09	1219	9	1115.7	360	RANDOM(ND)	5.724

Table B.41. Summary of results obtained for problem LA29 and case V (OO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	84	5047.49	1200	7	1086.2	227	CR(ND)	4.076
20x10	283	5304.76	1216	8	1084.15	285	RANDOM(ND)	5.464
20x10	571	5046.5	1219	10	1120.9	398	ODD(ND)	5.724
20x10	844	5014.97	1226	9	1116.95	431	Biased-RANDOM(ND)	6.331
20x10	22	5439.43	1202	5	1090.5	162	LRPT(ND)	4.250

Table B.42. Summary of results obtained for problem LA29 and case VI (OP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	842	5613.22	1227	9	1135.85	474	OST(ND)	6.418
20x10	492	5074.84	1235	9	1127.6	488	RANDOM(ND)	7.112
20x10	556	5244.95	1218	8	1107.75	344	RANDOM(ND)	5.637
20x10	843	5038.87	1224	10	1127.95	472	OST(ND)	6.158
20x10	9	5147.29	1206	5	1094.1	190	S/RPT(ND)	4.597

Table B.43. Summary of results obtained for problem LA29 and case VII (PS).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	1	5820.07	1212	8	1115.1	276	RANDOM(ND)	5.117
20x10	9	5494.59	1213	10	1129.9	349	WSPT+WOST(ND)	5.204
20x10	1	5546.1	1214	9	1084.45	361	RANDOM(ND)	5.291
20x10	2	5514.58	1216	10	1095	380	LRPT(ND)	5.464
20x10	3	5511.5	1207	7	1090.1	194	S/RPT(ND)	4.683

Table B.44. Summary of results obtained for problem LA29 and case VIII (PO).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	3	4705.85	1217	11	1129.45	445	JST(ND)	5.551
20x10	20	4961.64	1209	8	1106.9	288	RANDOM(ND)	4.857
20x10	198	5148.56	1214	7	1097.95	222	RANDOM(ND)	5.291
20x10	486	5090.6	1227	7	1114.15	378	LRPT(ND)	6.418
20x10	844	4805.33	1224	10	1120.3	384	LRPT(ND)	6.158

Table B. 45. Summary of results obtained for problem LA29 and case IX (PP).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	506	5648.31	1214	6	1107.6	236	OST(ND)	5.291
20x10	88	4749.79	1212	10	1096.1	348	RANDOM(ND)	5.117
20x10	225	4938.57	1217	9	1108.75	389	RANDOM(ND)	5.551
20x10	55	4960.6	1214	8	1102.35	294	RANDOM(ND)	5.291
20x10	16	4838.61	1210	9	1101.1	325	JST(ND)	4.944

APPENDIX C

RESULTS OF EXPERIMENT II

Table C.1. Summary of results obtained for problem FT6 and population size =44+4nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	187	71.12	55	0	51.5	0	CR(A)	0.000
6x6	116	72.12	55	0	51.5	0	LRPT(A)	0.000
6x6	117	59.82	57	1	49.5	2	RANDOM(ND)	3.636
6x6	188	64.49	55	0	51	0	LRPT(A)	0.000
6x6	93	56.47	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	111	72.83	55	0	51.5	0	LRPT(A)	0.000
6x6	102	70.42	56	1	51.5	1	RANDOM(A)	1.818
6x6	96	66.79	55	0	51	0	RANDOM(A)	0.000
6x6	185	62.83	57	1	49.5	2	Biased-RANDOM(ND)	3.636
6x6	188	69.37	55	0	51.5	0	OST(A)	0.000

Table C.2. Summary of results obtained for problem FT6 and population size =44+nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	79	29.22	55	0	51	0	OST(A)	0.000
6x6	60	27.73	55	0	51.5	0	COVERT(AS)	0.000
6x6	34	24.39	55	0	52	0	RANDOM(A)	0.000
6x6	73	22.35	58	4	53.667	9	RANDOM(ND)	5.455
6x6	33	24.55	56	1	51.5	1	LRPT(A)	1.818
6x6	80	27.41	55	0	51	0	CR(A)	0.000
6x6	79	28.29	56	1	52	1	OST(A)	1.818
6x6	80	28.79	55	0	51.5	0	LRPT(A)	0.000
6x6	78	28.72	55	0	51.5	0	LRPT(A)	0.000
6x6	30	40.81	57	1	52	2	S/RPT(A)	3.636

Table C.3. Summary of results obtained for problem FT6 and population size =44+2nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	77	49	55	0	51	0	ODD(A)	0.000
6x6	81	38.5	55	0	51	0	Biased-RANDOM(A)	0.000
6x6	116	32.46	57	1	49.5	2	RANDOM(ND)	3.636
6x6	116	37.79	56	1	52	1	LRPT(A)	1.818
6x6	68	35.37	57	1	49.5	2	RANDOM(ND)	3.636
6x6	81	40.92	55	0	51	0	COVERT(AS)	0.000
6x6	116	43.28	55	0	51.5	0	Biased-RANDOM(A)	0.000
6x6	13	64.1	57	2	53	3	OST(A)	3.636
6x6	65	38.23	55	0	51	0	Biased-RANDOM(A)	0.000
6x6	99	38.23	55	0	51	0	ODD(A)	0.000

Table C.4. Summary of results obtained for problem FT10 and population size =44+4nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	444	568.36	964	4	895.1	96	RANDOM(ND)	3.656
10x10	444	551.67	968	3	839.3	53	WSPT+WOST(ND)	4.086
10x10	444	588.64	964	4	895.1	96	OCR(ND)	3.656
10x10	444	549.42	964	4	887.9	96	RANDOM(ND)	3.656
10x10	420	537.94	960	4	819	92	RANDOM(ND)	3.226
10x10	444	598.36	968	3	819.4	99	WSPT+WOST(ND)	4.086
10x10	443	590.06	964	4	887.9	96	RANDOM(ND)	3.656
10x10	444	575.18	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	444	554.97	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	444	562.43	964	4	895.1	96	RANDOM(ND)	3.656

Table C.5. Summary of results obtained for problem FT10 and population size =44+nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	144	178.61	976	5	880.6	170	OST(ND)	4.946
10x10	119	171.64	978	4	900.8	152	OST(ND)	5.161
10x10	144	187.13	977	5	858.9	105	Biased-RANDOM(ND)	5.054
10x10	131	181.42	976	4	853.8	123	Biased-RANDOM(ND)	4.946
10x10	144	180.76	981	5	896.7	185	RANDOM(ND)	5.484
10x10	128	167.96	974	4	900.1	136	WSPT+WOST(ND)	4.731
10x10	144	175.76	968	3	825	97	ATC(ND)	4.086
10x10	144	174.82	968	3	836.2	50	SPT(ND)	4.086
10x10	114	177.03	976	5	891.8	116	COVERT(ND)	4.946
10x10	144	193.33	987	4	872.6	133	COVERT(ND)	6.129

Table C.6. Summary of results obtained for problem FT10 and population size =44+2nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	244	298.35	964	4	895.1	96	RANDOM(ND)	3.656
10x10	244	318.3	968	3	839.3	53	SPT(ND)	4.086
10x10	244	314.78	960	3	825.4	61	Biased-RANDOM(ND)	3.226
10x10	244	362.56	968	4	912.2	116	Biased-RANDOM(ND)	4.086
10x10	200	324.33	985	5	878.8	192	OST(ND)	5.914
10x10	244	330.6	975	4	906.1	140	Biased-RANDOM(ND)	4.839
10x10	244	342.02	973	4	833.4	149	Biased-RANDOM(ND)	4.624
10x10	244	329	964	4	895.1	96	MODD(ND)	3.656
10x10	244	314.45	964	4	895.1	96	Biased-RANDOM(ND)	3.656
10x10	244	325.38	979	3	841.5	132	LRPT(ND)	5.269

Table C.7. Summary of results obtained for problem FT20 and population size =44+4nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	442	1658.58	1182	2	886.35	34	EDD(ND)	1.459
20x5	7	1739.66	1178	3	911.3	33	LWR(ND)	1.116
20x5	443	1594.22	1197	2	881.8	41	SRT(ND)	2.747
20x5	360	1642.11	1182	1	860.4	17	Biased-RANDOM(ND)	1.459
20x5	444	1911.9	1182	2	905.4	34	SPT(ND)	1.459
20x5	444	1723.18	1182	1	867.65	17	Biased-RANDOM(ND)	1.459
20x5	444	1742.95	1203	2	917.3	43	SPT(ND)	3.262
20x5	444	2318.18	1185	2	903.75	38	Biased-RANDOM(ND)	1.717
20x5	440	1755.69	1203	2	927.8	63	SPT(ND)	3.262
20x5	439	1702.64	1190	1	863.35	25	Biased-RANDOM(ND)	2.146

Table C.8. Summary of results obtained for problem FT20 and population size =44+nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	144	598.2	1193	2	878.05	42	SPT(ND)	2.403
20x5	114	516.47	1193	2	858.2	40	SPT(ND)	2.403
20x5	144	525.3	1198	2	871.35	59	SPT(ND)	2.833
20x5	2	512.79	1198	2	907.05	57	EDD(ND)	2.833
20x5	144	556.45	1184	2	926.25	31	COVERT(ND)	1.631
20x5	144	540.41	1204	4	875.3	98	SPT(ND)	3.348
20x5	144	548.49	1193	2	893.45	40	SPT(ND)	2.403
20x5	144	544.31	1182	2	929.6	23	LWR(ND)	1.459
20x5	44	622.19	1202	3	909.15	79	SPT(ND)	3.176
20x5	115	536.18	1193	2	891.15	40	SPT(ND)	2.403

