

**Dynamic Flexible Job Shop Scheduling with Simulation
Optimization by Using Genetic Algorithm**

M. Sc. Thesis

In

Industrial Engineering

University of Gaziantep

Supervisor

Assist. Prof. Dr. Faruk GEYİK

by

Ayşe Tuğba DOSDOĞRU

July 2012

©2012[Ayşe Tuğba DOSDOĞRU].

T.C.
UNIVERSITY OF GAZİANTEP
GRADUATE SCHOOL OF
NATURAL & APPLIED SCIENCES
(INDUSTRIAL ENGINEERING DEPARTMENT)

Name of the thesis: Dynamic Flexible Job Shop Scheduling With Simulation
Optimization by Using Genetic Algorithm

Name of the student: Ayşe Tuğba Dosdoğru

Exam date: 17.07.2012

Approval of the Graduate School of Natural and Applied Sciences


Prof. Dr. Ramazan KOÇ
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of
Master of Science

Prof. Dr. Trkay DERELİ
Head of Department



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

Assist. Prof. Dr. Faruk GEYİK
Supervisor



Examining Committee Members

Title and Name-Surname

Prof. Dr. Trkay DERELİ

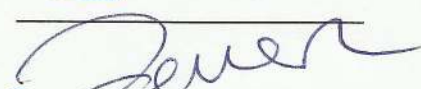
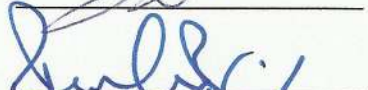
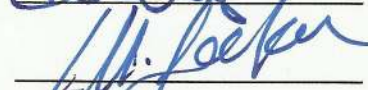
Assoc. Prof. Dr. Serap U. Sekiner

Asst.Prof. Dr. Faruk GEYİK

Asst.Prof. Dr. Mustafa GKEN

Asst.Prof. Dr. mer AKGBEK

Signature


I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name and Last name

Ayze Tugba DOSDOGRU

Signature



ABSTRACT

Dynamic Flexible Job Shop Scheduling with Simulation Optimization by Using Genetic Algorithm

DOSDOĞRU, Ayşe Tuğba

M. Sc. in Industrial Eng.

Supervisor: Assist. Prof. Dr. Faruk GEYİK

July 2012, 97 pages

In most real life manufacturing problems, certain operation of a part can be processed on more than one machine which makes the considered system (i.e. job shops) flexible. On one hand, flexibility provides alternative part routings which most of the time relaxes shop floor operations. On the other hand, increased flexibility makes operation machine pairing decisions (i.e., the most suitable part routing) much more complex. Thus, manufacturing systems must be scheduled by considering the flexibility to improve effectiveness and performance.

The aim of the study is to develop a system that generates the best feasible part routings in a dynamic flexible job shop scheduling environment. For this purpose both the best feasible process plan for each part and the best feasible machine for each operation in a dynamic flexible job shop scheduling environment must be determined, respectively. In this respect, a genetic algorithm is adapted to determine best part processing plan for each part and then select appropriate machines for each operation of each part according to the determined part processing plan. Genetic algorithm solves to the optimization phase of solution methodology. Then these machine-operation pairings are utilized by discrete-event system simulation model to estimate their performances. These two phases of the study follow each other iteratively. The goal of the proposed methodology is to find the solution that minimizes total of average flow times for all parts. The results show that the objective function improves as the considered level of flexibility increases.

Keywords: Flexible job shop scheduling, genetic algorithm, simulation optimization.

ÖZ

Genetik Algoritma Kullanılarak Benzetim Optimizasyonlu Dinamik Esnek Üretim Atölyesi Çizelgeleme

DOSDOĞRU, Ayşe Tuğba
Yüksek Lisans Tezi, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Yrd. Doç. Dr. Faruk GEYİK
Temmuz 2012, 97 sayfa

Gerçek hayat üretim problemlerinin çoğunda, bir parçanın belirli bir operasyonu birden fazla makinede işlenebilmektedir. Bu opsiyon göz önüne alınan sisteme (örn., atölye tipi üretim sistemi) esneklik katmaktadır. Esneklik bir taraftan üretim süreçlerini rahatlatan alternatif parça rotaları sağlarken diğer bir taraftan da parça-makine seçimlerinde (örn., en uygun parça rotasının seçimi) karmaşıklığa sebep olabilmektedir. Etkinliğin ve verimliliğin artırılması için, üretim sistemleri esneklik göz önüne alınarak çizelgelenmelidir.

Ele alınan çalışmanın amacı, bir dinamik esnek atölye tipi üretim ortamında olurlu en iyi parça rotalarının üretilmesini sağlayan bir sistem geliştirmektir. Bu amaçla, bir dinamik esnek atölye tipi üretim ortamında, sırasıyla hem her bir parça için olurlu en iyi proses planı hem de her bir operasyon için en iyi olurlu makinenin belirlenmesi gerekmektedir. Bu bağlamda, her bir parça için en iyi parça proses planı ve sonra belirlenen parça proses planına göre her bir parçanın her bir operasyonu için uygun makinenin seçilmesini sağlayan genetik algoritma geliştirilmiştir. Geliştirilen algoritma çözüm metodolojisinin optimizasyon aşamasını oluşturmaktadır. Elde edilen makine-operasyon ikilileri, performanslarının tahmini amacıyla, kesikli olay sistem simülasyonu modeline beslenmiştir. Çalışmanın bu iki aşaması birbirini ardışık bir şekilde izlemektedir. Önerilen metodolojinin amacı tüm parçalar için toplam ortalama akış süresini minimize eden çözümü bulmaktır. Sonuçlar, amaç fonksiyonunun göz önüne alınan esneklik seviyesinin artan değerleri için iyileştildiğini göstermiştir.

Anahtar Kelimeler: Esnek üretim atölyesi çizelgeleme, genetik algoritma, benzetim optimizasyonu.

ACKNOWLEDGMENT

I would like to sincerely thank to my supervisor, Assist. Prof. Dr. Faruk GEYİK to his guidance, advice, suggestions and encouragement throughout the study. His invaluable help of constructive comments and suggestions throughout the experimental and thesis works have contributed to the success of this research.

My thanks go to other members of my thesis committee, Prof. Dr. Trkay DERELİ, Assoc. Prof. Dr. Serap Ulusam SEÇKİNER, and Assist. Prof. Dr. mer AKGBEK. I am much indebted for using their precious times to read this thesis and gave their valuable criticisms about it.

I am deeply grateful to Assist. Prof. Dr. Mustafa GÇKEN for his important support and crucial contribution, which made him a backbone of this research and so to this thesis.

Finally, very special thanks to my family for their inseparable support and prayers.

CONTENTS

	Page
ABSTRACT	v
ÖZ	vii
ACKNOWLEDGMENT	vii
CONTENTS	viii
LIST OF TABLES	xi
LIST OF FIGURES	xii
ABBREVIATIONS	xiv
CHAPTER 1:	
INTRODUCTION	1
1.1. Introduction.....	1
1.2. Aim of the Thesis	3
1.3. Outline of the Thesis	4
CHAPTER 2:	
LITERATURE REVIEW AND BACKGROUND	6
2.1. Review of Job Shop Scheduling.....	6
2.2. Review of Flexible Job Shop Scheduling	12
2.3. Shop Configurations and Scheduling Environments	16
2.4. Problem Definition	18
2.4.1 Job Shop Scheduling Problem	18
2.4.2 Flexible Job Shop Scheduling.....	19
2.5. Objective Functions in Scheduling	20

CHAPTER 3:

METHODOLOGY	23
3.1. Scheduling Approximations	23
3.1.1 Genetic Algorithm.....	26
3.1.1.1 Representation	26
3.1.1.2 Fitness Function	27
3.1.1.3 Selection	27
3.1.1.4 General Crossover Procedures.....	28
3.1.1.5 General Mutation Procedures	28
3.2. Simulation Optimization.....	29
3.3. Experimental Design.....	32

CHAPTER 4:

SOLVING THE FLEXIBLE JOB SHOP SCHEDULING PROBLEM BY USING GENETIC ALGORITHM.....	33
4.1. Problem Formulation	33
4.2. Proposed GA for FJSSP	35
4.2.1. Chromosome Representation	35
4.2.2. Decoding process.....	37
4.2.3. Initial Population	38
4.2.4. Generation of new candidate solutions	40
4.2.5. Crossover	40
4.2.6. Mutation	42
4.3. Applying the proposed GA to Benchmark Problems and Computational Results	44
4.4. Experimental Design and Analysis.....	45
4.4.1. Framework of the proposed GA for the Experimental Design	47
4.5. Results of Analysis	48

CHAPTER 5:

APPLICATION OF SIMULATION OPTIMIZATION BY USING GA TO DYNAMIC FJSSP	56
5.1. System Characteristics and Problem Definition.....	56
5.1.1. System Characteristics	57
5.1.2. Problem Definition of dynamic FJSSP	58
5.2. Proposed GA for dynamic FJSSP.....	60
5.2.1. Chromosome Representation for the Considered Problem.....	60
5.2.2. Generation of Initial Population for the Considered Problem	60
5.2.3. Crossover for the proposed GA.....	61
5.2.4. Mutation for the Proposed GA	62
5.3. The methodology of Optimization via Simulation	62
5.3.1. Problem Formulation	62
5.4. Computational Results and Discussions	64

CHAPTER 6:

CONCLUSION	68
REFERENCES	71
APPENDIX A	79
APPENDIX B.....	96

LIST OF TABLES

LIST OF TABLES	Page
Table 2.1. Characteristics of static and dynamic environment.....	18
Table 2.2. An example of JSSP	19
Table 2.3. An example of FJSSP	20
Table 3.1 Summary of other dispatching rules.....	24
Table 4.1 Operation processing time table with deterministic times of FJSSP	35
Table 4.2 Operation processing times with stochastic times	35
Table 4.3 Computational results and comparisons	45
Table 4.4 Part $\times$ Machine Groups.....	46
Table 4.5 Number of operation levels.....	46
Table 4.6 Flexibility levels for each machine sizes	46
Table 4.7 Best makespans over five runs.	47
Table 4.8 Analysis of Variance for Makespan	48
Table 5.1 Part-operation plans data	59
Table 5.2 Machine-operation suitability data.....	59
Table 5.3 Processing times distributions for each part	60
Table 5.4 Operation sequences and machine numbers of best results	66

LIST OF FIGURES

LIST OF FIGURES	Page
Figure 3.1 A binary chromosome representation	27
Figure 3.2 An example of uniform crossover operator.....	28
Figure 3.3 An example for inversion mutation	29
Figure 4.1 Chromosome structure of proposed GA.....	36
Figure 4.2 Machine and processing time matrix used in decoding process.....	37
Figure 4.3 An example Gantt chart to set the operation into the idle time.	38
Figure 4.4 An example Gantt chart to set the operation to the end of the machine. ..	38
Figure 4.5 Illustration of crossover operations.....	42
Figure 4.6 Illustration of mutation operation which selects a machine with a shorter processing time	43
Figure 4.7 Illustration of mutation operation for machine selection and operation sequence parts.....	43
Figure 4.8 Comparison between random strategy and combined strategy.	48
Figure 4.9 Interaction Effect of Part Number, Machine Number, and Flexibility Level on Makespan.....	50
Figure 4.10 Interaction Effect of Part Number, Operation Number, and Flexibility Level on Makespan.....	51
Figure 4.11 Interaction Effect of Machine Number, Operation Number, and Flexibility Level on Makespan.....	52
Figure 4.12 Interaction Effect of Part Number and Flexibility Level on Makespan..	53
Figure 4.13 Interaction Effect of Operation Number and Flexibility Level on Makespan	54
Figure 4.14 Interaction Effect of Part Number, Machine Number, Operation Number, and Flexibility Level on Makespan	55
Figure 5.1 Illustration of the proposed chromosome representation	60

Figure 5.2 An illustration of crossover operator.....	61
Figure 5.3 An illustrative of mutation operator.....	62
Figure 5.4 Optimization via Simulation Methodology	64
Figure 5.5 Total of average flow times obtained with full flexibility.....	67
Figure 5.6 Total of average flow times results obtained with partial flexibility	67

ABBREVIATIONS

JSSP	Job Shop Scheduling Problem
FJSSP	Flexible Job Shop Scheduling Problem
GA	Genetic Algorithm
PSO	Part Swarm Optimization
ACO	Ant Colony Optimization
BA	Bee Algorithm
AIS	Artificial Immune Systems
VNS	Variable Neighbourhood Search
POX	Precedence preserving order based crossover
MC	Monte Carlo
FIFO	First in First Out
SPT	Shortest Processing Time
MST	Minimum Slack Time
SIRO	The Service in Random Order
ERD	The Earliest Release Date First
EDD	The Earliest Due Date First
WSPT	The Weighted Shortest Processing Time First
AL	Approach by Localization
ANOVA	Analysis of Variance

CHAPTER 1

INTRODUCTION

1.1. Introduction

Scheduling is an important process in the planning and managing of manufacturing and service systems which deals with allocation of resources to tasks in a limited time period. Pinedo (2008) explains the scheduling in manufacturing systems as *“Orders that are released in a manufacturing setting have to be translated into jobs with associated due dates. These jobs often have to be processed on the machines in a workcenter in a given order or sequence”*.

One of the most well known and studied problem is *Job Shop Scheduling Problem* (JSSP) in which finding optimum solution is still a challenging problem. JSSP deals with sequencing operations of a set of jobs on a set of machine which one is specific for an operation. If an operation can be processed at more than one machine, the problem converts to *Flexible Job Shop Scheduling problem* (FJSSP). FJSSP deals with two sub-problems. One of them is assigning operations to machines selected out of alternative machines and the other one is sequencing of operations on each machine. So, FJSSP becomes more difficult to find a feasible solution due to consider these two sub-problems (Zhang, et al., 2011).

JSSPs and FJSSPs are handled with different forms of parameters and in varied scheduling environments. Scheduling environments can be classified into two main categories as *static* and *dynamic*. In the static scheduling environment, all shop parameters are predetermined and already known. All jobs are released to the shop floor at once and, jobs' processing times are known at the beginning. In the static scheduling environment, jobs' processing times can be defined either deterministic or

stochastic. In the dynamic scheduling environment, jobs are released to the shop floor constantly. Processing times are to be uncertain and unexpected events such as machine breakdowns and/or preemptions can occur at an arbitrary point in time.

Numerous methods has been proposed to cope with sequencing and scheduling problems to date. These methods can be classified into two main categories: (1) Exact and (2) Approximation methods (Bondal, 2008). Exact methods, such as branch and bound, linear programming and integer programming have been developed to small-sized problem and guarantee global optimum. However, analytical methods cannot find the optimum when the problem size increased. Therefore, a large part of studies pay attention to approximation methods such as heuristics and artificial intelligence techniques to cope with large-sized problems. They could produce reasonably good schedules in a reasonable computational time, and could get near optimal solution easily (Zhang, et al., 2011). The intelligent optimization algorithms such as genetic algorithm (GA), particle swarm optimization (PSO) and ant colony (ACO) are relatively easy to implement and they could conveniently be adapted for different kinds of scheduling problems. For this reason, they become progressively popular in recent years (Zhang, et al., 2010).

GA is one of the most used meta-heuristic algorithms and has been applied successfully in the area of scheduling. GA is developed by John Holland in 1960s, and David Goldberg who is the one of his student, is applied a GA to the control of gas-pipeline transmission successfully (Yamada, 2003). In recent years, GA is applied to the FJSSP successfully with different GA procedures and hybrid approaches (Kacem, et al., 2002, Pezzella, et al., 2008, Zhang, et al., 2011).

In spite of being a powerful method to solve wide range of scheduling problems, GA's usage for solving real life problems is inadequate. This inability can mainly attributed to uncertainties in the structure of real-life problems. In this case, other approaches which successfully consider the uncertainties of real life problems should also be taken into account together with meta-heuristics.

In general, real life systems are often said to be probabilistic and models of such systems are known to be probabilistic models as well. It is well known that simulation is an indispensable tool for analyzing such complex real life systems. Also, more recently simulation together with meta-heuristics is an appropriate tool

optimizing such complex systems. Fu (2002) reported that optimization and simulation is become more popular and there is a large body of research literature relevant to combining them. Simulation optimization helps to find the values for the input parameters while optimizing the performance of the considered system (Medaglia, 2000).

1.2. Aim of the Thesis

The main purpose of this study is to develop an effective methodology for minimizing total average flow time of all parts in a dynamic and stochastic FJSSP. Achieving an optimal or near optimal solution is a challenging problem when considered system dynamic and stochastic. While meta-heuristic algorithms have been applied for static scheduling problems successfully, the complexity of the problem increases as the size of the problem increases. This thesis proposes a simulation optimization methodology to cope with the solution complexity of the dynamic and stochastic FJSSP regardless of the size of the problem. As the name suggests, the proposed methodology is composed of two consecutive stages; (1) Simulation, and (2) Optimization. Owing to simulation stage the complexity of the real life problem can be easily modelled. Besides, owing to optimization stage, various combinations of input variables can be generated to be tested in simulation model of the real system. By this way, obtaining high quality solutions in a reasonable solution time is aimed.

In optimization stage, a GA is needed for optimization phase of the proposed method. For this purpose, a GA is developed for static stochastic FJSSP. The processing times are considered to be stochastic to make the problem more realistic. A full factorial design is utilized to evaluate the performance of the stochastic flexible job shop and investigate relationship between the considered factors (parts, jobs, operations, and flexibility). 81 experiments are generated to investigate all factor level combinations.

Finally, the GA is adopted to dynamic and stochastic FJSSP and integrated with simulation. The integration of GA and simulation can be considered the most important part of the study. GA accounts for optimization phase of the simulation optimization methodology. Then the outputs of the GA are fed into the simulation as inputs to estimate the performance of the system.

1.3. Outline of the Thesis

In the context of this thesis, different types of FJSSP is considered. First, a GA is adapted to solve the general FJSSP and then adapted for use in different type of scheduling environments. The detailed explanation is given below.

First, a GA is proposed to solve static deterministic FJSSP. Initial population is generated by mixed initial strategies and, various crossover and mutation procedures are applied. The algorithm is tested with most used benchmark problems in the literature.

Then, static deterministic FJSSP is converted to static stochastic FJSSP type. Here, processing time is sampled from predetermined probability distribution. An experimental design is composed to evaluate the impact of flexibility level to the shop performance.

Finally, the system considered as dynamic stochastic FJSSP. The operation processing times are considered to be random in order to represent real life problems more closely which is the same as in the second section. Additionally, to provide the system being dynamic, jobs arrive to the shop constantly. Thesis outline is organized as follows:

A literature review of JSSPs and FJSSPs and background of scheduling problems are given in chapter 2. Applied methods and problem types are emphasized briefly. The problem definitions of JSSP and FJSSP are provided here. Shop configurations and scheduling environments are explained and the most commonly used objective functions are given. The most commonly used scheduling approximations in the literature and the general information about the methodologies used in our study is given chapter 3.

In chapter 4, the FJSSP is solved by using GA. The detailed explanations of the main framework of the GA and the procedures for GA are given. The algorithm is tested with well known benchmark problems from the literature and the results are given in this section. Afterwards, an experimental design is utilized for stochastic FJSSP. The problem statement and the GA are explained and the results of the analysis are given in this section.

Finally, simulation optimization methodology for dynamic FJSSP is presented in chapter 5. System characteristics and problem definition together with the

methodologies for simulation optimization is given. Computational results are also given in this section.

Chapter 6 contains the conclusions and the recommendations for future studies.

CHAPTER 2

LITERATURE REVIEW AND BACKGROUND

JSSP is studied in a great number of researches for several decades. Due to large number of studies, studies after millennium are reviewed. Therefore, the most recent methods, problem types and problem environments were taken into consideration. In this respect, a comprehensive review of JSSP and FJSSP literature are given, respectively.

2.1. Review of Job Shop Scheduling

Most of the JSSPs are handled in a static environment with known processing times, and ignored machine breakdowns or unexpected events.

Chong, et al. (2006) handled the JSSP as deterministic JSSP. There are a set of machines and a set of jobs. Each job has a pre-determined operation sequence. And, each operation can be processed on a determined machine within a processing time period. The objective function is to minimize the makespan (C_{max}) and a disjunctive graph is used to represent the JSSP. They proposed a honey bee colony algorithm to solve JSSP. The honey bee colony algorithm is tested on 82 benchmark problems. They used Ant Colony (ACO) and Tabu Search (TS) algorithm to compare their algorithm. Their algorithm is found to be comparable to ACO, but TS is found to be more efficient.

Sha, et al. (2006) used a hybrid PSO for deterministic JSSP. They used Giffler and Thompson's heuristic to decode a particle position into a schedule. TS is also applied for comparison purposes. All considered heuristics are applied to same set of benchmark problems. The computational results showed that the hybrid PSO performs better than the general PSO and the other traditional meta-heuristics.

Lian, et al. (2006) is used PSO to solve deterministic JSSP with some new algorithm operators. Makespan (C_{max}) is used as objective function. They used simulation to evaluate the algorithm with three instances. The results are compared with traditional GA. The results revealed that their algorithm is more efficient than the traditional GA.

Zhang, et al. (2007) proposed a TS with a new enhanced neighbourhood structure for JSSP. This neighbourhood structure can obtain a new set of neighbour solutions by a small mutation to a given solution. The objective is to find a solution with a minimum makespan. The computational results show that their proposed TS gives better solutions than an estimation approach and an exact evaluation approach. Geyik, et al. (2004) investigated the parameters and strategies of tabu search such as initial solution, neighbourhood structure, tabu list, aspiration criterion, elite solutions lists, intensification, diversification and the number of iteration. Benchmark problems are solved with tabu search to test the parameters and strategies.

Essafi, et al. (2008) solved deterministic JSSP using a genetic algorithm combined with an iterated local search heuristic. The minimization of the total weighted tardiness considered as objective function. Their study proved that using an iterative local search considerably enhanced the quality of genetic algorithms and decrease the role of the schedule builder.

Lei (2008) proposed a Pareto archive particle swarm optimization and, the objective of the study is to minimize both makespan and total tardiness of jobs simultaneously. Pareto is an archive system which maintains optimal solutions to a multi-objective problem. Their proposed algorithm is tested on 18 benchmark problems and results showed that the algorithm performs better than a multi-objective particle swarm optimization.

Zhang, et al. (2008) and Pan, et al. (2009) proposed a hybrid GA. Zhang, et al. (2008) constructed a schedule using a new full active schedule procedure based on the operation-based representation. A local search heuristic is used to improve the obtained schedules while getting away from local optimum. In the study, a new crossover operator, called the precedence operation crossover (POX) is used which maintains feasibility of schedule after the crossover. Pan, et al. (2009) used a hybrid GA and determined their problem to be no-wait JSSP. No-wait JSSP is a job shop

with a constraint that a job cannot wait between operations. A local search is applied to solve no-wait JSSP as a sub-problem in GA. Both studies' results showed that hybrid GA performs better than general GAs.

Roshanaei, et al. (2009) proposed a local search-based algorithm called variable neighbourhood search (VNS). In the study, set-up times are also considered which makes JSSP with sequence-dependent set-up times. Besides, the objective is to minimize makespan. VNS uses three advanced neighbourhood search techniques. The VNS achieved effective results in terms of computational time and solution quality than any other well known algorithms.

Fiğlalı, et al. (2009) proposed an ACO algorithm to solve general JSSP. ACO parameters are set by an experimental design on different sized and randomly generated JSSPs. The effect of ACO parameters to the job shop performance is investigated by the results. The results revealed that, ACO parameters have a significant effect on makespan value.

Manikas, et al. (2009) used GA to solve multi-criteria sequence-dependent JSSP. Apart from the literature, sequence-dependent setup times, staggered release dates (i.e., all jobs are not release to the system at the same time) and recirculation are included to JSSP. They used three criteria for objective function: makespan, earliness, tardiness and costumer and/or job rank. The fitness function is evaluated by using weights which are determined for each criterion. These weights are determined according to a manufacturer's needs. Due to being a novel problem in the literature, they did not have the chance to compare their solution quality with benchmark problems. Thus, an experimental design is used to test the different levels of the factors considered in the study.

A hybrid immune simulated annealing algorithm is proposed by Zhang, et al. (2010) in which the objective is to minimize total weighted tardiness. This study focused on the *bottleneck jobs* which processing sequence have a significant effect on the job performance and these jobs needs more intensive optimization. A fuzzy inference system is designed for evaluating the bottleneck level to determine the bottleneck job distribution. The main logic behind using the fuzzy inference system is to take variety in job shops into account where critical jobs can be changed. Simulated annealing is used to determine solution space and the immune mechanism is applied

to increase the solution quality. Computational results showed that the proposed approach is effective and robust.

