Development of a Multi-Objective Scheduling System for Complex Job Shops in a Manufacturing Environment

---

A thesis submitted for the degree of

Doctor of Philosophy

By

Abdalla Omar Dagroum Ali

Department of Mechanical and Construction

Engineering

Northumbria University

Newcastle upon Tyne

June 2016

DECLARATION OF ORIGINALITY

Title of Thesis: "*Development of a Multi-Objective Scheduling System for Complex Job Shops in a Manufacturing Environment*"

I declare that the thesis hereby submitted for the degree of Doctor of Philosophy at the University of Northumbria is my own work except as cited in the references and has not been previously submitted for any degree.

Name: Abdalla Ali

Signature:

Date:

Abstract

In many sectors of commercial operation, the scheduling of workflows and the allocation of resources at an optimum time is critical; for effective and efficient operation. The high degree of complexity of a “Job Shop” manufacturing environment, with sequencing of many parallel orders, and allocation of resources within multi-objective operational criteria, has been subject to several research studies. In this thesis, a scheduling system for optimizing multi-objective job shop scheduling problems was developed in order to satisfy different production system requirements. The developed system incorporated three different factors; setup times, alternative machines and release dates, into one model. These three factors were considered after a survey study of multi-objective job shop scheduling problems.

In order to solve the multi-objective job shop scheduling problems, a combination of genetic algorithm and a modified version of a very recent and computationally efficient approach to non-dominated sorting solutions, called “efficient non-dominated sort using the backward pass sequential strategy”, was applied. In the proposed genetic algorithm, an operation based representation was designed in the matrix form, which can preserve features of the parent after the crossover operator without repairing the solution. The proposed efficient non-dominated sort using the backward pass sequential strategy was employed to determine the front, to which each solution belongs. The proposed system was tested and validated with 20 benchmark problems after they have been modified. The experimental results show that the proposed system was effective and efficient to solve multi-objective job shop scheduling problems in terms of solution quality.

Acknowledgements

First of all, I thank Allah for giving me strength and ability to complete this study. I would also like to thank all the people who contributed in some way to the work described in this thesis. I am sincerely grateful to my first supervisor Dr. Phil Hackney for the continuous support of my Ph.D. study, for his patience and motivation. His guidance helped me in the time of research and writing of this thesis. Besides my first supervisor, I would like to thank my second supervisor Dr Martin Birkett for his insightful comments, encouragement and his assistance with my thesis and papers. I would also like to thank all the members of staff at Northumbria University for their assistance. Thanks must also go to all my friends who supported me during my time here and for all the fun we have had in the last four years.

Thanks to the people and the government of my country, the Libyan Government. I could not have gone through the doctoral program overseas without Libyan Government financial support. I would like to express my full appreciation to the staff in the Ministry of Higher Education and Science Research of Libya and to the staff in the culture attaché in the Libyan embassy in London. I would also like to express my full appreciation to staff in the Zawia University in general and in the Regdaleen Engineering Faculty in particular.

Finally, I must express my very profound gratitude to my mother for her pray for me and unceasing attention, to my father for his unfailing support, and to my brothers and sisters and their families for their spiritually support throughout writing this thesis and my life in general.

Thank you all.

List of Publications

1. Ali, A., P. Hackney, D. Bell and M. Birkett (2014). Dynamic job shop scheduling with alternative routes based on genetic algorithm. Engineering Optimization 2014, CRC Press: 827-832.
2. Ali, A., P. Hackney, M. Birkett and D. Bell (2015). Genetic Algorithms for Minimizing the Number of Tardy Jobs in Job Shop Scheduling With Machine Setup Issue. Proceedings of the 13th International Conference on Manufacturing Research Bath, UK, ICMR.
3. Ali, A., P. Hackney, D. Bell and M. Birkett (2015). Genetic Algorithms for Solving Bicriteria Dynamic Job Shop Scheduling Problems with Alternative Routes. Proceedings of the International Conference on Engineering & MIS 2015. Istanbul, Turkey, ACM: 1-8.
4. Ali, A., P. Hackney, M. Birkett and D. Bell (2016). Efficient Nondominated Sorting with Genetic Algorithm for solving Multi-objective job shop scheduling problems. 1st International Conference on Multidisciplinary Engineering Design Optimization. Belgrade, IEEE.

Contents

Chapter 1. Introduction 1

1.1 Background 2

1.2 Job Shop Scheduling 3

1.3 Aim and Objectives of the Thesis 5

1.4 Thesis outline 6

Chapter 2. Literature Review 8

2.1 Introduction 9

2.2 Factors for Describing JSSPs 9

2.2.1 Time Parameters 10

2.2.2 Setup Time 12

2.2.3 Arrival Patterns 13

2.2.4 Process Paths 14

2.3 Performance Measures 15

2.3.1 Process-Focused Performance Criteria 16

2.3.2 Customer-focused Due Date Criteria 17

2.3.3 Cost-based Criteria 17

2.4 Optimization Methods 20

2.4.1 Exact Methods 20

2.4.2 Constructive Methods 21

2.4.3 Iterative Methods 23

2.5 Multi-objective Optimization and Pareto-optimal Solutions 30

2.5.1 Solving Multi-objective Optimization Problems 32

2.6 Previously Existing Work and Gap of Knowledge 35

2.7 Summary 38

Chapter 3. Problem Description, Complexity and Formulation 40

3.1 Introduction 41

3.2 Problem Description and Complexity 41

3.2.1 Deterministic Dynamic Release Date 44

3.2.2 Alternative Machines 44

3.2.3 Machine Setup Time 48

3.2.4 Jobs with Various Lengths and Recirculation 49

3.2.5 Priorities among jobs expressed by weights 49

3.2.6 Multi-Objective Optimization 50

3.2.7 Effect of Time Uncertainty on the Optimal Solution 53
3.3 Mathematical Formulation 53
3.3.1 Indices 53
3.3.2 Parameters 54
3.3.3 Decision Variables 54
3.3.4 Constraints 55
3.3.5 Objective Function 56
3.4 Summary 58
Chapter 4. Research Methodology 59
4.1 Introduction 60
4.2 An Introduction to Genetic Algorithms in Job shop Scheduling Problems 60
4.2.1 Representation 61
4.2.2 Initial Population 63
4.2.3 Fitness Evaluation Function 63
4.2.4 Genetic Operators 63
4.2.5 Control Parameters 68
4.3 Proposed System 68
4.3.1 Solution Representation 70
4.3.2 Creation of Initial Population 72
4.3.3 Design of Fitness Evaluation Function 72
4.3.4 Genetic Operators 78
4.3.5 Solving Scheduling Problems with Machine Setup Time 83
4.3.6 Solving Scheduling Problems with Alternative Machines 84
4.3.7 Representation of Processing Time Uncertainty 86
4.4 Summary 86
Chapter 5. Computational Results and Discussion 88
5.1 Introduction 89
5.2 Computational Results for Classical JSSP 89
5.3 Experimental Plan 91
5.3.1 Incorporating Job / Machine Release Time 91
5.3.2 Incorporating Machine Setup Time 93
5.3.3 Incorporating Alternative Machines 96
5.3.4 Incorporating Jobs with Recirculation and Various Numbers of Operations 98
5.3.5 Scheduling System with Multi-Objective Optimization 99

5.4 The Effect of Uncertainty on the Optimal Makespan 108
5.4.1 Change of Processing Times with the Same Ratio 109
5.4.2 Change of Processing Times with Different Ratios 110
5.4.3 Time Uncertainty with Slack Time 112
5.5 Summary 115
Chapter 6. Conclusions, Contributions to knowledge and Future Work 116
6.1 Introduction 117
6.2 Summary of the Research work 117
6.3 Conclusion 121
6.4 Contributions 122
6.5 Future Research Directions 123
Appendices 124
REFERENCES 141

Table of Figures

Figure 1-1. The layout process of thesis objectives 6
Figure 2-1. Factors for Describing JSSPs 10
Figure 2-2. Common Performance measures 18
Figure 2-3. Percentage of included factors in MO-JSSPs from the available literature 37
Figure 3-1. Single Processing Route for J1 45
Figure 3-2. Alternative Processing Routes for J1 45
Figure 3-3. Alternative Processing Routes for J2 47
Figure 3-4. Different Machine Matrices for J1 & J2 48
Figure 4-1. Proposed JSS System 60
Figure 4-2. General Layout of GA 61
Figure 4-3. Classification of JSSPs Representation in GA 62
Figure 4-4. One-point Crossover 65
Figure 4-5. Two-point Crossover 65
Figure 4-6. Uniform crossover 66
Figure 4-7. Shift Mutation 67
Figure 4-8. Pairwise Interchange Mutation 67
Figure 4-9. Inversion Mutation 68
Figure 4-10. The Layout of the Proposed System for Solving MO-JSSPs 69
Figure 4-11. Operation Based Matrix Representation 70
Figure 4-12. Chromosome Based Matrix Representation 71
Figure 4-13. 3-Dimensional Matrix for Scheduling Evaluation 73
Figure 4-14. Proposed ENS-BPSS for Finding the Front of a Solution 77
Figure 4-15. Two-parents / One-row Crossover 80
Figure 4-16. Two-parents / Two-rows Crossover 80
Figure 4-17. Three-parents / One-row Crossover 81
Figure 4-18. Shift Mutation (Insertion Neighbourhood) 82
Figure 4-19. Pairwise Interchange Mutation (Swap Neighbourhood) 82
Figure 4-20. Inversion Mutation 83
Figure 4-21. Machine Selection Based on the Processing Time 85
Figure 5-1. LA1 with Job and Machine Release Date 93
Figure 5-2. LA1 with Machine Setup times and Release time 95
Figure 5-3. Gantt chart LA1 with Alternative Machines 97
Figure 5-4. Jobs with Recirculation and Different Numbers of Operations for LA1 99
Figure 5-5. Pareto Front for M_LA1 101
Figure 5-6. Pareto Front for M_LA2 101
Figure 5-7. Pareto Front for M_LA3 102
Figure 5-8. Pareto Front for M_LA4 102
Figure 5-9. Pareto Front for M_LA5 102
Figure 5-10. Pareto Front for M_LA6 102
Figure 5-11. Pareto Front for M_LA7 102
Figure 5-12. Pareto Front for M_LA8 102
Figure 5-13. Pareto Front for M_LA9 103
Figure 5-14. Pareto Front for M_LA10 103

Figure 5-15. Pareto Front for M_LA11 103
Figure 5-16. Pareto Front for M_LA12 103
Figure 5-17. Pareto Front for M_LA13 103
Figure 5-18. Pareto Front for M_LA14 103
Figure 5-19. Pareto Front for M_LA15 104
Figure 5-20. Pareto Front for M_LA16 104
Figure 5-21. Pareto Front for M_LA17 104
Figure 5-22. Pareto Front for M_LA18 104
Figure 5-23. Pareto Front for M_LA19 104
Figure 5-24. Pareto Front for M_LA20 104
Figure 5-25. Gant Chart of the Optimal Makespan for LA01 114

Table of Tables

Table 2-1. Common Performance Measures for Scheduling Problems. 19
Table 2-2. Previous research in MO-JSSP 35
Table 4-1. Types of Crossover Operators for JSSPs 66
Table 4-2. Comparison Between Actual and NSGAII Fronts 75
Table 5-1. Obtained Results for Benchmark JSSPs Using Operation Based Matrix Representation 90
Table 5-2. Releas Times for Jobs and Machines 92
Table 5-3. Jobs Setup Groups and Setup Times in Each Machine 94
Table 5-4. The Alternative Machines Processing Times 97
Table 5-5. Change of the Processing Times with the Same Ratio 110
Table 5-6. Change of the Processing Times with Different Ratios 111

Acronyms

AM	Alternative Machines
AIS	Artificial Immune System
ANN	Artificial Neural Network
BPSS	Backward Pass Sequential Strategy
BS	Beam Search
BCO	Bee Colony Optimization
BA	Bees Algorithm
BB	Branch and Bound
DP	Dynamic Programing
EDD	Earliest Due Date
ENS	Efficient Non-dominated Sort
ENS-SS	Efficient Non-dominated Sort using Sequential Strategy
ENS-BPSS	Efficient Non-dominated Sort using the Backward Pass Sequential Strategy
ES	Elitism Selection
FCFS	First Come First Served
FL	Fuzzy Logic
GA	Genetic Algorithms
GRASP	Greedy Randomized Adaptive Search Procedure
IP	Integer Programming
JRD	Job Release Date
JSS	Job Shop Scheduling
JSSP	Job Shop Scheduling Problem
MINSLACK	Minimum Slack
MIP	Mixed Integer Programming
MOGA	Multi-Objective Genetic Algorithm
MO-JSSP	Multi-objective Job Shop Scheduling Problem
MOMGA	Multi-Objective Messy Genetic Algorithm
NPGA	Niched Pareto Genetic Algorithm
NP-hard	Non-deterministic Polynomial-hard
NSGA	Non-dominated Sorting Genetic Algorithm

NSGA-II	Non-dominated Sorting Genetic Algorithm-II
PSO	Particle Swarm Optimization
PDR	Priority Dispatching Rule
RS	Rank Selection
RWS	Roulette Wheel Selection
SS	Sequential Strategy
ST	Setup Time
SPT	Shortest Processing Time
SA	Simulated Annealing
SUS	Stochastic Universal Sampling
SPEA	Strength Pareto Evolutionary Algorithm
SPEA2	Strength Pareto Evolutionary Algorithm
TS	Tabu Search
ACO	The Ant Colony Optimization
TRS	Tournament Selection
UPT	Unfixed Processing Time
VNS	Variable Neighbourhood Search
VEGA	Vector Evaluated Genetic Algorithm
WIP	Work-In-Process

List of Nomenclature

J_i	a total number of operations to complete job i
AR_i	Alternative Routes for job i
R^N	design space
n_{t_s}	Domination Count
d_i	due date of job i
g	equality constraint
$FIT(s)$	Fitness value of solution $s-th$
F_i	flow time of job i
KR	Front index $(KR = 1,2,...,LR)$
h	inequality constraint
i	job index $(i = 1,2,...,n)$
k	machine index $(k = 1,2,...,m)$
$Mach$	Machine Matrix
$C_{\max}$	Maximum completion time (makespan)
$F_{\max}$	Maximum flow time
$L_{\max}$	Maximum lateness
$T_{\max}$	Maximum tardiness
n	number of jobs
m	number of machines
M	number of objectives
$\sum_{i=1}^{n} U_i$	Number of tardy jobs
f	objective function
R^M	objective space
j	operation index
O_{ij}	Operation number j of job i $(O_{i1}, O_{i2}, ..., O_{ij})$
N	population of size
P_c	Probability of crossover

P_m	Probability of mutation
p_{ijk}	Processing time of operation O_{ij} on machine k
G_{ijk}	Setup group for operation j of job i on machine k
St_{Gk}	Setup time for a set of operations that belongs to setup group G on machine k
FR	Solution front
t_s	Solution number s
s	Solution or individual index $(s = 1,2,...,N)$
SP	sorted population
C_{ijk}	The completion time for operation j of job i on machine k.
C_i	The completion time for the last operation of job i.
E_i	the Earliness of job i
w_i	The importance weight of job i
L_i	the lateness of job i
$Ps(s)$	the probability of selecting the individual or solution $s-th$
$Pk(k)$	the probability of selecting the machine k from a set of alternative machines AM_{ij}
r_i	The release date or arrival time of job i
rt_k	The release time for machine k in the job shop
AM_{ij}	The set of machines that can process operation j of job i
S_{ijk}	The start time for operation j of job i on machine k.
T_i	the tardiness of job i
w_i	The weight of job i
w_v	The weight of objective v
tf	tightness factor
C_{total}	Total completion time
F_{tot}	Total flow time

L_{tot}	Total lateness
LR	Total number of fronts
$\sum p_i$	total processing times of job i (including setup times)
T_{tot}	Total tardiness
$\sum W_i$	total waiting times of job i
C_{w_tot}	Total weighted completion time
F_{w_tot}	Total weighted flow time
L_{w_tot}	Total weighted lateness
T_{w_tot}	Total weighted tardiness
x	vector of design variable

Chapter 1. Introduction

1.1 Background

It has become increasingly more apparent that to remain competitive in today's open market, manufacturing and service organisations need to manage their resources in an effective way. The market needs high quality, high product variety, and short lead-times. To meet these needs, planning and scheduling are two crucial topics which endeavour to increase efficiency, to improve resource utilisation and to reduce delivery time. In general, scheduling can be defined as an allocation of resources over a period of time to execute a number of tasks with the aim of optimizing certain objectives (Chaudhry 2012). It is a short-term execution plan that consists of a set of activities which need to be performed in manufacturing or service industries in order to manage the execution process and it defines, in a time manner, when each activity should be executed and how the organisation's resources should be used to satisfy the plan and meet the requirements. Failure to do so may lead to chaos in the system and cause a significant loss of the organisation's revenue. Hence, a good scheduling system assists enterprises to use their resources effectively and to achieve the strategic objectives that have been planned. However, practical scheduling problems are generally more difficult to solve to optimality because of the number and variety of tasks and jobs, the dynamic environment and the usually conflicting objectives (Fera, Fruggiero et al. 2013).

In manufacturing industries, production scheduling problems can be classified according to some criteria such as flow patterns (flow shop, job shop or open shop), processing mode (unit processing or batch processing), job release patterns (static or dynamic), and work centre configuration (single machine, identical parallel machines, uniform parallel machines, or unrelated parallel

machines) etc. Different patterns of scheduling problems put different constraints on how different operations can be scheduled on different machines. For instance, based on the flow pattern, in a flow shop environment, each job has to be processed on each machine in the same order, while in a job shop environment the system consists of a number of different machines and each job has a specified machine route in which some operations can be missing and some can be repeated. On the other hand, in an open shop environment, each job should be processed once on each machine with no order restriction for the machines. For all these environments, operation times for each job on different machines are usually not the same (Bayındır 2005). The focus of this research is on scheduling problems in job shop manufacturing environments.