Table C.9. Summary of results obtained for problem FT20 and population size =44+2nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	244	974.16	1193	1	885.35	28	SPT(ND)	2.403
20x5	33	917.26	1193	1	880.45	28	SPT(ND)	2.403
20x5	243	950.65	1182	2	924.15	21	MODD(ND)	1.459
20x5	184	881.78	1178	2	872.7	20	Biased-RANDOM(ND)	1.116
20x5	223	971.52	1193	2	869.85	40	MODD(ND)	2.403
20x5	244	899.24	1182	2	915.2	33	COVERT(ND)	1.459
20x5	164	1013.92	1185	2	899.55	31	SPT(ND)	1.717
20x5	244	896	1190	2	914.1	49	Biased-RANDOM(ND)	2.146
20x5	204	932.58	1193	2	886.25	41	SPT(ND)	2.403
20x5	12	967.46	1178	2	910.25	19	Biased-RANDOM(ND)	1.116

Table C.10. Summary of results obtained for problem LA25 and population size =44+4nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	11	1906.63	1003	5	933.667	86	RANDOM(ND)	2.661
15x10	642	1760.2	1007	5	932.467	84	Biased-RANDOM(ND)	3.071
15x10	1	1835.39	1003	5	930.867	86	RANDOM(ND)	2.661
15x10	643	1834.4	1003	5	930.867	87	Biased-RANDOM(ND)	2.661
15x10	33	1748.39	1003	5	933.667	86	Biased-RANDOM(ND)	2.661

Table C.11. Summary of results obtained for problem LA25 and population size =44+nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	6	615.66	1028	5	929.133	157	S/RPT(ND)	5.220
15x10	126	547.55	1007	5	925.667	105	WSPT+WOST(ND)	3.071
15x10	194	539.1	1007	7	940.667	125	Biased-RANDOM(ND)	3.071
15x10	194	517.95	1032	4	928.667	164	Biased-RANDOM(ND)	5.629
15x10	3	553.98	1008	6	940.6	135	S/RPT(ND)	3.173

Table C.12. Summary of results obtained for problem LA25 and population size =44+2nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	256	1023.21	1015	5	921.467	99	Biased-RANDOM(ND)	3.889
15x10	178	1036.28	1007	6	940.667	80	CR(ND)	3.071
15x10	343	952.51	1003	5	932.333	93	ATC(ND)	2.661
15x10	250	973.72	1012	5	923.267	131	WSPT+WOST(ND)	3.582
15x10	176	941.75	1010	3	905.267	78	WSPT+WOST(ND)	3.378

Table C.13. Summary of results obtained for problem LA29 and population size =44+4nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	37	4972.35	1202	10	1131.2	282	JST(ND)	4.250
20x10	1	4916.98	1191	6	1067.55	186	ODD(ND)	3.296
20x10	8	4810.48	1218	7	1093.05	311	RANDOM(ND)	5.637
20x10	522	5110.82	1229	7	1088.9	320	LRPT(ND)	6.592
20x10	15	4956.8	1212	7	1098.4	257	ATC(ND)	5.117

Table C.14. Summary of results obtained for problem LA29 and population size =44+nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	230	1437.84	1223	9	1080.05	297	JST(ND)	6.071
20x10	244	1482.6	1233	11	1154.85	601	LRPT(ND)	6.938
20x10	5	1351.39	1220	11	1135.5	437	LRPT(ND)	5.811
20x10	241	1464.2	1232	9	1140.05	455	LRPT(ND)	6.852
20x10	1	1474.7	1214	10	1118.3	256	JST(ND)	5.291

Table C.15. Summary of results obtained for problem LA29 and population size =44+2nm.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	35	2690.86	1211	6	1093.4	226	LRPT(ND)	5.030
20x10	442	2541.51	1231	6	1120.7	314	RANDOM(ND)	6.765
20x10	177	2588.97	1200	6	1058.35	218	Biased-RANDOM(ND)	4.076
20x10	93	2470.22	1221	12	1147.25	460	RANDOM(ND)	5.898
20x10	263	2624.73	1222	10	1130.9	353	RANDOM(ND)	5.984

APPENDIX D

RESULTS OF EXPERIMENT III

Table D.1. Summary of results obtained by CGA_Cmax approach for problem FT6.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	77	27.19	55	0	51	0	LRPT(A)	0.000
6x6	55	26.91	55	0	51.5	0	S/RPT(A)	0.000
6x6	80	22.57	55	0	51.5	0	COVERT(AS)	0.000
6x6	63	22.74	58	4	53.667	9	RANDOM(ND)	5.455
6x6	21	24.27	55	0	51.5	0	LRPT(A)	0.000

Table D.2. Summary of results obtained by UGA_Cmax approach for problem FT6.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	78	20.87	57	1	49.667	2	SRPT(ND)	3.636
6x6	75	21.64	57	1	49.5	2	RANDOM(ND)	3.636
6x6	78	20.65	57	1	49.5	2	RANDOM(ND)	3.636
6x6	63	21.14	57	1	49.667	2	RANDOM(ND)	3.636
6x6	71	20.33	57	1	49.667	2	Biased-RANDOM(ND)	3.636

Table D.3. Summary of results obtained by CGA_Cmax approach for problem FT10.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	144	179.12	968	3	823.7	97	ATC(ND)	4.086
10x10	14	168.84	976	5	895.6	184	SPT(ND)	4.946
10x10	144	180.32	976	4	894.4	133	JST(ND)	4.946
10x10	120	167.9	960	3	825.4	61	RANDOM(ND)	3.226
10x10	6	180.98	976	5	891.8	116	JST(ND)	4.946

Table D.4. Summary of results obtained by UGA_Cmax approach for problem FT10.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	144	169.5	973	4	833.4	149	OCR(ND)	4.624
10x10	144	168.96	968	3	839.3	53	WSPT+WOST(ND)	4.086
10x10	144	163.13	960	3	825.4	61	Biased-RANDOM(ND)	3.226
10x10	133	167.03	973	4	833.4	149	LAWINQ(ND)	4.624
10x10	96	188.67	976	7	934	178	Biased-RANDOM(ND)	4.946

Table D.5. Summary of results obtained by CGA_Cmax approach for problem FT20.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	140	531.96	1182	1	887.4	17	SRPT(ND)	1.459
20x5	134	537.5	1198	4	891	73	SPT(ND)	2.833
20x5	144	526.95	1193	2	861.7	51	SPT(ND)	2.403
20x5	26	506.09	1198	3	930.35	68	SPT(ND)	2.833
20x5	3	537.39	1200	3	882.35	59	SPT(ND)	3.004

Table D.6. Summary of results obtained by UGA_Cmax approach for problem FT20.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	144	468.68	1207	2	792.8	82	SPT(ND)	3.605
20x5	143	477.63	1192	2	856.7	52	MDD(ND)	2.318
20x5	7	529.87	1180	2	807.15	29	TWORK(ND)	1.288
20x5	130	510.15	1182	2	768.55	19	MDD(ND)	1.459
20x5	140	477.36	1210	2	805.25	78	SPT(ND)	3.863

Table D.7. Summary of results obtained by CGA Cmax approach for problem LA21.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15X10	191	579.63	1099	4	968.467	145	WSPT+WOST(ND)	5.067
15X10	189	586.61	1112	4	988.2	216	OCR(ND)	6.310
15X10	70	600.11	1097	5	979.267	127	ODD(ND)	4.876
15X10	190	568.26	1099	4	970.933	145	Biased-RANDOM(ND)	5.067
15X10	190	558.32	1103	4	960.2	129	RANDOM(ND)	5.449

Table D.8. Summary of results obtained by UGA Cmax approach for problem LA21.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15X10	84	599.35	1102	4	970.6	206	JST(ND)	5.354
15X10	1	609.89	1101	5	975.067	185	OST(ND)	5.258
15X10	76	555.03	1095	4	955.867	146	CR(ND)	4.685
15X10	25	577.1	1109	4	962.267	197	OCR(ND)	6.023
15X10	7	571.71	1094	4	951.4	144	S/RPT(ND)	4.589

Table D.9. Summary of results obtained by CGA Cmax approach for problem LA25.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	22	603.08	1003	5	933.667	86	RANDOM(ND)	2.661
15x10	188	616.1	1007	5	928.8	84	Biased-RANDOM(ND)	3.071
15x10	23	595.33	1003	5	928.533	93	Biased-RANDOM(ND)	2.661
15x10	60	577.05	1012	6	928.467	155	WSPT+WOST(ND)	3.582
15x10	107	596.43	1003	5	928.533	93	WSPT+WOST(ND)	2.661

Table D.10. Summary of results obtained by UGA Cmax approach for problem LA25.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	1	544.36	1034	5	882.667	138	JST(ND)	5.834
15x10	12	519.71	1051	5	865.6	234	ODD(ND)	7.574
15x10	194	527.89	1032	5	849.933	182	OCR(ND)	5.629
15x10	146	526.41	1042	6	926.333	218	CR(ND)	6.653
15x10	46	586.17	1029	3	875.467	127	ATC(ND)	5.322