Hybrid algorithms are studied in recent years progressively. Lian (2010) combined a local and global search while using PSO and to minimize the makespan. A local search is used to determine best position with the lowest fitness particle in the swarm in a particle (local search) and to find the best position of the whole colony (global search). Meeran, et al. (2011) combined GA and TS to solve a real life JSSP. First GA starts with a set of initial solutions and TS improves these solutions. Then, GA continues with the solutions obtained from TS. Cheng, et al. (2011) solved the multi-objective JSSP with a two stage algorithm which combines a dispatching rule based memetic algorithm in the first stage and a re-optimization procedure of shifting bottleneck in the second stage. Zhang, et al. (2012) used a hybrid artificial bee algorithm where a tree-based local search procedure is embedded into bee algorithm to enhance the solution quality. All these studies showed that hybrid algorithms are much more effective than the pure ones.

As clear from the above discussion, considerable amount of studies considered the JSSP as being static. In these studies, all jobs are ready at the beginning and random events are ignored. However, there are always random and unexpected events in real life systems (Fang, et al., 1997). To analyze real life problems accurately and reasonably these uncertainties should be considered.

Yoshitomi, et al. (2003) used a GA and the Monte Carlo (MC) method for stochastic JSSP. Job processing times are considered to be random variables that have stochastic distribution functions and the objective functions. Thus, the objective function is specified as expected value of makespan. First, GA is applied to stochastic JSSP. Crossover operator based on Giffler & Thompson's algorithm (Giffler & Thompson, 1960) is applied. Then, the solutions with higher frequencies are selected as best solutions and, MC is applied to find optimum schedule among these best solutions. Results showed that this GA-MC method needs less time than MC. Similarly, Lei, et al. (2008) solved JSSP with stochastic processing time using GA. Processing times are sampled from normal distribution. A new permutation-based representation method is used and the objective function is to minimize the makespan.

Gu, et al. (2009) proposed a new Quantum GA to solve stochastic JSSP with the objective of minimizing the expected value of makespan. The processing times of each job is sampled from normal distribution. Gu, et al. (2009) proposed a new parallel quantum GA. This new scheduling algorithm provides high convergence performance to either optimum or near optimum for stochastic JSSP. Gu, et al. (2010) proposed a competitive co-evolutionary quantum GA. New strategies called competitive hunter, cooperative surviving and the big fish eating small fish are developed in population growth process. The results revealed that this algorithm provides better performance than both quantum based GA and standard GA.

Fuzzy processing time and fuzzy due date are studied by Hu, et al. (2011) using differential evolution algorithm. A special fuzzy number defined for processing times and due dates. Then, the analytical formulas are determined for these objective functions. A fuzzy ranking concept is used to investigate the relation between fuzzy completion time and fuzzy due date. According to the experimental design results, this method is found comparable with state-of-the-art methods. The smallest makespan found in this study is not the best but reasonable for due dates of each jobs.

Lei (2011a) and Lei (2012) used interval theory (a special fuzzy number) to determine processing times for JSSP. Lei (2011a) applied a population based neighbourhood search (PNS) to find minimum interval makespan. It is reported that sometimes it can be difficult to determine an appropriate membership function or a probability distribution. That is why, interval theory is used in this study. The proposed algorithm is compared with particle swarm optimization with genetic operators (GPSO) and Simulated Annealing (SA). The results showed that the PNS is better than GPSO and SA. Lei (2012) used operation-based GA with interval theory. These studies claimed that interval theory is more easier to obtain interval processing time and to build schedule.

Lei (2011b) dealt with stochastic JSSP considering random breakdowns and repairs. GA is used to solve this problem and the objective is to minimize the stochastic makespan. Processing times and machine breakdowns are assumed to be random variables with exponential distribution. Random key representation is used to cope

with random events. It is reported that the proposed GA gives successful results for stochastic JSSP.

A multi-objective stochastic JSSP with exponential distribution is investigated by Lei (2011c). The study deals with two objective function: makespan and total tardiness. An effective ordered operation-based GA is proposed. Successful results are obtained with adopting a simplified external archive updating strategy and an efficient crossover.

A hybrid differential evolution algorithm Zhang, et al. (2012) and a two stage hybrid PSO Zhang, et al. (2012) used for stochastic JSSP with expected total tardiness objective function. Both of these studies, at first stage, applied meta-heuristics to stochastic JSSP with independent random variables with known expectation ($E(p_{jk})$) and variance ($var(p_{jk})$) where j denotes to jobs and k denotes to machines. At second stage, simulation is used to find optimum solution among a set of best solution obtained from the first stage.

The studies mentioned above are all related to *static* and/or *stochastic* JSSP. Besides them, some studies are handled *dynamic and/or stochastic* JSSP to represent real life problems more closely. Zhou, et al. (2009) studied ACO to measure its performance in a dynamic JSSP. Processing times, release dates and the job routes are generated randomly with different ranges. ACO is tested in different levels of machine utilization, different processing time distributions and different performance measures. These performance measures are determined as mean flow time, mean tardiness and total throughput. Different dispatching rules are applied to compare each other. These are FIFO, SPT and MST (minimum slack time). The computational results showed that ACO can perform well when the machine utilization is low and, when the variation of processing times is small.

Zandieh, et al. (2010) proposed a variable neighbourhood search (VNS) to solve dynamic JSSP with random job arrivals and machine breakdowns. The objective of the study is to minimize the mean flow time. The time between arrivals, mean time between failure and the mean time to repair are set as exponential distribution. The parameters of VNS are adjusted at any rescheduling point by Artificial Neural Network to increase efficiency and effectiveness. The proposed method compared with the SPT, FIFO and LIFO dispatching rules via simulation. And the results

showed that the proposed method performs better than the dispatching rules. Adibi, et al. (2010) studied an extended version of Zandieh, et al. (2010)'s study. A multi-objective scheduling in a dynamic JSSP is presented and a VNS is used. A multi-objective function composed of weighted makespan and tardiness is proposed.

2.2. Review of Flexible Job Shop Scheduling

Kacem, et al. (2002) applied two approaches to solve machine assignment problem and scheduling problem to minimize both makespan and total workload. First, they proposed an approach to assign each operation to a suitable machine while considering processing times and machine workloads called approach by localization (AL). Two assignment procedures are proposed for machine assignment. Assignment 1 is proposed to select a job with minimum processing time. At the end of this search step, found processing time is added to each machine workload. Assignment 2 is implemented in the same way as in the Assignment 1 procedure jobs are selected at random. Therefore, different solutions could be obtained in each run of the algorithm. Initial solutions are obtained with mixing these assignment procedures. Second, a GA is applied to generate new solutions from initial solutions obtained by AL. The results showed that using AL with GA enables better solutions. Another study which uses Kacem, et al. (2002)'s AL algorithms is Pezzella, et al. (2008). The AL is used to find initial solutions. The Most Work Remaining, the Most Operation Remaining and random selection is used to sequence these initial solutions. Afterwards, they solved the problems by using GA with different crossover and mutation operators. Computational results, revealed that their algorithms perform better than the others.

A combination of evolutionary algorithms and fuzzy logic is used to solve a Pareto optimality approach for FJSSP by Kacem, et al. (2002). Pareto optimality is a well known concept which related with multi-objective problems. Fuzzy logic is used to find a final set of near optimal solutions for selecting Pareto optimal solutions. After this step, AL and the controlled GA is used for resolution of the problem. The approach is tested with different problems by using simulation. The results showed that the proposed approach provides high quality solutions but does not guarantee the optimality.

Baykasoğlu, et al. (2004) presents a tabu search and grammars method to solve multi-objective FJSSP. The FJSSP data (part process plans, processing requirements and machine tool capabilities etc.) are represented by using context-free grammars. Later, the controls of the grammar is determined. Dispatching rules (Giffler & Thompson algorithm) are used to find feasible schedules. A selection probability is assigned to each machine according to processing times. FJSSP is solved by using tabu search algorithm. The authors claimed that using grammars simplifies solution complexity of the scheduling problems.

PSO and SA are combined to solve multi-objective FJSSP by Xia, et al. (2005). PSO is used to assign operations to machines and SA is used to schedule operations on each machine. The objectives of the study are determined to be minimizing makespan, the total workload of machines and the workload of the critical machine. Each particle's fitness is evaluated by SA, which makes SA a sub-algorithm. The algorithm is compared with other studies' algorithms and the results showed that the proposed method is effective for FJSSP.

Gao, et al. (2007) proposed a hybrid GA with multi-objective FJSSP. A local search procedure: bottleneck shifting is embedded to GA. The local search investigates neighbour solutions to improve each individual before added into the population. A two vector representation composed of machine assignment vector and operation sequence vector is used. The algorithm is tested by using simulation and, it is reported that according to the several computational results the proposed algorithm has superior performance than the other methods.

Liouane, et al. (2007) proposed ant systems and local search for FJSSP. Initial solutions are obtained by mixing dispatching rules (SPT, FIFO, etc.) and random solutions. Tabu search is used to improve solution quality. The results proved that ant systems and tabu search hybridization can find optimal solutions in FJSSPs.

Saidi-Mehrabad, et al. (2007) solved the FJSSP with sequence-dependent setup times by a two stage tabu search algorithm. *Sequence-dependent* setup time means that the setup depends on the previous processed operation on the machine. The first stage deals with the sequencing of operations and the second stage is related to selection of machines from alternative machine sets. Proposed algorithm is compared with branch and bound algorithm, and the computational results show that the proposed

algorithm dominates the branch and bound algorithm in terms of solution quality and time.

Ant colony optimization method is used to solve FJSSP with routing flexibility and separable setup times by Rossi, et al. (2007). Each operation has a setup time period with two independent activities: (1) sequence-dependent setup, (2) sequence-independent setup. The setup time which depends on the previous processed operation is called *sequence-dependent* setup time. However, the setup time which depends on the previous operation in the job routing is called *sequence-independent* setup time. First a disjunctive graph model and local search algorithm is applied to FJSSP with transportation and setup times, and combined them with selection of machines from alternative machine set. Afterwards, ACO is used to improve solution quality. The proposed method is found to be effective according to the benchmark problems.

Fattahi, et al. (2007) modelled a mixed integer linear programming for small-sized FJSSP. This mathematical formulas are coded in a software which uses branch and bound algorithm. The small-sized problems can be solved by this model and good solutions are achieved. But, it is reported that it can be hard to reach optimum solutions for medium and large-sized problems with branch and bound algorithm. Consequently, the upper and lower bounds of medium and large-sized problems obtained by mathematical model. Afterwards, heuristics are used to solve medium and large-sized problems considering pre-found upper and lower bounds. An integrated approach (integrated approach with simulated annealing, ISA and integrated approach with tabu search, ITS) is applied. These integrated approaches consider the assignment and sequencing problems together. Four hierarchical approaches are applied to FJSSPs in the study. These are HSA/SA, HSA/TS, HTS/TS and HTS/SA. These hierarchical approaches solve the assignment and sequencing problems separately. HSA/SA solves the assignment problem with hierarchical approach and SA, and solves the sequencing problem with SA. HSA/TS solves the assignment problem with hierarchical approach and SA, and solves the sequencing problem with TS. HTS/TS is used for the assignment problem with hierarchical approach and TS, and solves the sequencing problem with TS. HTS/SA is applied to the assignment problem with hierarchical approach and TS, and solves the sequencing problem with SA. As a result, hierarchical approaches are found to be

better than integrated approaches; however none of these algorithms can found optimum solutions for medium and large scale size problems. Both HSA/SA and HSA/TS can reach optimum solutions for all small size problems and HSA/TS is found better than the others. For medium and large-sized problems, HTS/SA and HTS/TS perform better than the others.

Zhang, et al. (2009) proposed PSO assigning operations to machines and sequencing of operations, and applied TS algorithm to schedule the problem according to the findings obtained from PSO. The objective is to minimize the makespan, the workload of the critical machine and the total workload of machines simultaneously. The PSO algorithm is hybridized by using GA procedures. A crossover is applied to update each particle. Additionally, a mutation procedure is also applied to enhance the diversity of each particle. An effective GA is proposed with different initial strategies by Zhang, et al. (2011). Global Selection (GS) where sum of the processing times of each machine are recorded and Local Selection (LS) where a machine with minimum processing time for each operation is selected are used to generate high quality solutions. Zhang, et al. (2009) and Zhang, et al. (2011) show that both the hybrid algorithms and local selections improve the solution quality.

Zhang, et al. (2012) expanded the FJSSP with transportation constraints and bounded processing times. In the problem, transportation resources supposed to be available to transport a job from one machine to another machine. All the loaded/empty transportation times are defined as machine dependent. The objective of the study is to minimize the makespan and the storage time which is the total waiting time before and after each machine during the production. Earliest and latest starting times are defined as bounds. A GA with TS is used to solve this problem. GA is used to solve the assignment problem with transportation and TS is used for both finding and improving the appropriate sequence on each machine.

As in JSSPs, FJSSPs can be defined as dynamic and stochastic FJSSPs. Gholami, et al. (2009) considered the breakdowns in FJSSP. An integration of GA and simulation is proposed for finding minimum expected makespan and minimum expected mean tardiness. GA is used for machine assignment problem and sequencing of operations. A simulator is used to evaluate the results obtained by GA. A breakdown algorithm where breakdowns are generated by exponential random numbers is embedded in

simulator algorithm. Al-Hinai, et al. (2011) used a hybrid GA for FJSSP with random breakdowns. The objective of the study is to minimize the effect of breakdowns on job shop performance. A two stage hybrid GA is proposed where at the first stage the makespan optimized with no expected disruptions and at the second stage the multi-objective optimized and the machine assignments and operations sequencing are integrated with the expected machine breakdowns.

Rajabinasab, et al. (2011) proposed a multi-agent based approach in a dynamic FJSSP with alternative process plans. Each job has alternative process plans and each job can be processed on a set of alternative machines. Random job arrivals and random machine breakdowns considered. Several objective functions are considered for measurement of the performance such as flow time based performance measures and due date based performance measures. Two types of agents (job agents and machine agents) are considered in the study. Machine assignment and operation sequencing are performed through negotiation and coordination between these two agents. A manager agent is used to make the coordination between job agent and machine agent easier. This multi-agent based approach is compared with common dispatching rules by using simulation. The proposed approach performs better than the common dispatching rules.

From the literature, it is apparent that FJSSP is much more complicated than the JSSP. Thus, FJSSP is studied less than JSSP. One of the main reasons for such complexity is FJSSP have to deal with both machine assignment and operations sequencing problems simultaneously. If a system is said to be flexible then it definitely becomes more complicated and difficult to solve. To tackle with this problem, both hybrid approaches and simulation-based approaches are found to be more useful and effective. Hybrid meta-heuristics are applied to static and stochastic problems generally. Simulation-based approaches with hybrid algorithms are encountered when the considered system dynamic.

2.3. Shop Configurations and Scheduling Environments

Shop configurations vary according to the machine types and their characteristics, production flow and number of resources. The most well known configurations defined in Pinedo (2008) are given in this section.

Single Machine Shop: There are only one machine to processed all jobs in the shop. Single machine is the simplest case of all possible machine environments and is a specific form of all other more complicated shop configurations.

Parallel Machine Shop: Parallel machine is a generalization of the single machine model. Jobs can be processed on any one of the parallel machines. These machines are assumed to be identical.

Flow Shop: There are m machines in series. Each job has to be processed on each one of the m machines. All jobs have to visit machines in the same order. If there are multiple parallel machines in one stage, the flow shop converts to *flexible flow shop*. In flexible flow shop, a job can visit one of these parallel machines.

Job Shop: There are m machines and each job has to be processed on each one of the m machines. In job shop, jobs can visit the machines in different orders. Therefore, there is a distinction between flow shop and job shop. A flow shop is a job shop in which each job visits the machines in the same order.

Flexible Job Shop: Flexible job shop is a generalization of job shop. Each job can be processed on one of the alternative machines. Alternative machine set consists of identical parallel machines.

In scheduling problems different type of scheduling environments are considered in the literature. These scheduling environments are classified into two main categories: static environment and dynamic environment.

In static environments, all jobs release at the same time to the shop and all parameters are known in advance (Büyükköprü, 2005). Static scheduling problems can be handled with deterministic or stochastic parameters. In stochastic scheduling problem, some parameters can be defined as random with a probabilistic distribution. By this way, the problem reflects real life problems more closely.

In dynamic environments, jobs arrive to the system constantly and, the finished jobs are moved out of the system as their completion of all operations. Similar to the static environments, dynamic environment can be considered to be either deterministic or stochastic. In static ones, job release times are known in advance. But in stochastic one, job release times sampled from a probabilistic distribution.

Random events such as machine breakdowns, repairs and due date changes may occur in dynamic environments (Ouelhadj, et al., 2009). Some characteristics are given in Table 2.1.

Table 2.1 Characteristics of static and dynamic environment

	Static Environment	Dynamic Environment
All jobs arrive at the same time	*	
Jobs arrive continuously		*
Deterministic processing times	*	*
Random processing times	*	*
Machine breakdowns and repairs		*
Unexpected events (i.e., Due date changes)		*

Scheduling problems vary according to shop configurations and scheduling environments. One of the most well known problems is JSSP in the literature. JSSP and, an extension version of it, the FJSSP is handled in the next section.

2.4. Problem Definition

Scheduling is a decision-making process that is used on a regular basis in many manufacturing and services industries. It deals with the allocation of resources to tasks and sequencing tasks over given time periods. The goal of the scheduling problems is to optimize one or more objectives (Pinedo, 2008).

In manufacturing systems, tasks usually refer to “jobs” and resources correspond to “machines”. In some cases, jobs have elementary tasks which are called “operations” (Baker, et al., 2009). Each job which has a known processing time has to be processed on a predefined machine in a given sequence.

2.4.1 Job Shop Scheduling Problem

Job shop scheduling problem is a well known and most studied problem in literature. In general, JSSP is denoted by $n \times m$ where n represents jobs and m corresponds to machines. Job j can be processed on machine i and, job i has its own predetermined route. The processing of job j on machine i is referred to as operation (i,j) and its processing time denoted by p_{ij} . General assumptions can be given as follows:

- There are N jobs $J=\{J_1, J_2, \dots, J_n\}$ indexed by j .
- There are M machines $M=\{M_1, M_2, \dots, M_m\}$ indexed by i .

- Operations $\{O_{i1}, O_{i2}, \dots, O_{in}\}$ indexed by (i,j) can be processed on more than one machine.
- Each job must be processed on each machine in a predefined operation route.
- A machine cannot perform more than one operation at a time.
- Pre-emption is not allowed. (Operation O_{ij} can be processed on machine i without any interruption)
- All jobs arrive at the same time t to the system. ($t=0$)
- Consecutive operations of parts can be processed on the same machine.
- The objective of JSSP is to find a feasible schedule with a desired objective function (i.e., minimization of makespan).

Table 2.2 gives an example of JSSP which shows the allocated machines of each operation and operations' processing times. For example, job 1 goes through machine 1, machine 3 and then machine 2, and the processing times of each operation is 4, 5 and 3 respectively.

Table 2.2. An example of JSSP

<i>Jobs</i>	<i>Operations</i>	<i>Machines</i>	<i>Processing times</i>
1	1	1	4
	2	3	5
	3	2	3
2	1	4	5
	2	1	6
	3	3	4

2.4.2 Flexible Job Shop Scheduling

FJSSP is a generalization of JSSP which represents real world manufacturing systems more closely. The main difference between JSSP and FJSSP is that one has to select a machine from a set of alternative machines for each operation in FJSSP. The other assumptions are similar to JSSP.

FJSSP has n jobs and m machines. There are a number of operations for each job which are allowed to be processed on a set of alternative machines. Therefore, FJSSP deals with two sub-problems: one of them is selecting a machine for each operation from the alternative machine set and the other one is sequencing operations.

Table 2.3 gives an example which has 3 jobs and 5 machines, and the operations column shows all of the operations of each job J_i . Note that, each operation can be processed at one of the two alternative machines among five machines. From Table 2.3, it is apparently seen that if an operation cannot be processed at a machine (i.e., infeasible), its processing time is represented by an asterisk. The other cells in Table 2.3 represent the processing times for all feasible operation & machine pairings.

Table 2.3. An example of FJSSP

<i>Jobs</i>	<i>Operations</i>	<i>Machine 1</i>	<i>Machine 2</i>	<i>Machine 3</i>	<i>Machine 4</i>	<i>Machine 5</i>
1	1	2	*	4	*	*
	2	*	5	*	*	7
2	1	7	*	3	*	*
	2	*	10	*	*	8
	3	*	6	*	*	4
3	1	*	*	1	3	*
	2	8	*	*	11	*

2.5. Objective Functions in Scheduling

The goal of manufacturing systems is scheduling of jobs on machines while achieving its objectives. The objectives can be defined by a manager or a researcher according to the problem types, shop configurations, etc.

Most used objective functions are given as follows (Pinedo, 2008):

Makespan: The makespan denoted by C_{max} and is defined as the time when the last job leaves the system, i.e.,

$$C_{max} = \max (C_1, \dots, C_n)$$

where C_j is the completion time of job j and n is the total number of jobs. The scheduler deals with minimizing the makespan. If each job has predetermined weights, where the weight of job j is w_j , the objective is converted to total weighted completion time. *Total weighted completion time* is given as follows:

$$\sum_{i=1}^n w_j C_j$$

Flow time: The flow time for job j , F_j , is given by the time span between the completion time C_j and the ready time r_j . The weighted completion time is corresponds to the *weighted flow time* (Metta, 2008).

$$F_j = C_j - r_j$$

Mean flow time: The average of flow time of all jobs in a system is defined as the mean flow time (Metta, 2008). Number of jobs is denoted as n .

$$\sum_{j=1}^n (F_j/n)$$

Maximum Lateness: It measures the worst violation of the due dates. The objective of the schedule is to minimize the maximum lateness. The lateness of job j , L_j , can be found as,

$$L_j = C_j - d_j$$

Where C_j is the completion time of job j and, d_j is the due date of job j . The maximum lateness, L_{max} is defined as,

$$L_{max} = \max(L_1, \dots, L_n).$$

Total Tardiness: The tardiness of job j , T_j , is defined as

$$T_j = \max(C_j - d_j, 0)$$

and the objective function is

$$\sum_{j=1}^n T_j$$

Suppose that different jobs have weights, where the weight of job j is w_j . And, the *total weighted tardiness* can be given as

$$\sum_{j=1}^n w_j T_j$$

Number of tardy jobs, U_j , or weighted number of tardy jobs can be considered as objective function which is given as follows:

Number of tardy jobs:

$$\sum_{j=1}^n U_j$$

Weighted number of tardy jobs:

$$\sum_{j=1}^n w_j U_j$$

CHAPTER 3

METHODOLOGY

In this study a GA is used for FJSSP. First, GA is adapted to solve static FJSSP and tested with benchmark problems from the literature. Generated problems are solved by using proposed GA to analyze the shop performance with respect to the considered factors. Finally, optimization via simulation by GA is used to solve the dynamic and stochastic FJSSP. Here with dynamic it is meant that jobs arrive to the shop constantly, and with stochastic it is meant that the processing times are sampled from a probability distribution. Therefore, different approaches are used in the study. The general information about the most commonly used methods in the literature and the methodologies of the study are given in subsections.

3.1. Scheduling Approximations

Various scheduling methods have been applied to the JSSPs and FJSSPs in the literature. Scheduling methods can be classified as exact methods and approximation methods.

Exact methods include some Operations Research methods such as Linear and Integer programming. These approaches were never accepted applicable for real life problems due to their failure to solve large size problems. Most successful results were obtained by Lagrangian Relaxation and branch and bound in this category (Martinis, 2003). Branch and bound finds exact value first by finding a feasible solution space in which the solution exists and then narrow down the search in the feasible solution space.

Heuristics methods such as dispatching rules and neighbourhood search are found to be more useful for dynamic scheduling environments. These methods are common in

real life applications which determine the sequence of jobs to be processed when the machines become available. Simulation can be used to evaluate the performance of dispatching rules. Pinedo (2005) gives the most common dispatching rules as follows:

The *Service in Random Order (SIRO)* rule: The next job is selected randomly from jobs waiting for processing when a machine is become available.

The *Earliest Release Date first (ERD)* rule: This rule minimizes the variation in the waiting times of the jobs at a machine which is known as *First Come First Served* rule.

The *Earliest Due Date first (EDD)* rule: The job with the earliest due date is selected to be processed next when a machine become available. This rules' objective is to minimize the maximum lateness.

The *Weighted Shortest Processing Time first (WSPT)* rule: The job with the highest ratio of weight (w_j) over processing time (p_j) is scheduled next when a machine become available. This rules' objective is to minimize the weighted sum of the completion times, i.e., $\sum_{i=1}^n w_j C_j$. If the all weights are equal, the WSPT rule converts to the *Shortest Processing Time first (SPT)* rule.