1.2 Job Shop Scheduling

Job Shop Scheduling Problems (JSSPs) have been around since the mid-1950s and occur mostly in manufacturing, where each customer order has its own specific characteristics and the order quantities are relatively small. The job might be produced only once, at irregular intervals, or periodically at regular intervals, based on market demands, policies and customer orders (Ebadian, Rabbani et al. 2009). For make-to-order job shop environments, scheduling of manufacturing operations spans between the order and delivery dates. Different jobs can have different routings, due dates, priorities, quantities, and resource requirements (Sawik 2006). Because of this large diversity of jobs involved, the Job Shop Scheduling (JSS) process tends to be a very complex problem mathematically (Browne, O'Kelly et al. 1982). Thus there is an increased need for an effective and reliable JSS system.

In general, each job in such an environment has a number of operations that must be performed on a number of machines to complete a job. The sequence of operations for each job is predefined based on the jobs technological requirements. Machines cannot handle more than one job at a time and operations of the same job cannot be started until their preceding operation is completed. Job Shop scheduling in this type of environment is considered to be one of the most important and complicated scheduling models existing in the practice and is amongst the hardest combinatorial optimization problems. They belong to a large class of intractable numerical problems known as Non-Deterministic Polynomial-hard (NP-hard). To find an optimal solution for such a problem the algorithm requires a number of computational steps that grows exponentially with the input (Jain and Meeran 1999).

Over the past decades, a large number of methods have been proposed to solve the JSSPs optimally with exact methods such as branch and bound algorithms, or near to optimality with approximation methods such as genetic algorithms. Yet there is still no efficient method which can guarantee an optimal solution, consistently, even for a single-objective and there is no work which shows that any of these methods outperform each other with regard to all problem aspects. Even though the JSSP with a single-objective has been widely studied, the research on the Multi-Objective Job Shop Scheduling Problem (MO-JSSP) is still relatively limited. Furthermore, the majority of previous works on MO-JSSPs do not incorporate some very important issues that reflect to the real job shop manufacturing environment, such as setup times, release times and alternative machines, whilst only a few studies have

dealt with time uncertainty, which closely describes and represents the real world problem.

In order to identify the gap of knowledge, a comprehensive literature review was conducted in this research. The survey study showed that although the JSSP is a popular topic, there still is no model that incorporates setup times, alternative machines, jobs and machines release times altogether in a multi objective job shop scheduling system. Therefore, the motivation of this work was to bring together all of these issues to acquire a more reliable, realistic and accurate scheduling system for the job shop manufacturing environment. Processing time uncertainty was also studied in order to identify the most important factors that affect the scheduling objective.

1.3 Aim and Objectives of the Thesis

The aim of this research was to develop a unique scheduling system capable of producing optimum work flow sequencing for multiple objectives within a complex job shop manufacturing environment. This aim was achieved through undertaking the following objectives:

- Conducting a comprehensive survey study on MO-JSSPs and a review existing production scheduling systems.
- Identifying and evaluating modelling techniques capable of simulating complex job shop production systems.
- Developing an optimization system capable of implementing multiple objectives for JSSPs.
- Producing MatLab codes to implement the developed system.

- Appling multiple datasets of benchmark JSSPs to optimize and validate the developed system. Then modifying these datasets to suit MO-JSSPs with all incorporated factors.
- Critically evaluating results from the benchmark problems to further develop the system.
- Disseminating of the results in journals, conference papers, thesis, and industrial presentations.

Figure 1-1 shows the layout process of thesis objectives.

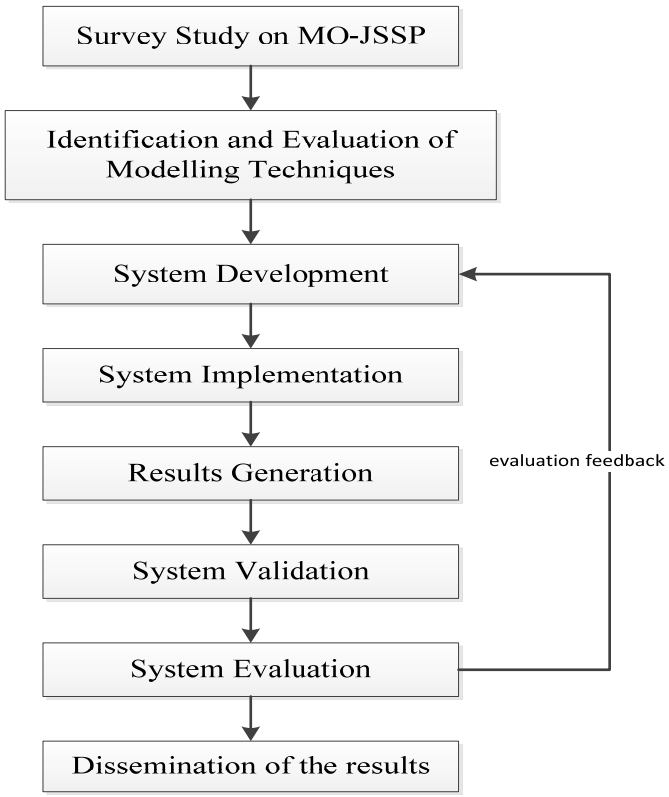

Figure 1-1. The layout process of thesis objectives

1.4 Thesis outline

Chapter 1 "Introduction":- An introduction to this PhD thesis, including background, scope, aim and objectives is presented in this chapter.

Chapter 2 "Literature Review":- A wide literature review of JSSPs, including factors of describing JSSPs, methods for solving JSSPs, multi objective techniques and previous work offered to solve the MO-JSSP is conducted in chapter 2.

Chapter 3 "Description, Complexity and Formulation":- In this chapter, JSSPs to be solved in this thesis are described and mathematical formulation is given. Also the complexity of the problem is explained.

Chapter 4 "Research Methodology":- The developed system for solving the MO-JSSP that has been described in chapter 3 is introduced in this chapter.

Chapter 5 "Computational Results and Discussion":- In this chapter experimental results and discussion on the research results with key observations in the research methodology are given.

Chapter 6 "Conclusion and Future Work":- The summary, conclusions and contributions to knowledge of this research and suggestions for further work are provided in this chapter.

Chapter 2. Literature Review

2.1 Introduction

The general goal of this research is to investigate and develop a new scheduling system for dealing with multiple objective, complex “Job Shop” manufacturing environments. To clearly identify the current knowledge and gaps, a comprehensive literature review is conducted. First, factors for describing JSSPs are identified and then existing methods for solving JSSPs, as well as multi-objective optimization problems, are specified. The previous works that have been done for solving MO-JSSPs are then reported. The chapter concludes with clearly defined gaps of knowledge which underpin the current work.

2.2 Factors for Describing JSSPs

JSSPs can be classified and considered according to the main factors that describe the problem in hand. The simplest form of the problem which can be found in the literature is the so called classical JSSP and can be traced back to the early 1950s (Jain and Meeran 1999). Since then, many assumptions have been made to reflect the problem in real world applications.

In this section, the most important factors for describing JSSPs are briefly investigated. These factors are used at the end of this chapter to distinguish between previous works that have been done for solving MO-JSSPs. The most popular methods for solving JSSPs and multi-objective optimization that have been reported in the literature are stated afterward. Figure 2-1 shows the main categories of factors that describe JSSPs.

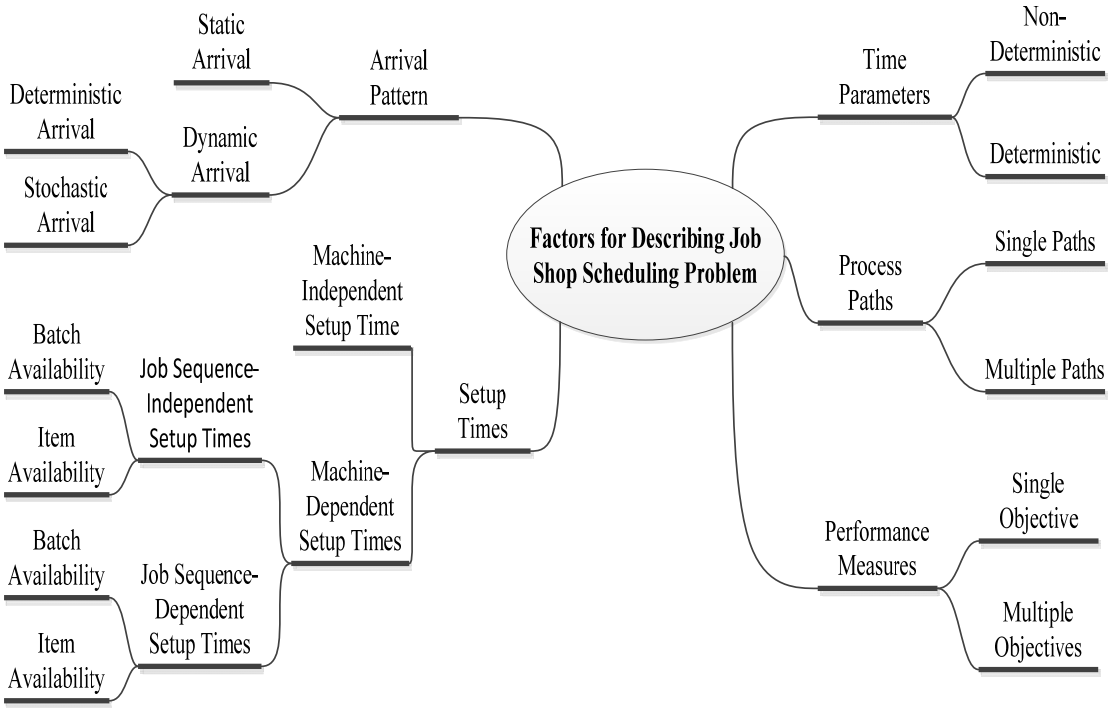

Figure 2-1. Factors for Describing JSSPs

2.2.1 Time Parameters

Time is an essential factor when dealing with a schedule, so that resources are allocated to perform a set of tasks over a period of time (Hollier 1975). In scheduling problems there are three main time parameters known as processing time (p_{ijk}), release date (r_i), and due date (d_i). Processing time is the main time parameter that is necessary to deal with any scheduling problem. The processing time p_{ijk} represents the time that has to be spent on machine k to execute operation j of job i. The release date r_i is the earliest time at which job i can start its processing. The due date d_i represents the date that job i is promised to the customer (Pinedo 2005). These three parameters can be presented by using single values, which is known as deterministic (crisp)

time, or by using multiple values such as fuzzy numbers, stochastic values or interval numbers to represent uncertainty (Chakraborty 2009).

Uncertainty of time parameters is a basic feature of manufacturing processes. Fuzzy and stochastic theories are two commonly used approaches to model the uncertainty of time parameters in scheduling problems. In a fuzzy theory approach, the times are given as fuzzy numbers. Two main methods exist in the literature to deal with fuzzy processing time, one by using triangular fuzzy numbers and the other by using the mean value of a fuzzy number so that processing times are de-fuzzed. Similarly, fuzzy double number and fuzzy triangular number are two main methods that are commonly used in the literature to represent fuzzy due date (Abdullah and Abdolrazzagh-Nezhad 2014). In the stochastic theory approach, time parameters are assumed to be random variables (Chakraborty 2009). For fuzzy and stochastic approaches, probability distribution and membership function need to be known in advance and a number of data are required to decide the distribution of stochastic variables or to build the fuzzy membership function (Chakraborty 2009). Interval number theory has some different features from fuzzy theory or stochastic theory. The lower and upper bounds of the interval are only needed to indicate time uncertainty (Lei 2011, Lei 2012). A survey of interval scheduling can be found in (Kolen, Lenstra et al. 2007). To the best of the author's knowledge, only one research article is available for solving JSSPs with the interval number, which was presented by Lei (2012). Because of its lower complexity the interval number is utilized in this research to evaluate the effect of processing time uncertainty on the optimal makespan.

2.2.2 Setup Time

Setup time can be defined as the time to prepare the required resources, such as machines, to execute different tasks. In manufacturing applications, setup activities often occur while shifting from one operation to another. For instance, cleaning up the machines; setting the required jigs and fixtures; setting the jobs in jigs and fixtures; positioning work in process; obtaining, adjusting and returning tools for an operation; and inspection of the material in manufacturing systems, are all considered as setup activities which need to be done before executing the task (Sharma and Jain 2015). In general, machine setup times can be classified into two main categories; machine independent setup times and machine dependent setup times, as shown in Figure 2-1. The case of machine independent setup times occurs when the setup times are included in the operation processing time or when there is no need for setup times. However, in the machine dependent setup times, the setup can be job sequence independent, where the preceding job does not have an effect on the duration of the setup time, or job sequence dependent, where the duration of the setup depends directly on the preceding job. These two classes can be further divided into setup times with batch availability or setup times with item availability (Sotskov, Tautenhahn et al. 1996, Allahverdi, Ng et al. 2008).

In real world industrial applications, scheduling problems involving machine setup times are receiving increasing attention. When setup times are properly incorporated to the scheduling decision, their addition has been shown to give significant improvements in reliability and accuracy of the overall scheduling system. Therefore, setup times must always be considered in solving scheduling problems (Kim and Bobrowski 1994).

JSSPs with setup time considerations was first studied by Wilbrecht and Prescott (1969). The study shows that for a fully loaded shop, especially in sequence-dependent setup times, setup times play a very important role in the performance of a job shop operation. A comprehensive review of scheduling problems with setup times has been given by Allahverdi, Gupta et al. (1999), Xiaoyan and Wilhelm (2006) , Allahverdi, Ng et al. (2008), Allahverdi (2015) and Sharma and Jain (2015). These studies also provide some directions for future research. According to Sharma and Jain (2015), forty two articles are JSSPs with sequence-dependent, non-batch setup times, only two articles related to JSSPs with sequence-independent, non-batch setup times and four articles each related JSSPs with sequence-dependent and sequence-independent batch setup times have been reported.

Although dealing with sequence-dependent setup times is more difficult, there are still limited researches in JSSPs with sequence-independent setup times. In addition, many issues such as multi objective JSSPs and dynamic JSSPs etc. with sequence-independent setup times have not been considered together. Therefore, the current research was confined to solve MO-JSSPs with machine set up times and job sequence-independent setup times.

2.2.3 Arrival Patterns

This refers to the time that the job arrives to the system, and it can be a static arrival or dynamic arrival. In a static arrival pattern, a determined number of jobs arrive at the same time and no further jobs arrive until the existing set of jobs has been finished. In a dynamic arrival pattern, jobs arrive at different times and are not fixed at one time; jobs can also continue to arrive indeterminately in the

future. Dynamic arrival patterns can be further classified as deterministic or stochastic based on the way of designing the job arrival times. Deterministic arrival patterns assume that the job arrival times are known in advance. In stochastic arrival patterns, job arrival times are random variables defined by a known probability distribution (Lin, Goodman et al. , Xhafa and Abraham 2008). Usually, in the stochastic arrival pattern, the scheduling problem is solved by an insertion method based on some dispatching rules or by rescheduling the remaining tasks with newly arrived jobs. Although the majority of previous work assumes that all jobs are available at the same time, in real manufacturing practice, jobs arrive dynamically at different times. In addition, some job's requirements, such as raw materials, might not be available at the time the order takes place. Therefore, in this research, dynamic job arrival with deterministic form was considered. It was also assumed that not all the machines are available at time zero when the schedule starts, which makes the scheduling system more reliable.

2.2.4 Process Paths

The classical JSSPs assume that only one machine type is capable to perform a particular type of operation i.e. no more than one machine is available for each type of job operation, thus each job will only have a single route. In real manufacturing practice, process paths can be very complicated and it is very uncommon for every job to be processed by every machine. The process path can change dynamically due to the availability of resources or sometimes the need for rework. It is common in real world scheduling problems to have a number of choices of machines on which certain operations can be executed. These machines can be identical with identical processing times for a given

operation. Otherwise, machines can be non-identical with different processing times and sometimes different resource requirements and processing characteristics. Also a machine can be dedicated for a specific type of operation or it can be capable of performing a variety of operations on the shop floor. The system with alternative machines is known as a flexible system and it can be fully flexible, so that any machine can execute any operation, or it can be partially flexible, where each operation can be executed on one or a subset of machines (Kacem, Hammadi et al. 2002). Wilhelm and Hyun-Myung (1985) investigated the influence of alternate operations in flexible manufacturing systems on system performance. The results showed that alternate operations can increase machine utilization and reduce in-process inventory. However, further decisions of machine allocation during the scheduling are required. Decisions need to be made between automated and manual equipment and between newer and older equipment etc. The process paths and even sometimes, the technological sequence of operations, may change based on the choice of machines. Ideally a good scheduling system needs to be very flexible regarding the types of process paths to be captured (Rodammer and White 1988). In this research a partially flexible JSS system where each operation can be executed on one or a subset of the machines in the shop floor is considered.

2.3 Performance Measures

Performance measures or scheduling objectives are criteria by which the performance of any solution can be measured (Oyetunji 2009). They show how ‘good’ a schedule can be (Hsu 2006). These scheduling objectives can be

categorized based on some broad criteria (Framiñán Torres, Leisten et al. 2014, Collier and Evans 2009) as follows:

2.3.1 Process-Focused Performance Criteria

These types of objectives are based on only the information about the start and end times of jobs and focus on shop performance such as equipment utilization and Work-In-Process (WIP) inventory. Two common objectives are used; flow time and makespan.