Table D.11. Summary of results obtained by CGA Cmax approach for problem LA27.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	4	1570.93	1286	5	1166.5	176	WSPT+WOST(ND)	4.130
20x10	66	1582.02	1278	3	1129.7	89	ATC(ND)	3.482
20x10	129	1530.61	1292	4	1134.4	170	S/RPT(ND)	4.615
20x10	46	1708.18	1305	8	1158.85	368	ATC(ND)	5.668
20x10	2	1646.94	1296	7	1147.65	156	JST(ND)	4.939

Table D.12. Summary of results obtained by UGA Cmax approach for problem LA27.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	26	1470.25	1308	6	1167.15	252	WSPT+WOST(ND)	5.911
20x10	1	1458.72	1299	6	1141.8	255	OST(ND)	5.182
20x10	244	1510.84	1296	6	1140.05	196	RANDOM(ND)	4.939
20x10	76	1440.92	1314	5	1135.65	276	MWR(ND)	6.397
20x10	5	1536.43	1310	6	1128.05	303	ATC(ND)	6.073

Table D.13. Summary of results obtained by CGA_Cmax approach for problem LA29.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	5	1514.46	1220	11	1130.7	398	JST(ND)	5.811
20x10	11	1525.67	1214	11	1124.1	440	Biased-RANDOM(ND)	5.291
20x10	3	1607.78	1208	7	1115.3	262	WSPT+WOST(ND)	4.770
20x10	240	1508.42	1221	8	1079.85	303	Biased-RANDOM(ND)	5.898
20x10	3	1580.76	1224	12	1151.45	576	JST(ND)	6.158

Table D.14. Summary of results obtained by UGA_Cmax approach for problem LA29.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	228	1519.46	1227	10	1071.6	407	LRPT(ND)	6.418
20x10	1	1463.11	1230	8	1101.35	425	A/OPN(ND)	6.678
20x10	243	1540.77	1244	10	1119	514	S/RPT(ND)	7.892
20x10	1	1371.49	1250	10	1105.8	636	RANDOM(ND)	8.413
20x10	148	1472.33	1229	9	1093.4	341	MWR(ND)	6.592

Table D.15. Summary of results obtained by CGA_Cmax approach for problem LA38.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15X15	1	1093.79	1268	5	1145.333	194	ODD(ND)	6.020
15X15	7	1153.76	1268	5	1144.333	190	COVERT(ND)	6.020
15X15	269	1059.46	1268	5	1145.333	194	JST(ND)	6.020
15X15	67	1060.12	1275	5	1140.533	213	Biased-RANDOM(ND)	6.605
15X15	3	1154.54	1300	7	1159.333	364	RANDOM(ND)	8.696

Table D.16. Summary of results obtained by UGA_Cmax approach for problem LA38.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15X15	267	1068.69	1303	6	1160.133	449	RANDOM(ND)	8.946
15X15	14	1108.07	1282	5	1144.067	259	Biased-RANDOM(ND)	7.191
15X15	7	1049.63	1292	5	1149.133	301	RANDOM(ND)	8.027
15X15	216	1110.54	1294	5	1159.267	377	Biased-RANDOM(ND)	8.194
15X15	8	1058.69	1292	6	1131.2	283	Biased-RANDOM(ND)	8.027

Table D.17. Summary of results obtained by CGA_Cmax approach for problem LA40.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15X15	269	1120.32	1278	5	1176.067	201	Biased-RANDOM(ND)	4.583
15X15	261	1028.09	1278	4	1148	139	RANDOM(ND)	4.583
15X15	269	1084.89	1273	5	1152.133	226	Biased-RANDOM(ND)	4.173
15X15	67	1083.57	1278	5	1162.2	139	Biased-RANDOM(ND)	4.583
15X15	244	1106.64	1278	5	1157.933	135	Biased-RANDOM(ND)	4.583

Table D.18. Summary of results obtained by UGA_Cmax approach for problem LA40.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15X15	269	1056.1	1288	5	1145.667	220	RANDOM(ND)	5.401
15X15	192	1046.88	1287	4	1125.933	193	Biased-RANDOM(ND)	5.319
15X15	269	1009.25	1294	6	1160.667	272	RANDOM(ND)	5.892
15X15	264	1126.9	1278	5	1130.667	190	ATC(ND)	4.583
15X15	255	1069.46	1278	5	1140.267	139	RANDOM(ND)	4.583

APPENDIX E

RESULTS OF EXPERIMENT IV

Table E.1. Summary of results obtained by CGA_TT approach for problem FT6.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	79	21.54	55	4	51	5.75	LRPT(A)	0.000
6x6	60	23.01	55	4	51	5.75	LRPT(A)	0.000
6x6	79	22.41	55	4	51	5.75	JST(A)	0.000
6x6	67	22.03	55	4	51	5.75	Biased-RANDOM(A)	0.000
6x6	34	21.64	55	4	51	5.75	OST(A)	0.000

Table E.2. Summary of results obtained by UGA_TT approach for problem FT6.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	80	22.3	58	4	50	8.13	LRPT(A)	41.391
6x6	48	24.88	55	4	51	5.75	WSPT+WOST(A)	0.000
6x6	80	23.02	58	4	50	8.13	OST(A)	41.391
6x6	80	23.45	55	4	50.167	7.04	JST(A)	22.435
6x6	80	23.01	58	4	50	8.13	OST(A)	41.391

Table E.3. Summary of results obtained by CGA_TT approach for problem FT10.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	144	166.04	1022	5	811	346	ODD(ND)	26.703
10x10	144	164.45	960	10	821.6	273.08	EDD(ND)	0.000
10x10	144	168.02	1003	5	809.6	332	EDD(ND)	21.576
10x10	144	164.22	960	10	821.6	273.08	EDD(ND)	0.000
10x10	144	160.77	1094	7	802	320	WSPT+WOST(ND)	17.182

Table E.4. Summary of results obtained by UGA_TT approach for problem FT10.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	144	186.36	1003	5	809.6	332	MDD(ND)	21.576
10x10	140	188.4	960	10	821.6	273.08	CR(ND)	0.000
10x10	144	176.76	960	10	821.6	273.08	JST(ND)	0.000
10x10	144	180.7	1003	5	809.4	332	MDD(ND)	21.576
10x10	142	181.09	960	10	821.6	273.08	Biased-RANDOM(ND)	0.000

Table E.5. Summary of results obtained by CGA_TT approach for problem FT20.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	101	406.39	1182	4	752.15	93.94	EDD(ND)	0.000
20x5	116	424.9	1182	4	753.25	123.9	A/OPN(ND)	31.893
20x5	144	436.11	1188	5	754.35	120.44	WSPT+WOST(ND)	28.209
20x5	141	411.23	1188	3	748.45	96.4	WSPT+WOST(ND)	2.619
20x5	140	446.44	1182	4	753.75	123.9	A/OPN(ND)	31.893

Table E.6. Summary of results obtained by UGA_TT approach for problem FT20.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	73	516.19	1204	6	751.25	214.97	CR(ND)	128.838
20x5	6	500.43	1218	9	755.3	245.77	A/OPN(ND)	161.624
20x5	29	499.49	1201	11	758.15	237.84	A/OPN(ND)	153.183
20x5	1	573.64	1224	7	765.1	268.01	A/OPN(ND)	185.299
20x5	9	484.38	1228	7	765.4	279.01	A/OPN(ND)	197.009

Table E.7. Summary of results obtained by CGA_TT approach for problem LA21.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15X10	83	577.15	1203	6	943.333	664.8	COVERT(ND)	17.154
15X10	169	595.88	1093	8	917.667	664.46	Biased-RANDOM(ND)	17.094
15X10	1	582.43	1205	4	951.933	641.16	Biased-RANDOM(ND)	12.988
15X10	192	572.93	1126	7	914.267	725.06	Biased-RANDOM(ND)	27.773
15X10	2	551.95	1192	4	938.133	567.46	RANDOM(ND)	0.000

Table E.8. Summary of results obtained by UGA_TT approach for problem LA21.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15X10	1	619.72	1126	5	913.6	707.66	Biased-RANDOM(ND)	24.707
15X10	27	705.14	1102	6	898.067	627.06	Biased-RANDOM(ND)	10.503
15X10	194	654.38	1177	8	946.6	669.96	Biased-RANDOM(ND)	18.063
15X10	194	613.47	1126	5	938.733	689.66	OCR(ND)	21.535
15X10	1	651.09	1114	8	938.6	644.06	Biased-RANDOM(ND)	13.499

Table E.9. Summary of results obtained by CGA_TT approach for problem LA25.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	42	546.29	1047	4	859.133	274.22	ATC(ND)	0.000
15x10	2	579.74	1068	6	869.6	331.24	Biased-RANDOM(ND)	20.794
15x10	194	530.91	1053	8	880.2	378.69	ATC(ND)	38.097
15x10	11	586.66	1055	6	868.2	350.6	LRPT(ND)	27.854
15x10	79	585.56	1056	4	851.467	343.22	MODD(ND)	25.162

Table E.10. Summary of results obtained by UGA_TT approach for problem LA25.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	26	636.54	1056	4	870.733	390.27	Biased-RANDOM(ND)	42.320
15x10	194	602.04	1133	3	878.667	423.19	Biased-RANDOM(ND)	54.325
15x10	10	597.42	1055	5	855.733	323.24	ATC(ND)	17.876
15x10	138	603.19	1333	4	871.733	420.17	WSPT+WOST(ND)	53.224
15x10	181	595.99	1061	4	853.067	339.22	RANDOM(ND)	23.704