Other dispatching rules are summarized in Table 3.1.

Table 3.1 Summary of other dispatching rules

Rule	Abbreviation	Objectives
The Longest Processing Time first	LPT	Load Balancing over Parallel Machines
The Shortest Setup Time first	SST	Makespan and Throughput
The Least Flexible Job first	LFJ	Makespan and Throughput
The Critical Path	CP	Makespan
The Largest Number of Successors	LNS	Makespan
The Shortest Queue at the Next Operation	SQNO	Machine Idleness

TS and SA are the heuristic methods which are known as neighbourhood search methods. These algorithms start with an initial schedule and try to obtain a better schedule with small changes among neighbours. Afterwards, the algorithm accepts or rejects the new schedule depending on an objective function. This procedure

continues to find the best schedule iteratively. *Schedule representation, neighbourhood design, search process* and *acceptance/rejection criterion* are the most important processes (Pinedo, 2008).

Simulated annealing and tabu search algorithms together with notations are given as follows:

S_k : a current schedule in k^{th} iteration

S_0 : best schedule found so far

$G(S_k)$: objective function value of S_k

$G(S_0)$: objective function value of S_0

S_c : a candidate schedule selected from the neighborhood

$P(S_k, S_c) = \exp\left(\frac{G(S_k) - G(S_c)}{\beta_k}\right)$: probability of a move from S_k to S_c

β_k : cooling parameter

Simulated Annealing

1. For $i = 1$ to N

1.1. Select an initial schedule S_I and β_I

Set $S_0 = S_I$

1.2. Select a candidate schedule S_c from the neighbourhood

1.2.1. if $G(S_0) < G(S_c) < G(S_k)$ then $S_{k+1} = S_c$ and go to step 1.3

1.2.2. if $G(S_c) < G(S_0)$ then $S_0 = S_{k+1} = S_c$ and go to step 1.3

1.2.3. if $G(S_c) < G(S_k)$ then generate a random number U_k from a

Uniform(0,1) distribution

1.2.3.1. if $U_k \leq P(S_k, S_c)$ then $S_{k+1} = S_c$ else $S_{k+1} = S_k$ and go to

step 1.3

1.3. Select $\beta_{k+1} \leq \beta_k$

1.3.1. $k = k + 1$

1.3.2. if $k = N$ then STOP else go to Step 1.2

Tabu Search

1. For $i = 1$ to N

1.1. Select an initial schedule S_I and β_I

Set $S_0 = S_I$

1.2. Select a candidate schedule S_c from the neighbourhood

1.2.1. if the move $S_k \rightarrow S_c$ is prohibited by a mutation on tabu list then set $S_{k+1} = S_k$ and go to step 1.4

1.2.2. if the move $S_k \rightarrow S_c$ is not prohibited by a mutation on tabu list then set $S_{k+1} = S_c$ and enter reverse mutation at the top of the tabu list

1.2.3. Push all other entries in the tabu list one position down and delete the entry at the bottom of the tabu list

1.3. if $G(S_c) < G(S_0)$ then $S_0 = S_c$ and go to step 1.4

1.4. $k = k + 1$

1.4.1. if $k = N$ then STOP else go to Step 1.2

In recent years meta-heuristics become more popular for solving sequencing and scheduling problems. Meta-heuristic algorithms inspired by natural life (i.e., ACO, BA, Artificial Immune System (AIS) and PSO).

One of the most common meta-heuristic, GA which is inspired by Darwin's evolutionary theory is used in this study. The details of the general GA is given in subsection 3.1.1.

3.1.1 Genetic Algorithm

GA is a population-based algorithm which consists of *individuals*. Each individual corresponds to a *chromosome* which holds *genes*. Each individual is evaluated by its *fitness*. *Selection*, *crossover* and *mutation* are the most important procedures of GA. These procedures apply iteratively and each iteration called *generation*. New generation consist of *offspring (children)* using two *parents* selected from current generation (Pinedo, 2008). General GA pseudo codes are given as follows.

General Genetic Algorithm

-
1. For $i=1$ to N
 - Generate initial schedules $S_1, \dots, S_n$
 - Evaluate fitness of each individual (f_i)
 2. For $i=1$ to N
 - 2.1. Select two parents S_1 and S_2 form current generation
 - 2.1.1. Apply crossover and set the changes as offspring (C_1 and C_2)
 - 2.1.2. Apply mutation to offspring and set the changes as offspring
 3. For $i=1$ to N
 - Evaluate fitness of each offspring (f_i)
 4. Select the best m new individual and place with worst m individual in the current generation
 - 4.1. $k=k+1$
 - if $k=\text{max iteration}$ then STOP else go to Step 1.2
-

3.1.1.1 Representation

Deciding how the problem will be represented with a string of symbols known as *genes* is the first step in GA. This string itself is the information about the solution. The representation is usually formed with binary, real-valued or integer-valued arrays. Binary encoding occurs by 1-0 strings and needs more memory space while computing (Figure 3.1). Integer encoding contains real values that represent the solution. For example, a string can be composed of job numbers or operation

numbers. The most important thing in the representation is construction of a structure that maintains the feasibility after the generation of new individuals. Most of the studies used integer encoding for combinatorial optimization problems (Innani, 2004).

1	1	0	1	0	0	1	1	0	1
---	---	---	---	---	---	---	---	---	---

Figure 3.1 A binary chromosome representation

3.1.1.2 Fitness Function

Each individual's fitness is evaluated by an objective function, i.e. makespan, which is used for comparison purpose with other individuals. According to the this comparison results an individual survives or deaths. The objective function is converted into each individual's fitness.

3.1.1.3 Selection

Selection is an important procedure to choose individuals for reproduction according to their fitness. All selected, select all but protect the best individuals, select the worst individuals, select the best individuals, roulette wheel selection, ranking and the tournament selection are some of the selection strategies. All individuals are selected for reproduction in all selected strategy. Select all, but protect the best individuals strategy selects the all individuals except the best individuals for reproduction. The best individuals are transferred to the new generation directly. Select the worst individuals strategy selects the worst individuals for reproduction and tries to improve them. Best individuals are protected. The worst individuals are selected according to a threshold value which is calculated from the fitness values of all individuals. Select the best individuals are similar to the select the worst ones. It selects the best individuals for reproduction (Chen, et al., 1999). The roulette wheel strategy selects the individuals according to a proportional to their fitness. Selection probability is determined according to the ratio of its fitness value to the total population fitness. The individual with high fitness has a high selection chance (Innani, 2004). Tournament selection selects n individuals randomly from the population and selects the best individual among these n individuals. Ranking

method orders the individuals according to its fitness and assigns a rank to each individual. Individual is selected according to its rank (Pezzella, et al., 2008).

3.1.1.4 General Crossover Procedures

After selecting the individuals for reproduction, crossover procedure begins. Crossover is applied using two individual by a crossover operator to generate new offspring (child). Crossover procedure can be applied as one-point crossover or multiple-point crossover. In one-point crossover, a point is selected randomly. From starting to this point of the chromosome is copied to the first offspring and remaining part of the chromosome is copied to the second offspring. And the vice versa is done for the second chromosome. In the two-point crossover procedure, the string between the two points is copied to the first offspring, and the remaining parts are copied to the second offspring.

Uniform crossover operator (Figure 3.2) creates the offspring by selecting a gene from the parent chromosome randomly.

There exist variety of crossover operators in the literature such as partially mapped crossover, order based crossover, cycle crossover, precedence preserving order based crossover (POX). Crossover operators must be selected according to the form of the representation.

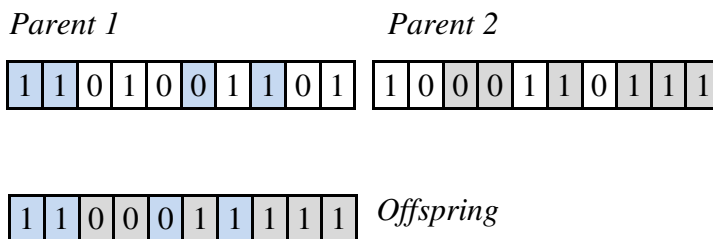


Figure 3.2 An example of uniform crossover operator

3.1.1.5 General Mutation Procedures

Mutation procedure is applied to create variability in the population and to keep diversity of the population. Mutation is usually applied to a single chromosome by exchanging a string position or a value of a string with a small probability. There are several mutation types such as insertion, displacement, reciprocal exchange and

scramble mutation. For example, the inversion mutation (Figure 3.3) selects two positions randomly and exchanges these two positions (Innani, 2004).

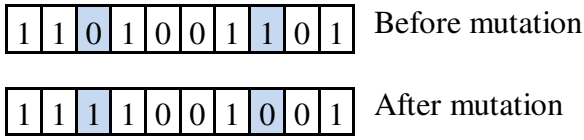


Figure 3.3 An example for inversion mutation

3.2. Simulation Optimization

Simulation is a powerful tool when integrated with other approaches in a wide range of application areas. In this study, an optimization model integrated with simulation is proposed for dynamic and stochastic FJSSP. Details of the proposed approach are given in Section 5.

Simulation is used to model of the operation of a real world process or system over time. Simulation helps to analyze the effects of changes to the existing system or to predict the performance of a new system (Banks, et al., 2010). It is easy to model the system more close to the real system without simplifying assumptions (i.e., deterministic processing times, deterministic interarrival times). Simulation enables to compare the different alternative strategies and analyzes the effect of these strategies to the system performance.

A system considered to study is represented by a simulation *model*. *Entities*, *attributes* and *activities* are the components of a model.

Simulation models can be classified mainly as static or dynamic, deterministic or stochastic, and discrete or continuous. *Static* simulation model represents a system at a certain point in time. *Dynamic* simulation model represents a system that changes over time. If simulation model doesn't contain random variables, it is called *deterministic* simulation model. The *stochastic* simulation model possessed one or more random variables as input. State variables change only at a discrete set of points in time in *discrete* simulation model. And, in the *continuous* simulation model, state variables change continuously over time (Banks, et al., 2010).

Many real life problems do incorporate uncertainty and result in uncertain solutions. This type of systems are said to be probabilistic and models of such systems are

known to be probabilistic models as well. Simulation is an indispensable tool for analyzing such systems. One of the limitations of simulation models in general is that they basically act as “black boxes” — they can only evaluate the model for the decision variables that are pre-specified. Thus, to use a simulation model for evaluating the performance of a process, one must first set the values of decision variables and then run a simulation to forecast the performance of that particular configuration. Adjusting true values of decision variables manually to get optimality gives rise to boredom and dissipation of time even for small problems. Moreover, it is often not clear how to adjust the decision variables from one simulation run to the next. In such cases, finding an optimal solution for a simulation model generally requires that you search in a heuristic or ad hoc fashion. This usually involves running a simulation for an initial set of decision variables, analyzing the results, changing one or more variables, re-running the simulation, and repeating this process until a satisfactory solution is obtained. As implicitly mentioned above, simulation itself can not automatically adjust the decision variables so as to reach an optimum solution. This was one of the main problems of simulation which left large scale models unresolved in the past.

Simulation modelling can require too many trial and error processes in many complex and uncertain systems. Therefore, using simulation can be very time consuming and it can be hard to achieve robust and efficient results. Using an optimization method with simulation experiments can be very challenging to cope with complex and uncertain systems (Paris, et al., 2001).

In parallel with the developments in simulation world, the techniques of optimization have evolved in a dizzying speed since 1990s. Particularly, after millennium, developments in the area of optimization have allowed for the creation of intelligent search methods capable of finding optimal or near optimal solutions to complex problems involving elements of uncertainty. Often, optimal solutions can be found among large sets of possible solutions even when exploring only a small fraction of them. But, it must be stated that increasing the number of decision variables increases the solution complexity of optimization via simulation methodology. Even with limited number of decision variables optimization via simulation is much more difficult than deterministic optimization setting due to inherent stochastic nature of simulation. For details readers can refer to (Banks, et al., 2010).

Although “simulation” and “optimization” each have distinguishing characteristics and are regarded as different disciplines in some cases it is a requirement that “simulation” and “optimization” should operate simultaneously. Until the end of the last millennium, optimization and simulation were kept pretty much separate in practice, even though there was a large body of research literature relevant to combining them (Fu, 2002). But, recent developments in both disciplines already herald a marriage between these two distinct disciplines. Moreover, in time, it already became a necessity to work hand in hand for these versatile disciplines (e.g., integration of optimization techniques into simulation practice). In the last decade, such cooperation appeared between well known optimization routines and simulation software packages. Nowadays, successful cooperations are made between commercial optimization routines (e.g., OptQuest, optimization technologies, Inc) and simulation software packages (e.g., Simio/Simio LLC, Arena/Rockwell software Inc.). The reader can refer to (Law, et al., 2000) for a list of other cooperation. OptQuest is known to be a standalone optimization routine that can be bundled with a number of commercial simulation languages. Briefly, OptQuest enhances the optimization capabilities of commercial simulation languages by searching optimal solutions to simulation models.

In simulation optimization terminology different keywords for the terms related to inputs and outputs are used. They all express the same meaning either intentionally or inadvertently used. For convenience some favourite sample naming for inputs and outputs from literature is given and then our naming convention in this study is given. The terms related to the inputs and outputs of a simulation optimization problem is well defined in Fu (2002) as follows:

“In the literature, there is a wide variety of terms used in referring to the inputs and outputs of a simulation optimization problem. Inputs are called (controllable) parameter settings, values, variables, (proposed) solutions, designs, configurations, or factors (in design of experiments terminology). Outputs are called performance measures, criteria, or responses (in design of experiments terminology). Some of the outputs are used to form an objective function, and there is a constraint set on the inputs.”

In his study Fu (2002) follows deterministic optimization common usage, and adopted “variables” and “objective function”, with the latter comprised of

performance measures estimated from simulation (consistent with discrete-event simulation common usage). In addition, he called “*configuration*” or a “*design*” as a particular setting of the variables. In this study, we adopt “*decision variables*” in referring to inputs and “*objective function*” in referring to outputs. In this respect, decision variables are to be process plan alternatives and alternative machines for each operation and objective function is to be total of average flow times.

3.3. Experimental Design

Experimental design provides a better understanding on experiment results. *Factors* are the input parameters and structural assumptions composed a model, and the *responses* are the output performance measures. The main issue is to determine which parameters and structural assumptions will be fixed aspects and which experimental factors are (Law, 2007).

Experimental design methods are useful for evaluation and comparison the basic design configurations and selecting the design parameters. *The factorial experimental design* is useful when there are two or more factors. In general, all combinations of factor levels are taken into account. *The analysis of variance* (ANOVA) is the primary tool for statistical data analysis. Detailed information can be found in (Montgomery, et al., 2002).

In the study, a four full factorial analysis is used to measure the effect of flexibility level on shop performance. The factorial design enables testing hypotheses concerning the effects of various levels of a factor and detection of interactions between factors by using ANOVA. The details of experimental design and results of analysis are given in section 4.4.

CHAPTER 4

SOLVING THE FLEXIBLE JOB SHOP SCHEDULING PROBLEM BY USING GENETIC ALGORITHM

JSSP deals with sequencing operations of a set of jobs on a set of machine which is specific for an operation. Thus, JSSP is one of the most difficult optimization problems that are known to be NP-Hard (Garey, et al., 1976). FJSSP is an extension of the classical JSSP which makes the problem much more complex even NP-Hard.

Each operation can be processed at more than one machine and operations can be processed in any order in FJSSP. FJSSP can be divided into two sub-problems: (1) the routing sub-problem that assigns each operation to a machine selected out of available machines set which is determined for each operation. (2) Scheduling sub-problem deals with sequencing the assigned operations on all machines to achieve a feasible schedule (Zhang, et al., 2011).

4.1. Problem Notification

The problem is composed of N jobs J_i ($i=1,2,...,N$) and M machines ($k=1,.....,M$). Each job J_i composed of a number of operations O_{ij} . FJSSP is formulated as follows:

N	Number of jobs
M	Number of machines
T	Number of total operations
P_{ijk}	Processing time of operation O_{ij} on machine k
i	Job index ($i=1,.....,N$).
k	Machine index ($k=1,.....,M$)
j	Operation index
O_{ij}	The j th operation of job i .
$\hat{J}$	The set of jobs
$\hat{A}$	The set of machines

$\hat{A}_{ij}$	The set of alternative machines on which operation O_{ij} can be processed ($\hat{A}_{ij} \subseteq \hat{A}$)
S_k	The start time of the idle time interval on machine k
E_k	The end time of the idle time interval on machine k
Tb_i	The end time of i th job's last operation processed
Tm_k	The end time of the last operation on machine k

Assumptions of FJSSP are given as follows:

- All jobs are released at time 0,
- All machines are available at time 0,
- Each part has more than one operation,
- Each operation can be processed on more than one machine.
- Operations can be processed according to the precedence constraints. $O_{i(j+1)}$ cannot be processed before O_{ij} .
- Operations' processing times can be defined as deterministic or stochastic.
- Pre-emption is not allowed. A machine cannot perform more than one operation at a time.
- Consecutive operations of parts can be processed on the same machine.
- The setup times and transportation times are not considered.

The problem consist of two sub-problems: one of them is the selection of a machine from a set of alternative machines $\hat{A}_{ij}$ for each job J_i and, the other one is sequencing operations O_{ij} on the machines to obtain a feasible schedule.

The objective is to find a schedule with minimum makespan, maximum completion time of jobs, $C_{max} = \{C_i \mid i=1, \dots, n\}$.

To explain the problem briefly, a small example is given in the Table 4.1. There are 3 jobs and 5 machines, and the operation column shows all of the operations of each job J_i . Note that, each operation can be processed at one of the two alternative machines among five machines. From Table 4.1, it is apparently seen that if an operation cannot be processed at a machine (i.e., infeasible), its processing time is represented by an asterisk. The other cells in Table 4.1 represent the processing time for all feasible operation & machine pairings.

Table 4.1 Operation processing time table with deterministic times of FJSSP

Job	Operation	Machine 1	Machine 2	Machine 3	Machine 4	Machine 5
J_1	O_{11}	4	*	6	*	*
	O_{12}	*	7	*	*	5
J_2	O_{21}	3	*	5	*	*
	O_{22}	*	10	*	*	14
	O_{23}	*	8	*	*	4
J_3	O_{31}	*	*	8	11	*
	O_{32}	7	*	*	4	*

Table 4.2 shows an example of FJSSP with stochastic times. Stochastic times are sampled from a probabilistic distribution. Triangular distribution is used in our analysis in order to represent real life problems more closely which is ignored to simplify the problem in most of the past studies.

Table 4.2 Operation processing times with stochastic times

Job	Operation	Machine 1	Machine 2	Machine 3	Machine 4	Machine 5
J_1	O_{11}	$Tria(2,4,6)$	*	$Tria(2,4,6)$	*	*
	O_{12}	*	$Tria(5,7,9)$	*	*	$Tria(5,7,9)$
J_2	O_{21}	$Tria(3,5,7)$	*	$Tria(3,5,7)$	*	*
	O_{22}	*	$Tria(10,12,14)$	*	*	$Tria(10,12,14)$
	O_{23}	*	$Tria(4,6,8)$	*	*	$Tria(4,6,8)$
J_3	O_{31}	*	*	$Tria(8,10,12)$	$Tria(8,10,12)$	*
	O_{32}	$Tria(6,8,10)$	*	*	$Tria(6,8,10)$	*

4.2. Proposed GA for FJSSP

GA is a suitable and applicable method for scheduling problems. GAs start with an initial set of (random) solutions. Primarily, a suitable chromosome structure must be constructed to represent a solution, that is, a schedule for the given FJSSP. The chromosomes evolve through successive iterations called generations (Moon, et al., 2008). A GA is adapted to select the machines among alternative machine sets for each operations and sequencing operations in a suitable order for determined FJSSP.

4.2.1. Chromosome Representation

Representation of a problem solution is the first but the most significant step in GA. An individual can consist of more than one chromosome and a chromosome composed of genes. In the study of (Chen, et al., 1999), individuals consist of two chromosomes. The first part of the individual illustrates the routing policy of the problem and the second one illustrates the operation sequence on each machine.

(Gao, et al., 2008) divided chromosome into two strings which represents machine assignment and operation sequencing, respectively. In the study of (Zhang, et al., 2011), the chromosome structure composed of two components where one of them stands for machine selection and the other one stand for operation sequencing, separately.

In a great deal of studies it is emphasized that integer encoding is more effective than binary (0-1) encoding, because binary encoding needs more memory space and increases the computational time. Thus, an integer based coding system is adopted to represent the problem solution. The chromosome structure is considered to be composed of two parts: (1) machine assignment and (2) operation sequencing parts. The first part of the individual consists of an array of integer values which represents selected machine numbers for each operation. These machines are selected from a certain machine set which are specific for each operation. The second one depicts the sequence of operations which consist of job index i . Thus, total number of i 's placed on the operations sequence part of the individual should be equal to the total number of operations of job i (see Figure 4.1). Chromosome representation is depicted over a small example in Figure 4.1.

Chromosome:

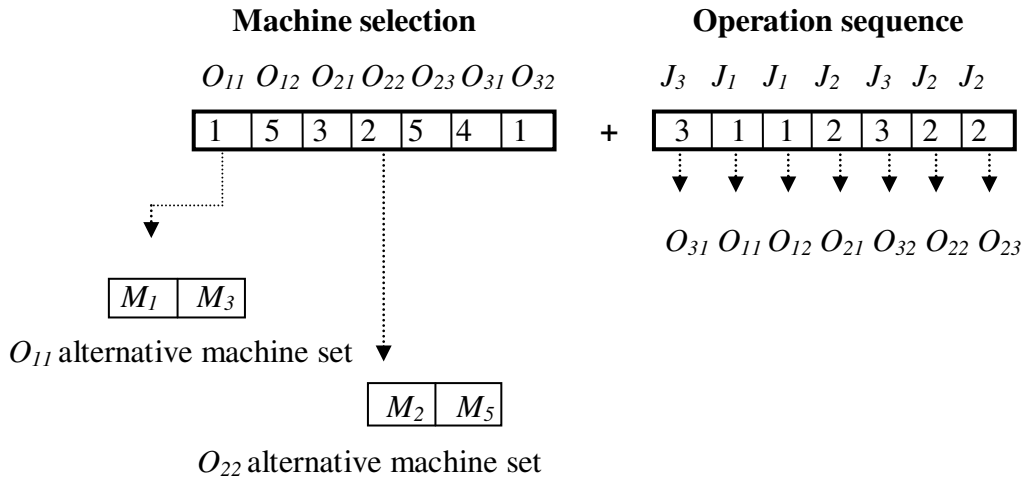


Figure 4.1 Chromosome structure of proposed GA

4.2.2. Decoding process

The decoding process evaluates the objective function of each individual using the information obtained by each chromosome and generates a schedule. The makespan (maximum completion time of jobs) is determined to be the objective function and active schedule is taken into account while calculating the makespan. The steps for decoding a chromosome to a feasible schedule are given step by step as follows:

Step 1:

Generate a machine matrix and processing time matrix. Record machine numbers of each operations of each job from machine selection part of the chromosome to the machine matrix. And record processing times of each operation to the processing time matrix. In each matrix, rows correspond to jobs, and columns correspond to operations of each job (Figure 4.2).

$$\text{Machine matrix: } \begin{bmatrix} 1 & 5 & \\ 3 & 2 & 5 \\ 4 & 1 & \end{bmatrix} \quad \text{Processing time matrix: } \begin{bmatrix} 4 & 7 & \\ 5 & 12 & 4 \\ 10 & 9 & \end{bmatrix}$$

Figure 4.2 Machine and processing time matrix used in decoding process

Step 2:

Starts with reading the operation sequence part of the chromosome and in this way determine O_{ij} . First, find the corresponding machine k for operation O_{ij} from machine matrix. Then, find corresponding processing time P_{ijk} from the processing time matrix for O_{ij} at machine k . Next, find the i th job's last operation's $O_{i(j-1)}$ ending time Tb_i and determine the last operation's ending time Tm_k on machine k .

If $Tb_i > Tm_k$ then record an idle time interval. Assign $S_k = Tm_k$ and $E_k = Tb_i$, update $Tb_i = Tb_i + P_{ijk}$ and $Tm_k = Tb_i$.

Else if $Tb_i \leq Tm_k$ then look for an appropriate idle time for O_{ij} . If there is an appropriate idle time and if $S_k \geq Tb_i$, assign $S_k = S_k + P_{ijk}$ and update Tb_i and E_k (Figure 4.3). If there is not an appropriate idle time then assign O_{ij} to the end of the last operation at machine k . Update Tb_i as $Tb_i = Tm_k + P_{ijk}$ and also update Tm_k (Figure 4.4). Note that this procedure applies for each gene placed on the operation sequence part of the chromosome.