FLOW TIME is the amount of time a job spends in the service or manufacturing system. Also known as throughput time or time spent in the system, including service.

$$F_i = \sum p_i + \sum W_i = C_i - r_i$$

Where; F_i is the flow time of job i, $\sum p_i$ is the total processing times of job i (including setup times), $\sum W_i$ is the total waiting times of job i, C_i is the completion time of job i and r_i is the release time for job i.

MAKESPAN : is the total amount of time required to complete a given set of jobs. The makespan $C_{\max}$ is important when the number of jobs is finite. A short makspan aims to achieve high equipment and resource utilization by getting jobs out of the job shop quickly.

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

2.3.2 Customer-focused Due Date Criteria

Due date related objectives are mainly concerned with customer satisfaction and service. Common scheduling objectives are minimizing the tardiness and the number of jobs being tardy or late.

LATENESS : is the difference between the completion time and the due date (either positive or negative).

$$L_i = C_i - d_i$$

Where; L_i is the lateness of job i and d_i is the due date of job i. If L_i is positive then it is called tardiness, when it is negative, it is called earliness. Thus

Job Earliness $E_i = \max(0, -L_i)$

Job Tardiness $T_i = \max(0, L_i)$

Number of tardy jobs $\sum_{i=1}^{n} U_i$ where $U_i = \begin{cases} 1; & C_i > d_i \\ 0; & otherwise \end{cases}$

2.3.3 Cost-based Criteria

Setup cost, inventory cost, processing cost, and material handling costs etc. are common in manufacturing. Cost-based criteria can be considered as the most important criteria, but it is usually hard to obtain an accurate cost value for each type of job, identify the relevant cost categories, and allocate costs to manufacturing parts. Usually, costs are considered implicitly in customer-focused due dates and process-focused performance criteria. In manufacturing scheduling, costs can be represented by weighted time parameters. For

instance, different jobs can have different weights based on the importance of different customers and the penalty associated to the tardy delivery (M'Hallah and Bulfin 2007). In this case the weighted number of tardy jobs is used as follows:

$$\sum_{i=1}^{n} w_i U_i$$

Where w_i is the importance weight of job i.

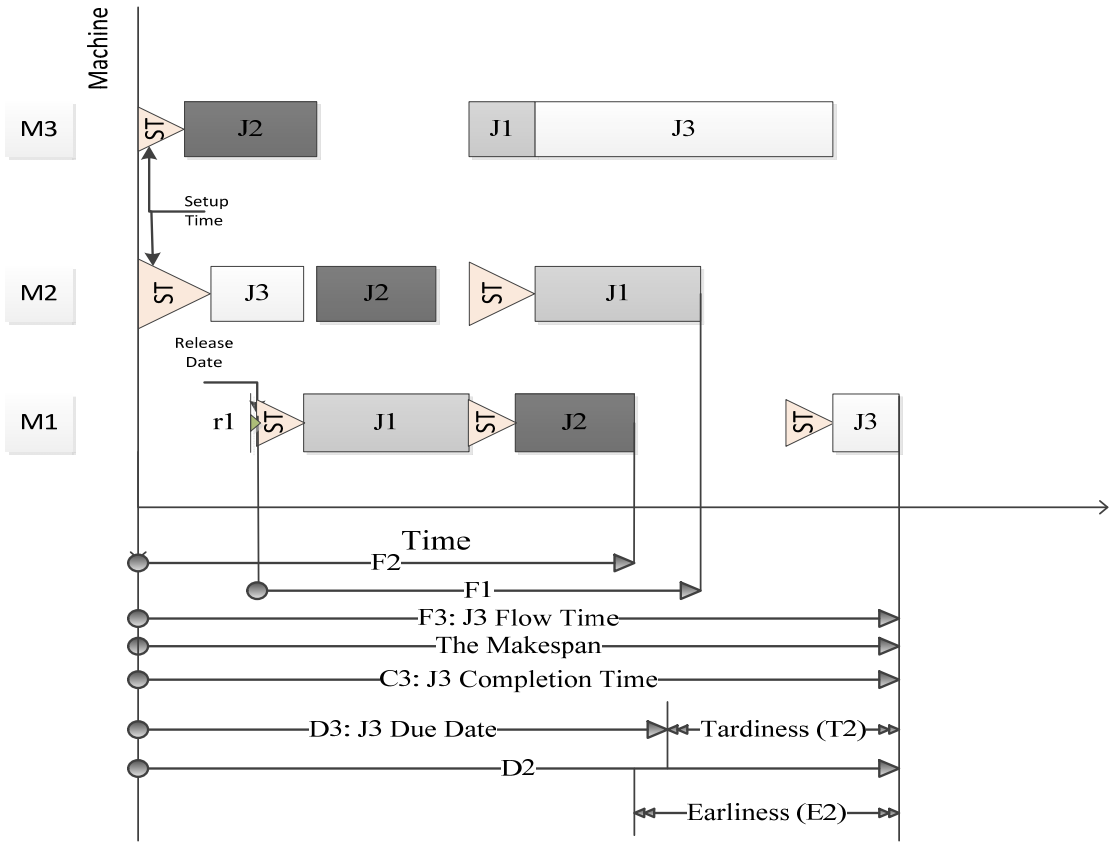

Figure 2-2. Common Performance measures

Table 2-1 summarizes the most commonly used objectives used as performance measures for JSSPs (Oyetunji 2009). A more detailed overview of performance measures for JSSPs can be found in (Pinedo 1995).

Table 2-1. Common Performance Measures for Scheduling Problems.

performance measures	Symbol and mathematical formula	Description
Total completion time	$C_{total} = \sum_{i=1}^{n} C_i$	The objective is to minimize the sum of all the completion times of the jobs.
Total weighted completion time	$C_{w\_tot} = \sum_{i=1}^{n} w_i C_i$	The objective is to minimize the sum of all the completion times multiplied by the relative weights of the jobs.
Maximum complete time	$C_{\max} = \max(C_1,...,C_n)$	The objective is to minimize the maximum completion time the last job leave the system.
Total flow time	$F_{tot} = \sum_{i=1}^{n} F_i = \sum_{i=1}^{n} (C_i - r_i)$	The objective is to minimize the sum of all the flow times of the jobs.
Total weighted flow time	$F_{w\_tot} = \sum_{i=1}^{n} w_i F_i = \sum_{i=1}^{n} w_i (C_i - r_i)$	The objective is to minimize the sum of all the flow times multiplied by the relative weights of the jobs.
Maximum flow time	$F_{\max} = \max(F_1, F_2,..., F_n)$	The objective is to minimize the longest flow times of the jobs.
Total lateness	$L_{tot} = \sum_{i=1}^{n} L_i = \sum_{i=1}^{n} (C_i - d_i)$	The objective is to minimize the sum of all lateness of the jobs.
Total weighted lateness	$L_{w\_tot} = \sum_{i=1}^{n} w_i L_i = \sum_{i=1}^{n} w_i (C_i - d_i)$	The objective is to minimize the sum of all lateness multiplied by the relative weights of the jobs.
Maximum lateness	$L_{\max} = \max(L_1, L_2,..., L_n)$	The objective is to minimize the maximum lateness
Number of tardy jobs	$\sum_{i=1}^{n} U_i$ where $U_i = \begin{cases} 1; & C_i > d_i \\ 0; & otherwise \end{cases}$	The objective is to minimize the total number of tardy jobs.
Total tardiness	$T_{tot} = \sum_{i=1}^{n} T_i$	The objective is to minimize the total tardiness.
Total weighted tardiness	$T_{w\_tot} = \sum_{i=1}^{n} w_i T_i$	The objective is to minimize the sum of all the tardiness multiplied by the relative weights of the jobs.
Maximum tardiness	$T_{\max} = \max(T_1, T_2,..., T_n)$	The objective is to minimize the longest of the tardiness of the jobs.

2.4 Optimization Methods

In this section different types of optimization methods for solving JSSPs are described. The optimization methods aim to find an optimal or near to optimal solution with short computational time. Distinction can be made between three different types of optimization methods: exact methods, constructive methods and iterative methods. Exact methods can guarantee finding an optimal solution for small size JSSPs; however, the computational time increases exponentially with problem size. Constructive methods have short computational time but do not guarantee the optimal or even a good solution. Iterative methods produce a good solution in reasonable time but do not guarantee an optimal solution. A brief review of these methods is given below. A research survey of various methods that have been used to solve JSSPs can be found in (Jain and Meeran 1999), (Çaliş and Bulkan 2015) and (Chaudhry and Khan 2015).

2.4.1 Exact Methods

The aim of exact methods is to find an exact solution by using enumerative algorithms which rely on more elaborate and sophisticated mathematical constructs. Branch and Bound (BB) is the main enumerative method, where the solution space is presented by a dynamically constructed tree of all feasible schedules that are implicitly searched. This method applies some rules and procedures to remove large portions of the tree from the search space and was the most popular technique to solve JSSPs for many years. However, the disadvantage of BB is that it has excessive computing requirements, since the number of branches or nodes is often very large, which limit its application in solving JSSPs (Jain and Meeran 1999). Mathematical programming methods

are another exact method that aim to find the optimal solution, where the function is defined by linear and nonlinear constraints (equalities and inequalities) (Pinedo 2005). Problems have been formulated using Integer Programming (IP), Mixed Integer Programming (MIP) (Bowman 1959, Balas 1965, Balas 1967) and Dynamic Programing (DP) (Srinivasan 1971). Yet, the use of mathematical programming methods has been limited because JSSPs belong to the class of NP-hard problems and have difficulties in the formulation of material flow constraints as mathematical inequalities (Jones, Rabelo et al. 2001).

In general, although exact methods provide an important accomplishment to the research field and have remarkable theoretical values, most of these methods are inadequate for practical application as they are unable to achieve a feasible solution or to solve instances with more than 100 operations (Jain and Meeran 1999).

2.4.2 Constructive Methods

Constructive methods such as shifting bottleneck based heuristics (Adams, Balas et al. 1988), insertion algorithms (Werner and Winkler 1995) and priority dispatching rules, build a solution from the problem data (Jain and Meeran 1999). In the Shifting Bottleneck procedure, the identified bottleneck machine is solved optimality and then the solution is introduced into the overall schedule to determine the optimal sequence for the remaining machines. In the Insertion Algorithm an operation is inserted into a feasible schedule, such that the length of the longest path passing through the operation is minimized, then the reinsertion strategy is applied to iteratively improve the initial solution. Priority

Dispatching Rules (PDRs) are constructive procedures that assign a priority to all the operations which are available to be scheduled and then select the operation with the highest priority. PDRs have a low computation burden and are very easy to implement. According to the performance criteria PDRs can be classified into three main classes; simple priority rules, combination dispatching rules, and weight priority rules. In the simple priority rules, indexes are based on information associated with the job such as processing times (e.g. Shortest Processing Time (SPT) rule, in which jobs with SPT will be processed first, then the one with the next SPT and so on), due dates (e.g. Earliest Due Date (EDD) rule, in which the jobs with EDD will be processed first, then the one with the next EDD and so on), and slack time (e.g. Minimum Slack (MINSLACK) rule). Combination dispatching rules are a combination of simple priority rules, wherein the applied specific rule will depend on the situation that exists on the shop floor. For instance, to avoid jobs with large processing times from waiting in the queue for a long time, SPT can be applied until the queue length reaches a certain number, and then switches to the First Come First Served rule (FCFS). In weight priority rules, scheduled jobs are built by using many pieces of information. Weights are assigned to these pieces of information such as processing time, current time, and due date, to reflect their relative importance to the job. Then an objective function is defined for each job, to rank them. However, although PDRs can be used to calculate a feasible schedule quickly and show reasonable performance, they do not usually produce schedules that are close to optimal and there are no existing rules that show superiority, as these rules only take into account the current machine situation and its immediate surroundings, which decreases the quality of the solution as the

problem dimensionality increases (Jain and Meeran 1999, Pinedo and CHAO 1999, Jones, Rabelo et al. 2001).

2.4.3 Iterative Methods

Iterative methods modify a schedule by repeatedly reordering the order of operations. The use of these methods has grown to overcome the insufficiency exhibited by PDRs. For JSSPs and many other real world scheduling problems, iterative methods have been shown to be suitable application tools and are very promising. They are more effective than the other methods, because they have successfully found the best-known solutions for the benchmark dataset of JSSPs even on large scale problems (Rodammer and White 1988, Jain and Meeran 1999). Iterative methods have two main classifications known as artificial intelligence and local search methods, which are both widely applied to JSSPs. They try to improve the schedule with regard to one or more criteria from complete schedules that can be built randomly or based on some heuristic. However, these methods do not guarantee an optimal solution, yet they try to find a better solution than the current one (Pinedo 2005). In the following section a brief description of the most frequently used iterative methods is given, including some advantages and disadvantages of each one.

Fuzzy Logic (FL)

FL is a method of multi valued logic derived from fuzzy set theory that was proposed by Zadeh (1965) to handle estimated rather than more precise data. It allows incorporation of uncertainty into a decision model. In contrast with “crisp logic”, where binary sets have binary logic of 0 or 1, the FL variables may have a membership value range between 0 and 1 and are not limited to the two truth

values. The method has been applied to solve many real JSSPs, such as imprecise data and machine breakdown, by regenerating a new schedule (Çaliş and Bulkan 2013). However, the number and type of inputs to the fuzzy scheduler affect the quality of the scheduling significantly. The types of input should be selected based on their ability to describe the general conditions in the job shop. On the other hand the number of inputs suitable for fuzzy scheduling determines the complexity of the inference engine (Bilkay, Anlagan et al. 2004).

Genetic Algorithms (GA)

GA is a directed random search technique that was developed by Holland (1975) and was first applied to JSSPs by Davis (1985). A GA is based on the principles of the natural evolution process and depends on the population of individuals. It has two main operators, crossover and mutation, which manipulate individuals in a population over a number of generations to gradually improve their fitness (Pham and Karaboga 1998, Werner 2011). This method is easy to understand, has no demand over complex knowledge of mathematics and chromosomes share information with each other (Abdullah and Abdolrazzagh-Nezhad 2014). On the other hand, there are still some disadvantages of using this method such as crossover operators cannot produce feasible solutions without losing their efficiency (Jain and Meeran 1999). Despite the fact that GA have no memory and search based on random techniques, some solutions can be replicated and, moreover, the computational time is high (Abdullah and Abdolrazzagh-Nezhad 2014).

Simulated Annealing (SA)

SA is an iterative method that was proposed by Kirkpatrick, Gelatt et al. (1983) for finding the global minimum of a cost function that might have a number of local minima (Bertsimas and Tsitsiklis 1993). SA was inspired from the physical process of cooling and recrystallization of metals. For scheduling problems, the current scheduling solution corresponds to the current state of the thermodynamic, while the objective function corresponds to the energy equation for the thermodynamic system, and the global optimum corresponds to the ground state. By sampling the probability distribution of the system, this method will randomly generate new schedules with the aim of optimising the objective function, which corresponds to lowering a global temperature as the iterations progress (Jones, Rabelo et al. 2001). SA has a good selection technique and gradually obtains good solutions. However, the drawback of the SA method is it cannot quickly achieve good solutions to JSSPs. Therefore, research has been diverted to combining SA with other methods, such as GA, with the intention of improving the results and reducing the time required for calculation. Combining SA with some other methods has made SA competitive with regard to solution quality but it still requires high computational time (Jain and Meeran 1999).

Artificial Immune System (AIS)

The AIS method is an adaptive system that was inspired by the natural immune system to solve real-world problems. AIS exploits the immune system's features of learning and memory for problem solutions. The distinctive feature of the AIS is its ability to provide robust solutions. This method has been applied to solve many optimization problems such as computer security, pattern recognition, machine learning and scheduling problems including JSSPs. However, despite

the robustness of AIS, an optimal solution may not be reached (Chaudhry and Khan 2015).

Tabu Search (TS)

TS is another iterative method proposed by Glover (1989,1990) for solving optimization problems and has many successful applications in solving different types of scheduling problems including JSSPs. In JSSPs, a move from one solution to another solution is made whilst avoiding duplicates or resembling previously achieved solutions by recording the search history in the Tabu List. However, the disadvantage of the TS technique is it does not ex-change information through the larger set of parallel solutions, which can lead to the production of poor solutions. Combining TS with other techniques such as GA can generate more promising results but it will increase the computational time (Coello Coello, Lamont et al. 2007).

Greedy Randomized Adaptive Search Procedure (GRASP)

GRASP was first introduced by Feo and Resende (1989) and was commonly applied to solve combinatorial optimization problems. Each iteration in this method consists of two phases; a construction phase that builds a solution from scratch and a local search phase to investigate its neighbourhood solutions to find a local optimal solution. GRASP has been applied successfully to a number of scheduling problems including JSSPs. However, the solution obtained by GRASP is not necessarily optimal and is often used to build initial solutions to be explored by other methods (Chaudhry and Khan 2015).

Ant Colony Optimization (ACO)

ACO is a population-based optimization method that was introduced by Colorni, Dorigo et al. (1991). It was inspired from studying ant colonies and is used to search for optimal solutions to difficult optimization problems based on the behaviour of ants seeking a path between a source of food and their colony. The ants deposit more pheromone on the paths that have the shortest distance from the colony to get more pheromone, resulting in other ants selecting those paths in later iterations (Fox, Xiang et al. 2007). Recently there have been a number of researchers who utilised ACO to solve JSSPs such as Xing, Chen et al. (2010) and Korytkowski, Rymaszewski et al. (2013), because of its ability to avoid premature convergence. However, ACO has poor performance when the JSSP to be solved is larger than 10 jobs and 10 machines. In addition, sequences are generated based on random decisions and there is no centralized process to lead ACO to good solutions (Abdullah and Abdolrazzagh-Nezhad 2014).