Table E.11. Summary of results obtained by CGA_TT approach for problem LA27.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	1	1457.33	1279	13	1076.35	637.53	MODD(ND)	15.681
20x10	1	1506.17	1570	10	1068.25	642.53	OCR(ND)	16.588
20x10	23	1446.29	1280	14	1085.45	551.11	ODD(ND)	0.000
20x10	242	1468.66	1329	12	1094.25	663.11	Biased-RANDOM(ND)	20.323
20x10	12	1435.15	1535	8	1084.3	657.18	OCR(ND)	19.247

Table E.12. Summary of results obtained by UGA_TT approach for problem LA27.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	108	1617	1544	13	1083.8	829.53	OCR(ND)	50.520
20x10	1	1538.8	1530	11	1082.55	729.96	OCR(ND)	32.453
20x10	13	1601.08	1600	13	1118.95	1015.14	A/OPN(ND)	84.199
20x10	5	1741.69	1469	11	1109.25	785.08	WSPT+WOST(ND)	42.454
20x10	31	1638.92	1375	10	1079.45	734.53	SPT(ND)	33.282

Table E.13. Summary of results obtained by CGA_TT approach for problem LA29.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	1	1426.91	1555	7	980	814	Biased-RANDOM(ND)	14.810
20x10	14	1374.07	1592	5	1023.6	740	SRPT(ND)	4.372
20x10	29	1371.27	1462	6	1029	798.67	Biased-RANDOM(ND)	12.647
20x10	1	1440.36	1592	6	1028	709	MODD(ND)	0.000
20x10	1	1518.8	1353	11	1080.25	747.34	EDD(ND)	5.408

Table E.14. Summary of results obtained by UGA_TT approach for problem LA29.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x10	5	1660.06	1558	8	1032.8	909	SRPT(ND)	28.209
20x10	1	1489.25	1647	9	1040.75	936.67	Biased-RANDOM(ND)	32.111
20x10	244	1635.24	1489	8	1045.1	918.34	Biased-RANDOM(ND)	29.526
20x10	4	1573.89	1592	6	1022.95	994.67	ATC(ND)	40.292
20x10	4	1521.16	1582	9	1052.35	1030.67	RANDOM(ND)	45.370

Table E.15. Summary of results obtained by CGA_TT approach for problem LA38.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15X15	13	1047.59	1429	10	1105	802.9	RANDOM(ND)	15.829
15X15	229	1072.7	1369	9	1092.667	789.26	MWR(ND)	13.861
15X15	269	1034.41	1351	12	1112	838.54	RANDOM(ND)	20.970
15X15	255	1088.35	1339	12	1115.6	858.54	Biased-RANDOM(ND)	23.855
15X15	3	1110.59	1364	11	1117.133	693.18	RANDOM(ND)	0.000

Table E.16. Summary of results obtained by UGA_TT approach for problem LA38.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15X15	269	1118.56	1364	13	1120.333	757.18	Biased-RANDOM(ND)	9.233
15X15	23	1173.27	1313	12	1118.8	795.28	JST(ND)	14.729
15X15	226	1131.63	1324	12	1120.867	759.54	Biased-RANDOM(ND)	9.573
15X15	41	1136.46	1353	10	1109.2	788.54	Biased-RANDOM(ND)	13.757
15X15	1	1144.38	1439	11	1099.867	781.9	EDD(ND)	12.799

Table E.17. Summary of results obtained by CGA_TT approach for problem LA40.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15X15	260	1091.86	1316	11	1107	367.81	Biased-RANDOM(ND)	0.000
15X15	1	1114.66	1316	11	1107	367.81	Biased-RANDOM(ND)	0.000
15X15	198	1044.85	1316	11	1107	367.81	RANDOM(ND)	0.000
15X15	84	1085.11	1316	11	1107	367.81	RANDOM(ND)	0.000
15X15	241	1033.53	1316	11	1107.267	367.81	Biased-RANDOM(ND)	0.000

Table E.18. Summary of results obtained by UGA_TT approach for problem LA40.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15X15	118	1197.54	1316	11	1107	367.81	Biased-RANDOM(ND)	0.000
15X15	207	1142.18	1316	11	1107.267	367.81	RANDOM(ND)	0.000
15X15	269	1217.04	1402	8	1124.733	502.75	Biased-RANDOM(ND)	36.687
15X15	268	1238.68	1316	11	1107.267	367.81	Biased-RANDOM(ND)	0.000
15X15	14	1180.57	1387	8	1116.8	514.34	Biased-RANDOM(ND)	39.839

APPENDIX F

RESULTS OF EXPERIMENT V

Table F.1. Summary of results obtained by CGA_WSPT approach for problem FT6.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
6x6	3	62.4	58.854	4	52.331	15.085	LRPT(A)
6x6	17	61.95	58.854	4	52.331	15.085	LRPT(A)
6x6	21	60.64	61.181	5	51.558	14.25	LRPT(A)
6x6	45	61.51	58.854	4	52.331	15.085	LRPT(A)
6x6	7	62.51	58.854	4	52.331	15.085	LRPT(A)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
6x6	3.913	33.333	0.430	17.998
6x6	4.093	33.333	4.291	26.960
6x6	0.301	66.667	2.750	31.003
6x6	4.093	33.333	4.291	26.960
6x6	4.093	33.333	4.291	26.960

Table F.2. Summary of results obtained by CGA_SIM approach for problem FT6.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
6x6	1	1939.15	61.251	3	52.107	18.396	LRPT(A)
6x6	18	1498.2	61.366	3	50.178	20.653	Biased-RANDOM(ND)
6x6	11	1521.54	61.366	3	50.178	20.653	Biased-RANDOM(ND)
6x6	7	1582.95	61.366	3	50.178	20.653	Biased-RANDOM(ND)
6x6	52	1493.15	61.366	3	50.178	20.653	LAWINQ(ND)

90 Percent confidence interval

Problem size	Makespan		Number Tardy		Average flow time		Total Tardiness	
6x6	55.683	66.819	2.727	3.273	47.370	56.844	16.724	20.068
6x6	55.787	66.945	2.727	3.273	45.616	54.740	18.775	22.531
6x6	55.787	66.945	2.727	3.273	45.616	54.740	18.775	22.531
6x6	55.787	66.945	2.727	3.273	45.616	54.740	18.775	22.531
6x6	55.787	66.945	2.727	3.273	45.616	54.740	18.775	22.531

Table F.3. Summary of results obtained by CGA_WSPT approach for problem FT10.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
10x10	133	446.98	1079.332	6	812.432	528.921	Biased-RANDOM(ND)
10x10	137	445.99	1036.885	6	825.459	452.772	EDD(ND)
10x10	139	445.67	1037.551	6	827.185	459.65	MDD(ND)
10x10	143	469.12	1071.895	6	814.87	501.372	A/OPN(ND)
10x10	131	462.69	1063.015	6	812.467	533.409	EDD(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
10x10	1.487	0.000	1.278	14.730
10x10	2.918	0.000	0.006	31.380
10x10	6.261	0.000	0.185	22.494
10x10	2.056	0.000	1.746	23.175
10x10	3.841	0.000	1.148	15.320

Table F.4. Summary of results obtained by CGA_SIM approach for problem FT10.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
10x10	14	3256.65	1063.514	6	822.95	620.289	EDD(ND)
10x10	19	3172.66	1068.047	6	825.506	659.827	A/OPN(ND)
10x10	26	3307.89	1106.85	6	825.656	593.052	MODD(ND)
10x10	1	3404.4	1094.401	6	829.347	652.616	Biased-RANDOM(ND)
10x10	74	3230.66	1105.48	6	821.901	629.91	MDD(ND)

90 Percent confidence interval

Problem size	Makespan		Number Tardy		Average flow time		Total Tardiness	
10x10	966.831	1160.197	5.455	6.545	748.136	897.764	563.899	676.679
10x10	970.952	1165.142	5.455	6.545	750.460	900.552	599.843	719.811
10x10	1006.227	1207.473	5.455	6.545	750.596	900.716	539.138	646.966
10x10	994.910	1193.892	5.455	6.545	753.952	904.742	593.287	711.945
10x10	1004.982	1205.978	5.455	6.545	747.183	896.619	572.645	687.175

Table F.5. Summary of results obtained by CGA_WSPT approach for problem FT20.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
20x5	132	1322.93	1240.885	8	761.133	439.272	EDD(ND)
20x5	9	1268.34	1232.552	6	758.724	426.703	A/OPN(ND)
20x5	120	1296.4	1253.168	8	775.512	518.12	A/OPN(ND)
20x5	139	1289.43	1259.168	7	774.198	426.838	A/OPN(ND)
20x5	141	1196.83	1259.168	9	772.465	463.978	A/OPN(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
20x5	0.130	11.111	1.343	21.136
20x5	0.628	40.000	1.771	23.561
20x5	0.175	0.000	1.031	0.176
20x5	1.513	22.222	0.166	20.784
20x5	1.439	10.000	0.198	13.832

Table F.6. Summary of results obtained by CGA_SIM approach for problem FT20.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
20x5	2	20558.94	1242.498	9	771.498	557	A/OPN(ND)
20x5	3	20070.82	1240.339	10	772.401	558.224	A/OPN(ND)
20x5	143	16459.8	1250.98	8	767.601	519.031	A/OPN(ND)
20x5	4	19649.82	1240.399	9	772.914	538.825	A/OPN(ND)
20x5	1	18856.8	1241.311	10	774.001	538.455	EDD(ND)