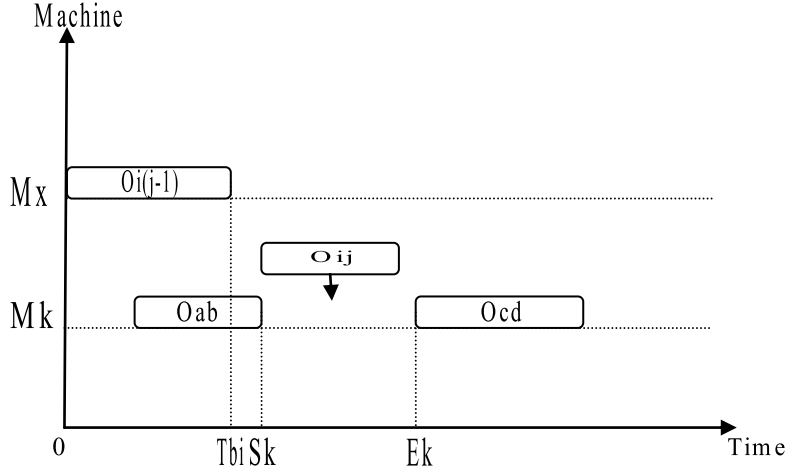


Figure 4.3 An example Gantt chart to set the operation into the idle time.

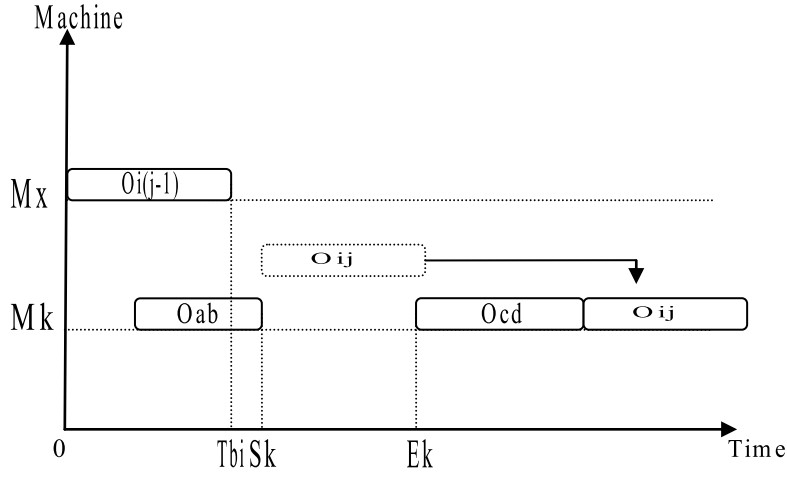


Figure 4.4 An example Gantt chart to set the operation to the end of the machine.

4.2.3. Initial Population

In literature, it is recognized that mixed strategies increases the quality of the solution. (Zhang, et al., 2011) tested two different mixed strategies; 1)Global Selection, 2)Local Selection and also considered the random selection to account for randomness. They reported that these strategies improve the quality of the initial solutions.

Due to the structure of the problem, proposed algorithm evaluates two sub-problems separately. First, it selects machines for each operation and then sequences the operations at each selected machine. In order to obtain more qualified solutions different strategies are combined while selecting machines. In this study, in addition to (Zhang, et al., 2011)'s strategies a new strategy which selects the machine with the

minimum workload is included. The motivation for including this strategy is to improve the quality of the initial solutions. The strategies and their details are given as follows:

Strategy 1:

Selecting a machine with minimum total processing time (Global Selection, (Zhang et. al., 2011))

```
1. For k=1 to M
    // Set total processing times of machines equal to 0
end
2. For j=1 to T
    // Check alternative machines' total processing times
    2.1. If alternative machines' total processing times are all equal
        // Select a machine randomly among them
    else
        // Select a machine with minimum total processing times
    2.2. // Add the operation's processing time to the selected machine's total
processing times
    end
end
end
```

Strategy 2:

Selecting a machine with minimum processing time (Local Selection, (Zhang et. al., 2011))

```
1. For j=1 to T
    // Check alternative machines' processing times
    1.1. If alternative machines' processing times are all equal
        // Select a machine randomly among them
    else
        // Select a machine with minimum processing times
    end
end
end
```

Strategy 3:

Selecting a machine with minimum workload (number of jobs)

```
1. For k=1 to M
    // Set workload of each machine equal to 0
End
2. For j=1 to T
    // Check alternative machines' number of workload
    2.1. If alternative machines' number of workload are all equal
        // Select a machine randomly among them
    else
        // Select a machine with minimum number of workload
    2.2. Assign Workload of selected machine = Workload of selected
machine + 1
    end
end
```

Strategy 4:

Selecting a machine randomly

```
1. For j=1 to T
    // Select a machine randomly for an operation
end
```

4.2.4. Generation of new candidate solutions

After generating initial solutions GA jumps to other steps. First, makespans of the initial solutions are evaluated. Hereafter, individuals are chosen for reproduction among the current population. And, chosen individuals are translated to the mating pool. In the literature, tournament approach is found to be more successful than other selection strategies (Pezzella, et al., (2008), Zhang, et al., (2011)). For this reason a tournament approach is utilized in this study as well. Two individuals are chosen randomly from current generation, and the best one among them is translated to the mating pool.

4.2.5. Crossover

Several crossover operators have been proposed for permutation representation, such as single-point crossover, cycle crossover, order crossover, and so on. (Gao, et al., 2007) proposed a two vector permutation representation which considers the operations' precedence constraint. (Gao, et al., 2008) applied order crossover to operation sequence chromosomes and used extended order crossover and uniform crossover to machine assignment chromosomes. (Zhang, et al., 2011) used two-point

crossover and uniform crossover for machine selection part and, precedence POX crossover for operation sequence part.

Crossover operator is applied to machine selection part and sequencing part of the chromosome separately. POX crossover is used for sequencing part of the crossover (Figure 4.5). The steps of POX crossover is given as follows:

POX

1. Select two individual from mating pool randomly and set them as *parent1* and *parent2*
 2. Determine two chromosomes as *child1* and *child2*
 3. Generate a random number $p[0,1]$ for each chromosome and determine a crossover probability p_r
 - 3.1. If $p < p_r$ then
//Copy the gene to the *child1* from *parent1* in the same position
Else
//Copy the gene to the *child2* from *parent1* in the same position
 - 3.2. Copy the genes that are not found in *child1* from *parent2* and do the same for *child2*
-

The uniform crossover is used for machine selection part (Figure 4.5). The steps of uniform crossover are given as follows:

Uniform Crossover

1. Select two individual from mating pool randomly and set them as *parent1* and *parent2*
 2. Determine two chromosomes as *child1* and *child2*
 3. Generate a random number $p[0,1]$ for each chromosome and determine a crossover probability p_r
 - 3.1. If $p < p_r$ then
//Copy the gene to the *child1* from *parent1* in the same position
Else
//Copy the gene to the *child2* from *parent1* in the same position
 - 3.2. Fill the empty genes in *child1* and *child2* from *parent2*
-

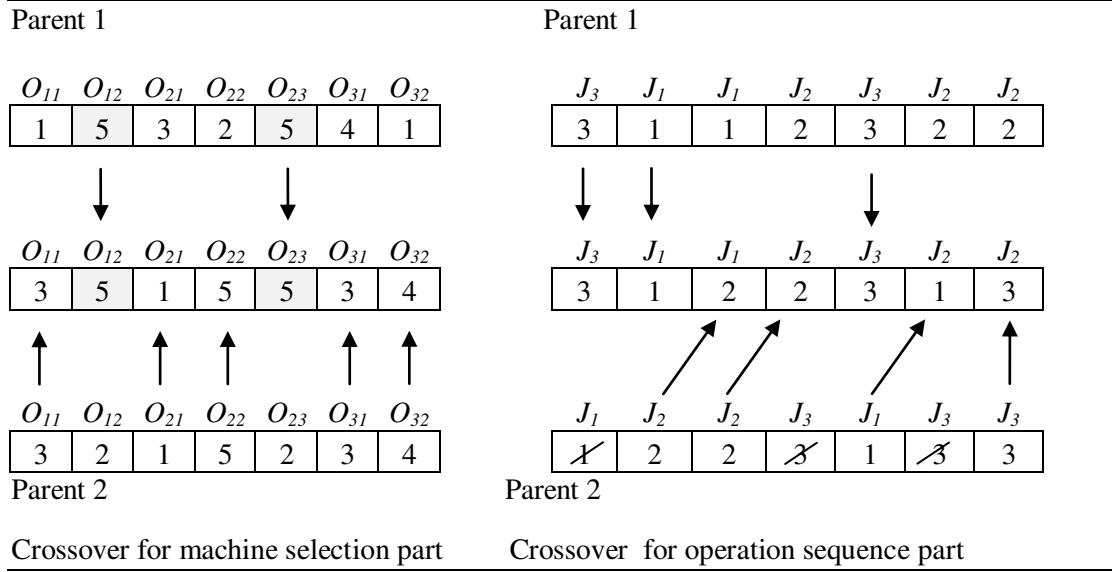


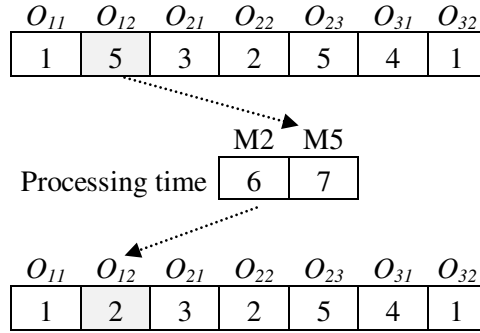
Figure 4.5 Illustration of crossover operations

4.2.6. Mutation

Mutation operator exchanges one gene at a time according to a mutation probability p_m to increase the variety of population. The decision whether to exchange each gene of the chromosome is dependent upon mutation probability.

Two mutation operators are used in the study. One of the mutation operators changes a gene with another machine if there is a machine with shorter processing time than selected machine's processing time in the alternative machine set. Otherwise, the gene should not be mutated. If there is more than one machine with shorter processing time, one of them is selected randomly (Figure 4.6).

Before mutation



After mutation

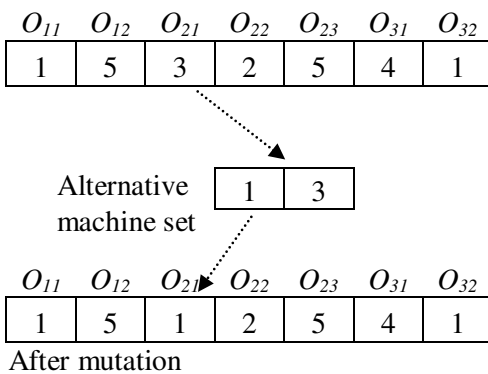
Mutation for machine selection part

Figure 4.6 Illustration of mutation operation which selects a machine with a shorter processing time

In other mutation operator, the machine is exchanged with another machine from alternative machine set if a gene of the chromosome is determined to be exchanged according to the mutation probability. While doing this, alternative machine set should be taken into account carefully (Figure 4.7).

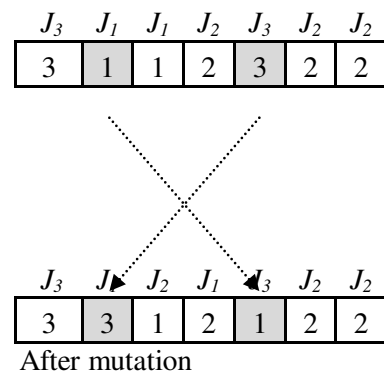
In operation sequence part, two genes are selected randomly and their positions are exchanged (Figure 4.7). The crossover and mutation procedures save the feasibility of the chromosomes. Therefore, new chromosomes do not need to control the feasibility. They are also ready for the decoding procedure.

Before mutation



Mutation for machine selection part

Before mutation



Mutation for operation sequence part

Figure 4.7 Illustration of mutation operation for machine selection and operation sequence parts

4.3. Applying the proposed GA to Benchmark Problems and Computational Results

The GA is tested with the Brandimerte's problem data set from literature (<http://www.idsia.ch/~monaldo/fjsp.html>) to understand how it works. The proposed GA is coded in Visual Basic® for Applications by Microsoft®. For convenience, the problem data sets are given in APPENDIX A.

The initial population is generated by mixing of strategy1, strategy2, strategy3 and strategy4. As shown in Zhang, et al. (2011)'s study, mixed strategy gives the near optimal solution more quickly than the single ones. According to the preliminary runs, an initial population with 40% strategy1, 20% strategy2, 30% strategy3 and 10% strategy4 gives the best solutions. Operations sequencing is selected randomly by considering the precedence constraint. After generating the initial solutions, predetermined number of individual is selected for reproduction. This parameter is determined as *select size* and, given in Table 4.3 for each problem instance. After reproduction, predetermined number of individual among new population, which is called *number of exchanges*, is translated to the current population. By the way, the population size for each problem instance is given in the same table. Other parameters of the GA is given in the below. The algorithm is run for five times for each instance and the best results among them are taken into account.

Parameters of the GA

Rate of initial strategy 1: 40%

Rate of initial strategy 2: 20%

Rate of initial strategy 3: 30%

Rate of initial strategy 4: 10%

Mutation probability: 0.01

Number of iteration: 100-300

The Brandimerte's problem data set (BRdata) consist of number of jobs between 10 and 20 and the number of operations of each job changes between 5 and 15. Number of machines ranges between 4 and 15.

Table 4.3 gives the computational results and the results of comparisons with other studies from literature. $n \times m$ column gives the number of jobs & number of machines

for each problem instance. *Flex* gives the average number of machines per operation. (LB, UB) column gives the best known solution, if it is known, otherwise, the lower and upper bound found so far. The maximum completion time, $C_{\max}$ (makespan), is considered as the measure of comparison. If the best known solution is found, an asterisk (i.e., *) is indicated near the solution. The results are compared with three studies from the literature. M&G column is the results of the study of Mastrolilli, et al. (2000), GENACE is the results of Ho, et al. (2004)'s study and the eGA is the results of Zhang, et al. (2011).

Table 4.3 Computational results and comparisons

Results of Brdata					Computational Results and parameters						
Problem	nxm	T0	Flex	LB, UB	M&G C_m	GENACE C_m	eGA C_m	Best Results	Pop Size	Select Size	Num of Exchange
Mk01	10x6	55	2,09	36, 42	40*	40*	40*	40*	100	50	20
Mk02	10x6	58	4,01	24, 32	26*	32	26*	27	200	50	20
Mk03	15x8	150	3,01	204, 211	204*	N/A	204*	204*	50	20	10
Mk04	15x8	90	1,91	48, 81	60*	67	60*	61	300	200	100
Mk05	15x4	106	1,71	168, 186	173*	176	173*	176	300	100	100
Mk06	10x15	150	3,27	33, 86	58*	67	58*	70	300	100	50
Mk07	20x5	100	2,83	133, 157	144*	147	144*	144*	200	100	20
Mk08	20x10	225	1,43	523	523*	523*	523*	523*	50	10	10
Mk09	20x10	240	2,53	299, 369	307*	320	307*	315	300	100	100
Mk10	20x15	240	2,98	165, 296	198*	229	198*	247	300	100	100

From Table 4.3, the GA obtained the best known solutions in problems Mk01, Mk03, Mk07 and Mk08. However, in Mk02, Mk04, Mk06 and Mk09 approximate solutions to the best known solutions are obtained. The algorithm cannot achieve the best known solution for the Mk10 problem since it is being the most challenging problem. As a result, from Table 4.3, it is apparently seen that not best but promising solutions obtained with proposed GA. It should be mentioned that the solutions are obtained with small GA parameters. Therefore, it can achieve the near optimal solutions in a limited time which is important for integrated systems.

4.4. Experimental Design and Analysis

An experimental design is utilized to analyze the main and the interaction effects of the factors considered (i.e., Part number, machine number, operation number, and flexibility levels) by using GA which is specifically designed for FJSSP. Stochastic processing times sampled from triangular distribution are considered in the problem.

A four full factorial design is utilized to evaluate the performance of the shop and investigate relationships between the considered factors. The design includes three level of each number of parts, number of jobs, number of operations, and level of flexibility. Thus, 81 experiments are necessary (i.e., $3*3*3*3=81$) to investigate all factor level combinations. Table 4.4 gives part $\times$ machine groups used in the experiments. Three levels of number of operations determined and are defined as low, medium, and high. Number of operations for each part is generated from uniform distribution. Parameters of uniform distribution are given in Table 4.5. Flexibility levels are adjusted according to the number of machines (Table 4.6).

Table 4.4 Part $\times$ Machine Groups

# parts	# machines
10	5
10	10
10	15
20	5
20	10
20	15
30	5
30	10
30	15

Table 4.5 Number of operation levels

Operation levels	Ranges
low	[3,5]
medium	[6,8]
high	[8,10]

Table 4.6 Flexibility levels for each machine sizes

# machines	Flexibility levels		
	1	2	3
5	2	3	5
10	4	6	10
15	6	9	15

Data is collected on one performance measure to evaluate the performance of the stochastic flexible job shop. The selected performance measure is makespan. The

experiments are performed using 20 replications of each treatment, thus minimizing the variability in the results. In addition, common random numbers are used between each experiment as another variance reduction technique.

The main and interaction effects of all factors will be discussed for the performance measure considered (i.e., makespan) in subsequent section. $\alpha=0.05$ was used in evaluating statistical significance.

4.4.1. Framework of the proposed GA for the Experimental Design

GA starts with an initial solution. As mentioned before, three different initial strategies and random selection is used. Individuals are generated by using these four strategies with a fixed percentage of population size. Initial population is generated by 30% strategy 1 and 20% strategy 2, 30% strategy 3, and 20% strategy 4. Note that, strategy 1 and 3 have much percentage in the population because they can give different solutions in each run and help reaching better solutions more quickly (Figure 4.8). Strategy 2 finds solutions by selecting a machine with minimum processing time from alternative machine set of jobs. And the random selection (strategy 4) is added because it keeps up randomness and more general solutions. In Table 4.7, the best makespan values together with iteration number for each strategy are summarized. And, the GA parameters are adjusted according to the problem size.

Table 4.7 Best makespans over five runs.

Strategy	Best makespan	Iteration number
1	47	30
2	54	8
3	44	80
4	44	84
Combined	44	16

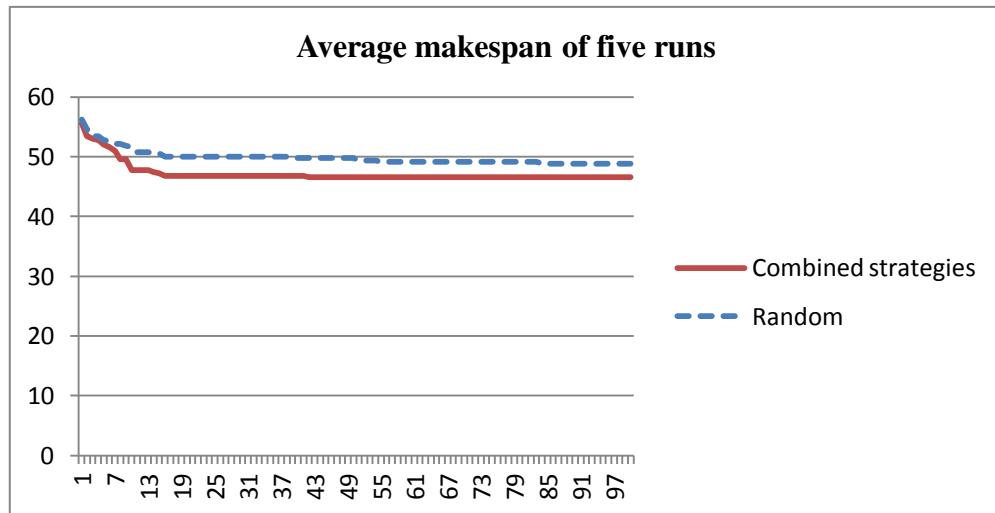


Figure 4.8 Comparison between random strategy and combined strategy.

Other GA parameters are given as follows:

Population size: 100-250

Number of iteration: 200-300

Mutation probability: 0.01

Selection type: Tournament approach

4.5. Results of Analysis

The analysis of variance results reported in Table 4.8 suggest that makespan performance of the shop is significantly affected by factors and some factor interactions.

According to Table 4.8, it is seen that all main factors have significant effect on makespan performance at 0.05 significance level. It is interesting to observe some of the factor interactions do not have significant effect on makespan (i.e., Part Number * Machine Number * Flexibility Level, Part Number * Operation Number * Flexibility Level, Machine Number * Operation Number * Flexibility Level, Part Number * Flexibility Level, Operation Number * Flexibility Level, Part Number * Machine Number * Operation Number * Flexibility Level). Below a detailed analysis of these insignificant interaction effects will be given with their interaction plots, respectively.

Table 4.8 Analysis of Variance for Makespan

<i>Source</i>	<i>Type III Sum of Squares</i>	<i>df</i>	<i>Mean Square</i>	<i>F</i>	<i>Sig.</i>
Corrected Model	13179375,067	80	164742,188	1311,185	,000
Intercept	35739734,193	1	35739734,193	284452,869	,000
Part Number (A)	3910127,201	2	1955063,600	15560,369	,000
Machine Number (B)	3633461,242	2	1816730,621	14459,376	,000
Operation Number (C)	4057699,119	2	2028849,559	16147,632	,000
Flexibility level (D)	14392,624	2	7196,312	57,276	,000
AB	824532,732	4	206133,183	1640,616	,000
AC	384669,226	4	96167,307	765,396	,000
AD	416,988	4	104,247	,830	,506
BC	293598,844	4	73399,711	584,189	,000
BD	3638,585	4	909,646	7,240	,000
CD	943,042	4	235,760	1,876	,112
ABC	54243,198	8	6780,400	53,965	,000
ABD	292,050	8	36,506	,291	,969
ACD	323,958	8	40,495	,322	,958
BCD	409,322	8	51,165	,407	,917
ABCD	626,936	16	39,184	,312	,996
Error	193365,780	1539	125,644		
Total	49112475,040	1620			
Corrected Total	13372740,847	1619			

a. R Squared = ,986 (Adjusted R Squared = ,985)

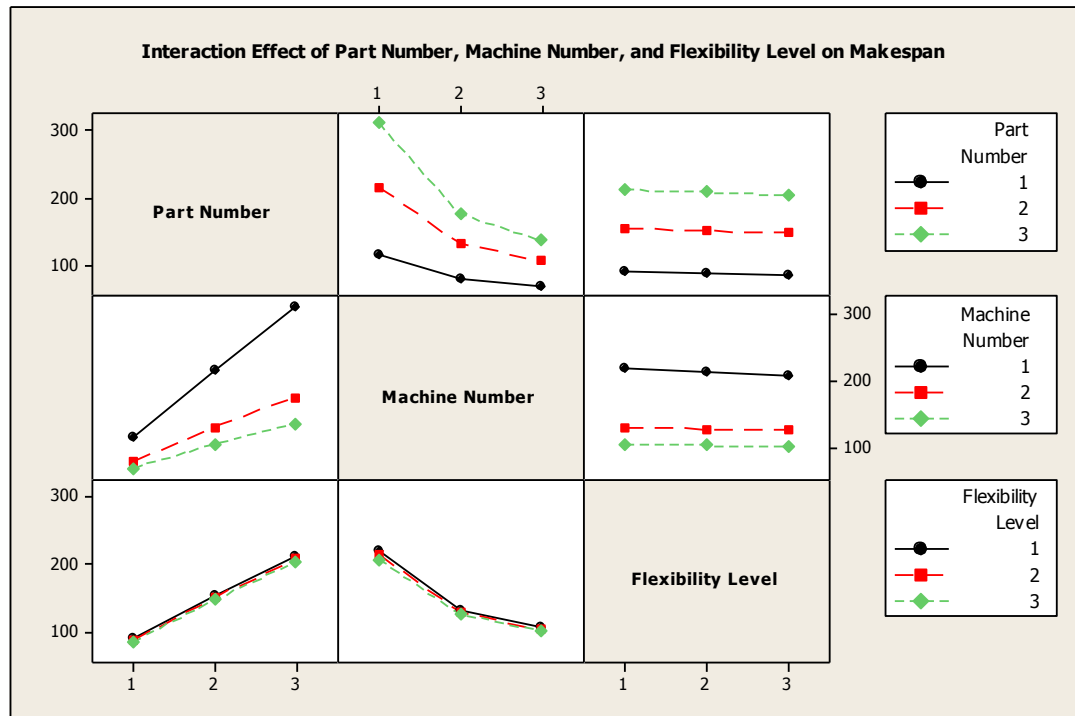


Figure 4.9 Interaction Effect of Part Number, Machine Number, and Flexibility Level on Makespan

Interaction Effect of Part Number, Machine Number, and Flexibility Level on Makespan is given in Figure 4.9. From Figure 4.9, it is noticeable that the interaction between flexibility level and the other two factors (i.e., Machine number and part number) has no significant effect on makespan performance of the shop. From flexibility level&machine number and flexibility level&part number interaction graphs it is clearly seen that as the level of the flexibility increases only marginal improvements gained on makespan performance at different levels of other factors which makes part number, machine number, and flexibility level interaction effect on makespan insignificant. A wide range of insights can be gained by looking into machine number and part number interaction graphs. For example, by looking into machine number interaction graphs it is clearly seen that makespan performance of the shop improves as the level of machine number increases. Also, by looking into machine number and part number interaction graph it is seen that at low level of machine number the makespan performances of the shop significantly differ for each level of part number whereas for higher levels of machine number it is observed that the difference between makespan performances for each level of part number get much more closed. Another significant insight can be gained by analyzing part number and machine number interaction graph. In this graph it is observed that

setting the machine number at its low level makes the makespan performance of the shop much more sensitive to different levels of part number. Further insights can be gained by analyzing Figure 4.9 thoroughly. As a result, from the Figure 4.9, it is obvious that the best makespan performance can be gained for high levels of machine number together with low level of part number regardless of the flexibility level.