Particle Swarm Optimization (PSO)

PSO is a population-based optimization method inspired by social behaviour of bird flocking and was introduced by Kennedy and Eberhart (1995) for optimization of continuous nonlinear functions. By having a population of candidate solutions, PSO optimises the problem by labelling the particles or the individual and moving them around in the search space, based on the mathematical formulae to find out the best solution (best position). Since the solution space of the JSSP is discrete, the PSO method is largely considered as unsuitable for the problem in hand. However, some researchers have tried to

modify the particle position representation, particle movement, and particle velocity to improve its suitability to JSSPs (Sha and Hsu 2006).

Variable Neighbourhood Search (VNS)

VNS is a framework for building heuristics proposed by Mladenovi and Hansen (1997), and designed for solving combinatorial and global optimization problems. It aims to escape from local optimum by changing the neighbourhood structure. In its basic form, VNS explores a set of neighbourhoods of the current solution, makes a local search from a neighbour solution to a local optimum, and moves to it only if an improvement was made (Moreno Pérez, Marcos Moreno-Vega et al. 2003). VNS is a simple method and requires few parameters, and overall, can provide very good solutions. It has been applied to solve many real practice problems such as location problems, data mining, vehicle routing problems and scheduling problems, including JSSPs. However, in the later search period, VNS may not be able to escape from local optimal regions. (Çaliş and Bulkan 2013).

Beam Search (BS)

BS is a heuristic search method for solving optimization problems. It is an adaptation of the best-first-search method that reduces its memory requirement. In BS only the most promising nodes are expanded and evaluated in the search tree and remaining nodes are cut off permanently, thereby trying to keep the combinatorial nature of the problem in check. One application of BS for solving JSSPs is presented by Sabuncuoglu and Bayiz (1999). The main disadvantage of a BS method is that the optimal solution may not be reached, as many nodes that can lead to an optimal solution are cut off (Zhang 1999).

Bee Inspired Methods

Bee inspired methods such as, the Bee Colony Optimization (BCO) (Teodorovic and Dell'Orco 2005) and Bees Algorithm (BA) (Pham, Ghanbarzadeh et al. 2006) were inspired by bees' behaviour of searching for food sources in nature and then, after they return to the hive, they indicate these sources to the other bees of the colony to identify promising locations. Bee inspired methods have fast convergence and high flexibility, smaller amounts of setting parameters, and memory of best solutions (Abdullah and Abdolrazzagh-Nezhad 2014). Chin Soon, Low et al. (2006) used BCO to solve the JSSP with the aim of minimizing the makespan. They compared the proposed approach with existing approaches such as ACO and TS and found that the performance of the algorithm is comparable to ACO algorithms but is less efficient than TS. However, in the later search period, the bee inspired methods have premature convergence, and sometimes need to apply local search algorithms to meet some requirements (Abdullah and Abdolrazzagh-Nezhad 2014).

Artificial Neural Network (ANN)

ANN is a data-driven modelling tool that is able to capture and represent complex and non-linear input/output relationships. A neural network is composed of a number of layers of processing elements or nodes. These nodes are linked by connections, with each connection having an associated weight. The weight of a connection is a measure of its strength and its sign is indicative of the excitation or inhibition potential. A neural network captures task-relevant knowledge as part of its training procedure. This knowledge is encoded in the architecture or topology of the network, the transfer functions used for nonlinear

mapping and a set of network parameters (weights and biases). There have been several applications of neural networks such as data processing, robotics, computer numerical control and scheduling problems, including JSSPs (Weckman, Ganduri et al. 2008). However, great computational burden and the empirical nature of model development are some key disadvantages of ANN (Tu 1996).

Hybridization Methods

Hybridization methods combine two or more optimization methods to take advantage of their strengths. A wide range of these methods have been proposed to solve JSSPs. For instance a hybridization of GA and TS to overcome the main drawback of these methods and generate more promising results has been proposed by Vilcot and Billaut (2008), however computational time has been shown to increase (Chaudhry and Khan 2015).

In conclusion, most of the previous work on testing the developed methods focused on benchmark problems and there is no work to show that any of these methods outperform each other with regard to all problem aspects (Çaliş and Bulkan 2015, Chaudhry and Khan 2015).

2.5 Multi-objective Optimization and Pareto-optimal Solutions

The classical JSS formulation tends towards optimizing a single objective such as minimizing the makespan, the number of tardy jobs or maximum tardiness. Real manufacturing environments noticeably involve multiple conflicting objectives to satisfy various manufacturing requirements. Despite the fact that

organisations normally look for minimizing the costs and maximizing the utilization of resources, scheduling objectives also include objectives directed towards satisfying customer demand, reducing confusion and improving schedule stability. A good scheduling system would need to capture and balance a considerable range of these objectives. Usually, a single objective optimization system cannot represent all of these complex problem aspects in their complete richness. However, developing a scheduling system that takes into account all objectives would require massive amounts of data and excessive time. An effective and economic scheduling system will consider only the most important objectives of a given environment, disregarding the least important features, and handling those in-between in an aggregate way (Rodammer and White 1988). In general, a single-objective optimization problem can be represented as:

$$\min f(x) \ : \ x \in R^N$$

Subject to $g(x) = 0 \ \ \& \ h(x) \leq 0$

Where f is the objective function, or evaluation function or cost function, x is a vector of design variable $x = \{x_1, x_2, ..., x_N\}$, R^N is the design space, g is an equality constraint, and h is an inequality constraint.

In the case of multi-objective optimization, generally the problem consists of a number of objectives and is associated with a number of equality and inequality constraints. Mathematically, the problem can be written as follows.

$$\min f(x) \ : \ x \in R^N \ \ \& \ \ f \in R^M$$

Where $f = \{f_1, f_2, ..., f_M\}$, and R^M is the objective space.

In the case of simultaneous optimization (maximization or minimization) of two or more conflicting objective functions, there is no single feasible solution that can optimize all objective functions at the same time. In this case, the notation of Pareto optimality needs to be introduced. A solution is called a Pareto optimal or a non-dominated solution if none of the objective functions can be improved in value without worsening at least one other objective value. Formally speaking, a vector $x^* \in R^N$ is Pareto optimal if there is no feasible vector $x \in R^N$ that would improve some objective function, without causing a simultaneous deterioration in at least one other objective function (Caramia and Dell'Olmo 2008, Rangaiah and Bonilla-Petriciolet 2013).

2.5.1 Solving Multi-objective Optimization Problems

In the literature, there are various methods for solving multi-objective optimization problems. These methods can be classified based on three main categories; aggregation approach, population-based non-Pareto approach and Pareto-based approach.

2.5.1.1 Aggregation Approach

The aggregation approach is the simplest approach for solving multi-objective optimization problems. It is easy to apply and does not require high computational procedures to implement. Moreover, all existing evolutionary algorithms can be applied directly with only slight changes (Jin 2003). In the aggregation approach, different objectives are combined into a single objective using a weighting or a goal-based method, thus converting the multi-objective

optimization problem into a single-objective optimization problem (Xiao-Juan, Chao-Yong et al. 2008). The goal programming-based method (Charnes, Cooper et al. 1955, Charnes and Cooper 1961), weighted sum method proposed by Fishburn (1967), goal attainment-based method proposed by Gembicki and Haimes (1975), and ε-constraint method proposed by Chankong and Haimes (1983) all belong to this category (Bandyopadhyay and Saha 2013). The weighted sum method is the most straightforward and commonly applied method. In this method different objectives are merged together in one objective function using some weights $w_v, v = 1,2,...M$ (where M is the number of objectives). The weights represent the relative importance of each objective, thus, to optimize a number of objective functions the following formula is used:

$$\sum_{v=1}^{M} w_v f_v(x)$$

Where $\sum_{v=1}^{M} w_v = 1, w_v > 0, v = 1,...,M$ and $x \in R^N$

The obtained solution of this approach is usually a Pareto-optimum solution. However, a prior knowledge is required to decide the weight for each objective and only one Pareto solution can be achieved from one run of optimization. In real world practice, decision makers may need different alternatives before making a decision (Srinivas and Deb 1994, Jin 2003).

2.5.1.2 Population-based Non-Pareto Approach

In the population-based non-Pareto approach, the optimization is carried out by using a sub-population for each objective, but without the Pareto dominance in the selection mechanism. In each sub-population, the solutions are ranked and selected based on one objective. Vector Evaluated Genetic Algorithm (VEGA)

proposed by Schaffer (1985) is a typical example belonging to this category. In this method, generation is produced by performing appropriate fraction or proportional selection according to each objective. Then the whole population is pooled together to achieve the mating of individuals of different subpopulations through crossover and mutation operators. Non-dominated solutions are specified by observing the population as it evolves but this data is not used by the algorithm itself (Gen and Cheng 2000). The disadvantage of this approach is that a Pareto solution can be disregarded during the evolving process, because the optimal trade-off for all the sub-objectives is usually not the optimal solution for each objective (Xiao-Juan, Chao-Yong et al. 2008).

2.5.1.3 Pareto-based Approach

The Pareto-based approach is currently the most popular way to evolutionary multi objective optimization. In this approach Pareto optimality is applied in the selection mechanism. All the non-dominated solutions are given the same fitness value, while other solutions are given inferior values. Multi-Objective Genetic Algorithm (MOGA) proposed by Fonseca and Fleming (1993), Non-dominated Sorting Genetic Algorithm (NSGA) proposed by Srinivas and Deb (1994), Niched Pareto Genetic Algorithm (NPGA) proposed by Horn, Nafpliotis et al. (1994), Strength Pareto Evolutionary Algorithm (SPEA) proposed by Zitzler and Thiele (1999), Non-dominated Sorting Genetic Algorithm-II (NSGA-II) proposed by Deb, Pratap et al. (2002), Strength Pareto Evolutionary Algorithm (SPEA2) proposed by Zitzler, Laumanns et al. (2002), Multi-Objective Messy Genetic Algorithm (MOMGA) (Van Veldhuizen 1999, Zydallis 2003) and a very recent method named Efficient Non-dominated Sort (ENS) proposed by Xingyi, Ye et al. (2015) all belong to this category. The main disadvantage of

these methods in this category is that they are computationally very expensive (Jin 2003, Xiao-Juan, Chao-Yong et al. 2008).

2.6 Previously Existing Work and Gap of Knowledge

In this section, a survey study of JSSPs based on multi-objective optimization that have previously appeared in the literature is conducted. Here, four main factors for describing the JSSPs are considered to distinguish between these works. These factors are Setup Time (ST), Alternative Machines (AM), Job Release Date (JRD), and Unfixed Processing Time (UPT). Table 2-2 shows the reference and authors for different researches, and the factors considered in their work.

Table 2-2. Previous research in MO-JSSP

Author(s) and reference	Factor of Describing the JSSP			
	ST	AM	JRD	UPT
Sakawa and Mori (1999)	-	-	-	✓
Ponnambalam, Ramkumar et al. (2001)	-	-	-	-
Baykasoğlu, özbakir et al. (2004)	-	✓	-	-
Low, Wu et al. (2005)	✓	-	-	-
Xia and Wu (2005)	-	✓	-	-
Suresh and Mohanasundaram (2006)	-	-	-	-
Lei (2008)	-	-	-	✓
Xing, Chen et al. (2009)	-	✓	-	-
Manikas and Chang (2009)	✓	-	-	-
Zhang, Shao et al. (2009)	-	✓	-	-
Li and Huo (2009)	-	✓	-	-
Huang (2010)	✓	-	-	-
Adibi, Zandieh et al. (2010)	-	-	✓	-
Li, Pan et al. (2010)	-	✓	-	-
Wang, Gao et al. (2010)	-	✓	-	-
Sha and Lin (2010)	-	-	-	-
Moslehi and Mahnam (2011)	-	✓	✓	-
Zheng, Li et al. (2011)	-	✓	-	✓
Kachitvichyanukul and Sitthitham (2011)	-	-	-	-

Table 2-2. Continued

Author(s) and reference	Factor of Describing the JSSP			
	ST	AM	JRD	UPT
Li, Pan et al. (2011)	-	✓	-	-
Tavakkoli-Moghaddam, Azarkish et al. (2011)	✓	-	✓	-
Ramkumar, Tamilarasi et al. (2012)	-	-	-	✓
Wang, Zhou et al. (2012)	-	✓	-	-
Li, Pan et al. (2012)	-	✓	-	-
Li, Pan et al. (2012)	-	✓	-	-
Frutos and Tohmé (2012)	-	-	-	-
Dalfard and Mohammadi (2012)	-	✓	✓	-
Lei (2012)	-	-	-	✓
Rahmati, Zandieh et al. (2012)	-	✓	-	-
Zhang, Gao et al. (2013)	-	-	✓	-
Wang, Wang et al. (2013)	-	✓	-	-
Shahsavari-Pour and Ghasemishabankareh (2013)	-	✓	-	-
Shao, Liu et al. (2013)	-	✓	-	-
Qiu and Lau (2013)	-	-	✓	-
Niu, Ong et al. (2013)	-	-	-	-
Li, Pan et al. (2014)	-	✓	-	-
Gao, Suganthan et al. (2014)	-	✓	-	-
Zhao, Tang et al. (2014)	-	-	✓	-
Su, Mengjie et al. (2014)	-	-	✓	✓
Gao, Suganthan et al. (2014)	-	✓	-	-
Jia and Hu (2014)	-	✓	-	-
Hosseinabadi, Siar et al. (2014)	-	✓	✓	-
Xue, Zhang et al. (2014)	-	✓	-	-
Karthikeyan, Asokan et al. (2014)	-	✓	-	-
Pérez and Raupp (2014)	-	✓	-	-
Yang and Gu (2014)	-	-	-	-
Shen and Yao (2015)	-	✓	✓	-
Huang and Süer (2015)	-	-	-	✓
Shivasankaran, Kumar et al. (2015)	-	✓	✓	-
Zhao, Gao et al. (2015)	-	✓	-	-
Singh, Singh et al. (2015)	-	✓	-	-
Zhang and Chiong (2016)	-	-	-	✓
Kaplanoğlu (2016)	-	✓	-	-

The fifty-three reviewed literatures have been categorized based on the four main factors for describing JSSPs with multi-objective optimization, as shown in Table 2-2. Figure 2-3 (a) shows the percentage of MO-JSSPs with alternative machines, setup time, release date and unfixed processing time. The results show that thirty-one journal articles (57%) considered alternative machines, eleven journal articles (20%) considered job release date, eight journal articles (15%) considered unfixed processing time and only four journal articles (8%) considered setup times. As shown in Figure 2-3 (b) none of the articles incorporated all four factors or even three factors together, as thirty-eight articles (72%) considered only one factor, mostly alternative machines, eight articles (15%) considered two factors and seven articles (13%) did not consider any of these four factors.

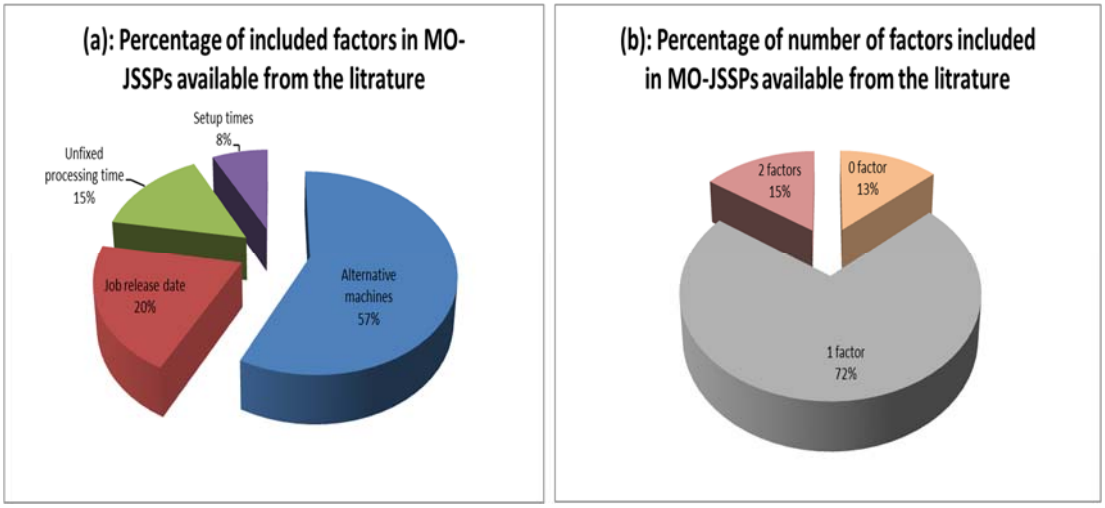

Figure 2-3. Percentage of included factors in MO-JSSPs from the available literature

After a comprehensive study made on the existing MO-JSSPs literature, a number of limitations have been found:

1. Although the JSSP is a popular topic, the number of researches in MO-JSSPs is still limited and there is still work to be done in this area.

2. At most, only two factors have been incorporated in one scheduling system, which implies that, there is a need to incorporate more factors to reflect real manufacturing practice.
3. Although JSSPs with job sequence dependent setup time are more complicated, there have still not been any studies made on MO-JSSPs with machine dependent setup time and job sequence independent setup time.