90 Percent confidence interval

Problem size	Makespan		Number Tardy		Average flow time		Total Tardiness	
20x5	1129.544	1355.452	8.182	9.818	701.362	841.634	506.364	607.636
20x5	1127.581	1353.097	9.091	10.909	702.183	842.619	507.476	608.972
20x5	1137.255	1364.705	7.273	8.727	697.819	837.383	471.846	566.216
20x5	1127.635	1353.163	8.182	9.818	702.649	843.179	489.841	587.809
20x5	1128.465	1354.157	9.091	10.909	703.637	844.365	489.505	587.405

Table F.7. Summary of results obtained by CGA_WSPT approach for problem LA21.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15x10	178	1574.88	1222.432	4	948.29	888.119	RANDOM(ND)
15x10	190	1629.59	1224.665	7	939.321	934.708	RANDOM(ND)
15x10	183	1657.93	1183.361	8	933.082	927.703	Biased-RANDOM(ND)
15x10	192	1728.84	1206.95	9	991.736	1363.652	WSPT+WOST(ND)
15x10	186	1630.62	1246.686	6	950.319	1193.703	LAWINQ(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
15x10	1.497	50.000	2.922	27.907
15x10	7.156	22.222	2.901	30.049
15x10	4.537	0.000	3.910	23.493
15x10	2.492	12.500	2.270	15.168
15x10	0.820	25.000	1.017	0.157

Table F.8. Summary of results obtained by CGA_SIM approach for problem LA21.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15x10	3	7138.78	1241.004	8	976.83	1231.91	WSPT+WOST(ND)
15x10	6	6795.44	1319.063	9	967.387	1336.242	Biased-RANDOM(ND)
15x10	1	7356.06	1239.6	8	971.052	1212.57	RANDOM(ND)
15x10	159	7182.11	1237.795	8	969.727	1184.059	JST(ND)
15x10	1	6752.1	1236.545	8	960.08	1191.837	Biased-RANDOM(ND)

90 Percent confidence interval

Problem size	Makespan		Number Tardy		Average flow time		Total Tardiness	
15x10	1128.185	1353.823	7.273	8.727	888.027	1065.633	1119.918	1343.902
15x10	1199.148	1438.978	8.182	9.818	879.443	1055.331	1214.765	1457.719
15x10	1126.909	1352.291	7.273	8.727	882.775	1059.329	1102.336	1322.804
15x10	1125.268	1350.322	7.273	8.727	881.570	1057.884	1076.417	1291.701
15x10	1124.132	1348.958	7.273	8.727	872.800	1047.360	1083.488	1300.186

Table F.9. Summary of results obtained by CGA_WSPT approach for problem LA25.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15x10	169	1575.48	1127.558	5	873.222	537.39	RANDOM(ND)
15x10	176	1518.09	1314.645	3	878.273	707.489	RANDOM(ND)
15x10	175	1636.72	1126.1	6	874.045	624.144	MODD(ND)
15x10	183	1539.34	1309.088	2	880.297	669.137	JST(ND)
15x10	184	1621.18	1245.386	4	878.136	725.607	MODD(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
15x10	3.791	28.571	3.587	39.887
15x10	2.483	50.000	2.820	15.689
15x10	7.788	14.286	3.632	31.802
15x10	8.328	66.667	1.173	14.165
15x10	3.836	33.333	1.703	4.995

Table F.10. Summary of results obtained by CGA_SIM approach for problem LA25.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15x10	20	8303.37	1171.99	7	905.707	893.971	Biased-RANDOM(ND)
15x10	7	7870.44	1282.798	6	903.759	839.141	Biased-RANDOM(ND)
15x10	173	7818.97	1221.203	7	906.983	915.192	Biased-RANDOM(ND)
15x10	77	8512.62	1208.444	6	890.747	779.563	Biased-RANDOM(ND)
15x10	12	8801.59	1199.373	6	893.349	763.758	WSPT+WOST(ND)

90 Percent confidence interval

Problem size	Makespan		Number Tardy		Average flow time		Total Tardiness	
15x10	1065.445	1278.535	6.364	7.636	823.370	988.044	812.701	975.241
15x10	1166.180	1399.416	5.455	6.545	821.599	985.919	762.855	915.427
15x10	1110.185	1332.221	6.364	7.636	824.530	989.436	831.993	998.391
15x10	1098.585	1318.303	5.455	6.545	809.770	971.724	708.694	850.432
15x10	1090.339	1308.407	5.455	6.545	812.135	974.563	694.325	833.191

Table F.11. Summary of results obtained by CGA_WSPT approach for problem LA38.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15X15	249	3244.34	1435.47	9	1154.895	1530.144	Biased-RANDOM(ND)
15X15	260	3175.63	1466.153	9	1158.001	1559.851	Biased-RANDOM(ND)
15X15	247	2974.93	1498.82	9	1145.1	1496.685	EDD(ND)
15X15	261	3161.18	1423.915	10	1176.191	1766.202	MDD(ND)
15X15	261	3052.7	1381.867	12	1177.306	1740.067	LRPT(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
15X15	1.747	18.182	0.285	5.145
15X15	3.224	25.000	0.065	3.467
15X15	8.608	25.000	0.974	1.836
15X15	0.406	9.091	1.543	16.141
15X15	2.123	9.091	2.401	22.759

Table F.12. Summary of results obtained by CGA_SIM approach for problem LA38.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15X15	2	15576.99	1410.821	11	1151.618	1455.264	Biased-RANDOM(ND)
15X15	3	15494.48	1420.365	12	1158.757	1507.585	Biased-RANDOM(ND)
15X15	140	15193.05	1380.032	12	1156.361	1469.699	Biased-RANDOM(ND)
15X15	18	15870.94	1429.715	11	1158.313	1520.735	Biased-RANDOM(ND)
15X15	1	15509.03	1411.839	11	1149.703	1417.47	Biased-RANDOM(ND)

90 Percent confidence interval

Problem size	Makespan		Number Tardy		Average flow time		Total Tardiness	
15X15	1282.565	1539.077	10.000	12.000	1046.925	1256.311	1322.967	1587.561
15X15	1291.241	1549.489	10.909	13.091	1053.415	1264.099	1370.532	1644.638
15X15	1254.575	1505.489	10.909	13.091	1051.237	1261.485	1336.090	1603.308
15X15	1299.741	1559.689	10.000	12.000	1053.012	1263.614	1382.486	1658.984
15X15	1283.490	1540.188	10.000	12.000	1045.185	1254.221	1288.609	1546.331

Table F.13. Summary of results obtained by CGA_WSPT approach for problem LA40.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15X15	263	3053.47	1360.974	12	1161.059	1090.126	Biased-RANDOM(ND)
15X15	231	3034.52	1409.571	10	1167.044	1244.586	Biased-RANDOM(ND)
15X15	254	2972.57	1387.883	11	1149.836	1081.458	RANDOM(ND)
15X15	250	3109.28	1444.59	12	1176.202	1367.004	A/OPN(ND)
15X15	248	3101.04	1406.917	9	1159.811	1175.426	RANDOM(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
15X15	4.174	20.000	0.245	15.814
15X15	0.464	16.667	0.144	5.488
15X15	0.782	0.000	1.590	18.903
15X15	3.070	9.091	0.854	11.369
15X15	1.854	18.182	1.010	9.015

Table F.14. Summary of results obtained by CGA_SIM approach for problem LA40.

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15X15	92	17617.69	1420.25	10	1163.909	1294.898	Biased-RANDOM(ND)
15X15	237	16575.97	1416.144	12	1168.725	1316.857	Biased-RANDOM(ND)
15X15	42	17193.16	1398.818	11	1168.411	1333.53	RANDOM(ND)
15X15	1	20077.25	1401.565	11	1166.247	1227.459	Biased-RANDOM(ND)
15X15	2	20720.26	1433.487	11	1171.639	1291.889	Biased-RANDOM(ND)

90 Percent confidence interval

Problem size	Makespan		Number Tardy		Average flow time		Total Tardiness	
15X15	1291.136	1549.364	9.091	10.909	1058.099	1269.719	1177.180	1412.616
15X15	1287.404	1544.884	10.909	13.091	1062.477	1274.973	1197.143	1436.571
15X15	1271.653	1525.983	10.000	12.000	1062.192	1274.630	1212.300	1454.760
15X15	1274.150	1528.980	10.000	12.000	1060.225	1272.269	1115.872	1339.046
15X15	1303.170	1563.804	10.000	12.000	1065.126	1278.152	1174.445	1409.333

APPENDIX G

RESULTS OF EXPERIMENT VI

Table G.1. Summary of results obtained for problem FT6 (Case I: PP1 and O = Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	79	64.32	4678.900	2	3417.858	1024.863	ATC(A)	102686.300
6x6	80	57.89	4678.900	2	3417.858	1024.863	ATC(A)	102486.300
6x6	69	58.66	4678.900	2	3417.858	1024.863	ATC(A)	102486.300
6x6	80	61.13	4678.900	2	3417.858	1024.863	ATC(A)	102486.300
6x6	70	53.83	4720.967	2	3459.925	1108.997	ATC(A)	110899.700

Table G.2. Summary of results obtained for problem FT6 (Case II: PP1 and O < Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
11x6	110	269.41	5000.212	0	2873.28	0	MDD(ND)	0.000
11x6	110	267.11	5000.212	0	2871.666	0	MDD(ND)	0.000
11x6	110	254.64	5000.212	0	2899.289	0	MDD(ND)	0.000
11x6	110	264.69	4625.698	0	2896.844	0	A/OPN(ND)	0.000
11x6	110	261.5	5000.212	0	2942.759	0	MDD(ND)	0.000