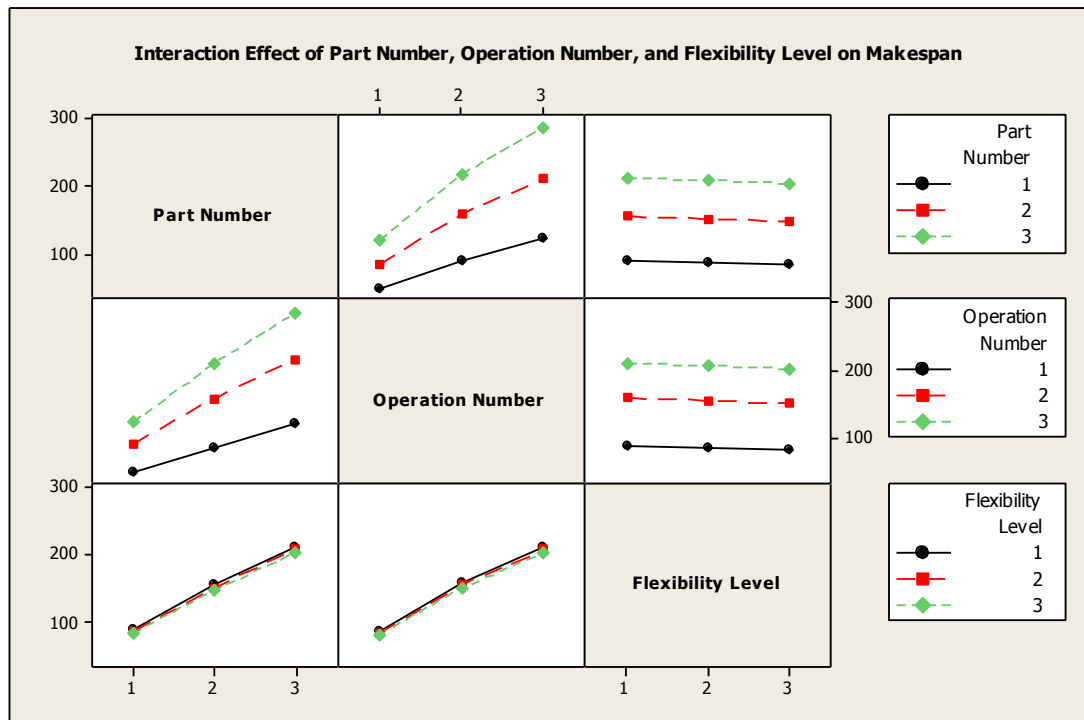


Figure 4.10 Interaction Effect of Part Number, Operation Number, and Flexibility Level on Makespan

Interaction effect of part number, operation number, and flexibility level on makespan is given in Figure 4.10. From Figure 4.10, it is noticeable that the interaction between flexibility level and the other two factors (i.e., Part number and operation number) has no significant effect on makespan performance of the shop. From flexibility level&operation number and flexibility level&part number interaction graphs it is clearly seen that as the level of the flexibility increases only marginal improvements gained on makespan performance at different levels of other factors which makes part number, machine number, and flexibility level interaction effect on makespan insignificant. A wide range of insights can be gained by looking into machine number and part number interaction graphs. For example, by looking into operation number interaction graphs it is clearly seen that makespan

performance of the shop deteriorates as the level of operation number increases. In addition to this, as the level of the part number increases, the makespan performance of the shop gets much more deteriorated for each level of operation number. Note that, as the level of part number increases, the gap between the makespan performances of the shop gets larger as the level of the operation numbers increases. Further insights can be gained by analyzing Figure 4.10 thoroughly. As a result, from the Figure 4.10, it is obvious that the best makespan performance can be gained for low levels of operation number and part number regardless of the flexibility level.

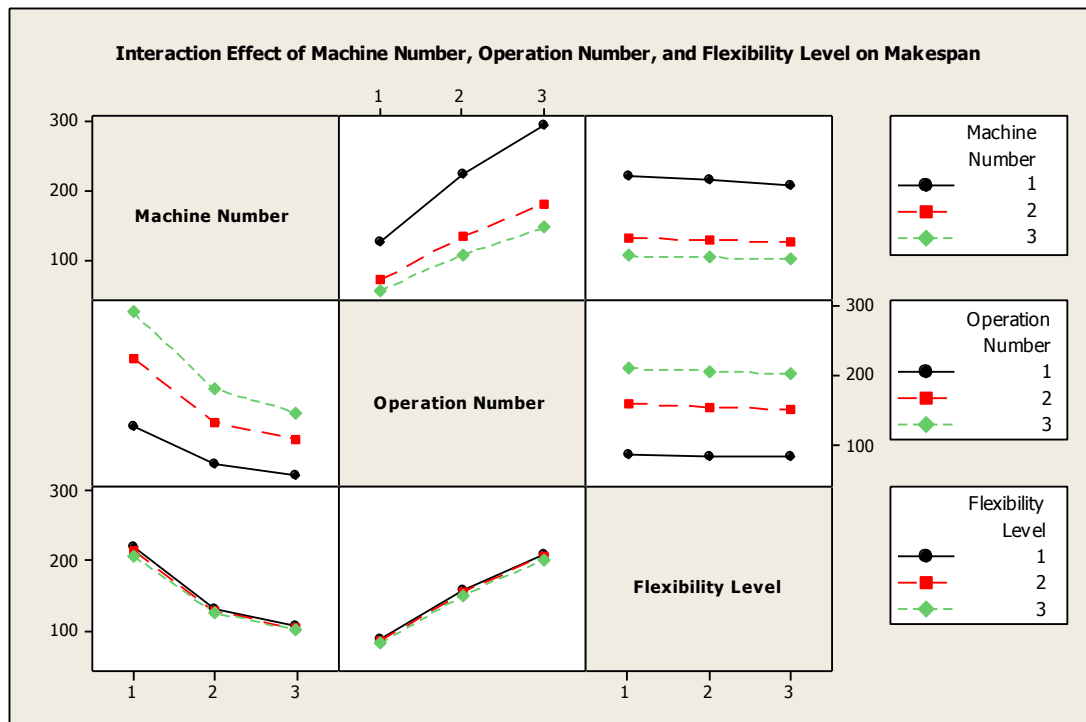


Figure 4.11 Interaction Effect of Machine Number, Operation Number, and Flexibility Level on Makespan

Interaction effect of machine number, operation number, and flexibility level on makespan is given in Figure 4.11. From Figure 4.11, it is noticeable that the interaction between flexibility level and the other two factors (i.e., Machine number and operation number) has no significant effect on makespan performance of the shop. From flexibility level&operation number and flexibility level&machine number interaction graphs it is clearly seen that as the level of the flexibility increases only marginal improvements gained on makespan performance at different levels of other factors which makes machine number, operation number, and flexibility level interaction effect on makespan insignificant. By looking into operation number interaction graphs it is clearly seen that makespan performance of

the shop deteriorates as the level of operation number increases. Also, it is apparently seen that at low level of machine number the makespan performances of the shop significantly deteriorates while increasing the level of operation number from low to high whereas for higher levels of machine number it is observed that the difference between makespan performances for each level of part number get much more closed. Further insights can be gained by analyzing Figure 4.11 thoroughly. As a result, from the Figure 4.11, it is obvious that the best makespan performance can be gained for high levels of machine number together with low level of operation number regardless of the flexibility level.

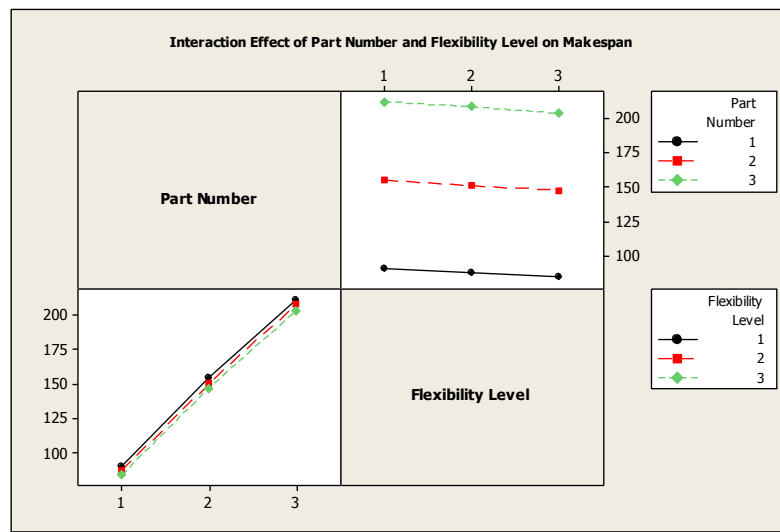


Figure 4.12 Interaction Effect of Part Number and Flexibility Level on Makespan

Interaction effect of flexibility level and part number on makespan performance is given in Figure 4.12. From figure 4.12 it is clearly seen that as the level of the flexibility increases only marginal improvements gained on makespan performance at different levels of part number which makes flexibility level and part number interaction effect on makespan insignificant.

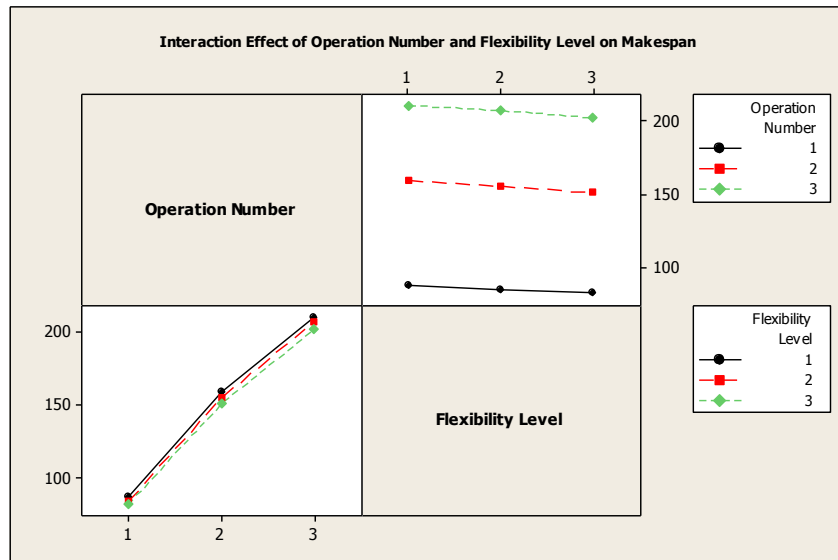


Figure 4.13 Interaction Effect of Operation Number and Flexibility Level on Makespan

Interaction effect of flexibility level and operation number on makespan performance is given in Figure 4.13. From figure it is clearly seen that as the level of the flexibility increases only marginal improvements gained on makespan performance at different levels of operation number which makes operation number and flexibility level interaction effect on makespan insignificant.

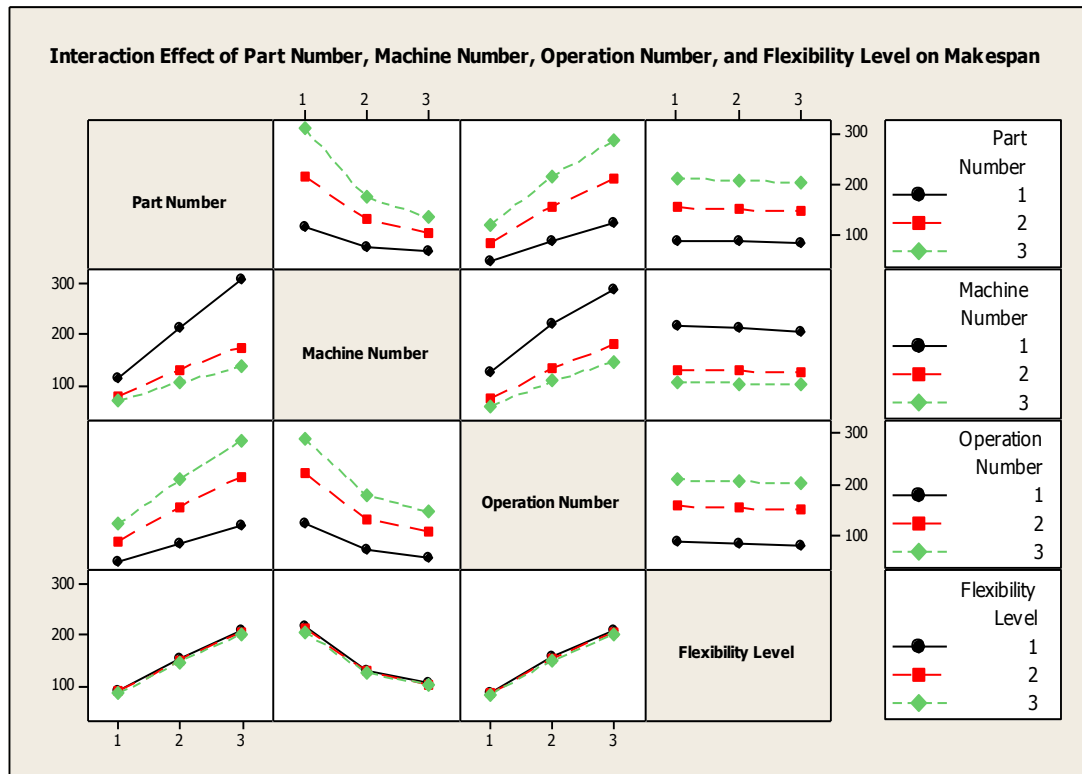


Figure 4.14 Interaction Effect of Part Number, Machine Number, Operation Number, and Flexibility Level on Makespan

Figure 4.14 summarizes all of the discussions made above. It is apparently seen from the figure that flexibility level has negligible effect on makespan performance of the shop. Makespan performance of the shop deteriorates as the level of operation number and part number increases. Also, makespan performance of the shop improves as the level of machine number increases.

CHAPTER 5

APPLICATION OF SIMULATION OPTIMIZATION BY USING GA TO DYNAMIC FJSSP

This part of the study presents an optimization via simulation approach to solve dynamic FJSSPs. The study deals with both determining the best process plan for each part and then finding the best machine for each operation in a dynamic FJSSP environment. In this respect, a GA is adapted to determine best part processing plan for each part and then select appropriate machines for each operation of each part according to the determined part processing plan. Genetic algorithm solves the optimization phase of solution methodology. Then these machine-operation pairings are utilized by discrete-event system simulation model to estimate their performances. These two phases of the study follow each other iteratively. The goal of methodology is to find the solution that minimizes total of average flow times for all parts.

5.1. System Characteristics and Problem Definition

Having flexibility option enables job shops to respond faster to various changes such as machine breakdowns, demand fluctuation and product mix. Due to its great flexibility on the shop floor and the efficiency of large volume production, the production scheduling and control in flexible manufacturing systems becomes very complex as the number of jobs, operations, parts and machines increases (Siwamogsatham, et al., 2004). System characteristics and definition of such a challenging problem is given below.

5.1.1. System Characteristics

General job shop scheduling assumptions are applied here for the purpose of model standardization (Baker, 1974). The required parameters of the model are as follows:

- Parts dynamically arrive to the shop and the arrival rate is exponentially distributed with a mean of 130 unit of time.
- At each arrival only one part arrives at the system (i.e., no batches considered).
- Jobs are released to the shop as soon as they arrive to the system.
- FIFO dispatching rule considered in front of all machine queues.
- There are N parts $P=\{p_1, p_2, \dots, p_N\}$ indexed by j .
- There are M machines $M=\{m_1, m_2, \dots, m_M\}$ indexed by k .
- Alternative process plans are predetermined for each part.
- Operations $\{O_{i1}, O_{i2}, \dots, O_{iN}\}$ indexed by i can be processed on more than one machine.
- Operations' processing times are sampled from triangular distribution.
- Pre-emption is not allowed.
- A machine cannot perform more than one operation at a time.
- Consecutive operations of parts can be processed on the same machine.
- The warm-up period for the shop is determined to be 20% of the total simulation run after preliminary runs. The data are then collected for the remaining 80% of the total simulation run.
- Common random numbers are used within each replication as a variance reduction technique, i.e., for each replication we use the same stream of random numbers for both system configurations we wish to compare. By this way, comparison of different methods under similar experimental conditions enabled.
- The objective is to find a solution with minimum total of average flow times which is represented by Eq. (5.2).
- The setup times and transportation times assumed to be zero and are excluded from the model.
- Machine breakdowns are neglected.

F_j can be calculated by using Eq. (5.1). For more details the reader should refer to Baykasoglu, et al. (2008).

$$F_j = \sum_{m \in S\{j\}} (s_{jm} + t_{jm} + w_{jm} + p_{jm}) \quad (5.1)$$

where

F_j : Flowtime of each part.

j : Part type

m : Station no

p_{jm} : Processing time of part j at station m .

s_{jm} : Setup time needed by part j at station m .

t_{jm} : Transportation time necessary for moving part j from station m to the next station on its route.

w_{jm} : Queue waiting time of part j at station m 's queue.

$S\{j\}$: The set of stations which are placed on part j 's route.

$$\sum_{j \in N} F_j \quad (5.2)$$

where

N : Number of parts to be produced.

5.1.2. Problem Definition of dynamic FJSSP

The FJSSP is addressed in two phases; (1) Optimization, (2) Simulation. Optimization phase composed of two consecutive stages; (1) Selection of appropriate process plan for each part by using GA, and then (2) Matching the most suitable machine-operation pairs by using GA. In the simulation phase, new sets of values for decision variables (i.e., obtained machine-operation pairings for all parts) are generated by GA from the first phase evaluated by running the discrete-event simulation model of dynamic FJSS. This is an iterative process that successively generates new sets of values for the decision variables, not all of them improving, but which, over time, provides a highly efficient trajectory to the best solutions. The process continues until some termination criterion is satisfied — usually stopping after a number of simulations or when the GA determines the objective value has stopped improving. Its ultimate goal is to find the solution that optimizes (maximizes or minimizes) the value of the model's objective. By this way finding the best

process plan for each part and then machine-operation pairs that minimizes the total of average flow times achieved in an iterative manner.

Meanwhile, it should be noted that while obtaining the total of average flow times, the system (i.e., flexible job shop) dynamics (i.e., random arrivals, random operation processing times) are taken into account. In related literature, limited number of studies handled both of “dynamic” and “stochastic” FJSSP. Most of these studies are applied optimization techniques and simulation separately. In our study, an effective optimization tool (GA) and simulation are integrated to solve and to evaluate FJSSP. From this point of view, to our best knowledge this study is a premier one that solves “dynamic” and “stochastic” FJSSP simultaneously (i.e., iterative process).

A sample instance which takes place in the study of Baykasoglu, et al. (2008) is utilized with minor modifications as a test bed for the proposed approach in this study. In its modified form, the problem includes random interarrival times and random processing times to further reflect stochastic nature that real world problems inherent. The process plans for each part is given in the Table 5.1 and appropriate machine sets for each operation is given in Table 5.2. Processing time distributions for each part is shown in Table 5.3.

Table 5.1 Part-operation plans data

Parts	Process Plans	Process plans and processing sequences
1	1	Op1, Op2, Op3
	2	Op1, Op3, Op2
2	1	Op2, Op3, Op1
	2	Op3, Op2, Op1
3	1	Op1, Op3
	2	Op1, Op2

Table 5.2 Machine-operation suitability data

Machines	Operation1	Operation2	Operation3
M1	*		
M2	*		*
M3		*	
M4			*
M5		*	

Table 5.3 Processing times distributions for each part

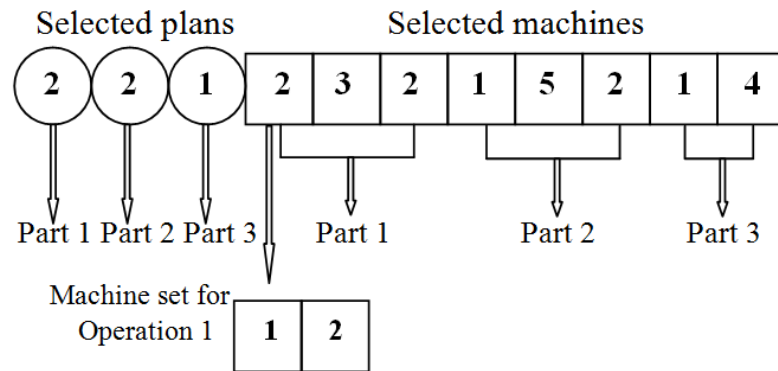
	<i>Machine 1</i>	<i>Machine 2</i>	<i>Machine 3</i>	<i>Machine 4</i>	<i>Machine 5</i>
<i>Part 1</i>	Tria(7,9,11)	Tria(10,12,14)	Tria(8,10,13)	Tria(10,15,17)	Tria(6,8,10)
<i>Part 2</i>	Tria(10,12,14)	Tria(12,14,16)	Tria(10,12,14)	Tria(12,15,18)	Tria(4,9,13)
<i>Part 3</i>	Tria(8,10,13)	Tria(13,15,17)	Tria(7,9,11)	Tria(10,15,20)	Tria(7,10,12)

5.2. Proposed GA for dynamic FJSSP

GA is developed to select the appropriate process plan for each part and then to match the most appropriate machine-operation pairs among machine sets, respectively.

5.2.1. Chromosome Representation for the Considered FJSSP

An integer based coding system is adopted to represent the problem solution. The chromosome structure is considered to be composed of two parts: (1) Process plan selection denoted by circles in Figure 5.1 and (2) machine-operation matching denoted by squares in Figure 5.1 (i.e., operation sequence). The first part of the chromosome locates integer coding for the selection of process plan for each part. Then, the second part of the chromosome locates integer coding for the machines. The proposed chromosome representation is depicted in Figure 5.1.

**Figure 5.1** Illustration of the proposed chromosome representation

5.2.2. Initial Population Generation for the Considered FJSSP

The algorithm begins with the selection of process plans for each part. Then, according to the selected process plan corresponding machine is assigned to each operation. Plans and machines are assigned randomly for the initial population. Then, the initial population is fed into the simulation model to be evaluated. Note that,

simulation model takes these values as a control (since they control the inputs to the model) and after completion of the simulation run the responses sent (i.e., outputs) to GA. The GA module evaluates the responses from the simulation run, analyzes and integrates these with responses from previous simulation runs, and determines a new set of values for the controls which are then evaluated by running the simulation model etc. Intermediate steps of the evaluation process of GA are given in subsequent sections.

As mentioned before, chromosomes are translated to the mating pool by tournament approach. Two individual are selected from the population and, the best one among them is translated to the mating pool.

5.2.3. Crossover for the proposed GA

Crossover operator designed to take features from both parents and it's performed in two stages. In the first stage, crossover operation is related to the first three genes of chromosomes (i.e., process plan selection). In the second stage crossover operation is related to the rest of genes (i.e., machine-operation assignment). First of all, process plans selected randomly for each part from both selected parents. In machine-operation assignment section, for each part type, genes are replaced according to the process plan section. Genes are randomly exchanged between individuals having same process plan sequence. This is implemented for each part separately. A sample crossover operator is depicted in Figure 5.2.