As a result, the current research considers MO-JSSPs with alternative machines, job and machine release time and machine dependent setup time with job sequence independent setup time. The effect of processing time uncertainty on the optimal makespan for classical JSSPs is also studied.

2.7 Summary

In real job shop manufacturing practice, there are a number of factors for describing the scheduling problem. Disregarding some of these factors can have a major impact on the scheduling system reliability. In this chapter, the most important factors for describing the JSSPs were illustrated. Four main factors were then used to distinguish between different works that have been undertaken for solving MO-JSSPs. These factors are setup time, alternative machines, job release date and unfixed processing time. Based on these factors, the knowledge gaps for solving MO-JSSPs in the literature were identified, which provide the evidence to support the contributions of this thesis. The chapter also presented the most known methods for solving JSSPs. For JSSPs, iterative methods were considered to be suitable application tools and are very promising. However, none of these methods outperforms each other

with regard to all problem aspects. Some basic concepts about multi-objective optimization and three popular approaches for multi-objective optimization were also discussed. These three approaches are; aggregation approach, population-based non-Pareto approach and Pareto-based approach. Currently the most popular approach to evolutionary multi objective optimization is Pareto-based approach. In the next chapter, the problem description, issues and challenges related to solution methodologies as well as mathematical formulation of the problem are presented.

Chapter 3. Problem Description, Complexity and Formulation

3.1 Introduction

In this chapter, JSSPs with multiple objectives are discussed in detail. Issues and challenges related to solution methodologies are also addressed. Finally, mathematical formulations for JSSPs with considered factors are presented.

3.2 Problem Description and Complexity

JSS is common practice in the manufacturing environment for many small and medium-sized companies. The term describes the work flow of the products on the shop floor. In such an environment the process characteristics on the machines, such as operations, setup times, processing times, and routings often differ from one product to another. Because of these variations, JSSPs are extremely difficult to solve to optimality, both in practice and in theory, and have been considered as a member of a large class of intractable numerical problems known as NP-hard. For instance, in the classical JSSP, where n jobs need to be processed by m machines, the possible number of sequences is $(n!)^m$. Accordingly, a 20×5 problem can have at most 8.5×10^{91} possible solutions, which is a very large number and requires, even for the fastest computer, an enormous computational time to evaluate all the possibilities. Therefore, JSSPs are extremely difficult to solve to optimality (SMITH 1966, Jain and Meeran 1999).

Generally, the JSSP consists of n jobs to be processed by m machines, while minimising some functions of completion time and due date of the jobs, subject to some technological rules and constraints. In what follows, common features for all JSSPs are presented followed by various forms of JSSPs that exist in the literature.

(i) Every job has a predetermined sequence of operations based on its technological requirements.

(ii) Each machine can perform only one operation of any job at a time and it becomes available to perform other operations once it has completed the currently assigned operation (resources constraints).

(iii) An operation of a job can be performed by only one machine at a time.

(iv) Tasks of the same job cannot be processed concurrently and cannot be started until the precedence operation is finished.

(v) The jobs are independent; that is, there are no precedence constraints among the jobs and they can be operated in any sequence.

In its simplest form, which is known as the classical JSSP, the assumption is made that; each job must be processed on each machine only once. There are no alternate machines for each job's operations, i.e., there is only one machine for each type of job operation. Processing time of an operation for a job varies but is known in advance. All jobs and machines are available at time zero. The setup times and transportation times are either negligible or are included in the processing times and no pre-emption is allowed. However, in real manufacturing environment, machine preparation can be required before starting the task and operations can be processed on alternative machines. Moreover, time uncertainty, dynamic job arrival, and multi-objectives optimization, etc. are all features of real manufacturing practice and need to be considered in the scheduling model in order to achieve a more accurate and reliable scheduling system. Still, when all or some of these factors are

incorporated in the scheduling system, the problem complexity will also increase.

The motivation of this research is to develop a scheduling system that considers many of these factors, to reflect real world job shop manufacturing practice and acquire a more reliable and accurate scheduling system that benefits both practitioners and researchers. In this research the following factors are considered;

- Deterministic dynamic job arrival, in which jobs are available for their process at various times.
- Deterministic dynamic machine release date, in which not all machines are available at time zero.
- Alternative machines, in which some or all of the job's operations can be executed on a number of alternative machines
- Machine setup times with job sequence independed setup times.
- A job does not necessarily visit every machine and it may visit a machine more than once.
- Priorities among jobs, expressed by weights, and
- Scheduling based on multi-objectives

The effect of the processing time uncertainty on the optimal makespan is also studied. The following sections provide a summary of these factors and the problem complexity.

3.2.1 Deterministic Dynamic Release Date

The release date of a job or machine refers to the first point in time where that job or machine is available. The job release date is helpful when modelling a job's earliest starting time due to the order placement time or the availability of raw materials. Likewise, the machine release date is helpful when machines have unavailability windows due to maintenance or breakdowns. In this research, deterministic release times of both jobs and machines are considered. Deterministic release times mean that; the time for a job or machine to be released is known in advance.

3.2.2 Alternative Machines

JSSPs with alternative machines allow some or all of the job's operations to be executed on a number of alternative machines. These machines can be identical or non-identical. The alternative machines or alternative routing is beneficial when the capacity problem is an important issue. Although incorporation of alternative machines makes the scheduling system more complicated, it significantly increases the performance of idle machines and can reduce the pressure on other overloaded machines. The benefit of using alternative machines are that; the total manufacturing time is shortened, the work-in-process is reduced, lead time is reduced, and overall machine utilization is improved (Wilhelm and Hyun-Myung 1985, Chaudhry 2012). In this research, the case of alternative machine tools is allowed for some operations (partial flexible system). Since the system needs further decisions of machine allocation throughout scheduling, it is considered to be much more complex than the classical JSSP.

3.2.2.1 Complexity of Alternative Machines

To show the complexity of alternative machines, let's assume that a job $(J1)$ has three different machining operations, which are turning, milling, and drilling, that have to be processed on machines $M1,1$, $M2,1$, $M3,1$ successively as shown in Figure 3-1.

Figure 3-1. Single Processing Route for J1

It can be seen that only one possible route is available to finish $J1$ ($M1,1 \rightarrow M2,1 \rightarrow M3,1$). If one more machine $M2,2$ is added to $M21$ as an alternative machine for the milling operation, and $M3,2$ is added to $M3,1$ as an alternative machine for the drilling operation, the possible number of routes will increase from 1 to 4 as shown in Figure 3-2.

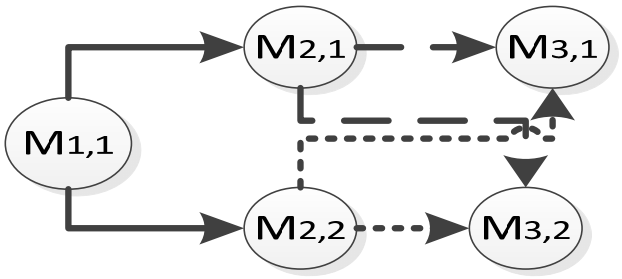

Figure 3-2. Alternative Processing Routes for J1

The possible alternative routes to process $J1$ are now 4 as follows:

$$M1,1 \rightarrow M2,1 \rightarrow M3,1$$

$$M1,1 \rightarrow M2,1 \rightarrow M3,2$$

$$M1,1 \rightarrow M2,2 \rightarrow M3,1$$

$$M1,1 \rightarrow M2,2 \rightarrow M3,2$$

In general, the total number of alternative routes AR_i for job i that has a total number of operations J_i, and each operation O_{ij} that can be executed by one machine or set of alternative machines AM_{ij}, is equal to multiplying AM_{ij} of all O_{ij}

$$AR_i = AM_{i1} \times AM_{i2} \times \ldots \times AM_{iJ_i}$$

For instance, in the first case (single route), only one machine is capable of processing each operation. Consequently, the number of alternative routes in the first case is equal to one (1*1*1=1). In the second case, there is only one machine for operation 1, two machines capable to execute operation 2, and two machines capable to execute operation 3. Consequently, the number of alternative routes in the second case is equal to four (1*2*2=4).

Now, let's consider another job $J2$ to see the influence of alternative machines on the overall scheduling decision. Similarly as in $J1$, $J2$ has 3 different operations but with a different order on the machines, so that milling on $M2,1$ or $M2,2$ is first, drilling on $M3,1$ or $M3,2$ is second and turning on $M1,1$ is last, as shown in Figure 3-3.

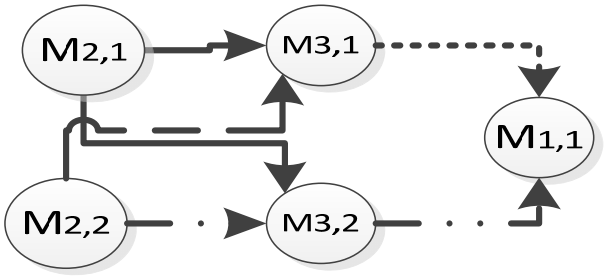

Figure 3-3. Alternative Processing Routes for J2

The number of possible alternative routes for processing $J2$ is also 4 and as follows:

$$M2,1 \rightarrow M3,1 \rightarrow M1,1$$

$$M2,1 \rightarrow M3,2 \rightarrow M1,1$$

$$M2,2 \rightarrow M3,1 \rightarrow M1,1$$

$$M2,2 \rightarrow M3,2 \rightarrow M1,1$$

In its simplest case, the possible sequences for scheduling n jobs on m machines are $(n!)^m$. By considering the alternative machines, the number of possible sequences will increase dramatically, which can already be large, even for some small size problems. For instance, scheduling $J1$ and $J2$ on 3 different machines will result in 8 different sequences. Considering the alternative machines for processing $J1$ and $J2$ will result in 4*4=16 different matrix for the machines, as shown in Figure 3-4, this will increase the possible number of sequences to 16*8= 128.

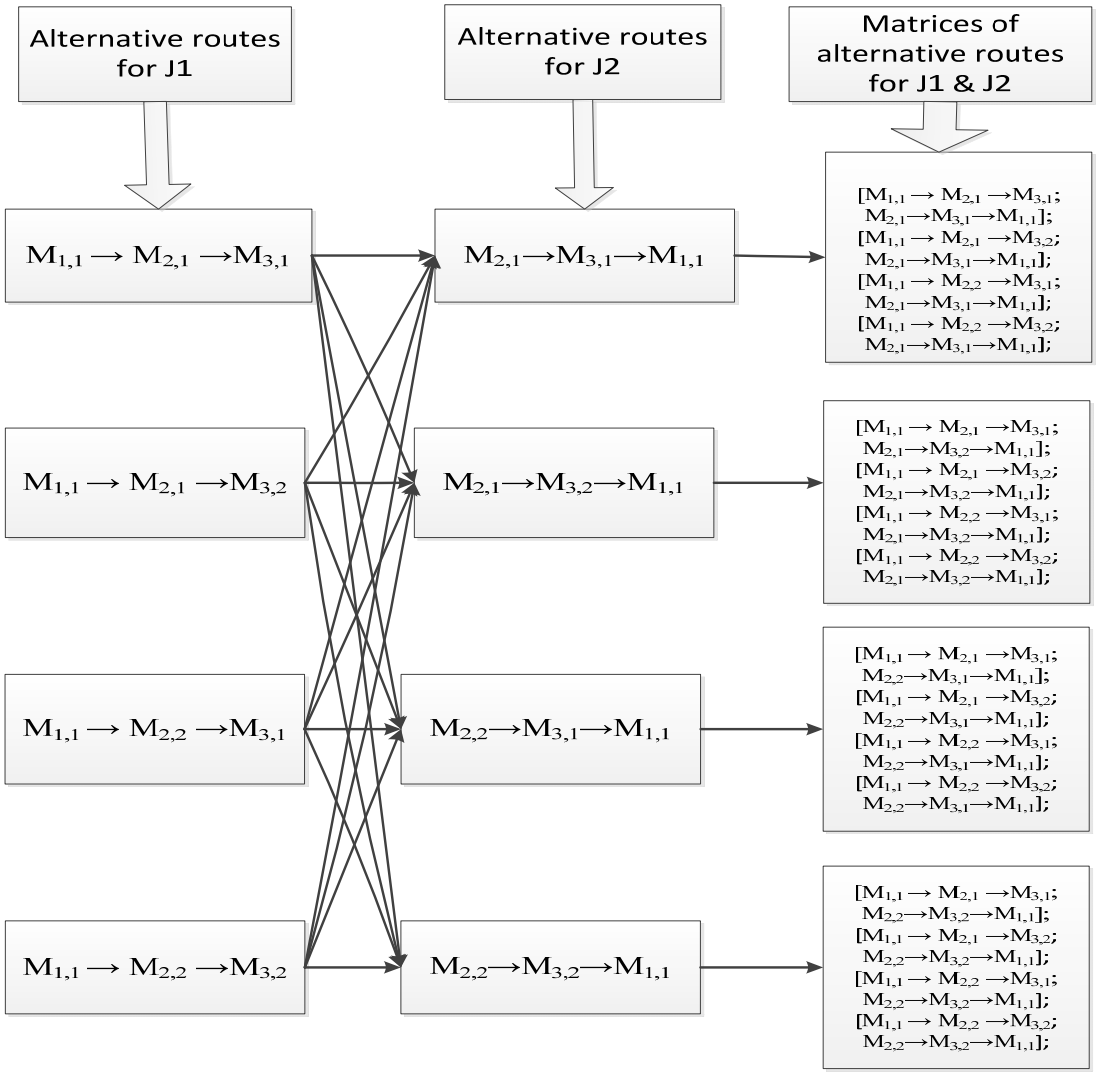

Figure 3-4. Different Machine Matrices for J1 & J2

3.2.3 Machine Setup Time

In this research the JSSP with machine dependent, job sequence independent, and item availability setup times is considered. For this type of problem the setup takes place on a machine when the job has to be processed on that machine first and when a job has to be processed after a job of another group (Sotskov, Tautenhahn et al. 1996). In this research we distinguish the presented work from Sotskov, Tautenhahn et al. (1996), so that the group is made based on the machines and operations, i.e. a job can belong to one group on one machine and to a different group on another machine. By doing so, the job-

based setup group and operation-based setup group can be solved. Furthermore, different batches can be formed based on these groups from the resulting schedule. An anticipatory setup time in which the setup can be started before the corresponding job becomes available on the machine is considered. The application of this type of scheduling problem can be found in many small and medium-size manufacturing companies in which different tools are required to process operations of different jobs on the same machine, such as different milling cutters in a milling machine or different cutting tools in a turning machine. Therefore, developing a scheduling system that considers the setup time will bring benefits to such manufacturing companies. However, the decision complexity will increase, as the decision-making will also take into account the machine setup time.

3.2.4 Jobs with Various Lengths and Recirculation

In a real job shop manufacturing environment, not all jobs have to visit all machines (jobs with different numbers of operations), while some jobs need to visit some machines more than once (scheduling with recirculation). This decision depends on the job operational requirements and the scheduling system must consider all these technological requirements. Therefore, in this research, a scheduling system in which some operations can be repeated and some can be missed is considered to reflect to the real job shop manufacturing environment.

3.2.5 Priorities among jobs expressed by weights

The weight w_i of job i is a priority factor, denoting the importance of job i relative to the other job in the system. It may represent the cost of keeping the

job in the system. The weight can be a holding or inventory cost, or it can be the amount of value already added to the job (Pinedo and CHAO 1999). In this research, minimizing the weighted number of tardy jobs was considered. Different jobs can have different weights with respect to the importance of different customers. These weights are usually decided based on the penalty associated to each job of being tardy.

3.2.6 Multi-Objective Optimization

Since the study of JSSPs started, a large number of researches have been published. The majority of these publications considered only a single objective (most frequently the makespan). However, in real manufacturing practice, many industries have to consider multiple objectives to satisfy the overall performance of the system, with different conflicting objectives. Therefore, a good and flexible scheduling system must take into account multiple objectives to satisfy different aspects of the decision makers. In this research, a scheduling system for solving JSSPs with multi-objective optimization was developed. Three practical performance measures: the maximum completion time or the makespan, the weighted number of tardy jobs and the maximum tardiness, were considered simultaneously in this work.

3.2.6.1 Complexity of Multi Objective Optimization

Solving MO-JSSPs is considered to be more complex than solving JSSPs with a single objective because the objectives are often conflicting or even contradictory. The appearance of conflicting objectives of the feasible solutions is a common difficulty for JSSPs with multi-objective optimization that does not allow a simultaneous optimal solution for all objectives (Srinivas and Deb 1994).

Hence, optimizing one objective usually leads to deterioration of one or more other objectives. For instance, to increase the throughput of the system, the input rate of products has to be increased, but that will cause higher work in progress (WIP) inventory. Therefore, it is necessary to find multiple trade-off solutions between different objectives. Due to the great difficulty but also necessity, more attention has been given lately to MO-JSSPs. Generally, there are two main approaches to handle the multi-objective; firstly by aggregating them to one objective to find the optimal weighted-sum solution and secondly by optimizing them simultaneously to find a set of non-dominated solutions. If the objectives are combined into a single objective using weights, the difficulty is to assign weights to each objective, since it requires knowledge and the order of importance for each objective in advance, which may be difficult in today's unstable market condition. If all objectives are optimized simultaneously, the problem is to develop an effective search algorithm for some further steps and the significant increase of time and space complexity (Lei 2008).