Table G.3. Summary of results obtained for problem FT6 (Case III: PP2 and O = Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
6x6	80	61.02	4870.012	3	3435.844	241.243	EDD(ND)	24124.300
6x6	80	54.43	4870.012	3	3435.844	241.243	EDD(ND)	24124.300
6x6	76	56.8	4870.012	3	3435.844	241.243	RANDOM(ND)	24124.300
6x6	76	59.38	4870.012	3	3435.844	241.243	MODD(ND)	24124.300
6x6	80	54.82	4870.012	3	3435.844	241.243	Biased-RANDOM(ND)	24124.300

Table G.4. Summary of results obtained for problem FT6 (Case IV: PP2 and O < Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
11x6	110	277.65	4148.081	0	2525.526	0	TWORK(ND)	0.000
11x6	110	279.08	4398.677	0	2807.144	0	Biased-RANDOM(ND)	0.000
11x6	110	272.81	4647.518	0	2799.299	0	EDD(A)	0.000
11x6	110	272.22	4405.831	0	2917.759	0	SRPT(ND)	0.000
11x6	110	271.67	4161.727	0	2784.773	0	SRPT(ND)	0.000

Table G.5. Summary of results obtained for problem FT10 (Case I: PP1 and O = Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	142	437.76	10377.882	3.333	5279.406	843.464	Biased-RANDOM(ND)	794.173
10x10	144	437.65	10243.996	4.333	5338.428	1514.745	WSPT+WOST(ND)	204.567
10x10	142	455.88	10377.882	3.333	5279.406	843.464	MODD(ND)	142.564
10x10	129	548.49	10392.997	2	5232.494	657.41	MODD(A)	2266.487
10x10	144	487.91	10392.997	2	5232.494	657.41	MODD(A)	59.143

Table G.6. Summary of results obtained for problem FT10 (Case II: PP1 and O < Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
19x10	227	2949.28	7477.108	1	4696.928	679.125	ATC(ND)	619.954
19x10	228	3024.53	7456.205	1.667	4628.617	726.145	Biased-RANDOM(ND)	46.005
19x10	232	2965.37	7509.317	1	4738.849	742.555	ATC(ND)	113.544
19x10	232	2972.67	7769.071	1	4770.877	619.724	ATC(ND)	2130.828
19x10	231	3032.05	7653.914	1	4726.632	542.068	Biased-RANDOM(ND)	31.221

Table G.7. Summary of results obtained for problem FT10 (Case III: PP2 and O = Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
10x10	2	442.64	10333.062	3	5319.798	1618.428	MODD(A)	1615.727
10x10	35	463.52	10185.158	4	5317.002	1260.086	MODD(A)	153.363
10x10	140	495.76	9672.101	2.667	5241.074	1758.459	MODD(A)	405.698
10x10	115	450.22	10050.545	4.333	5368.325	1590.944	MODD(A)	5626.940
10x10	139	466.59	10148.79	2.667	5258.093	641.122	MODD(A)	55.200

Table G.8. Summary of results obtained for problem FT10 (Case IV: PP2 and O < Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
19x10	229	3021.13	7978.923	1	4932.674	94.329	ATC(ND)	0.000
19x10	230	2989.65	8048.686	1	4953.34	497.344	ATC(ND)	0.000
19x10	230	3061.39	7895.034	1	4871.786	347.729	ATC(ND)	0.000
19x10	228	3278.61	7939.48	0.667	4801.138	27.78	Biased-RANDOM(ND)	0.000
19x10	231	3180.9	8055.037	1.333	5004.182	413.095	ATC(ND)	0.000

Table G.9. Summary of results obtained for problem FT20 (Case I: PP1 and O = Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	1	1425.43	12229.579	17	6417.905	58524.599	MDD(A)	0.000
20x5	1	1179.85	12128.903	18	6312.958	56343.504	SRPT(ND)	0.000
20x5	2	1296.9	12229.579	17	6413.127	58429.031	MDD(A)	5.300
20x5	144	1242.58	11918.124	17	6370.781	57627.613	MDD(ND)	3.628
20x5	20	1393.35	12251.797	17	6473.464	59635.769	MDD(A)	3.397

Table G.10. Summary of results obtained for problem FT20 (Case II: PP1 and O < Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
37x5	31	9237.75	13961.862	26.333	5740.116	84347.725	SPT(ND)	44.124
37x5	2	9458.61	14474.099	25	5694.326	81870.47	SRPT(ND)	45.306
37x5	1	9268.01	13961.862	24.333	5739.904	84172.942	SRPT(ND)	51.695
37x5	1	9293.67	14516.519	24	5799.659	86133.535	SRPT(ND)	54.888
37x5	171	9604.88	13961.862	26	5748.889	84597.425	SPT(ND)	46.676

Table G.11. Summary of results obtained for problem FT20 (Case III: PP2 and O = Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
20x5	122	1089.23	12333.526	19.667	6544.912	60467.992	ATC(ND)	3.321
20x5	143	1082.14	13683.754	19	6534.704	60311.277	SRPT(ND)	7.042
20x5	3	1111.75	12675.144	17.333	6293.53	55488.338	ATC(ND)	0.000
20x5	11	1044.57	12729.337	18	6288.605	55610.257	ATC(ND)	0.000
20x5	143	1054.68	12568.266	19	6391.188	57676.273	MODD(ND)	0.000

Table G.12. Summary of results obtained for problem FT20 (Case IV: PP2 and O < Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
37x5	1	8126.72	14822.514	26.667	5870.063	90304.46	SRPT(ND)	54.302
37x5	3	7738.06	14728.141	24.333	5759.24	86169.473	SRPT(ND)	52.936
37x5	2	7586.36	14822.514	27	5911.186	90049.376	SRPT(ND)	62.285
37x5	203	7736.2	14822.514	26	5902.435	90764.624	SRPT(ND)	63.216
37x5	50	8026.05	14792.063	25.333	5836.836	89672.107	SRPT(ND)	55.475

Table G.13. Summary of results obtained for problem LA21 (Case I: PP1 and O = Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	115	1488.32	11782.555	10.333	7189.398	13557.195	Biased-RANDOM(ND)	78.881
15x10	183	1475.13	11595.666	12.667	7312.769	13644.937	SPT(ND)	94.010
15x10	194	1544.84	10972.215	13	7404.054	15040.765	MODD(ND)	156.984
15x10	192	1656.05	11472.059	12.333	7257.854	13618.968	EDD(ND)	103.802
15x10	185	1535	11048.617	12	7313.665	13698.016	SPT(ND)	89.098

Table G.14. Summary of results obtained for problem LA21 (Case II: PP1 and O < Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
28x10	312	13134.5	10583.701	9.333	6271.694	8116.891	MDD(ND)	7.099
28x10	158	13340.2	10389.481	9.667	6294.417	8628.599	MDD(ND)	22.685
28x10	324	12297.16	10485.72	11.667	6334.354	7857.786	MDD(ND)	34.257
28x10	199	13447.41	10315.424	9.333	6438.975	8147.594	MDD(ND)	21.925
28x10	280	12590.13	10415.377	10.333	6414.264	7552.052	MDD(ND)	4.255

Table G.15. Summary of results obtained for problem LA21 (Case III: PP2 and O = Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
15x10	180	1527.8	11518.854	12.333	7152.286	10391.181	MODD(ND)	37.107
15x10	189	1474.37	12164.899	13	7255.654	12180.41	SPT(ND)	73.187
15x10	194	1510.18	11354.351	12	7226.579	12301.23	MODD(ND)	110.177
15x10	192	1545.66	11755.656	11.667	7100.836	11025.144	WSPT+WOST(ND)	64.987
15x10	193	1594.82	11925.236	11.333	7239.288	12609.367	SPT(ND)	74.070

Table G.16. Summary of results obtained for problem LA21 (Case IV: PP2 and O < Q).

Problem size	No. of alternatives	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome	Percentage of error
28x10	243	14472.87	10789.83	10.667	6227.124	7578.889	SPT(ND)	0.000
28x10	282	14538.83	11055.534	9	6323.238	7033.108	Biased-RANDOM(ND)	0.000
28x10	252	14097.4	10721.009	9.333	6063.088	5852.796	Biased-RANDOM(ND)	0.000
28x10	182	14402.23	10494.373	6.667	6208.898	6682.451	Biased-RANDOM(ND)	0.000
28x10	279	13361.94	10897.655	9.667	6344.725	7243.858	Biased-RANDOM(ND)	0.000

APPENDIX H

RESULTS OF EXPERIMENT VII

Table H.1. Summary of results obtained by simulating the final best solution obtained by the CGA_WSPT approach for problem FT6.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
6x6	82	0.22	60.831	4	53.37	23.253	LRPT(A)
6x6	133	0.33	65.432	4	54.351	30.192	LRPT(A)
6x6	182	0.39	65.78	3	52.56	24.808	LRPT(A)
6x6	75	0.16	62.275	4	54.04	27.542	LRPT(A)
6x6	152	0.33	64.139	4	54.337	30.806	LRPT(A)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
6x6	0.686	33.333	2.424	26.402
6x6	6.626	33.333	8.316	46.187
6x6	7.193	0.000	4.747	20.118
6x6	1.481	33.333	7.697	33.356
6x6	4.519	33.333	8.288	49.160