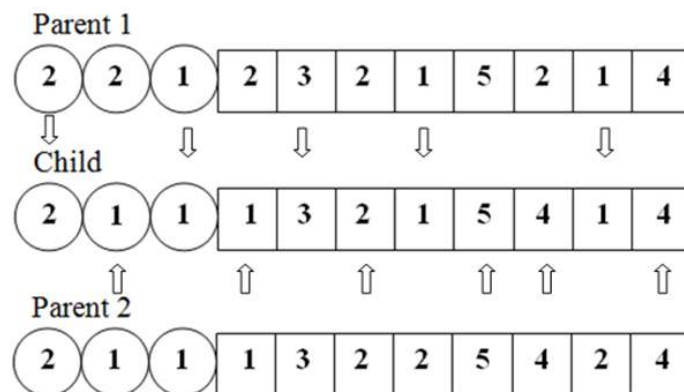


Figure 5.2 An illustration of crossover operator

5.2.4. Mutation for the Proposed GA

In general, mutation is applied with small probability because large probability may disorder the good individuals (Zhang, et al., 2011). Mutation probability is usually determined to be 0.05. A probability value for each gene in the chromosome is generated and compared with the mutation probability value. If the mutation probability value bigger than the gene's probability value, the gene is changed accordingly. See Figure 5.3 for how this mechanism works.

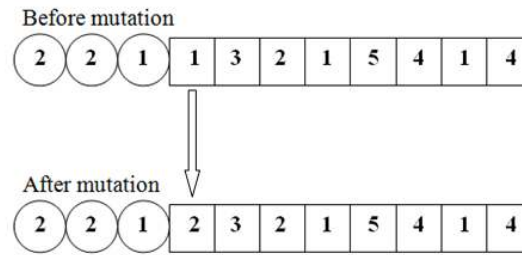


Figure 5.3 An illustration of a mutation operator

5.3. The methodology of Optimization via Simulation

As mentioned before, the aim of this study is to determine the best process plan for each part and then determine the best machine-operation pairing along the part's routing while minimizing the total of average flow times. In this respect, the problem can be interpreted as optimizing part routing in a FJSSP. For this purpose a computer model of dynamic FJSS is coded in ARENA. Note that this model will be evaluated in simulation stage of the proposed methodology. Furthermore, a problem specific GA is coded for the optimization stage of the proposed methodology. It should be emphasized that this is a "*specific*" optimization routine like other optimization tools. Then, this optimization routine is linked with simulation model. By this way (1) the problem transformed into resolvable format and (2) the user given the capability of managing his/her optimization routine easily.

5.3.1. Problem Formulation

There has been many optimization formulations offered in literature. Their common purpose is to find a "*configuration*" or "*design*" that minimizes the objective function. We adopted Fu (2002)'s notation to represent our "*objective function*" or "*fitness function*" (in GA terminology). As mentioned, the total of average flow

times is considered as objective function to be minimized. In equation (5.3) objective function is given:

$$\min_{\theta \in \Theta} J(\theta) = E[L(\theta, w)] \quad (5.3)$$

Where $\theta \in \Theta$ represents the (vector of) input variables, $J(\theta)$ is the objective function, w represents a sample path (simulation replication), and L is the sample performance measure. We will use $\hat{f}$ to represent an estimate for $J(\theta)$, e.g., $L[\theta, \omega]$ would provide one such estimator that is unbiased. The constraint set Θ defined to be finite (e.g., an operation of any part can be processed on a certain number of machines).

$\theta \in \Theta$: A representation of a problem solution (i.e., the vector of decision variables) which is called as chromosome in this study.

Θ : All possible chromosomes.

$L[\theta, 2]$: Response of simulation model after replication 2 with chromosome representation θ .

Once the optimization problem is described (by means of selecting decision variables, the objective, and possibly imposing constraints), simulation model is called every time a different set of decision variables' values needed to be evaluated. The GA module (e.g., optimization routine) evaluates the responses from the current simulation run, analyzes and integrates these with responses from previous simulation runs, and determines a new set of values for the decision variables which are then fed into the simulation model to be evaluated. This is a multi stage iterative process that successively generates new sets of values for the decision variables, not all of them improving, but which, over time, provides a highly efficient trajectory to the best solutions. The process continues until some termination criterion is satisfied — usually stopping after a number of simulations or when the GA module determines the objective value has stopped improving. Its ultimate goal is to find the solution that optimizes (maximizes or minimizes) the value of the model's objective. This somewhat complex process is depicted in Figure 5.4 to increase its intelligibility.

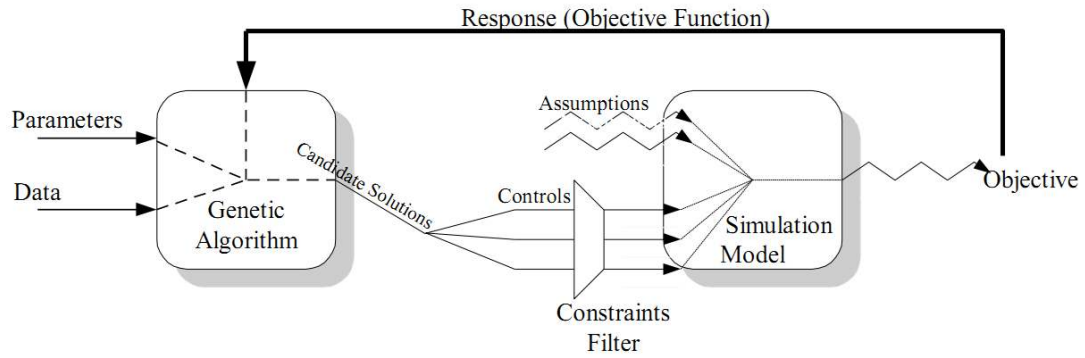


Figure 5.4 Optimization via Simulation Methodology

5.4. Computational Results and Discussions

The GA coded using Visual Basic® for Applications by Microsoft®. A Pseudo code of GA is given in APPENDIX B. After preliminary runs, the parameter of the GA is determined to be as follows:

Population size (P_s): 10

Number of generations (G_{max}): 20

Mutation probability (p_m): 0.05

Chromosome length (C_l : number of total parts + number of total operations): 11

Selection type: Tournament approach

The hypothetical simulation model coded using ARENA. A Pseudo code of simulation model is given in APPENDIX B. In simulation experiments two levels of flexibility taken into account to show the efficiency of the GA. (1) Full flexibility (i.e., all machines are considered to be available for each operation to be chosen as alternatives), (2) partial flexibility (i.e., for each operation a machine has to be chosen from among several “*available alternatives*”). It should be noted that increasing the level of the flexibility increases the number of alternatives which complicates the job of GA.

Our solution approach starts up with five different initial solutions determined by GA, for each level of flexibility, to guarantee to obtain near optimal solutions. For each generation of all initial solutions final machine-operation pairings are fed into simulation model. Then, ten independent replications are conducted for one simulation run. At the end of each simulation run an average of ten replications fed into GA as an objective function. This iterative procedure repeats itself until 20

iterations completed. Note that, common random numbers (i.e., the same random number streams) are used for each simulation run to improve estimates of differences in performance.

Figure 5.5 shows an average “*total of average flow times*” of 5 runs with different initial solutions and the best “*total of average flow times*” among 5 runs with full flexibility. Figure 5.6 shows an average “*total of average flow times*” of 5 runs with different initial solutions and the best “*total of average flow times*” among 5 runs with partial flexibility.

Figure 5.4 shows the selected machine-operation pairings of the best results. From figure 5.4, note that all operations of part 1 are processed on only machine 5 with full flexibility. This is a result of system characteristics which enable process of consecutive operations of a part on the same machine. In this case, GA strives only for coming up with a solution for the system that specifies optimal machine-operation pairs (i.e., part routings) while minimizing the objective function.

Table 5.4 Operation sequences and machine numbers of best results

<i>Full Flexibility</i>			Total average part flow times
	<i>Part 1</i>		96
	Operation Sequence	Machine Sequence	
	O11 O12 O13	5_5_5	
	<i>Part2</i>		
	Operation Sequence	Machine Sequence	
	O23 O22 O21	1_5_5	
	<i>Part3</i>		
	O31 O33	3_3	
<i>Partial Flexibility</i>			Total average part flow times
	<i>Part 1</i>		106
	Operation Sequence	Machine Sequence	
	O11 O12 O13	1_5_2	
	<i>Part2</i>		
	Operation Sequence	Machine Sequence	
	O22 O23 O21	5_2_1	
	<i>Part3</i>		
	O31 O32	1_3	

GA achieves the best when “*total of average flow times*” is equal to 96, at 4th generation in the fifth run with full flexibility. And with the partial flexibility, our algorithm improves the best “*total of average flow times*” very quickly, that the best “*total of average flow times*”, equal to 106, is achieved at 2nd generation in first run. The very early convergence to optimum can be explained by limited choice of alternative machines with partial flexibility (see Table 5.2 for alternative machines with partial flexibility).

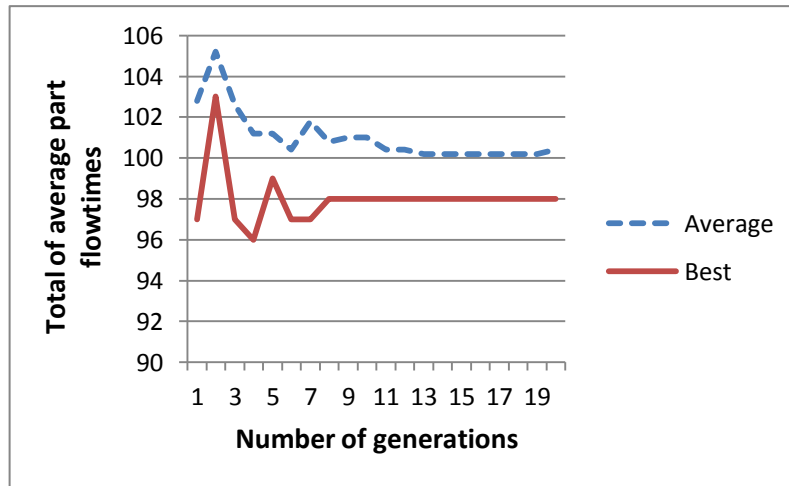


Figure 5.5 Total of average flow times obtained with full flexibility

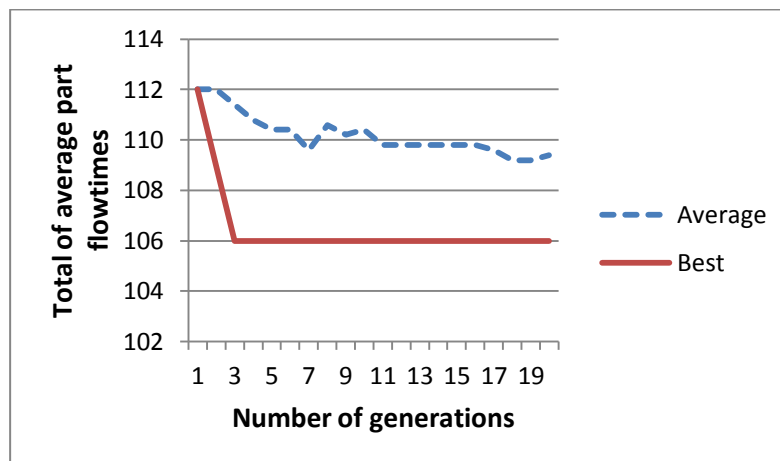


Figure 5.6 Total of average flow times results obtained with partial flexibility

CHAPTER 6

CONCLUSION

FJSSP is handled as static, stochastic and dynamic scheduling problem within the context of this study. FJSSP deals with two sub problems: (1) selecting a machine from alternative machine set for each operation, and (2) sequencing the operations on all machines to achieve a feasible schedule. First, the GA is adapted to static FJSSP. In static FJSSP, all jobs and machines are ready at time 0. And, the processing times of all operations are already known. The maximum completion times of jobs, C_{max} (makespan), is determined as the objective function to be minimized. Initial population is generated with a mixed initial strategy. Four strategies are used to generate initial population. This mixed strategy helps achieving near optimal solution more quickly. The algorithm is tested with Brandimerte's data set (BRdata) (Brandimerte, 1993). Ten problem instances are run for five times and the best results among them are selected. The proposed GA obtains the best known solution in problems of Mk01, Mk03, Mk07 and Mk08. Besides, the proposed GA can find the solutions with small parameter size as well. Therefore the algorithm can be considered a promising candidate for further studies.

An experimental design is generated for static stochastic FJSSPs to measure the impact of flexibility on shop performance by using an efficient GA. A four full factorial design is utilized to evaluate the performance of the shop and investigate relationships between the considered factors. Three levels of each number of parts, number of jobs, number of operations, and level of flexibility are included to the experimental design. A total of 81 experiments performed to investigate all factor level combinations. The algorithm is run within 20 replications of each experiment to evaluate the performance of the static stochastic flexible job shop. Results revealed that makespan performance of the shop is significantly affected by the considered

factors and some factor interactions. It is seen that all main factors have significant effect on makespan performance at 0.05 significance level. Makespan performance of the shop improves as the level of machine number increases. It is observed that setting the machine number at its low level makes makespan performance of the shop becomes much more sensitive to different levels of part number. The best makespan performance observed for high levels of machine number together with low level of part number regardless of the flexibility level. As the level of the flexibility increases only marginal improvements gained on makespan performance at different levels of part number which makes flexibility level and part number interaction effect on makespan insignificant. It is apparently seen that flexibility level has negligible effect on makespan performance of the shop. Makespan performance of the shop deteriorates as the level of operation number and part number increases. Also, makespan performance of the shop improves as the level of machine number increases.

In Section 5 an optimization via simulation methodology is proposed to solve a dynamic stochastic FJSSP which inherents considerable complexity. By the proposed approach the best process plan for each part and then the best machine-operation pairing along the part's routing is determined while minimizing the total of average flow times. By this aim, an optimization via simulation approach is adopted to a dynamic stochastic FJSSP with random inter-arrivals and processing times. The GA module is used as an optimization routine and it is iterated by the objective function obtained from simulation. This synergic integration is aimed at tackling two common challenges in designing and optimizing complex, dynamic, and stochastic real-world production systems. This includes problem formulation/representation in terms of objective function and constraints (i.e., simulation phase of the methodology), searching the often complex and large problem domain (i.e., optimization phase of the methodology). Note that, in most cases it is impractical to search complex and large problem domain.

One of the most important findings of the study is that the objective function (i.e., total of average flow times) improves as the level of flexibility increases. Also, it is shown that good solutions can be obtained using GA as a tool for systematically guiding the recursive process towards convergence to an optimum solution by considering only a limited number of alternative configurations of the system. Without the help of GA it would be impractical to evaluate all possible system configurations resulting from the combination of design factors, as the number of possibilities could be in thousands and in many cases in millions.

The main contribution of this study is that the optimization via simulation methodology is used in an iterative manner automatically. Thus, this methodology can be modified and used for most of the other industrial and service problems. For further studies, the methodology must be extended and compared with other methods. Furthermore, some production constraints or other performance measures together with machine breakdowns and other random events can be included in this problem.

Other meta-heuristic algorithms, such as simulated annealing, artificial immune systems, differential evolution, bee algorithms etc., can be used in optimization phase of the methodology.

It would be interesting to use this methodology for other similar problem types in this area, i.e., shop configuration problems, part-machine grouping etc. Also, it is possible to apply this methodology to many real life problems exist in service and manufacturing systems.

REFERENCES

- Adibi, M. A., Zandieh, M., & Amiri, M. (2010). Multi-objective scheduling of dynamic job shop using variable neighborhood search. *Expert Systems with Applications* , 37, 282-287.
- Al-Hinai, N., & ElMekkaway, T. Y. (2011). Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *Int J Production Economics* , 132, 279-291.
- Baker, K. R. (1974). *Introduction the theory of scheduling*. New York: Wiley.
- Baker, K. R., & Trietsch, D. (2009). *Principles of Sequencing and Scheduling* (1st ed.). Hoboken, New Jersey: John Wiley & Sons.
- Banks, J., Carson, J. S., Nelson, B. L., & Nicol, D. M. (2010). *Discrete Event Systems Simulation* (3rd ed.). Englewood Cliffs, NJ: Prentice Hall.
- Baykasoğlu, A., & Ozbakir, L. (2008). Analysing the effect of flexibility on manufacturing systems performance. *Journal of Manufacturing Technology Management* , 19 (2), 172-193.
- Baykasoğlu, A., Göçken, M., & Unutmaz, Z. D. (2008). New approaches to due date assignment in job shops. *European Journal of Operational Research* , 187, 31-45.
- Baykasoğlu, A., Özbakır, L., & Sönmez, A. İ. (2004). Using multiple objective tabu search and grammars to model and solve multi-objective flexible job shop scheduling problems. *Journal of Intelligent Manufacturing* , 15, 777-785.
- Bondal, A. A. (2008). *Artificial Immune Systems Applied to Job Shop Scheduling*. Ohio: Ohio University.

Brandimerte, P. (1993). Routing and scheduling in a flexible job shop by taboo search. *Annals of Operations Research* , 41, 157-183.

Büyükköprü, E. (2005). *Flexible Job Shop Scheduling via Simulation under sequence dependent setup times*. İzmir: Dokuz Eylül University.

Chen, H., Ihlow, J., & Lehmann, C. (1999). A Genetic Algorithm for flexible job shop scheduling. *International Conference on Robotics & Automation* (s. 1120-1125). Detroit, Michigan: Proceedings of the 1999 IEEE.

Cheng, H.-C., Chiang, T.-C., & Fu, L.-C. (2011). A two-stage hybrid memetic algorithm for multiobjective job shop scheduling. 38, 10983-10998.

Chong, C. S., Low, M. Y., Sivakumar, A. I., & Gay, K. L. (2006). A Bee colony optimization algorithm to job shop scheduling. *Proceedings of the 2006 Winter Simulation Conference*, (s. 1954-1961).

Essafi, I., Mati, Y., & Dauzere-Peres, S. (2008). A genetic local search algorithm for minimizing total weighted tardiness in the job shop scheduling problem. *Computers & Operations Research* , 35, 2599-2616.

Fang, J., & Xi, Y. (1997). A rolling horizon job shop rescheduling strategy in the dynamic environment. *Int J Adv Manuf Technol* , 13, 227-232.

Fattahi, P., Mehrabad, M. S., & Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *J Intell Manuf* , 18, 331-342.

Fiğlalı, N., Özkale, C., Engin, O., & Fiğlalı, A. (2009). Investigation of Ant System parameter interactions by using design of experiments for job-shop scheduling problems. *Computers & Industrial Engineering* , 56 , 538-559.

Fu, M. C. (2002). Optimization for Simulation: Theory vs. Practice . *INFORMS Journal on Computing* , 3 (14), 192–215.

Gao, J., Gen, M., Sun, L., & Zhao, X. (2007). A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. *Computers & Industrial Engineering* , 53, 149-162.

- Gao, J., Sun, L., & Gen, M. (2008). A Hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research* , 35, 2892-2907.
- Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flow shop and job shop scheduling. *1*, 117-129.
- Geyik, F., & Cedimoğlu, İ. H. (2004). The strategies and parameters of tabu search for job shop scheduling. *Journal of Intelligent Manufacturing* , 15, 439-448.
- Gholami, M., & Zandieh, M. (2009). Integrating simulation and genetic algorithm to schedule a dynamic flexible job shop. *J Intell Manuf* , 20, 481-498.
- Giffler, J. & Thompson, G.L. (1960). Algorithms for solving production scheduling problems, *Operations Research*, 8, 487-503.
- Gu, J., Gu, M., Cao, C., & Gu, X. (2010). A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem. *Computers & Operations Research* , 37, 927-937.
- Gu, J., Gu, X., & Gu, M. (2009). A novel parallel quantum genetic algorithm for stochastic job shop scheduling. *J Mathematical analysis and applications* , 355, 63-81.
- Ho, N. B., & Tay, J. C. (2004). GENACE: an efficient cultural algorithm for solving the flexible job shop problem. *IEEE international conference on robotics and automation*, (s. 1759-1766).
- Hsueh-Chien Cheng, T.-C. C.-C. (2011). A two-stage hybrid memetic algorithm for multiobjective job shop scheduling. 38, 10983-10998.
- Hu, Y., Yin, M., & Li, X. (2011). A novel objective function for job shop scheduling problem with fuzzy processing time and fuzzy due date using differential evolution algorithm. *Int J Adv Manuf Technol* , 56, 1125-1138.
- Innani, A. D. (2004). *Applying data mining to job shop scheduling using regression analysis*. Ohio: Ohio University.

Kacem, I., Hammadi, S., & Borne, P. (2002). Approach by localization and multiobjective evolutionary optimization for flexible job shop scheduling problems. *IEEE transactions on systems, Man, And Cybernetics* , 1-12.

Kacem, I., Hammadi, S., & Borne, P. (2002). Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation* , 60, 245-276.

Law, A. M. (2007). *Simulation Modelling and Analysis* (4th ed.). New York: McGraw Hill.

Law, A. M., & Kelton, W. D. (2000). *Simulation Modelling and Analysis* (3rd ed.). New York: McGraw Hill.

Lei, D. (2008). A Pareto archive particle swarm optimization for multi objective job shop scheduling. *Computers & Industrial Engineering* , 54, 960-971.

Lei, D. (2012). Interval job shop scheduling problems. *Int J Adv Manuf Technol* , 60, 291-301.

Lei, D. (2011a). Population based neighbourhood search for job shop scheduling with interval processing time. *Computers & Industrial Engineering* , 61, 1200-1208.

Lei, D. (2011b). Scheduling stochastic job shop subject to random breakdown to minimize makespan. *Int J Adv Manuf Technol* , 55, 1183-1192.

Lei, D. (2011c). Simplified multi-objective genetic algorithms for stochastic job shop scheduling. *Applied Soft Computing* , 11 , 4991-4996.

Lei, D., & Xiong, H. (2008). Job Shop Scheduling with Stochastic Processing Time Through Genetic Algorithm. *Proceedings of the Seventh International Conference on Machine Learning and Cybernetics*, (s. 941-946). Kunming.

Lian, Z. (2010). A Local and Global Search Combine Particle Swarm Optimization Algorithm for Job Shop Scheduling to Minimize Makespan. *Discrete Dynamics in Nature and Society* , 1-11.

- Lian, Z., Jiao, B., & Gu, X. (2006). A similar particle swarm optimization algorithm for job shop scheduling to minimize makespan. *Applied Mathematics and Computation* , 183, 1008-1017.
- Liouane, N., Saad, I., Hammadi, S., & Borne, P. (2007). Ant systems & Local Search Optimization for flexible job shop scheduling production. *Int. J. Computers, Communication & Control* , 2, 174-184.
- Manikas, A., & Chang, Y.-L. (2009). Multi-criteria sequence-dependent job shop scheduling using genetic algorithms. *Computers & Industrial Engineering* , 56, 179-185.
- Martinis, P. (2003). *A hybrid method for selecting scheduling schemes in a manufacturing environment*. Knoxville: The University of Tennessee.
- Mastrolilli, M., & Gambardella, L. (2000). Effective neighborhood functions for the flexible job shop problem. *Journal of Scheduling* , 3, 3-20.
- Medaglia, A. L. (2000). *Simulation Optimization using soft computing*. North Carolina State University.
- Meeran, S., & Morshed, M. S. (2011). A Hybrid genetic tabu search algorithm for solving job shop scheduling problems a case study. *J Intell Manuf* , DOI 10.1007/s10845-011-0520-x.
- Metta, H. (2008). *Adaptive, Multi-Objective Job Shop Scheduling Using Genetic Algorithms*. Kentucky: University of Kentucky.
- Montgomery, D. C., & Runger, G. C. (2002). *Applied Statistics and Probability for Engineers* (3 ed.). New York: Wiley.
- Moon, I., Cha, B. C., & Bae, H. C. (2006). Hybrid genetic algorithm for group technology economic lot scheduling problem. *International Journal of Production Research* , 44 (21), 4551-4568.
- Moon, I., Lee, S., & Bae, H. (2008). Genetic Algorithms for job shop scheduling problems with alternative routings. *International Journal of Production Research* , 2695-2705.