Simultaneous optimization of the objectives means identifying the Pareto front or non-dominated solutions. In a simple approach, to identify the first front of non-dominated solutions in a population of size N, each solution needs to be compared with all other solutions in the population to decide if it is dominated by some other solutions or not. That means each solution requires $O(MN)$ comparisons, where M is the number of objectives. When this process is continued to find all members of the first non-dominated level in the population, the total complexity is $O(MN^2)$ (Deb, Pratap et al. 2002). Usually a good selection approach requires identifying all Pareto fronts and finding solutions in the other fronts will also require comparing each solution with those solutions

not included in the previous fronts. This technique, however, can be computationally expensive and requires a large storage space, especially when the number of individuals in the population is large. For instance, NSGA has a time complexity of $O(MN^3)$ and space complexity of $O(N)$. This time complexity has been reduced in NSGAII to $O(MN^2)$ but storage space has increased to $O(N^2)$. To avoid many unnecessary comparisons, Xingyi, Ye et al. (2015) proposed a new method called Efficient Non-dominated Sort (ENS). Instead of comparing each solution with all other solutions in the population, in ENS, a solution to be assigned to a front, will only be compared with solutions that have already been assigned to a front, in that way many unnecessary comparisons can be avoided. ENS using Sequential Search Strategy (ENS-SS) has a time complexity of $O(MN\sqrt{N})$ in the best case and $O(MN^2)$ in the worst case and has a space complexity of $O(1)$. Since ENS-SS has less time and space complexity, it has been employed in this research to determine the front to which each solution belongs, but instead of starting with the first front, the proposed algorithm starts the comparison with the last created front so far, and this has been termed as the Backward Pass Sequential Strategy (BPSS). Efficient Non-dominated Sorting using the Backward Pass Sequential Strategy (ENS-BPSS) can reduce the number of comparisons needed for N solutions with M objectives to $O(M(N-1))$ when there are fronts and there exists only one solution in each front. In this case, and because a solution can never be dominated by any succeeding solution in the sorted population, each solution in the sorted population will only be compared with the directly preceding solution.

3.2.7 Effect of Time Uncertainty on the Optimal Solution

In deterministic JSSPs, the duration of each task in each job is known in advance and is given as a single value. Therefore, the start and end date for each task can be calculated in a single pass calculation. However, in a real manufacturing system, the duration of the task is often uncertain and it can be defined as a variable value, where each value has a different occurring probability. In this case, the number of different combinations of durations for forming the schedule can be very large. Studying each combination individually can be a massive task. Therefore, in this research, two different scenarios were considered to study the effect of time uncertainty on the optimal solution for a single objective (the makespan). In the first scenario, the ratios of change in the processing time for all operations were considered to be the same. In the second scenario, the ratios of change were varied from one operation to another.

3.3 Mathematical Formulation

To solve an optimization problem, it is useful first to formulate the problem in a manner reflecting the situation being modelled. In this section, the mathematical formulas to describe the problem are presented. The notation, parameters, decision variables and objective function are given below:

3.3.1 Indices

n: Total number of jobs.

m: Total number of machines.

i: Job index $(i = 1,2,...,n).$

k: Machine index $(k = 1,2,...,m).$

O_i: Total number of operations required to complete job i

j: Operation index $(j = 1,2,...,O_i).$

O_{ij}: Operation number j of job i $(O_{i1}, O_{i2}, ..., O_i)$

AM_{ij}: The set of machines that can process operation j of job i.

3.3.2 Parameters

p_{ijk}: Processing time of operation O_{ij} on machine k

r_i: The release date of job i in the job shop

rt_k: The release time for machine k in the job shop

w_i: The importance weight of job i

d_i: Due date for job i

G_{ijk}: Setup group for operation j of job i on machine k

St_{Gk}: Setup time for a set of operations that belongs to setup group G on machine k.

3.3.3 Decision Variables

S_{ijk}: The start time for operation j of job i on machine k.

C_{ijk}: The completion time for operation j of job i on machine k.

C_i: The completion time for the last operation of job i.

$X_{ijk} = 1$: If operation j of job i is processed on machine k, otherwise $X_{ijk} = 0$.

$Y_{ijaqk} = 1$: If operation j of job i precedes operation q of job a on machine k, otherwise $Y_{ijaqk} = 0$.

$Z_{ijk} = 1$: If operation j of job i is processed on machine k as the first, otherwise $Z_{ijk} = 0$.

$Q_{jqk} = 0$: If two different operations j and q from the same setup group G are processed consecutively on machine k, otherwise $Q_{jqk} = 1$.

3.3.4 Constraints

3.3.4.1 Release time constraints:

Release time constraints ensure that a job cannot be started before its arrival date and before the machine release time.

$$S_{ijk} \geq \max(r_i, rt_k)$$

3.3.4.2 Initial Setup Constraint:

Initial setup constraint ensures that the first job on machine k cannot be processed until the machine setup has been completed.

$$(S_{ijk} - St_{Gk} - \max(r_i, rt_k)) * Z_{ijk} \geq 0$$

3.3.4.3 Operation Precedence Constraint:

Precedence constraint ensures that Operation j of job i cannot be started before its preceding operation is completed

$$S_{ij} \geq C_{i,j-1}$$

3.3.4.4 Processing Time Requirement

Processing time requirement ensures that the difference between the start time and the completion time of operation j on machine k is equal to the required processing time of operation j on machine k.

$$(C_{ijk} - S_{ijk}) * X_{ijk} = p_{ijk}$$

3.3.4.5 Capacity and Setup Requirement Constraints

These constraints ensure that two different operations cannot be processed at the same time on the same machine and that machine setup must take place whenever an operation has to be processed after an operation of another group on that machine.

$$(C_{pqk} - C_{ijk} - p_{pqk}) * Y_{ijaqk} + St_{Gk} * Q_{jpk} \geq 0$$

$$(C_{ijk} - C_{pqk} - p_{ijk}) * (1 - Y_{ijaqk}) + St_{Gk} * Q_{jqk} \geq 0$$

3.3.4.6 Operational Constraint

This constraint ensures that every job is processed by only one machine in each stage

$$\sum_{k=1}^{m} X_{ijk} = 1 \quad \forall i, j$$

3.3.5 Objective Function

The objective of the addressed JSSP in this research is to minimise simultaneously, three practical performance measures;– the maximum completion time or the makespan, the maximum tardiness and the weighted number of tardy jobs.

Makespan: The first objective $f_1(x)$ is to minimize the makespan $C_{\max}$ or the maximum completion time of the last job leave the system. The makespan is important when the number of jobs is finite. A short makspan aims to achieve high equipment and resource utilization by getting jobs out of the job shop quickly. Therefore, the makespan objective was used in this research to increase equipment and resource utilization.

$$f_1(x) = C_{\max} = \max(C_1, ..., C_n)$$

Maximum tardiness: The second objective $f_2(x)$ is to minimize the longest tardiness $T_{\max}$ of all jobs in the system. In scheduling problems, job tardiness is an important issue because it may cause customer dissatisfaction and may impose additional penalty cost. Specially, when the penalties go up exponentially, maximum tardiness will be of great importance. Therefore, maximum tardiness was considered in this research in order to avoid high penalty cost.

$$f_2(x) = T_{\max} = \max(T_1, T_2, ..., T_n)$$

Weighted number of tardy jobs: The third objective $f_2(x)$ is to minimize the weighted number of tardy jobs $\sum_{i=1}^{n} w_i U_i$. In a real manufacturing environment, different weights are assigned to different customers, based on the importance of different customers and the penalty associated to tardy delivery. Therefore, in this research various weights were assigned to different jobs to show the importance of different customers.

$$f_3(x) = \sum_{i=1}^{n} w_i U_i \ : U_i = \begin{cases} 1; & if \quad C_i > d_i \\ 0; otherwise \end{cases}$$

Thus the multi-objective problem can be formulated as:

$$\min \ \ F = [f_1(x), f_2(x), f_3(x)]$$

Note that, any of these three objectives can be replaced with any other due date or completion time related objectives.

3.4 Summary

JSSP is the most generalized and complex scheduling problem and is considered to be NP-hard. The problem complexity can be further increased by including different factors for describing real world JSSPs. However, in order to achieve a more reliable scheduling system, these factors need to be included in the scheduling model. In this chapter, the JSSP including the factors that are to be included in the proposed system, were described. The proposed system is one of the main contributions of this thesis, since it incorporates the release date, alternative machines and setup time in one model. The chapter also presented the problem complexity and mathematical formulations of the problems. In the next chapter a proposed system for solving JSSPs, including the aforementioned factors is introduced.

Chapter 4. Research Methodology

4.1 Introduction

In this chapter, the proposed system for solving JSSPs, including the factors that have been mentioned in chapter 3, is introduced. The proposed system in this research consists of two methods; a Genetic Algorithm (GA) and an Efficient Nondominated Sort using the Backward Pass Sequential Strategy (ENS-BPSS), Figure 4-1 depicts the proposed system. In the following sections, a brief introduction to GA in solving JSSPs is provided, followed by a detailed description of the proposed scheduling system design for solving MO-JSSPs with job release date, setup time and alternative machines.

Figure 4-1. Proposed JSS System

4.2 An Introduction to Genetic Algorithms in Job shop Scheduling Problems

GA is a very popular method that was invented by Holland (1975), and belongs to the class of evolutionary algorithms. It is a directed random search technique based on the mechanics of natural selection and Darwin's main principle: survival of the fittest (Darwin 1859). GA has been applied to many optimization problems in the last few decades and has proven to give excellent results for complex applications (1988, Shah and Kusiak 2004).

The first step before employing GA operators is to represent the problem in a suitable way. A fitness function is also required to give each solution a figure of

merit. For reproduction, parents must be selected and recombined to generate offspring in each run. Figure 4-2 represents the general process of the GA. More details for these aspects are explained in the following sections.

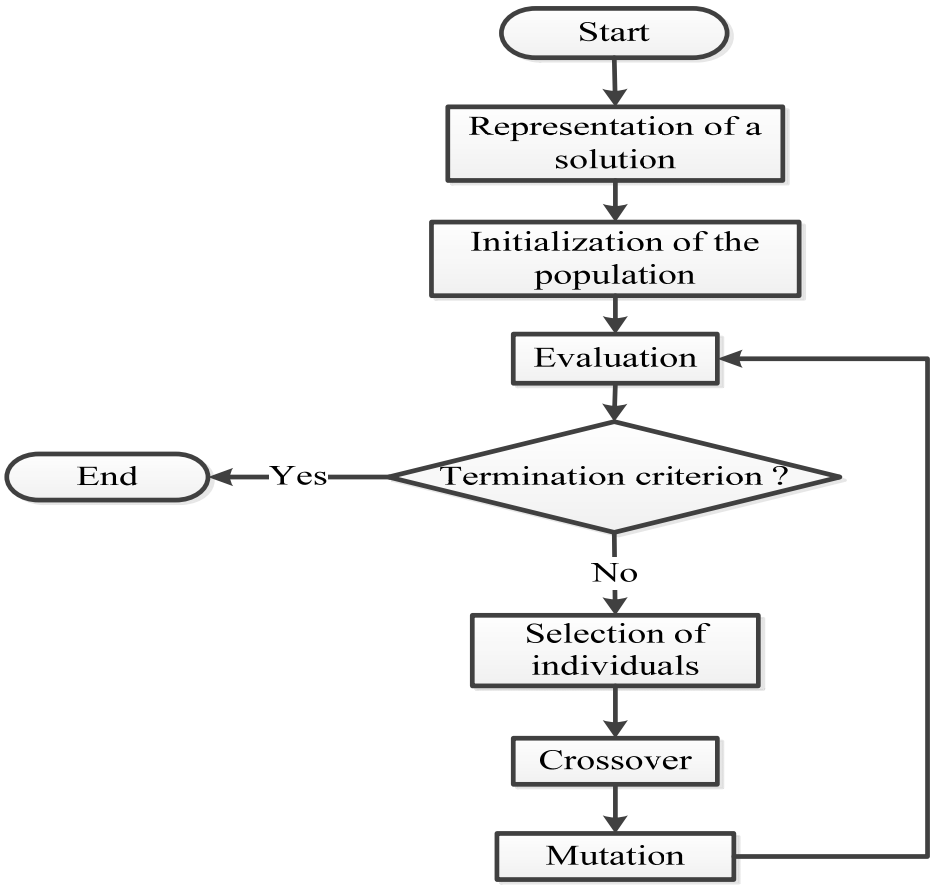

Figure 4-2. General Layout of GA

4.2.1 Representation

The method of representing the problem has a major impact on the performance of the GA. Different representation schemes can cause different performance in terms of accuracy and computational time (Pham and Karaboga 1998). In the past, a number of schemes have been proposed to represent JSSPs in GA. Generally, there are two main based-approaches for encoding that have been used as shown in Figure 4-3.

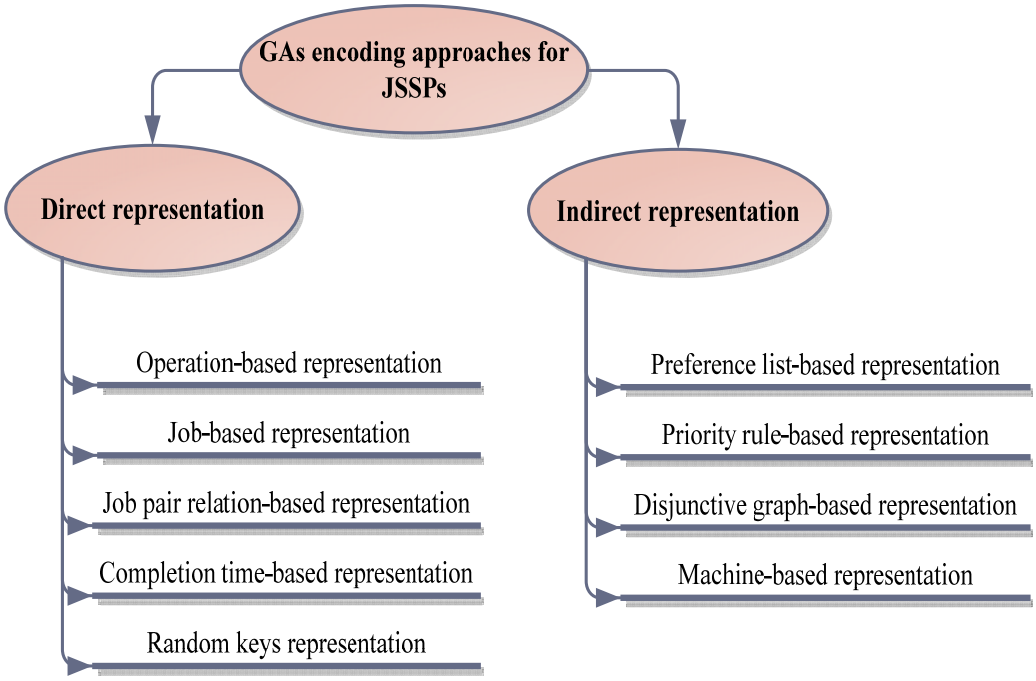

Figure 4-3. Classification of JSSPs Representation in GA

In the first approach, the solution can be encoded directly into a chromosome, and then GA operators can be utilized to discover a better solution by evolving these chromosomes. Operation-based representation (Fang, Ross et al. 1993, Gen, Tsujimura et al. 1994), job pair relation-based representation (Nakano and Yamada 1991), completion time-based representation (Yamada and Nakano 1992), and random keys representation (Bean 1994, Norman 1995) belong to the direct approach. In the second approach, the job assignment is encoded into a chromosome using a sequence of priority dispatching rules, and then the GA operators can be utilized to discover a better solution using these rules, through evolving these chromosomes (Cheng, Gen et al. 1996). Preference list-based representation (Davis 1985, Kobayashi, Ono et al. 1995), priority rule-based representation (Dorndorf and Pesch 1995), disjunctive graph-based representation (Tamaki and Nishikawa 1992), and machine-based representation (Dorndorf and Pesch 1995) belong to the indirect representation.

In this research, a modified version of operation based representation is proposed. This method applies the same procedures as in an operation based representation but it represents the solution in a matrix form, which can preserve features of the parent after the crossover operator without repairing the solution.

4.2.2 Initial Population

Once the solution has been represented in the GA, the next step is to create a number of possible solutions. These solutions can be created randomly or by using some heuristic rules or prior knowledge. The randomly created solutions method is preferred when no prior knowledge exists or for evaluating the performance of an algorithm. While in the second method, a prior knowledge of the given optimization problem can be used to converge to an optimal solution in less time than in the first method (Pham and Karaboga 1998).

4.2.3 Fitness Evaluation Function

The GA evaluates the quality of solutions based on the information produced by the fitness evaluation function. This unit works as an interface between the GA and the optimisation problem. It can be simple or complex depending on the optimization problem that needs to be tackled (Pham and Karaboga 1998).

4.2.4 Genetic Operators

Genetic operators are employed in GA to guide the search towards an optimal solution of a given problem. Selection, crossover and mutation are three common genetic operators which describe the algorithm. Sometimes another reproduction operator called inversion is applied. Crossover and mutation operators can work independently from each other, so it is not obligatory to

employ both of them in GA (Pham and Karaboga 1998). In what follows, a brief description for selection, crossover and mutation is presented:-

4.2.4.1 Selection of Individuals

GA uses a selection mechanism to select individuals from the population and insert them into a mating pool. The aim is to create the basis of the next generation from the current generation. Usually the individuals with better fitness are more likely to be selected for mating and reproduction. Hence the selection procedure determines which of the individuals in the current generation can be chosen to reproduce new individuals or solutions for the next generation, in the hope that the next generation will have individuals with better fitness. There are many methods to select individuals with greater fitness, for instance Roulette Wheel Selection (RWS), Elitism Selection (ES), Rank Selection (RS), Stochastic Universal Sampling (SUS), and Tournament Selection (TRS). Amongst all these methods, RWS and TRS are very common selection approaches in JSSPs. In RWS, the probability of selection is proportional to an individual's fitness. Thus the probability $Ps(s)$ of selecting the solution or individual $s-th$ with fitness $FIT(s)$ is given as:

$$Ps(s) = \frac{FIT(s)}{\sum_{u=1}^{N} FIT(u)}.$$

In TRS a number of individuals are selected from the population at random. These individuals are compared with each other and the best one is selected to be the parent (Pham and Karaboga 1998, Werner 2011).