Table H.2. Summary of results obtained by simulating the final best solution obtained by the CGA_TT approach for problem FT6.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
6x6	141	0.39	62.612	4	55.726	36.13	LRPT(A)
6x6	158	0.33	61.004	4	54.057	26.248	LRPT(A)
6x6	130	0.33	62.25	5	55.792	35.048	JST(A)
6x6	72	0.16	64.768	5	57.991	48.005	Biased-RANDOM(A)
6x6	126	0.28	61.916	4	54.904	30.678	OST(A)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
6x6	2.222	33.333	6.945	96.401
6x6	0.590	33.333	7.730	27.090
6x6	1.441	66.667	11.188	69.699
6x6	5.544	66.667	15.571	132.436
6x6	0.896	33.333	9.418	48.540

Table H.3. Summary of results obtained by simulating the final best solution obtained by the CGA_SIM approach for problem FT6.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
6x6	91	0.22	61.251	3	52.107	18.396	LRPT(A)
6x6	58	0.11	61.366	3	50.178	20.653	Biased-RANDOM(ND)
6x6	58	0.11	61.366	3	50.178	20.653	Biased-RANDOM(ND)
6x6	58	0.11	61.366	3	50.178	20.653	Biased-RANDOM(ND)
6x6	58	0.11	61.366	3	50.178	20.653	LAWINQ(ND)

90 Percent confidence interval

Problem size	Makespan		Number Tardy		Average flow time		Total Tardiness	
6x6	55.683	66.819	2.727	3.273	47.370	56.844	16.724	20.068
6x6	55.787	66.945	2.727	3.273	45.616	54.740	18.775	22.531
6x6	55.787	66.945	2.727	3.273	45.616	54.740	18.775	22.531
6x6	55.787	66.945	2.727	3.273	45.616	54.740	18.775	22.531
6x6	55.787	66.945	2.727	3.273	45.616	54.740	18.775	22.531

Table H.4. Summary of results obtained by simulating the final best solution obtained by the CGA_WSPT approach for problem FT10.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
10x10	31	0.27	1096.485	7	849.298	773.306	Biased-RANDOM(ND)
10x10	21	0.16	1076.879	7	848.158	819.863	EDD(ND)
10x10	41	0.33	1108.936	7	855.383	835.797	MDD(ND)
10x10	46	0.33	1119.78	7	854.213	817.696	A/OPN(ND)
10x10	27	0.22	1092.847	7	842.864	749.833	EDD(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
10x10	3.100	16.667	3.202	24.669
10x10	0.827	16.667	2.744	24.254
10x10	0.188	16.667	3.600	40.931
10x10	2.319	16.667	2.998	25.295
10x10	1.143	16.667	2.551	19.038

Table H.5. Summary of results obtained by simulating the final best solution obtained by the CGA_TT approach for problem FT10.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
10x10	26	0.17	1080.194	7	845.635	767.79	ODD(ND)
10x10	43	0.32	1114.563	7	862.784	820.893	EDD(ND)
10x10	27	0.16	1095.852	7	865.518	911.314	EDD(ND)
10x10	57	0.5	1053.114	8	858.779	727.646	EDD(ND)
10x10	68	0.49	1114.494	7	853.187	772.393	WSPT+WOST(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
10x10	1.568	16.667	2.757	23.779
10x10	4.355	16.667	4.516	24.410
10x10	0.994	16.667	4.828	53.665
10x10	3.773	33.333	3.549	11.497
10x10	0.815	16.667	3.807	22.620

Table H.6. Summary of results obtained by simulating the final best solution obtained by the CGA_SIM approach for problem FT10.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
10x10	11	0.05	1063.514	6	822.95	620.289	EDD(ND)
10x10	11	0.11	1068.047	6	825.506	659.827	A/OPN(ND)
10x10	11	0.06	1106.85	6	825.656	593.052	MODD(ND)
10x10	11	0.11	1094.401	6	829.347	652.616	Biased-RANDOM(ND)
10x10	11	0.05	1105.48	6	821.901	629.91	MDD(ND)

90 Percent confidence interval

Problem size	Makespan		Number Tardy		Average flow time		Total Tardiness	
10x10	966.831	1160.197	5.455	6.545	748.136	897.764	563.899	676.679
10x10	970.952	1165.142	5.455	6.545	750.460	900.552	599.843	719.811
10x10	1006.227	1207.473	5.455	6.545	750.596	900.716	539.138	646.966
10x10	994.910	1193.892	5.455	6.545	753.952	904.742	593.287	711.945
10x10	1004.982	1205.978	5.455	6.545	747.183	896.619	572.645	687.175

Table H.7. Summary of results obtained by simulating the final best solution obtained by the CGA_WSPT approach for problem FT20.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
20x5	42	0.88	1257.596	9	773.072	678.294	EDD(ND)
20x5	48	0.99	1260.387	9	773.813	758.119	A/OPN(ND)
20x5	60	1.2	1282.671	11	799.228	1072.514	A/OPN(ND)
20x5	63	1.32	1307.754	11	798.149	989.403	A/OPN(ND)
20x5	41	0.83	1275.525	11	789.949	798.436	A/OPN(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
20x5	1.215	0.000	0.204	21.776
20x5	1.616	10.000	0.183	35.809
20x5	2.533	37.500	4.120	106.638
20x5	5.430	22.222	3.265	83.622
20x5	2.756	10.000	2.060	48.283

Table H.8. Summary of results obtained by simulating the final best solution obtained by the CGA_TT approach for problem FT20.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
20x5	58	1.21	1266.774	9	776.307	688.183	EDD(ND)
20x5	48	1.04	1264.428	10	776.646	691.183	A/OPN(ND)
20x5	58	1.21	1245.166	11	793.17	890.768	WSPT+WOST(ND)
20x5	62	1.21	1296.153	9	779.686	753.737	WSPT+WOST(ND)
20x5	48	0.99	1245.404	9	776.6	650.28	A/OPN(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
20x5	1.954	0.000	0.623	23.552
20x5	1.942	0.000	0.550	23.818
20x5	0.465	37.500	3.331	71.621
20x5	4.495	0.000	0.876	39.885
20x5	0.330	10.000	0.336	20.768

Table H.9. Summary of results obtained by simulating the final best solution obtained by the CGA_SIM approach for problem FT20.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
20x5	44	0.88	1242.498	9	771.498	557	A/OPN(ND)
20x5	45	0.93	1240.339	10	772.401	558.224	A/OPN(ND)
20x5	28	0.61	1250.98	8	767.601	519.031	A/OPN(ND)
20x5	49	0.98	1240.399	9	772.914	538.825	A/OPN(ND)
20x5	48	0.94	1241.311	10	774.001	538.455	EDD(ND)

90 Percent confidence interval

Problem size	Makespan		Number Tardy		Average flow time		Total Tardiness	
20x5	1129.544	1355.452	8.182	9.818	701.362	841.634	506.364	607.636
20x5	1127.581	1353.097	9.091	10.909	702.183	842.619	507.476	608.972
20x5	1137.255	1364.705	7.273	8.727	697.819	837.383	471.846	566.216
20x5	1127.635	1353.163	8.182	9.818	702.649	843.179	489.841	587.809
20x5	1128.465	1354.157	9.091	10.909	703.637	844.365	489.505	587.405

Table H.10. Summary of results obtained by simulating the final best solution obtained by the CGA_WSPT approach for problem LA21.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15x10	11	0.22	1289.558	9	1001.358	1634.079	RANDOM(ND)
15x10	18	0.33	1251.812	9	992.416	1479.511	RANDOM(ND)
15x10	11	0.22	1262.005	9	991.884	1488.649	Biased-RANDOM(ND)
15x10	15	0.33	1236.718	10	1002.529	1521.685	WSPT+WOST(ND)
15x10	11	0.22	1308.552	9	991.294	1697.678	LAWINQ(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
15x10	3.912	12.500	2.511	32.646
15x10	5.098	0.000	2.587	10.722
15x10	1.807	12.500	2.145	22.768
15x10	0.087	25.000	3.383	28.514
15x10	5.823	12.500	3.251	42.442

Table H.11. Summary of results obtained by simulating the final best solution obtained by the CGA_TT approach for problem LA21.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15x10	11	0.22	1279.249	11	1013.927	1497.299	COVERT(ND)
15x10	25	0.49	1258.602	9	971.286	1368.973	Biased-RANDOM(ND)
15x10	11	0.22	1277.601	9	1009.629	1511.519	Biased-RANDOM(ND)
15x10	12	0.22	1249.007	9	977.428	1368.801	Biased-RANDOM(ND)
15x10	11	0.22	1226.564	8	965.896	1007.412	RANDOM(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
15x10	3.082	37.500	3.798	21.543
15x10	4.584	0.000	0.403	2.449
15x10	3.066	12.500	3.973	24.654
15x10	0.906	12.500	0.794	15.602
15x10	0.807	0.000	0.606	15.474

Table H.12. Summary of results obtained by simulating the final best solution obtained by the CGA_SIM approach for problem LA21.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15x10	11	0.22	1241.004	8	976.83	1231.91	WSPT+WOST(ND)
15x10	11	0.22	1319.063	9	967.387	1336.242	Biased-RANDOM(ND)
15x10	11	0.22	1239.6	8	971.052	1212.57	RANDOM(ND)
15x10	11	0.22	1237.795	8	969.727	1184.059	JST(ND)
15x10	11	0.22	1236.545	8	960.08	1191.837	Biased-RANDOM(ND)