- Ouelhadj, D., & Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *J Sched* , 12, 417-431.
- Pan, J. C.-H., & Huang, H.-C. (2009). A Hybrid genetic algorithm for no-wait job shop scheduling problems. *Expert Systems with Applications* , 36, 5800-5806.
- Paris, J. L., Tautou-Guillaume, L., & Pierreval, H. (2001). Dealing with design options in the optimization of manufacturing systems: An evolutionary approach. *Int Journal of Production Research* , 39 (6), 1081-1094.
- Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A Genetic Algorithm for the flexible job shop scheduling problem. *Computers & Operations Research* , 35, 3202-3212.
- Pinedo, M. L. (2005). *Planning and Scheduling in Manufacturing and Services*. New York: Springer.
- Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems* (3th ed.). New York: Prentice Hall.
- Rajabinasab, A., & Mansour, S. (2011). Dynamic flexible job shop scheduling with alternative process plans: an agent based approach. *Int J Adv Manuf Technology* , 54, 1091-1107.
- Roshanaei, V., Naderi, B., Jolai, F., & Khalili, M. (2009). A variable neighborhood search for job shop scheduling with set-up times to minimize makespan. *Future Generation Computer Systems* , 25, 654-661.
- Rossi, A., & Dini, G. (2007). Flexible job shop scheduling with routing flexibility and separable setup times using ant colony optimisation method. *Robotics and Computer-Integrated Manufacturing* , 23, 503-516.
- Rui Zhang, S. S. (2012). A two-stage hybrid particle swarm optimization algorithm for stochastic job shop scheduling. *Knowledge-Based Systems* , 27, 393-406.
- S Meeran, M. S. (2011). A Hybrid genetic tabu search algorithm for solving job shop scheduling problems a case study. *J Intell Manuf* , DOI 10.1007/s10845-011-0520-x.

- Saidi-Mehrabad, M., & Fattahi, P. (2007). Flexible job shop scheduling with tabu search algorithms. *Int.J.Advance Manufacturing Technology* , 32, 563-570.
- Sha, D. Y., & Hsu, C.-Y. (2006). A hybrid particle swarm optimization for job shop scheduling problem. *Computers & Industrial Engineering* , 51 , 791- 808.
- Siwamogsatham, T., & Saygin, C. (2004). Auction-based distributed scheduling and control scheme for flexible manufacturing system. *International Journal of Production Research* , 42 (3), 542-572.
- Xia, W., & Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering* , 48, 409-425.
- Yamada, T. (2003). *Studies on metaheuristics for job shop and flow shop scheduling problems*. Kyoto: Kyoto University.
- Yoshitomi, Y., & Yamaguchi, R. (2003). A genetic algorithm and the monte carlo method for stochastic job shop scheduling. *International Transactions in Operational Research* , 10, 577-596.
- Zandieh, M., & Adibi, M. A. (2010). Dynamic job shop scheduling using variable neighbourhood search. *Int J Production Research* , 48, 2449-2458.
- Zhang, C. Y., Li, P., Guan, Z., & Rao, Y. (2007). A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research* , 34, 3229-3242.
- Zhang, C. Y., Rao, Y., & Li, P. (2008). An effective hybrid genetic algorithm for the job shop scheduling problem. *Int J Adv Manuf Technol* , 39, 965-974.
- Zhang, G., Gao, L., & Shi, Y. (2011). An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications* , 38, 3563-3573.
- Zhang, G., Shao, X., Li, P., & Gao, L. (2009). An effective hybrid particle swarm optimization algortihm for multi-objective flexible job shop scheduling problem. *Computers & Industrial Engineering* , 56, 1309-1318.

Zhang, Q., Manier, H., & Manier, M. A. (2012). A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Computers & Operations Research* , 39, 1713-1723.

Zhang, R., & Wu, C. (2010). A Hybrid immune simulated annealing algorithm for the job shop scheduling problem. *Applied Soft Computing* , 10 , 79-89.

Zhang, R., Song, S., & Wu, C. (2012). *Applied Soft Computing* , doi:10.1016/j.asoc.2012.02.024.

Zhang, R., Song, S., & Wu, C. (2012). A Hybrid artificial bee colony algorithm for the job shop scheduling problem. *Int J Production Economics* , <http://dx.doi.org/10.1016/j.ijpe.2012.03.035>.

Zhang, R., Song, S., & Wu, C. (2012). A hybrid differential evolution algorithm for job shop scheduling problems with expected total tardiness criterion. *Applied Soft Computing* , doi:10.1016/j.asoc.2012.02.024.

Zhang, R., Song, S., & Wu, C. (2012). A two-stage hybrid particle swarm optimization algorithm for stochastic job shop scheduling. *Knowledge-Based Systems* , 27, 393-406.

Zhou, R., Nee, A. Y., & Lee, H. P. (2009). Performance of an ant colony optimisation algorithm in dynamic job shop scheduling problems. *Int J Production Research* , 47 , 2903-2920.

APPENDIX A

Data set explanations:

In the first line there are (at least) 2 numbers: the first is the number of jobs and the second the number of machines (the 3rd is not necessary, it is the average number of machines per operation)

Every row represents one job: the first number is the number of operations of that job, the second number (let's say $k \geq 1$) is the number of machines that can process the first operation; then according to k , there are k pairs of numbers (machine, processing time) that specify which are the machines and the processing times; then the data for the second operation and so on...

Example: Fisher and Thompson 6x6 instance, alternate name (mt06)

```
6 6 1
6 1 3 1 1 1 3 1 2 6 1 4 7 1 6 3 1 5 6
6 1 2 8 1 3 5 1 5 10 1 6 10 1 1 10 1 4 4
6 1 3 5 1 4 4 1 6 8 1 1 9 1 2 1 1 5 7
6 1 2 5 1 1 5 1 3 5 1 4 3 1 5 8 1 6 9
6 1 3 9 1 2 3 1 5 5 1 6 4 1 1 3 1 4 1
6 1 2 3 1 4 3 1 6 9 1 1 10 1 5 4 1 3 1
```

first row = 6 jobs and 6 machines 1 machine per operation

second row: job 1 has 6 operations, the first operation can be processed by 1 machine that is machine 3 with processing time 1.

Table A1: Mk01 problem data

10	6 2																																			
6	2	1	5	3	4	3	5	3	3	5	2	1	2	3	4	6	2	3	6	5	2	6	1	1	1	3	1	3	6	6	3	6	4	3		
5	1	2	6	1	3	1	1	1	2	2	2	6	4	6	3	6	5	2	6	1	1															
5	1	2	6	2	3	4	6	2	3	6	5	2	6	1	1	3	3	4	2	6	6	6	2	1	1	5	5									
5	3	6	5	2	6	1	1	1	2	6	1	3	1	3	5	3	3	5	2	1	2	3	4	6	2											
6	3	5	3	3	5	2	1	3	6	5	2	6	1	1	1	2	6	2	1	5	3	4	2	2	6	4	6	3	3	4	2	6	6	6		
6	2	3	4	6	2	1	1	2	3	3	4	2	6	6	6	1	2	6	3	6	5	2	6	1	1	2	1	3	4	2						
5	1	6	1	2	1	3	4	2	3	3	4	2	6	6	6	3	2	6	5	1	1	6	1	3	1											
5	2	3	4	6	2	3	3	4	2	6	6	6	3	6	5	2	6	1	1	1	2	6	2	2	6	4	6									
6	1	6	1	2	1	1	5	5	3	6	6	3	6	4	3	1	1	2	3	3	4	2	6	6	6	2	2	6	4	6						
6	2	3	4	6	2	3	3	4	2	6	6	6	3	5	3	3	5	2	1	1	6	1	2	2	6	4	6	2	1	3	4	2				

Table A2: Mk02 problem data

10 6

6	6	3	3	4	5	1	3	6	6	2	2	5	3	2	6	5	3	4	6	1	1	5	6	3	3	4	3	2	6	6	5	1	2	6	2	6	3	5	6	3	3	2	2	1	5	4																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
6	5	6	1	5	6	1	3	2	4	4	2	2	6	3	5	6	1	5	2	2	2	4	3	3	3	3	2	2	1	5	4	6	3	3	4	5	1	3	6	6	2	2	5	3																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
6	6	1	1	5	6	3	3	4	3	2	6	6	5	6	5	3	4	6	2	4	6	6	3	6	1	2	3	3	2	2	1	5	4	5	3	5	1	4	2	3	6	3	5	2	6	4																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
	1	1	5	2	4	5	5	3	3	6	3	5	6	3	1	4	4	6	3	6	5	3																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													

Table A3: Mk03 problem data

15	8	3																																									
10	4	7	15	8	11	4	5	5	19	2	3	18	4	5	4	8	18	7	3	6	11	3	16	4	5	7	2	1	7	2	3	19	2	5	6	6	3	3	4				
	5	5	2	8	18	1	5	2	1	1	17	5	5	10	2	10	1	12	8	5	3	14	3	7	15	6	2	8	19														
10	4	8	18	7	3	6	11	3	16	1	1	17	2	2	1	4	13	5	5	10	2	10	1	12	8	5	3	14	5	4	11	1	9	2	18	6	18	3	13				
	2	6	15	7	13	4	7	15	8	11	4	5	5	19	4	5	7	2	1	7	2	3	19	4	4	11	1	7	6	13	8	3	3	7	15	6	2	8	19				
10	2	3	3	5	5	4	5	7	2	1	7	2	3	19	2	3	18	4	5	2	5	6	6	3	4	4	11	1	7	6	13	8	3	3	7	15	6	2	8				
	19	5	4	11	1	9	2	18	6	18	3	13	3	4	5	5	2	8	18	1	1	17	2	2	1	4	13																
10	2	3	18	4	5	2	3	3	5	5	5	4	11	1	9	2	18	6	18	3	13	4	4	11	1	7	6	13	8	3	2	6	15	7	13	4	5	7	2				
	1	7	2	3	19	1	5	2	4	8	18	7	3	6	11	3	16	1	1	17	2	5	6	6	3																		
10	2	6	15	7	13	3	7	15	6	2	8	19	1	5	2	4	7	15	8	11	4	5	5	19	5	4	11	1	9	2	18	6	18	3	13	4	5	7	2				
	1	7	2	3	19	3	4	5	5	2	8	18	2	5	6	6	3	2	3	3	5	5	5	5	10	2	10	1	12	8	5	3	14										
10	2	2	1	4	13	2	6	15	7	13	2	3	18	4	5	4	8	18	7	3	6	11	3	16	5	4	11	1	9	2	18	6	18	3	13	5	5	10	2				
	10	1	12	8	5	3	14	4	4	11	1	7	6	13	8	3	4	7	15	8	11	4	5	5	19	2	5	6	6	3	2	3	3	5	5								
10	5	5	10	2	10	1	12	8	5	3	14	4	4	11	1	7	6	13	8	3	2	2	1	4	13	1	1	17	2	6	15	7	13	4	5	7	2	1	7				
	2	3	19	1	5	2	5	4	11	1	9	2	18	6	18	3	13	2	3	18	4	5	3	7	15	6	2	8	19														
10	3	7	15	6	2	8	19	1	1	17	4	7	15	8	11	4	5	5	19	2	6	15	7	13	5	5	10	2	10	1	12	8	5	3	14	4	4	11	1				
	7	6	13	8	3	5	4	11	1	9	2	18	6	18	3	13	2	2	1	4	13	2	3	18	4	5	2	3	3	5	5												
10	1	1	17	5	5	10	2	10	1	12	8	5	3	14	4	8	18	7	3	6	11	3	16	3	7	15	6	2	8	19	2	6	15	7	13	4	4	11	1				
	7	6	13	8	3	1	5	2	2	2	1	4	13	5	4	11	1	9	2	18	6	18	3	13	4	7	15	8	11	4	5	5	19										
10	1	1	17	2	6	15	7	13	3	4	5	5	2	8	18	5	4	11	1	9	2	18	6	18	3	13	4	4	11	1	7	6	13	8	3	2	3	18	4				
	5	2	5	6	6	3	3	7	15	6	2	8	19	4	8	18	7	3	6	11	3	16	5	5	10	2	10	1	12	8	5	3	14										
10	2	2	1	4	13	3	7	15	6	2	8	19	4	8	18	7	3	6	11	3	16	2	3	18	4	5	2	5	6	6	3	1	1	17	2	3	3	5	5				
	3	4	5	5	2	8	18	5	5	10	2	10	1	12	8	5	3	14	5	4	11	1	9	2	18	6	18	3	13														
10	4	4	11	1	7	6	13	8	3	3	4	5	5	2	8	18	4	8	18	7	3	6	11	3	16	1	1	17	5	4	11	1	9	2	18	6	18	3	13				
	3	7	15	6	2	8	19	1	5	2	2	3	3	5	5	4	7	15	8	11	4	5	5	19	2	2	1	4	13														

Table A3 (Continued)

10	5	5	10	2	10	1	12	8	5	3	14	1	5	2	2	3	18	4	5	4	5	7	2	1	7	2	3	19	2	6	15	7	13	4	8	18	7	3	6
	11	3	16	4	7	15	8	11	4	5	5	19	5	4	11	1	9	2	18	6	18	3	13	2	5	6	6	3	4	4	11	1	7	6	13	8	3		
10	4	8	18	7	3	6	11	3	16	3	4	5	5	2	8	18	2	2	1	4	13	4	5	7	2	1	7	2	3	19	2	5	6	6	3	2	3	18	4
	5	2	6	15	7	13	1	5	2	5	4	11	1	9	2	18	6	18	3	13	1	1	17																
10	5	5	10	2	10	1	12	8	5	3	14	2	5	6	6	3	2	6	15	7	13	4	7	15	8	11	4	5	5	19	4	8	18	7	3	6	11	3	16
	1	1	17	5	4	11	1	9	2	18	6	18	3	13	3	4	5	5	2	8	18	2	3	18	4	5	4	5	7	2	1	7	2	3	19				

Table A4: Mk04 problem data

15	8	2																																																		
8	1	1	6	2	1	6	7	9	2	6	7	3	1	2	4	2	7	5	3	1	8	3	9	8	9	3	2	3	4	8	3	2	2	5	5	6	7	2	6	1	4	7										
7	1	6	1	2	6	1	4	7	1	1	6	2	6	7	3	1	3	2	3	4	8	3	2	1	6	2	1	7	2																							
6	1	6	1	3	2	3	4	8	3	2	3	3	2	7	1	4	4	2	4	2	7	5	2	1	7	3	7	2	4	4	3	1																				
5	1	7	2	1	1	6	2	1	6	7	9	2	6	7	3	1	2	4	5	5	7																															
7	1	7	2	2	1	6	7	9	2	4	4	3	1	3	1	8	3	9	8	9	2	1	7	3	7	3	2	3	4	8	3	2	2	4	5	5	7															
9	1	6	2	2	4	4	3	1	3	3	2	7	1	4	4	2	6	1	4	7	2	4	5	5	7	3	1	8	3	9	8	9	2	1	7	3	7	1	6	1	2	1	6	7	9							
5	2	5	5	6	7	2	1	7	3	7	2	6	1	4	7	1	6	2	2	6	7	3	1																													
6	2	4	5	5	7	2	5	5	6	7	3	2	3	4	8	3	2	1	6	2	1	6	1	2	1	6	7	9																								
9	1	1	6	2	1	6	7	9	2	4	4	3	1	3	1	8	3	9	8	9	2	4	2	7	5	2	6	1	4	7	1	7	2	2	1	7	3	7	3	2	3	4	8	3	2							
5	2	5	5	6	7	1	1	6	1	7	2	2	4	5	5	7	2	1	6	7	9																															
4	3	1	8	3	9	8	9	1	1	6	3	2	3	4	8	3	2	2	4	2	7	5																														
6	2	4	2	7	5	1	6	1	1	1	6	2	1	7	3	7	3	1	8	3	9	8	9	1	7	2																										
4	1	6	2	2	6	7	3	1	2	6	1	4	7	2	5	5	6	7																																		
3	2	5	5	6	7	1	6	1	2	4	2	7	5																																							
6	2	4	5	5	7	1	7	2	3	1	8	3	9	8	9	3	2	3	4	8	3	2	3	3	2	7	1	4	4	1	1	6																				

Table A5: Mk05 problem data

15 4 1,5

6	2	3	5	2	7	2	1	8	4	8	2	1	6	2	5	1	3	7	2	4	5	2	6	2	4	5	1	5											
5	1	3	7	2	1	6	2	5	1	4	6	2	4	5	2	6	2	1	8	2	6																		
8	2	4	7	3	9	2	3	5	2	7	2	4	5	1	5	2	1	8	4	8	2	1	6	2	5	1	4	6	2	1	8	2	6	2	4	9	3	6	
7	2	4	5	1	5	2	4	7	3	9	2	1	8	4	8	1	4	8	2	1	8	2	6	2	4	5	2	6	1	4	6								
6	2	3	7	1	5	2	4	6	2	7	2	4	7	3	9	1	3	8	2	3	5	2	7	2	1	8	2	6											
9	1	4	6	2	4	5	2	6	1	3	8	2	3	7	1	5	2	4	6	2	7	1	4	8	2	1	8	2	6	2	1	8	4	8	2	4	5	1	5
5	1	3	8	2	4	7	3	9	2	1	6	2	5	2	4	6	2	7	1	3	7																		
8	2	3	7	1	5	1	3	8	2	4	7	3	9	2	4	5	1	5	1	3	7	1	4	8	2	4	9	3	6	2	1	6	2	5					
9	2	3	5	2	7	1	4	8	2	4	5	2	6	2	1	6	2	5	1	4	6	2	1	8	4	9	2	1	8	4	8	2	1	8	2	6	1	3	7
9	2	1	8	2	6	2	1	8	4	8	2	1	8	4	9	2	4	9	3	6	2	1	6	2	5	1	3	8	1	3	7	1	4	6	2	4	5	2	6
7	2	1	8	2	6	2	1	8	4	8	2	1	6	2	5	1	3	7	1	4	6	1	3	8	2	4	9	3	6										
6	1	4	8	1	3	7	2	4	7	3	9	2	1	6	2	5	1	3	8	2	1	8	4	8															
7	1	4	8	2	4	9	3	6	2	1	8	4	8	2	4	6	2	7	2	4	6	2	7	2	1	8	2	6	2	3	7	1	5						
7	2	1	6	2	5	2	3	7	1	5	2	1	8	4	8	2	1	8	2	6	2	4	5	1	5	2	4	6	2	7	1	4	6						
7	1	3	8	2	1	8	4	9	2	4	9	3	6	1	3	7	2	4	5	2	6	2	1	8	2	6	2	1	6	2	5								

Table A6: Mk06 problem data

10 15

15	4	2	8	6	3	7	2	9	5	2	9	7	1	2	5	7	4	1	4	9	1	2	7	10	4	2	1	1	8	2	3	7	5	3	8	5	8	5	1	3
	8	8	2	5	3	8	10	9	3	5	6	1	1	6	2	5	2	5	1	9	9	1	5	7	4	6	2	10	6	1	2	2	7	9	5	6	2	4	8	7
	2	5	2	1	5	8	4	2	1	8	3	7	3	10	2	8	9	4	5	3	7	5	3	7	9	3	3	9	4	5	8	1	1							
15	5	1	3	8	8	2	5	3	8	10	9	5	7	4	1	4	9	1	2	7	10	4	3	5	6	1	1	6	2	5	2	1	5	8	4	2	1	8	3	7
	2	4	8	7	2	2	10	6	1	2	3	10	2	8	9	4	5	2	7	9	5	6	3	7	5	3	7	9	3	3	7	5	3	8	5	8	3	9	4	5
	8	1	1	2	9	7	1	2	2	1	1	8	2	4	2	8	6	3	7	2	9	5	5	2	5	1	9	9	1	5	7	4	6							
15	2	1	1	8	2	2	7	9	5	6	2	10	6	1	2	2	4	8	7	2	5	2	1	5	8	4	2	1	8	3	7	3	9	4	5	8	1	1	2	9
	7	1	2	3	7	5	3	7	9	3	5	7	4	1	4	9	1	2	7	10	4	4	2	8	6	3	7	2	9	5	5	1	3	8	8	2	5	3	8	10
	9	3	10	2	8	9	4	5	5	2	5	1	9	9	1	5	7	4	6	3	5	6	1	1	6	2	3	7	5	3	8	5	8							
15	3	5	6	1	1	6	2	5	2	5	1	9	9	1	5	7	4	6	5	1	3	8	8	2	5	3	8	10	9	5	2	1	5	8	4	2	1	8	3	7
	2	4	8	7	2	2	10	6	1	2	3	7	5	3	8	5	8	2	9	7	1	2	3	7	5	3	7	9	3	3	9	4	5	8	1	1	4	2	8	6
	3	7	2	9	5	2	1	1	8	2	5	7	4	1	4	9	1	2	7	10	4	2	7	9	5	6	3	10	2	8	9	4	5							
15	3	10	2	8	9	4	5	2	1	1	8	2	3	9	4	5	8	1	1	2	9	7	1	2	3	7	5	3	8	5	8	5	2	1	5	8	4	2	1	8
	3	7	3	5	6	1	1	6	2	3	7	5	3	7	9	3	4	2	8	6	3	7	2	9	5	2	10	6	1	2	5	7	4	1	4	9	1	2	7	10
	4	2	7	9	5	6	5	2	5	1	9	9	1	5	7	4	6	5	1	3	8	8	2	5	3	8	10	9	2	4	8	7	2							
15	3	7	5	3	8	5	8	5	1	3	8	8	2	5	3	8	10	9	2	7	9	5	6	3	5	6	1	1	6	2	5	2	5	1	9	9	1	5	7	4
	6	2	4	8	7	2	2	9	7	1	2	5	2	1	5	8	4	2	1	8	3	7	5	7	4	1	4	9	1	2	7	10	4	4	2	8	6	3	7	2
	9	5	2	1	1	8	2	3	7	5	3	7	9	3	2	10	6	1	2	3	9	4	5	8	1	1	3	10	2	8	9	4	5							
15	3	5	6	1	1	6	2	3	10	2	8	9	4	5	3	7	5	3	8	5	8	5	1	3	8	8	2	5	3	8	10	9	2	1	1	8	2	2	9	7
	1	2	5	2	1	5	8	4	2	1	8	3	7	3	7	5	3	7	9	3	5	7	4	1	4	9	1	2	7	10	4	3	9	4	5	8	1	1	2	10
	6	1	2	4	2	8	6	3	7	2	9	5	2	7	9	5	6	2	4	8	7	2	5	2	5	1	9	9	1	5	7	4	6							
15	5	7	4	1	4	9	1	2	7	10	4	3	7	5	3	7	9	3	3	7	5	3	8	5	8	2	1	1	8	2	3	5	6	1	1	6	2	5	2	5
	1	9	9	1	5	7	4	6	3	10	2	8	9	4	5	3	9	4	5	8	1	1	2	9	7	1	2	4	2	8	6	3	7	2	9	5	5	1	3	8
	8	2	5	3	8	10	9	2	4	8	7	2	2	10	6	1	2	5	2	1	5	8	4	2	1	8	3	7	2	7	9	5	6							

Table A6 (Continued)

15	4	2	8	6	3	7	2	9	5	3	9	4	5	8	1	1	3	7	5	3	8	5	8	5	7	4	1	4	9	1	2	7	10	4	5	2	1	5	8	4
	2	1	8	3	7	2	4	8	7	2	2	9	7	1	2	3	10	2	8	9	4	5	5	1	3	8	8	2	5	3	8	10	9	2	10	6	1	2	5	2
	5	1	9	9	1	5	7	4	6	3	7	5	3	7	9	3	2	7	9	5	6	2	1	1	8	2	3	5	6	1	1	6	2							
15	2	1	1	8	2	4	2	8	6	3	7	2	9	5	3	10	2	8	9	4	5	3	7	5	3	8	5	8	3	7	5	3	7	9	3	2	10	6	1	2
	2	7	9	5	6	3	9	4	5	8	1	1	5	7	4	1	4	9	1	2	7	10	4	5	2	5	1	9	9	1	5	7	4	6	5	1	3	8	8	2
	5	3	8	10	9	3	5	6	1	1	6	2	5	2	1	5	8	4	2	1	8	3	7	2	4	8	7	2	2	9	7	1	2							