4.2.4.2 Crossover

A crossover operator recombines the information from selected solutions in order to generate new solutions, which are hopefully better than the selected ones. Normally this operator has a probability rate which is referred to as the Probability of crossover (P_C) and gives the possibility of applying a crossover when producing an offspring from a number of selected parents. In the literature there are many types of crossover such as n-point crossover (most commonly one-point or two-point crossover) and uniform crossover (Pham and Karaboga 1998, Werner 2011).

1. ***One-point Crossover:*** One cut point is determined for producing two new strings; the genes of the two parents after the cut point are then swapped (Werner 2011). Figure 4-4 shows one point crossover for operation based representation.

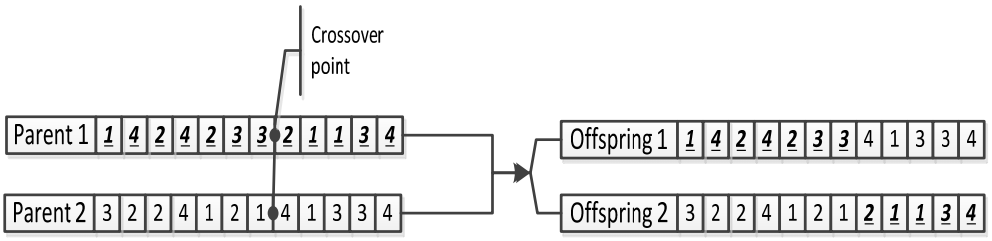

Figure 4-4. One-point Crossover

2. ***Two-point Crossover:*** Two cut points are determined; the genes of the two parents between the first and second cut points are swapped to generate two new strings as shown in figure Figure 4-5 (Werner 2011).

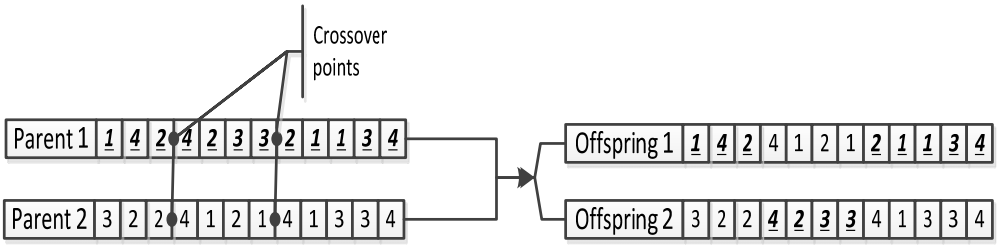

Figure 4-5. Two-point Crossover

3. ***Uniform Crossover:*** A string with a decimal number between 0 and 1 or a bit mask of the numbers 0 is 1 are randomly generated, and for producing the offspring any two corresponding genes from two randomly selected parents are exchanged if the corresponding component of the generated decimal number is more than or equal to P_C, or the bit mask is equal to 1. Otherwise these genes do not exchange with each other as shown in Figure 4-6 (Werner 2011).

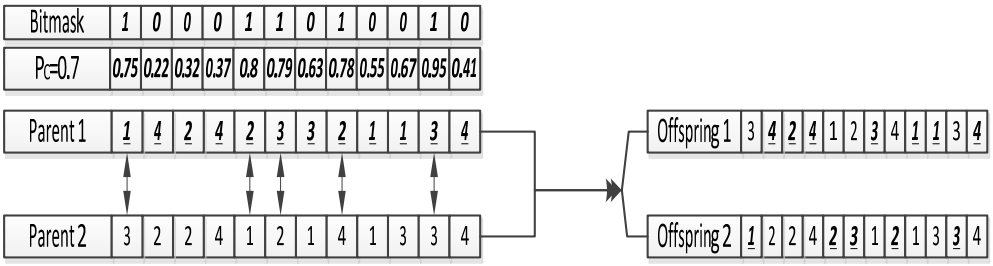

Figure 4-6. Uniform crossover

However, applying these crossover operators to individuals often produces infeasible offspring. To overcome this problem, several operators have been proposed to repair infeasible offspring so that all offspring can lead to feasible individuals. Table 4-1 shows the most commonly used crossover operators for JSSPs (Werner 2011).

Table 4-1. Types of Crossover Operators for JSSPs

Crossover Operator	**Authors**
Linear Order Crossover (LOX)	Wang (1984)
Partially Mapped Crossover (PMX)	Goldberg and Robert Lingle (1985)
Order Crossover (OX)	Davis (1985)
Cycle Crossover (CX)	Oliver, Smith et al. (1987)
Order-Based Crossover (OBX)	Syswerda (1990)
Position-Based Crossover (PBX)	Syswerda (1990)
Partial Schedule Exchange Crossover	Gen, Tsujimura et al. (1994)
Subsequence Exchange Crossover (SXX)	Kobayashi, Ono et al. (1995)
Job-based Order Crossover (JOX)	Ono, Yamamura et al. (1996)

4.2.4.3 Mutation

Mutation usually works with a single chromosome to keep diversity in a population and to prevent the GA from trapping in local optima. It alters one or more gene values in a chromosome from their initial position. Generally there are three common types of mutation operator; shift mutation insertion, pairwise interchange mutation and inversion mutation (Werner 2011).

1. ***Shift Mutation (Insertion Neighbourhood):*** One job is randomly selected from an individual and then shifted to a different position. All remaining jobs between these two positions will be pushed one position towards the selected job position as shown in Figure 4-7 (Werner 2011).

Figure 4-7. Shift Mutation

2. ***Pairwise Interchange Mutation (Swap Neighbourhood):*** Two jobs are randomly selected from an individual and then swap their positions (Werner 2011). Figure 4-8 shows pairwise interchange mutation.

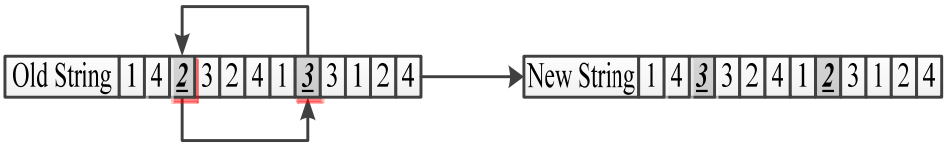

Figure 4-8. Pairwise Interchange Mutation

3. ***Inversion Mutation:*** Two points are randomly selected from an individual and then the jobs in that segment are reinserted in the reverse order as shown in Figure 4-9 (Werner 2011).

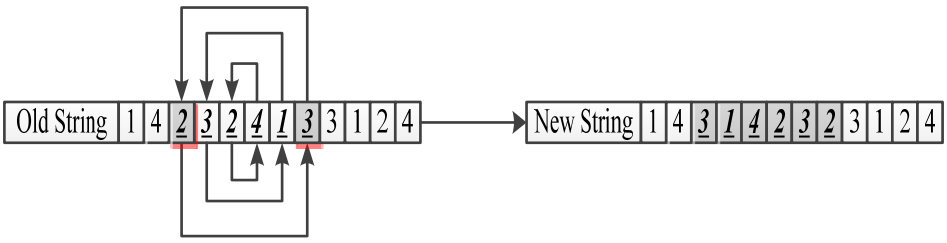

Figure 4-9. Inversion Mutation

4.2.5 Control Parameters

The number of individuals in the population, which is known as population size, and the rate of crossover and mutation are very important control parameters for the GA. The probability of convergence to a global optimal solution is greater when using a large population size, rather than using a small population size. However, the computation time per iteration increases when the number of individuals in the population increases. If the crossover rate is low, it can reduce the speed of convergence to the global optimal solution area. On the other hand, too high a crossover rate might lead to saturation around one solution. Mutation rate can cause instability if it is too high, but it is usually very hard for a GA to find a global optimal solution with too low a mutation rate (Pham and Karaboga 1998).

After a general introduction of GA has been given, the proposed system for solving JSSPs that was introduced in chapter 3 is presented in the following sections.

4.3 Proposed System

This section presents the proposed system for solving MO-JSSPs with the considered factors from chapter 3. In the proposed system, GA and ENS-BPSS were combined together to solve the MO-JSSP. In the beginning, the

representation method of the chromosome was presented, and then generation of the initial population and design of the fitness evaluation function was introduced. The mechanisms of GA operators (selection, crossover and mutation) were also explained. Figure 4-10 shows the layout of the proposed System for solving MO-JSSPs.

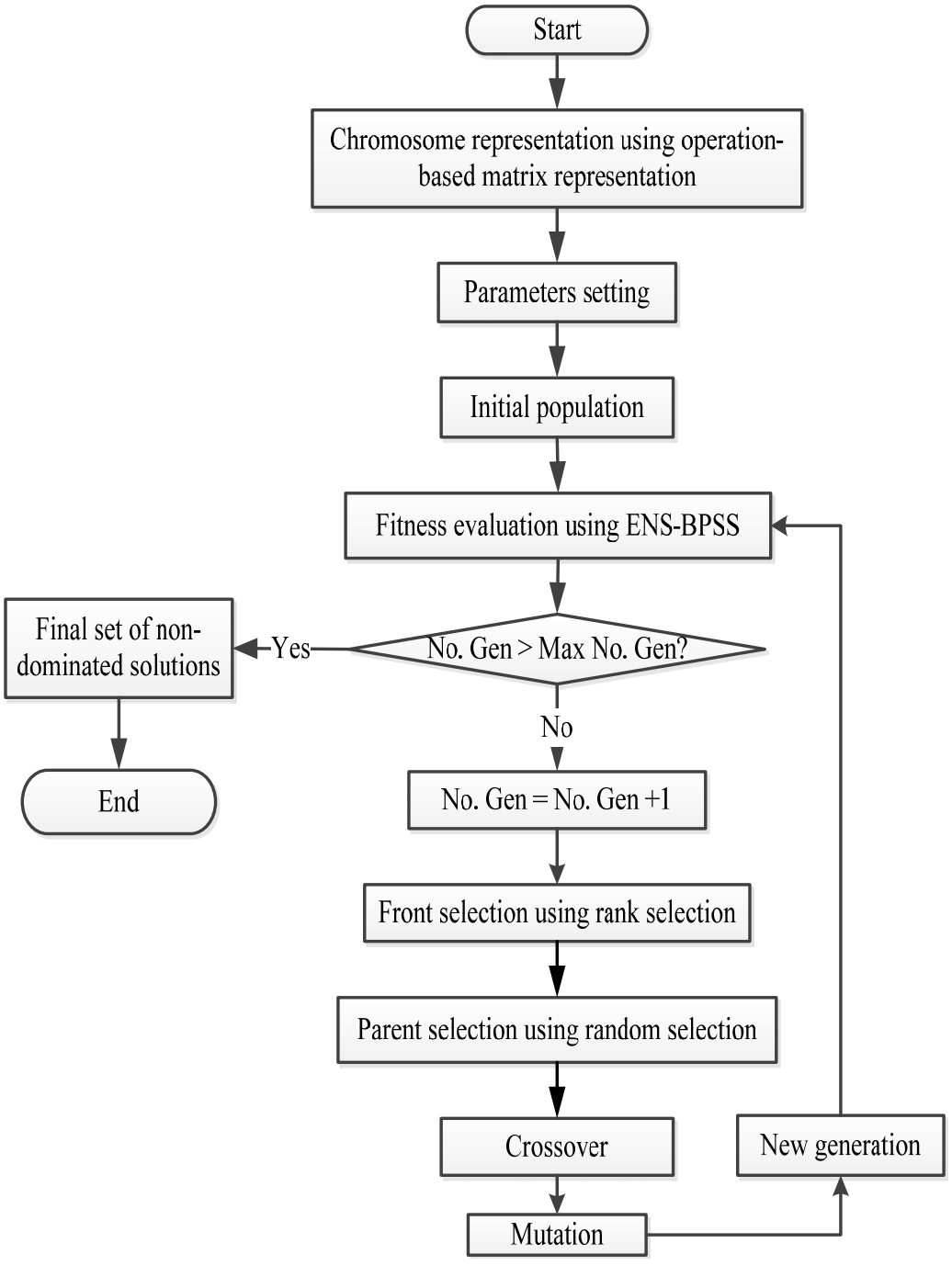

Figure 4-10. The Layout of the Proposed System for Solving MO-JSSPs

4.3.1 Solution Representation

In this research, a new solution representation method termed an Operation-based Matrix Representation is introduced. This method applies similar techniques to the operation-based representation that was proposed by Gen, Tsujimura et al. (1994), to represent operations in the schedule or the individual. Each gene in this method stands for a sequence of one operation, and each integer number in the gene represents a job type. The first occurrence of a job in a chromosome stands for the first operation of that job in the corresponding machine, while the second occurrence stands for the second operation, and so on (Cheng, Gen et al. 1996). Instead of using a vector to represent the chromosome, the chromosome or the solution is presented in matrix form so that, for n jobs and m machines, the solution can have a matrix size $m \times n$, where each job cannot appear more than once in each row. For instance if we have five jobs that need to be processed on four machines, one chromosome or solution can be presented as shown in Figure 4-11.

$$Chromosome = \begin{bmatrix} 1 & 3 & 5 & 2 & 4 \\ 4 & 1 & 2 & 3 & 5 \\ 3 & 5 & 4 & 2 & 1 \\ 2 & 5 & 4 & 1 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} O_{11} & O_{32} & O_{53} & O_{23} & O_{44} \\ O_{41} & O_{12} & O_{22} & O_{33} & O_{54} \\ O_{31} & O_{51} & O_{42} & O_{24} & O_{14} \\ O_{21} & O_{52} & O_{43} & O_{13} & O_{34} \end{bmatrix}$$

Figure 4-11. Operation Based Matrix Representation

The decoding procedure can be described as follows: first decode the chromosome to a list of order operations starting from the genes in the first column then the genes in the second column and so on. The schedule can then

be generated using a one-pass heuristic based on the list. The first operation or gene in the first column is scheduled first, and then the second operation in the first column is scheduled second and so on, until all operations in the first column are scheduled. Then the operations in the second column are scheduled with the same procedure as in the first column and so on. Each operation under treatment is allocated in the best available processing time. These procedures are continuously repeated until all operations are scheduled.

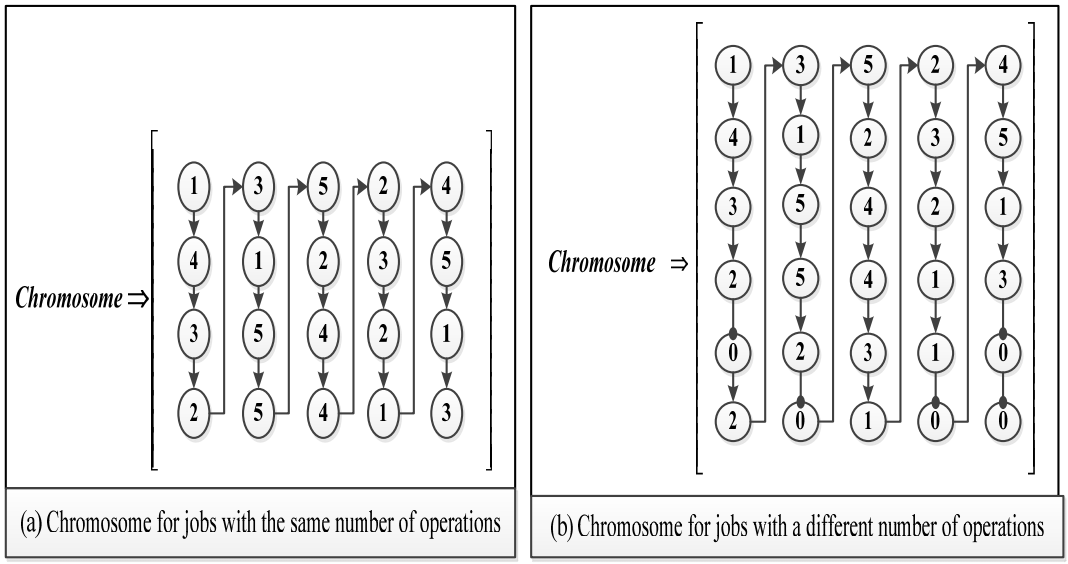

Figure 4-12. Chromosome Based Matrix Representation

Figure 4-12 (a) shows the chromosome based matrix representation when all jobs have the same number of operations. If some jobs need more numbers of operations, such as in recirculation or additional operations, then dummy operations can be used for matrix balance, as shown in Figure 4-12 (b). To generate feasible solutions from any crossover operation, all corresponding rows in matrices in the population should contain the same type of jobs, regardless of their position.

4.3.2 Creation of Initial Population

In this research, individuals in the initial population were created randomly. This technique allows the search process to cover a wide space, which can be useful for multi-objective optimization. Randomly created solutions must take into account that, each job cannot appear more than once in each row and all corresponding rows of different matrices must have the same elements, regardless of the job order in the row. Individuals created in this way always produce feasible solutions.

4.3.3 Design of Fitness Evaluation Function

The aim of the proposed scheduling system was to minimize the multiple objectives based on job completion time and due date whilst considering release time and setup time. In order to achieve that, the start and completion time for each operation and machine have to be calculated to attain the required information from each schedule. This information can then be used to evaluate each solution regarding the performance measurement and compare it with other solutions.