90 Percent confidence interval

Problem size	Makespan		Number Tardy		Average flow time		Total Tardiness	
15x10	1128.185	1353.823	7.273	8.727	888.027	1065.633	1119.918	1343.902
15x10	1199.148	1438.978	8.182	9.818	879.443	1055.331	1214.765	1457.719
15x10	1126.909	1352.291	7.273	8.727	882.775	1059.329	1102.336	1322.804
15x10	1125.268	1350.322	7.273	8.727	881.570	1057.884	1076.417	1291.701
15x10	1124.132	1348.958	7.273	8.727	872.800	1047.360	1083.488	1300.186

Table H.13. Summary of results obtained by simulating the final best solution obtained by the CGA_WSPT approach for problem LA25.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15x10	25	0.44	1174.343	7	907.783	951.044	RANDOM(ND)
15x10	19	0.39	1330.665	5	884.953	852.01	RANDOM(ND)
15x10	28	0.49	1172.62	8	915.093	1012.242	MODD(ND)
15x10	30	0.55	1290.549	6	921.266	1132.23	JST(ND)
15x10	17	0.33	1286.116	6	906.009	950.422	MODD(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
15x10	0.201	0.000	0.229	6.384
15x10	3.731	16.667	2.081	1.534
15x10	3.978	14.286	0.894	10.604
15x10	6.794	0.000	3.426	45.239
15x10	7.232	0.000	1.417	24.440

Table H.14. Summary of results obtained by simulating the final best solution obtained by the CGA_TT approach for problem LA25.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15x10	27	0.49	1179.467	7	908.574	978.29	ATC(ND)
15x10	24	0.49	1140.697	9	928.719	993.595	Biased-RANDOM(ND)
15x10	36	0.66	1141.13	9	937.18	1089.149	ATC(ND)
15x10	30	0.61	1139.83	9	926.073	957.079	LRPT(ND)
15x10	22	0.44	1143.975	7	894.271	868.435	MODD(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
15x10	0.638	0.000	0.317	9.432
15x10	11.077	50.000	2.762	18.406
15x10	6.557	28.571	3.329	19.008
15x10	5.678	50.000	3.966	22.771
15x10	4.619	16.667	0.103	13.706

Table H.15. Summary of results obtained by simulating the final best solution obtained by the CGA_SIM approach for problem LA25.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15x10	11	0.22	1171.99	7	905.707	893.971	Biased-RANDOM(ND)
15x10	11	0.22	1282.798	6	903.759	839.141	Biased-RANDOM(ND)
15x10	11	0.16	1221.203	7	906.983	915.192	Biased-RANDOM(ND)
15x10	11	0.22	1208.444	6	890.747	779.563	Biased-RANDOM(ND)
15x10	11	0.22	1199.373	6	893.349	763.758	WSPT+WOST(ND)

90 Percent confidence interval

Problem size	Makespan		Number Tardy		Average flow time		Total Tardiness	
15x10	1065.445	1278.535	6.364	7.636	823.370	988.044	812.701	975.241
15x10	1166.180	1399.416	5.455	6.545	821.599	985.919	762.855	915.427
15x10	1110.185	1332.221	6.364	7.636	824.530	989.436	831.993	998.391
15x10	1098.585	1318.303	5.455	6.545	809.770	971.724	708.694	850.432
15x10	1090.339	1308.407	5.455	6.545	812.135	974.563	694.325	833.191

Table H.16. Summary of results obtained by simulating the final best solution obtained by the CGA_WSPT approach for problem LA38.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15X15	11	0.27	1426.277	10	1157.978	1639.654	Biased-RANDOM(ND)
15X15	11	0.33	1438.849	11	1174.828	1863.863	Biased-RANDOM(ND)
15X15	19	0.44	1499.826	11	1174.543	1832.773	EDD(ND)
15X15	15	0.39	1441.106	11	1187.351	1994.602	MDD(ND)
15X15	13	0.33	1424.848	12	1198.998	2161.14	LRPT(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
15X15	1.096	9.091	0.552	12.671
15X15	0.281	8.333	0.553	1.386
15X15	5.259	8.333	0.244	10.196
15X15	0.195	8.333	1.326	19.768
15X15	3.412	0.000	3.707	51.590

Table H.17. Summary of results obtained by simulating the final best solution obtained by the CGA_TT approach for problem LA38.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15X15	13	0.33	1534.48	11	1173.494	1789.46	RANDOM(ND)
15X15	26	0.66	1455.08	12	1178.982	1881.49	MWR(ND)
15X15	14	0.38	1423.996	12	1184.445	1858.496	RANDOM(ND)
15X15	15	0.38	1419.917	12	1168.841	1735.621	Biased-RANDOM(ND)
15X15	11	0.33	1434.953	12	1189.127	1905.281	RANDOM(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
15X15	8.765	0.000	1.900	22.965
15X15	1.413	0.000	0.202	2.345
15X15	0.063	0.000	1.090	11.743
15X15	1.278	0.000	0.253	4.218
15X15	4.145	0.000	2.853	33.643

Table H.18. Summary of results obtained by simulating the final best solution obtained by the CGA_SIM approach for problem LA38.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15X15	11	0.27	1410.821	11	1151.618	1455.264	Biased-RANDOM(ND)
15X15	25	0.66	1434.812	12	1181.363	1838.386	Biased-RANDOM(ND)
15X15	11	0.28	1424.895	12	1171.679	1663.195	Biased-RANDOM(ND)
15X15	22	0.6	1438.303	12	1171.807	1665.383	Biased-RANDOM(ND)
15X15	11	0.28	1377.84	12	1156.139	1425.649	Biased-RANDOM(ND)

90 Percent confidence interval

Problem size	Makespan		Number Tardy		Average flow time		Total Tardiness	
15X15	1282.565	1539.077	10.000	12.000	1046.925	1256.311	1322.967	1587.561
15X15	1304.375	1565.249	10.909	13.091	1073.966	1288.760	1671.260	2005.512
15X15	1295.359	1554.431	10.909	13.091	1065.163	1278.195	1511.995	1814.395
15X15	1307.548	1569.058	10.909	13.091	1065.279	1278.335	1513.985	1816.781
15X15	1252.582	1503.098	10.909	13.091	1051.035	1261.243	1296.045	1555.253

Table H.19. Summary of results obtained by simulating the final best solution obtained by the CGA_WSPT approach for problem LA40.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15X15	16	0.44	1422.082	12	1205.19	1757.294	Biased-RANDOM(ND)
15X15	19	0.5	1435.509	12	1182.14	1501.492	Biased-RANDOM(ND)
15X15	18	0.49	1487.657	12	1209.293	1871.847	RANDOM(ND)
15X15	22	0.55	1483.567	13	1215.632	1900.967	A/OPN(ND)
15X15	17	0.44	1399.145	10	1171.435	1357.204	RANDOM(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
15X15	0.129	20.000	3.547	35.709
15X15	1.367	0.000	1.148	14.021
15X15	3.695	0.000	2.348	26.831
15X15	5.851	18.182	4.235	54.870
15X15	2.378	9.091	0.306	2.531

Table H.20. Summary of results obtained by simulating the final best solution obtained by the CGA_TT approach for problem LA40.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15X15	22	0.61	1395.379	12	1196.604	1599.825	Biased-RANDOM(ND)
15X15	26	0.71	1391.05	12	1186.663	1543.734	Biased-RANDOM(ND)
15X15	28	0.77	1407.151	12	1193.629	1579.636	RANDOM(ND)
15X15	32	0.82	1409.616	12	1188.354	1538.251	RANDOM(ND)
15X15	19	0.5	1406.42	12	1179.505	1440.542	Biased-RANDOM(ND)

Percentage of error

Problem size	Makespan	Number Tardy	Average flow time	Total Tardiness
15X15	1.751	20.000	2.809	23.548
15X15	1.772	0.000	1.535	17.229
15X15	1.917	0.000	1.022	7.032
15X15	0.574	9.091	1.896	25.320
15X15	1.870	9.091	0.381	3.454

Table H.21. Summary of results obtained by simulating the final best solution obtained by the CGA_SIM approach for problem LA40.

Problem size	Number of replicates	CPU time (Sec.)	Makespan	Number Tardy	Average flow time	Total Tardiness	The origin of the best chromosome
15X15	11	0.27	1420.25	10	1163.909	1294.898	Biased-RANDOM(ND)
15X15	11	0.33	1416.144	12	1168.725	1316.857	Biased-RANDOM(ND)
15X15	19	0.5	1434.649	12	1181.55	1475.86	RANDOM(ND)
15X15	11	0.27	1401.565	11	1166.247	1227.459	Biased-RANDOM(ND)
15X15	17	0.5	1433.224	11	1175.026	1392.447	Biased-RANDOM(ND)

90 Percent confidence interval

Problem size	Makespan		Number Tardy		Average flow time		Total Tardiness	
15X15	1291.136	1549.364	9.091	10.909	1058.099	1269.719	1177.180	1412.616
15X15	1287.404	1544.884	10.909	13.091	1062.477	1274.973	1197.143	1436.571
15X15	1304.226	1565.072	10.909	13.091	1074.136	1288.964	1341.691	1610.029
15X15	1274.150	1528.980	10.000	12.000	1060.225	1272.269	1115.872	1339.046
15X15	1302.931	1563.517	10.000	12.000	1068.205	1281.847	1265.861	1519.033