Table A7: Mk07 problem data

20	5	3																																									
5	2	2	4	1	15	2	3	18	1	15	1	2	4	1	4	18	5	3	8	5	2	4	5	1	7	2	7																
5	2	1	3	5	13	5	3	8	5	2	4	5	1	7	2	7	2	2	4	1	15	3	1	8	5	1	2	5	3	1	3	5	13	3	2								
5	5	2	18	5	1	4	19	1	9	3	3	1	4	18	2	4	11	3	9	1	2	4	3	5	12	3	14	4	19														
5	2	2	4	1	15	4	4	10	3	10	2	17	5	8	4	5	18	3	13	2	2	1	5	5	4	10	5	15	1	2	3	9	2	16	2	3	15	1	6				
5	3	1	3	5	13	3	2	2	3	18	1	15	5	2	18	5	1	4	19	1	9	3	3	3	5	12	3	14	4	19	1	4	5										
5	5	3	8	5	2	4	5	1	7	2	7	2	3	18	1	15	2	1	15	5	7	2	2	7	1	17	2	2	4	1	15												
5	1	4	5	2	1	15	5	7	2	2	4	1	15	3	1	3	5	13	3	2	4	4	6	2	17	3	15	5	7														
5	4	4	6	2	17	3	15	5	7	3	3	18	1	2	4	15	4	2	14	4	14	3	19	5	15	1	2	4	2	2	7	1	17										
5	5	2	18	5	1	4	19	1	9	3	3	4	4	6	2	17	3	15	5	7	3	1	8	5	1	2	5	4	2	14	4	14	3	19	5	15	2	1	17	5	15		
5	2	1	15	5	7	4	4	10	3	10	2	17	5	8	2	3	15	1	6	1	4	5	5	3	16	5	17	4	10	2	10	1	7										
5	1	4	18	3	1	8	5	1	2	5	5	3	8	5	2	4	5	1	7	2	7	2	1	15	5	7	2	1	17	5	15												
5	3	5	12	3	14	4	19	4	4	10	3	10	2	17	5	8	2	3	15	1	6	5	3	8	5	2	4	5	1	7	2	7	5	3	16	5	17	4	10	2	10	1	7
5	2	1	17	5	15	1	4	18	4	2	17	5	19	4	5	3	12	3	3	18	1	2	4	15	3	1	8	5	1	2	5												
5	2	5	1	3	5	3	3	18	1	2	4	15	4	4	10	3	10	2	17	5	8	2	3	18	1	15	5	3	8	5	2	4	5	1	7	2	7						
5	5	3	8	5	2	4	5	1	7	2	7	2	5	1	3	5	3	5	12	3	14	4	19	5	3	16	5	17	4	10	2	10	1	7	2	1	17	5	15				
5	5	4	10	5	15	1	2	3	9	2	16	2	4	11	3	9	1	2	4	2	1	15	5	7	1	4	5																
5	5	3	8	5	2	4	5	1	7	2	7	4	2	14	4	14	3	19	5	15	3	3	18	1	2	4	15	2	3	15	1	6	5	2	18	5	1	4	19	1	9	3	3
5	1	2	4	3	1	8	5	1	2	5	2	5	1	3	5	2	3	18	1	15	2	1	15	5	7																		
5	3	1	3	5	13	3	2	4	4	6	2	17	3	15	5	7	4	5	18	3	13	2	2	1	5	1	4	18	2	1	3	5	13										
5	1	4	5	2	2	4	1	15	1	4	18	2	1	15	5	7	5	4	10	5	15	1	2	3	9	2	16																

Table A8: Mk08 problem data

20	10																																											
10	2	7	18	4	5	2	5	7	7	7	7	1	3	19	1	7	14	2	4	5	10	12	1	1	10	1	10	18	2	7	10	8	19	2	3	11	8	9	2	3	5	8	12	
12	1	2	5	2	7	18	4	5	2	3	5	8	12	1	1	10	1	10	19	2	3	15	4	19	1	7	14	1	5	9	2	5	14	9	5	1	1	19	2	7	10	8	19	
	1	1	16																																									
14	2	5	14	9	5	1	1	19	1	1	10	1	3	19	2	7	18	4	5	2	4	5	10	12	2	3	5	8	12	1	10	10	1	5	9	1	1	7	2	7	10	8		
	19	1	1	10	1	10	19	1	10	18																																		
10	1	10	10	2	5	7	7	7	1	7	14	1	1	10	1	10	18	2	3	15	7	13	2	10	14	5	7	2	3	11	8	9	1	9	11	1	5	9						
12	1	5	9	2	5	14	9	5	2	7	18	4	5	2	3	11	8	9	1	1	10	1	9	11	1	1	7	1	7	14	2	4	5	10	12	2	3	15	4	19	1	8		
	18	1	10	19																																								
10	2	3	15	7	13	1	3	19	1	5	9	1	10	19	2	3	5	8	12	2	7	18	4	5	2	8	14	10	9	2	4	5	10	12	1	10	18	1	1	7				
12	1	1	10	1	10	18	1	1	7	1	5	9	2	8	14	10	9	2	7	10	8	19	2	3	15	4	19	2	10	14	5	7	1	8	18	1	10	19	1	1	19	1		
	1	10																																										
11	1	1	10	1	7	14	1	1	10	2	3	15	4	19	2	5	14	9	5	2	7	18	4	5	1	3	19	1	1	19	2	4	5	10	12	1	5	9	1	10	19			
14	2	7	10	8	19	2	8	14	10	9	1	1	19	1	10	19	2	10	14	5	7	1	2	5	2	4	5	10	12	2	5	7	7	7	1	1	16	1	1	7	1	9	11	
	1	3	19	1	1	10	1	10	18																																			
11	1	10	19	2	10	14	5	7	1	8	18	2	3	11	8	9	1	1	7	1	1	10	2	5	14	9	5	2	3	15	4	19	1	10	18	1	3	19	1	1	19			
11	2	5	14	9	5	1	1	10	1	8	18	2	3	15	4	19	2	7	10	8	19	2	3	5	8	12	2	3	11	8	9	2	8	14	10	9	1	10	10	1	9	11	1	
	3	19																																										
10	1	10	19	2	3	11	8	9	2	5	7	7	7	1	1	16	1	7	14	2	7	18	4	5	2	4	5	10	12	1	1	10	1	8	18	2	5	14	9	5				
11	2	10	14	5	7	1	10	19	2	7	10	8	19	2	3	15	4	19	1	1	19	1	8	18	2	8	14	10	9	2	3	11	8	9	1	10	18	2	5	14	9	5		
	1	2	5																																									
11	1	1	10	2	5	7	7	7	1	1	10	1	9	11	1	7	14	2	3	15	7	13	2	8	14	10	9	1	1	16	2	3	5	8	12	2	5	14	9	5	1	2	5	
11	2	5	14	9	5	2	5	7	7	7	1	7	14	1	10	10	2	7	10	8	19	2	3	15	4	19	2	7	18	4	5	1	1	7	2	3	11	8	9	1	1	19		
	1	8	18																																									
11	1	2	5	2	7	10	8	19	1	10	10	1	9	11	1	8	18	2	10	14	5	7	2	5	14	9	5	1	1	10	1	1	19	2	3	15	7	13	2	8	14	10	9	

Table A8 (Continued)

13	1	10	10	2	5	14	9	5	1	5	9	1	10	19	1	1	10	2	3	5	8	12	1	2	5	2	10	14	5	7	1	1	10	2	8	14	10	9	2	3	15	7	13	
	1	1	16	1	7	14																																						
11	2	3	15	7	13	1	2	5	1	10	19	1	3	19	1	8	18	1	1	7	1	5	9	1	7	14	2	7	18	4	5	1	1	10	2	5	14	9	5					
10	2	7	10	8	19	1	2	5	2	3	11	8	9	1	9	11	2	4	5	10	12	1	10	18	2	7	18	4	5	2	8	14	10	9	2	3	5	8	12	1	10	19		
10	1	10	18	1	10	10	1	7	14	1	9	11	2	3	15	7	13	1	2	5	2	8	14	10	9	2	3	5	8	12	1	5	9	1	1	16								

Table A9: Mk09 problem data

20 10

12	2	2	10	1	11	1	8	17	1	8	14	1	1	10	2	2	16	10	18	2	9	6	2	12	4	7	9	4	11	3	10	1	16	2	5	19	1	7	1	9	11	1	4	
	16	1	2	5	5	7	9	9	9	4	6	8	14	6	16																													
13	1	8	17	2	5	6	4	11	2	2	10	1	11	2	5	9	8	8	2	2	16	3	11	4	1	8	5	14	10	15	6	12	4	6	10	8	15	7	5	2	8	2	5	
	19	1	7	4	7	9	4	11	3	10	1	16	1	1	10	4	1	16	3	11	7	17	4	7	1	4	16	4	3	11	5	8	7	11	9	17								
11	4	6	10	8	15	7	5	2	8	2	5	9	8	8	2	2	16	10	18	2	2	10	1	11	5	7	9	9	9	4	6	8	14	6	16	1	4	16	2	5	19	1	7	
	1	1	10	2	5	6	4	11	2	2	16	3	11	1	3	14																												
11	4	1	8	5	14	10	15	6	12	2	5	19	1	7	4	4	11	8	16	9	15	1	6	1	8	14	1	4	16	1	8	17	4	1	16	3	11	7	17	4	7	4	10	
	6	8	13	5	5	2	8	1	3	14	4	7	9	4	11	3	10	1	16	1	1	10																						
14	1	8	17	1	4	16	1	5	9	4	10	6	8	13	5	5	2	8	4	1	16	3	11	7	17	4	7	2	2	16	10	18	4	6	10	8	15	7	5	2	8	1	8	
	14	2	5	6	4	11	4	2	5	7	13	10	10	5	11	5	7	9	9	9	4	6	8	14	6	16	2	5	9	8	8	4	1	8	5	14	10	15	6	12	2	5	19	
	1	7																																										
11	4	2	5	7	13	10	10	5	11	2	2	16	10	18	1	1	10	1	3	14	1	5	9	5	7	9	9	9	4	6	8	14	6	16	1	8	17	1	8	14	1	2	5	
	4	6	10	8	15	7	5	2	8	4	4	11	8	16	9	15	1	6																										
14	1	8	14	1	8	17	2	5	9	8	8	1	4	16	1	1	10	4	2	5	7	13	10	10	5	11	1	2	5	2	5	6	4	11	5	7	9	9	9	4	6	8	14	
	6	16	4	4	11	8	16	9	15	1	6	5	2	8	1	19	8	13	6	14	10	18	4	6	10	8	15	7	5	2	8	4	1	16	3	11	7	17	4	7	2	2	16	
	10	18																																										
13	1	1	10	4	10	6	8	13	5	5	2	8	1	5	9	4	7	9	4	11	3	10	1	16	1	9	11	4	2	5	7	13	10	10	5	11	4	6	10	8	15	7	5	
	2	8	1	2	5	5	2	8	1	19	8	13	6	14	10	18	5	7	9	9	9	4	6	8	14	6	16	2	2	10	1	11	4	1	16	3	11	7	17	4	7	2	5	
	6	4	11																																									
11	1	8	17	1	2	5	1	1	10	1	4	16	2	5	6	4	11	4	7	9	4	11	3	10	1	16	5	2	8	1	19	8	13	6	14	10	18	1	9	11	2	9	6	
	2	12	2	2	10	1	11	2	5	9	8	8																																
12	1	4	16	4	4	11	8	16	9	15	1	6	1	3	14	4	2	5	7	13	10	10	5	11	1	9	11	5	7	9	9	9	4	6	8	14	6	16	2	5	6	4	11	
	4	1	16	3	11	7	17	4	7	2	2	10	1	11	2	2	16	3	11	4	1	8	5	14	10	15	6	12	1	1	10													
10	1	9	11	1	5	9	5	2	8	1	19	8	13	6	14	10	18	1	4	16	4	4	11	8	16	9	15	1	6	2	5	9	8	8	4	7	9	4	11	3	10	1	16	

Table A9 (Continued)

	1	3	14	1	1	10	4	1	16	3	11	7	17	4	7																																											
11	4	10	6	8	13	5	5	2	8	4	4	11	8	16	9	15	1	6	1	4	16	2	9	6	2	12	4	6	10	8	15	7	5	2	8	4	7	9	4	11	3	10	1	16	1													
	16	1	2	5	1	8	14	5	7	9	9	9	4	6	8	14	6	16	2	5	6	4	11	2	2	16	10	18																														
11	1	2	5	1	3	14	2	9	6	2	12	1	5	9	4	2	5	7	13	10	10	5	11	4	1	16	3	11	7	17	4	7	2	2	10	1	11	1	8	17	2	5	19	16	1													
	1	7	1	1	10	4	7	9	4	11	3	10	1	16																																												
10	4	3	11	5	8	7	11	9	17	1	1	10	2	2	16	10	18	2	2	10	1	11	4	6	10	8	15	7	5	2	8	4	4	11	8	16	9	15	1	6	1	4	16	1														
	4	1	16	3	11	7	17	4	7	4	7	9	4	11	3	10	1	16	2	2	16	3	11																																			
12	1	1	10	4	4	11	8	16	9	15	1	6	4	2	5	7	13	10	10	5	11	5	2	8	1	19	8	13	6	14	10	18	2	5	6	4	11	2	9	6	2	12	1	16	1													
	2	5	4	10	6	8	13	5	5	2	8	1	4	16	2	2	16	3	11	2	2	10	1	11	4	6	10	8	15	7	5	2	8																									
14	1	8	17	4	4	11	8	16	9	15	1	6	1	3	14	2	9	6	2	12	1	8	14	4	6	10	8	15	7	5	2	8	4	7	9	4	11	3	10	1	16	4	2	16	1													
	5	7	13	10	10	5	11	4	1	8	5	14	10	15	6	12	2	2	10	1	11	1	4	16	4	3	11	5	8	7	11	9	17	2	5	19	1	7	4	10	6	8	13	16	1													
	5	5	2	8																																																						
13	5	2	8	1	19	8	13	6	14	10	18	1	9	11	4	7	9	4	11	3	10	1	16	1	8	17	4	10	6	8	13	5	5	2	8	2	5	6	4	11	1	1	10	16	1													
	4	6	10	8	15	7	5	2	8	2	2	10	1	11	2	2	16	10	18	4	1	16	3	11	7	17	4	7	1	3	14	2	5	19	1	7																						
11	5	2	8	1	19	8	13	6	14	10	18	5	7	9	9	9	4	6	8	14	6	16	2	5	6	4	11	4	10	6	8	13	5	5	2	8	1	3	14	4	3	11	5	16	1													
	8	7	11	9	17	1	9	11	2	2	10	1	11	4	2	5	7	13	10	10	5	11	1	8	14	4	1	8	5	14	10	15	6	12																								
13	1	3	14	2	2	10	1	11	4	7	9	4	11	3	10	1	16	2	2	16	10	18	2	2	16	3	11	4	4	11	8	16	9	15	1	6	4	1	16	3	11	7	17	16	1													
	4	7	4	2	5	7	13	10	10	5	11	4	10	6	8	13	5	5	2	8	2	5	9	8	8	1	2	5	4	6	10	8	15	7	5	2	8	1	5	9																		
13	4	1	16	3	11	7	17	4	7	4	2	5	7	13	10	10	5	11	4	6	10	8	15	7	5	2	8	1	3	14	2	5	6	4	11	4	4	11	8	16	9	15	1	16	1													
	6	1	5	9	1	1	10	1	8	17	2	9	6	2	12	5	2	8	1	19	8	13	6	14	10	18	2	2	16	3	11	2	2	16	10	18																						

Table A10: Mk10 problem data

20	15																																									
12	2	6	5	2	5	2	7	11	6	11	1	2	5	4	8	10	3	18	4	10	9	7	2	7	9	1	7	4	1	8	7	14	9	12	4	7	3	4	13	8		
	8	2	6	5	3	8	1	19	9	13	10	19	2	16	5	2	16	10	9	3	12	4	11	5	15	2	9	10	10	5	3	7	5	2	8	4	7	4	1	6		
	6	13	5	11	10	7																																				
13	2	7	11	6	11	4	2	16	10	9	5	9	8	16	2	6	5	2	5	2	2	11	1	9	2	3	12	7	15	4	4	11	10	14	5	10	7	15	4	3		
	8	1	12	5	5	13	11	5	3	8	1	19	9	13	10	19	2	16	3	4	13	8	8	2	6	4	8	10	3	18	4	10	9	7	4	1	16	5	11	10		
	17	3	6	2	9	10	10	5	2	5	11	2	11																													
11	4	3	8	1	12	5	5	13	11	2	2	11	1	9	2	7	9	1	7	2	6	5	2	5	4	1	6	6	13	5	11	10	7	2	9	10	10	5	5	3		
	8	1	19	9	13	10	19	2	16	4	8	10	3	18	4	10	9	7	4	2	16	10	9	5	9	8	16	2	3	12	7	15	2	2	5	9	19					
11	4	4	11	10	14	5	10	7	15	5	3	8	1	19	9	13	10	19	2	16	1	5	15	1	2	5	2	9	10	10	5	2	7	11	6	11	4	1	16	5		
	11	10	17	3	6	2	10	13	6	11	2	2	5	9	19	3	4	13	8	8	2	6	4	8	10	3	18	4	10	9	7											
14	2	7	11	6	11	2	9	10	10	5	4	5	11	7	8	10	11	2	16	2	10	13	6	11	4	1	16	5	11	10	17	3	6	2	7	9	1	7	4	3		
	8	1	12	5	5	13	11	1	2	5	4	2	16	10	9	5	9	8	16	3	1	15	2	19	9	9	4	1	6	6	13	5	11	10	7	2	2	11	1	9		
	4	4	11	10	14	5	10	7	15	5	3	8	1	19	9	13	10	19	2	16																						
11	3	1	15	2	19	9	9	2	7	9	1	7	4	8	10	3	18	4	10	9	7	2	2	5	9	19	4	5	11	7	8	10	11	2	16	4	1	6	6	13		
	5	11	10	7	2	7	11	6	11	1	2	5	3	7	5	2	8	4	7	4	3	8	1	12	5	5	13	11	1	5	15											
14	1	2	5	2	7	11	6	11	2	2	11	1	9	2	9	10	10	5	4	8	10	3	18	4	10	9	7	3	1	15	2	19	9	9	3	7	5	2	8	4		
	7	4	2	16	10	9	5	9	8	16	4	1	6	6	13	5	11	10	7	1	5	15	4	7	13	10	19	6	18	4	8	4	3	8	1	12	5	5	13	11		
	4	1	16	5	11	10	17	3	6	2	7	9	1	7																												
13	4	8	10	3	18	4	10	9	7	2	10	13	6	11	4	5	11	7	8	10	11	2	16	3	4	13	8	8	2	6	5	2	16	10	9	3	12	4	11	5		
	15	3	1	15	2	19	9	9	4	3	8	1	12	5	5	13	11	3	7	5	2	8	4	7	4	7	13	10	19	6	18	4	8	4	1	6	6	13	5	11		
	10	7	2	6	5	2	5	4	1	16	5	11	10	17	3	6	4	2	16	10	9	5	9	8	16																	
11	2	7	11	6	11	3	7	5	2	8	4	7	4	8	10	3	18	4	10	9	7	2	9	10	10	5	4	2	16	10	9	5	9	8	16	3	4	13	8	8		
	2	6	4	7	13	10	19	6	18	4	8	5	2	16	10	9	3	12	4	11	5	15	4	1	8	7	14	9	12	4	7	2	6	5	2	5	2	2	11	1		
	9																																									

Table A10 (Continued)

12	2	9	10	10	5	1	5	15	2	2	5	9	19	3	1	15	2	19	9	9	5	2	16	10	9	3	12	4	11	5	15	4	1	6	6	13	5	11	10	7
	4	2	16	10	9	5	9	8	16	4	1	16	5	11	10	17	3	6	2	6	5	2	5	2	3	12	7	15	4	4	11	10	14	5	10	7	15	4	8	10
	3	18	4	10	9	7																																		
10	5	2	16	10	9	3	12	4	11	5	15	4	5	11	7	8	10	11	2	16	4	7	13	10	19	6	18	4	8	2	9	10	10	5	1	5	15	2	2	11
	1	9	3	4	13	8	8	2	6	2	2	5	9	19	4	8	10	3	18	4	10	9	7	4	1	16	5	11	10	17	3	6								
11	2	10	13	6	11	1	5	15	2	9	10	10	5	4	1	8	7	14	9	12	4	7	4	3	8	1	12	5	5	13	11	3	4	13	8	8	2	6	3	7
	5	2	8	4	7	1	2	5	4	1	6	6	13	5	11	10	7	4	2	16	10	9	5	9	8	16	2	7	9	1	7									
11	3	7	5	2	8	4	7	2	2	5	9	19	4	1	8	7	14	9	12	4	7	4	5	11	7	8	10	11	2	16	3	1	15	2	19	9	9	4	1	16
	5	11	10	17	3	6	2	6	5	2	5	2	7	11	6	11	5	3	8	1	19	9	13	10	19	2	16	4	8	10	3	18	4	10	9	7	3	4	13	8
	8	2	6																																					
10	2	5	11	2	11	4	8	10	3	18	4	10	9	7	2	7	9	1	7	2	6	5	2	5	4	3	8	1	12	5	5	13	11	1	5	15	2	9	10	10
	5	4	1	16	5	11	10	17	3	6	3	4	13	8	8	2	6	2	3	12	7	15																		
12	4	8	10	3	18	4	10	9	7	1	5	15	3	1	15	2	19	9	9	4	7	13	10	19	6	18	4	8	4	2	16	10	9	5	9	8	16	4	1	8
	7	14	9	12	4	7	3	7	5	2	8	4	7	2	10	13	6	11	2	9	10	10	5	2	3	12	7	15	2	6	5	2	5	4	3	8	1	12	5	5
	13	11																																						
14	2	7	11	6	11	1	5	15	2	2	5	9	19	4	1	8	7	14	9	12	4	7	1	2	5	4	3	8	1	12	5	5	13	11	3	4	13	8	8	2
	6	3	1	15	2	19	9	9	4	4	11	10	14	5	10	7	15	2	6	5	2	5	2	9	10	10	5	2	5	11	2	11	5	3	8	1	19	9	13	10
	19	2	16	2	10	13	6	11																																
13	4	7	13	10	19	6	18	4	8	5	2	16	10	9	3	12	4	11	5	15	3	4	13	8	8	2	6	2	7	11	6	11	2	10	13	6	11	4	2	16
	10	9	5	9	8	16	4	8	10	3	18	4	10	9	7	4	3	8	1	12	5	5	13	11	2	6	5	2	5	2	7	9	1	7	4	1	16	5	11	10
	17	3	6	2	2	5	9	19	5	3	8	1	19	9	13	10	19	2	16																					
11	4	7	13	10	19	6	18	4	8	4	1	6	6	13	5	11	10	7	4	2	16	10	9	5	9	8	16	2	10	13	6	11	2	2	5	9	19	2	5	11
	2	11	5	2	16	10	9	3	12	4	11	5	15	2	6	5	2	5	3	1	15	2	19	9	9	1	2	5	4	4	11	10	14	5	10	7	15			
13	2	2	5	9	19	2	6	5	2	5	3	4	13	8	8	2	6	2	7	9	1	7	2	3	12	7	15	1	5	15	4	1	16	5	11	10	17	3	6	3
	1	15	2	19	9	9	2	10	13	6	11	2	2	11	1	9	3	7	5	2	8	4	7	4	3	8	1	12	5	5	13	11	4	5	11	7	8	10	11	2

Table A10 (Continued)

16																																								
13	4	1	16	5	11	10	17	3	6	3	1	15	2	19	9	9	4	3	8	1	12	5	5	13	11	2	2	5	9	19	4	2	16	10	9	5	9	8	16	1
	5	15	4	5	11	7	8	10	11	2	16	4	8	10	3	18	4	10	9	7	2	7	11	6	11	4	1	8	7	14	9	12	4	7	4	7	13	10	19	6
	18	4	8	2	3	12	7	15	2	7	9	1	7																											

APPENDIX B

Figure B1. The pseudo code of the proposed GA

Procedure: Genetic Algorithm

Step 1. Initialize the GA parameters: Stopping criterion: (number of generations) g_{max} ; Population size: P_s ; Chromosome length: C_l ; Mutation probability: p_m

Step 2. for $k=1$ to P_s

2.1.for $j=1$ to N

 // Select a plan for part j and Set into chromosome

 end

2.2. for $i=1$ to total number of operation

 // Select a machine for operation O_{ij} and Set into chromosome via operation sequence

 end

end

Step 3. $g \leftarrow 1$

3.1. // Select parents using Tournament approach from population

Step 4. Crossover:

4.1. // Generate a random number

4.2. // Select two parent from mating pool for crossover via random number

4.3. // Generate new individual via crossover operator

Step 5. Mutation:

5.1. if $p < p_m$ then

5.2. // Mutate the gene

Step 6. // Get new population

Step 7. // Get objective function value for each individual

Step 8. $g \leftarrow g+1$

end

Figure B2. Pseudo Code of FJSSP Simulation Model

```
BEGIN
CREATE      Jobs every 130 minutes exponentially
ASSIGN      myJobType=DISC(0.26,1,0.74,2,1,3)
            myJobSequence=PartSequences(myJobType)
STATION     Order Release
ROUTE Jobs to the imminent station of myJobSequence after
vTransferTime(Order Release, myJobSequence)
//      vTransferTime(Order Release, myJobSequence) is
predetermined.
A: PROCESS delay jobs by "ProcessTime" minutes
    //Seize appropriate machine according to the
myJobSequence
    //Delay "ProcessTime"
    //Release the machine
DECIDE      if myJobSequence completed
            If false, GoTo A
STATION "Exit System"
Collect Statistics
Send Related Statistics to the Database
DISPOSE
END
```