In this work a 3-dimensional matrix $[X, Y, Z]$ was used to handle the start time, end time, machine ID, setup group for each job and machine setup time, as shown in Figure 4-13. The Z-axis refers to the machine ID or machine number and each machine has a list of gaps, where X and Y represent the gaps list. Initially there is only one gap. The first column of the gap list in all machines refers to the gap start time and is set to be equal to the machine release time. The second column is the gap end time, which is initially set to be equal to infinity. The third column of the gap list is the machine setup time and the fourth

column is the setup group. When a job has to be inserted, the required gap must be identified and then the job divides the gap into two, left and right gaps, these gaps are then added to the gap list and the previous gap is deleted. Jobs are inserted into the best available gap length with respect to all constraints. These procedures are continued until all jobs are scheduled and all information such as start time and end time for the operations and machine setup is obtained.

Once resources have been allocated to existing jobs with the best available time, three performance measures are calculated. The performance measures that have been used in this work, as mentioned in chapter 3, are makespan, maximum tardiness and weighted number of tardy jobs. After calculating the performance measures for each schedule, the fitness assignment procedure is applied using Pareto dominance.

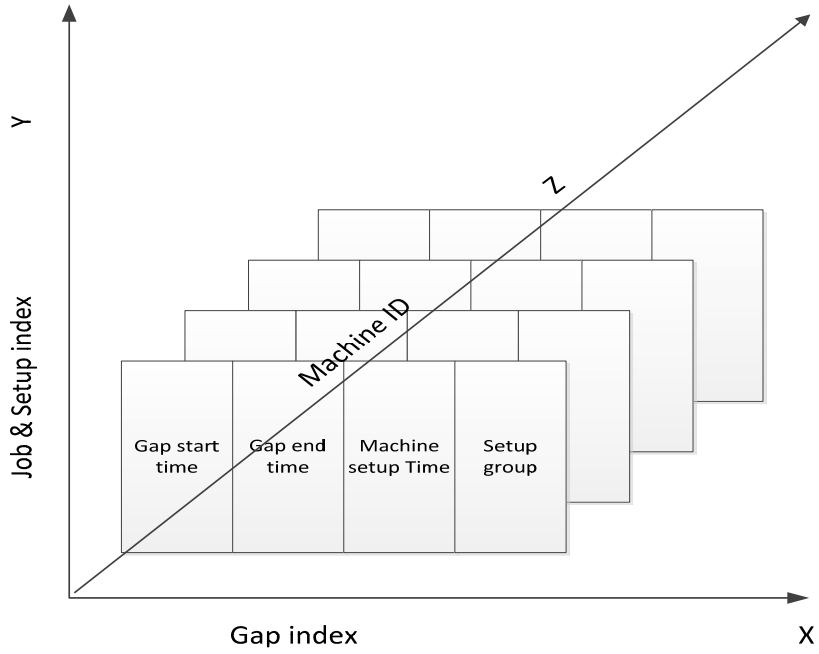

Figure 4-13. 3-Dimensional Matrix for Scheduling Evaluation

4.3.3.1 Efficient Nondominated Sort Using the Backword Pass Sequential Strategy for Fitness Assignment

The fitness assignment procedure works by exploiting the concept of Pareto dominance. In the previous methods such as NSGA-II (Deb, Pratap et al. 2002), in order to identify each front, each solution must be compared with all other solutions in the population, to find out how many solutions dominate a solution s and how many solutions are dominated by solution s, which requires at least $O(MN^2)$ number of comparisons. In NSGA-II, to design each front, a domination count n_{t_s}, the number of solutions which dominate the solution t_s, is needed. The domination count determines the number of or level of the non-dominated front. However, this method might assign some solutions to the lower non-dominated front or level from its actual front. For instance, using an example in Table 4-2, in which 5 solutions with 3 different objective functions [obj1, obj2 obj3]. These solutions belong to 3 different fronts based on their objective functions. In this example, in the case of minimization, two solutions (t_3 &t_4) belong to the second front but t_3 is dominated by just one solution from the first front (t_2) and t_4 is dominated by two solutions (t_1, t_2). However, by using NSGA-II; t_3 will be assigned to the second front and t_4 will be assigned to the third front, since n_{t_3} =1 and n_{t_4} =2. This will also cause solutions dominated by t_4 to step back one front.

Table 4-2. Comparison Between Actual and NSGAII Fronts

	First front	Second front	Third front
Actual fronts	t_1 = [201,40,15] t_2 = [209,33,12]	t_3 = [218,35,16] t_4 = [210,41,20]	t_5 = [220,45,19]
fronts by using NSGAII	t_1 = [201,40,15] t_2 =[209,33,12]	t_3 =[218,35,16]	t_4 = [220,45,19] t_5 = [210,41,20]

Because of these deficiencies of NSGA-II and some other techniques, the proposed system in this research adopted a very recent method that was proposed by Xingyi, Ye et al. (2015) and termed as ENS-SS. This method has less computational time and can identify each non-dominated front more precisely. Instead of comparing each solution with all other solutions in the population and then determining the front number of all solutions on the same front at once, the ENS-SS approach determines the front each solution belongs to one by one. Compared to NSGAII, the space complexity of ENS-SS can be reduced from $O(N^2)$ to $O(1)$, while the time complexity or number of comparisons can be reduced from $O(MN^2)$ to $O(MN\sqrt{N})$ in the best case. For more details of ENS-SS, readers are encouraged to refer to (Xingyi, Ye et al. 2015). In the following section, the proposed ENS with the new comparison strategy called BPSS is described.

For the case of minimizing three objectives; the makespan $(f_1 = C_{\max})$, the maximum tardiness $(f_2(x) = T_{\max})$ and the total weighted number of tardy jobs $(f_3(x) = \sum_{i=1}^{n} w_i U_i)$, first all solutions are sorted in ascending order according to the value of $C_{\max}$. When two or more solutions have the same value of $C_{\max}$, they are sorted according to the $T_{\max}$. If $C_{\max}$ and $T_{\max}$ are also the same in

two or more solutions, then they are sorted according to the $\sum_{i=1}^{n} w_i U_i$. Otherwise, if all values in all objectives are the same, they can be sorted arbitrarily. These procedures continue until all individuals in the population are sorted. By doing this, any succeeding solution in the sorted list can never dominate any preceding solution, as there is at least one objective value in the preceding solution that is less than the objective value of the succeeding solution or there exists two or more solutions with the same values in all objectives. Consequently, only two possible relationships can exist between any two solutions instead of three relationships. The preceding solution in the list dominates the succeeding solution or two solutions do not dominate each other. This step has $O(N \log N)$ time complexity and $O(1)$ space complexity (Xingyi, Ye et al. 2015).

Once all solutions are sorted, the proposed ENS-BPSS starts to assign solutions from the sorted population, starting from the first solution in the list and ending with the last one, one after another to their fronts. A similar procedure as in the SS was used in BPSS to determine the front to which each solution belongs, but instead of starting with the first front, the proposed BPSS starts the comparison with the last created front so far and ends where it finds its dominant solution. ENS-BPSS can reduce the number of comparisons needed when there are N fronts and there exists only one solution in each front to $O(M(N-1))$, since the solution in the sorted population can never dominate a preceding solution, so that each solution will only be compared with the direct preceding solution in this case. In what follows, the BPSS within the ENS

framework is introduced. Figure 4-14 illustrates the proposed ENS-BPSS for finding the front of a solution.

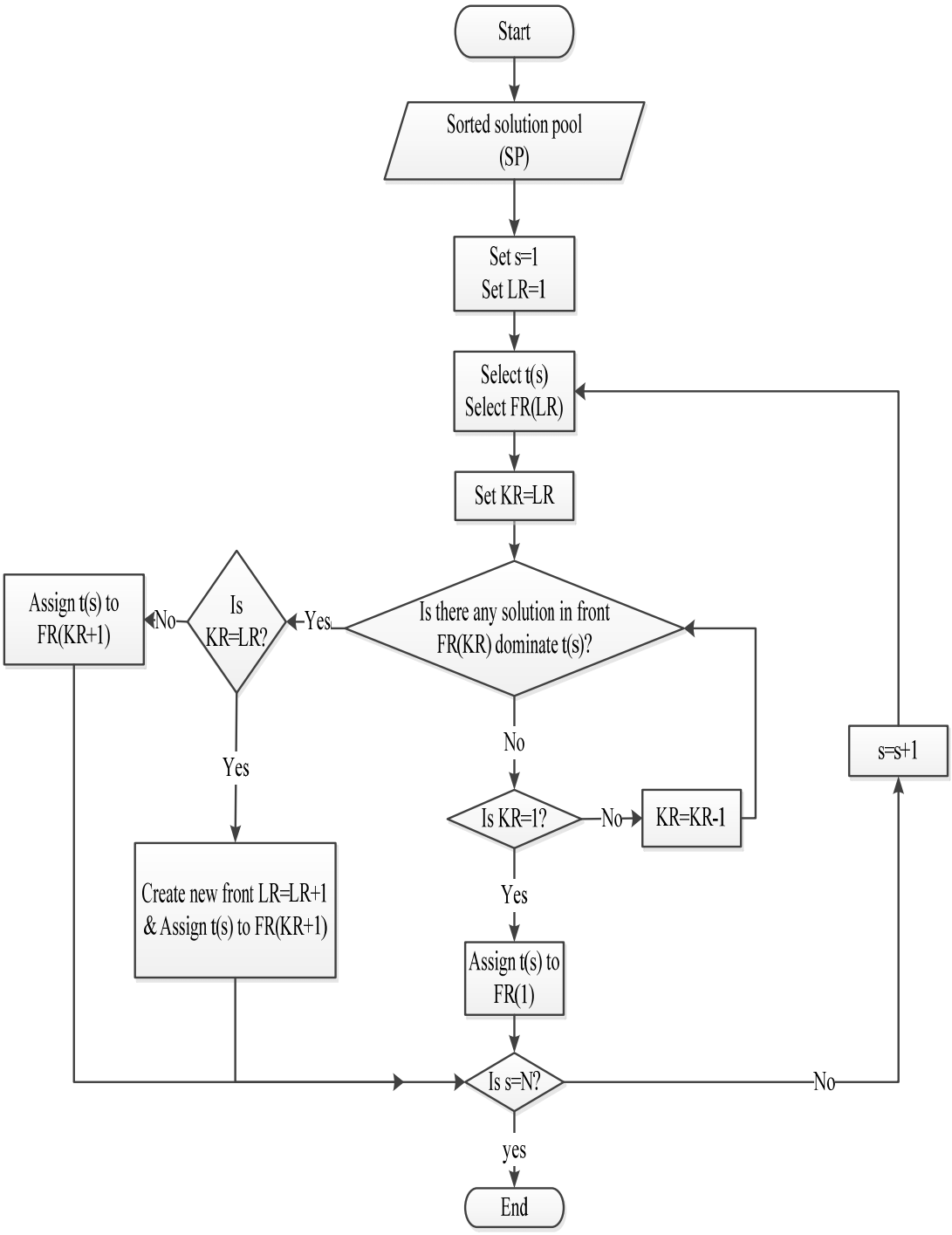

Figure 4-14. Proposed ENS-BPSS for Finding the Front of a Solution

For solution t_S that belongs to the sorted population SP, where $1 \le s \le N$ and N is the population size; - check whether a solution exists that has been

assigned to the last created front so far FR_{LR} that dominates t_S. If t_S is dominated by any solution in FR_{LR}; - create a new front FR_{LR+1} and assign t_S to this new front and set $LR = LR + 1$. If such a solution does not exist so that no solution in FR_{LR} dominates t_S; start comparing t_S with solutions assigned to FR_{LR-1}. If t_S is dominated by any solution in FR_{LR-1}; assign t_S to FR_{LR}, otherwise check the solution in FR_{LR-2} and so on. This procedure is continued until such a solution that is a dominated solution t_S is found in front KR where $1 \le KR \le LR$. If such a solution has been found, then assign t_S to the front FR_{KR+1}, otherwise, if such a solution has not been found in any front, t_S is assigned to the first front FR_1. Similarly, as in SS, solutions assigned to an existing front are also sorted in ascending order according to the value of $C_{\max}$, $T_{\max}$ and $\sum_{i=1}^{n} w_i U_i$. Therefore, the comparisons between t_S and the solutions assigned to any front would start with the last assigned solution in the front and end with the first assigned one.

In the case when there are LR fronts and there exists only one solution in each front, only one comparison is needed between each solution and its direct preceding solution. Consequently, the total number of comparisons in this case is reduced to $O(M(N-1))$.

4.3.4 Genetic Operators

In this research the three main elements of genetic operators (selection, crossover and mutation) are used to guide the search towards the optimal

solutions. More details of these three operators are given in the following sections.

4.3.4.1 Individuals Selection Method

In order to determine which of the chromosomes or individuals in the current generation can be selected to reproduce offspring, two selection mechanisms were used in this research. In the first, the rank selection was used to determine the Pareto front number to be selected for each parent. Rank selection ranks the front so that the first front will have a fitness value equal to LR (total number of fronts), the second front will have a fitness value equal to LR-1 and so on, so that the worst front will have a fitness value of 1. In the second selection, after determining the Pareto front number for each parent, random selection was used to select the parent from the selected front. The first selection method ensures that chromosomes in the higher level, or with a better Pareto front, have a better chance to be selected, while the second selection method ensures that chromosomes in the same level, or same front, have an equal chance to be selected. These procedures are applied each time when two parents need to be selected for mating as well as when one parent needs to be selected for mutation. Individuals in the first front are always maintained to form the next generation.

4.3.4.2 Crossover Procedures

Relating to the chromosome representation method in this research, three forms of crossover were introduced, which are based on two classifications; the number of rows to be exchanged between two or more parents and the number of parents to be contributed to produce new offspring. Three different types of

crossover were introduced; two-parents / one-row crossover, two-parents / two-rows crossover, and three-parents / one-row crossover (harmony crossover).

1. ***Two-parent / One-row Crossover:*** In this type of crossover, after two parents have been identified, one row from each parent is selected and then these two rows exchange their genes as shown in Figure 4-15.

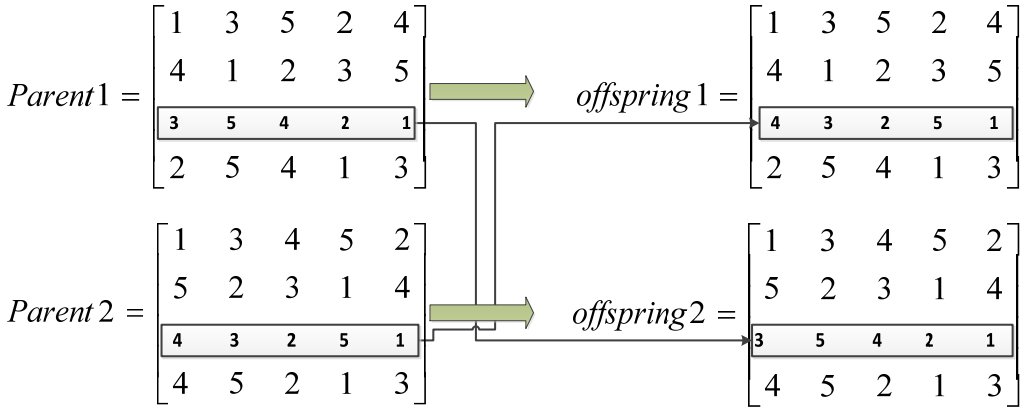

Figure 4-15. Two-parents / One-row Crossover

2. ***Two-parent / Two-row Crossover:*** Two rows from each of the two selected parents are randomly selected and then the genes in these rows are exchanged as shown in Figure 4-16.

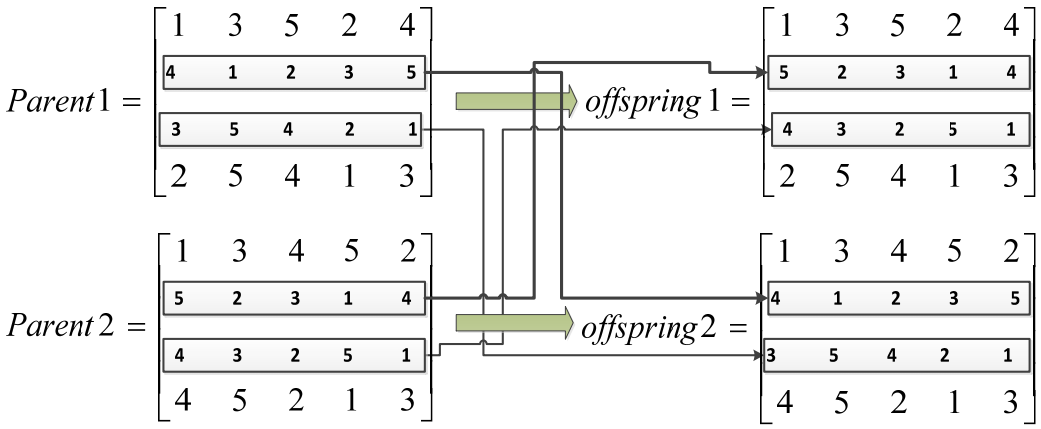

Figure 4-16. Two-parents / Two-rows Crossover

3. ***Three-parent / One-row Crossover***: in this type of crossover two parents are identified as main parents and the third parent only contributes to each offspring with gens taken from one row. After the main two parents have been identified, one row from each parent is selected and then these two rows exchange their genes. Another row with a different row index, is randomly selected from the third parent and inserted into that row index in the two offspring. These procedures are depicted in Figure 4-17.

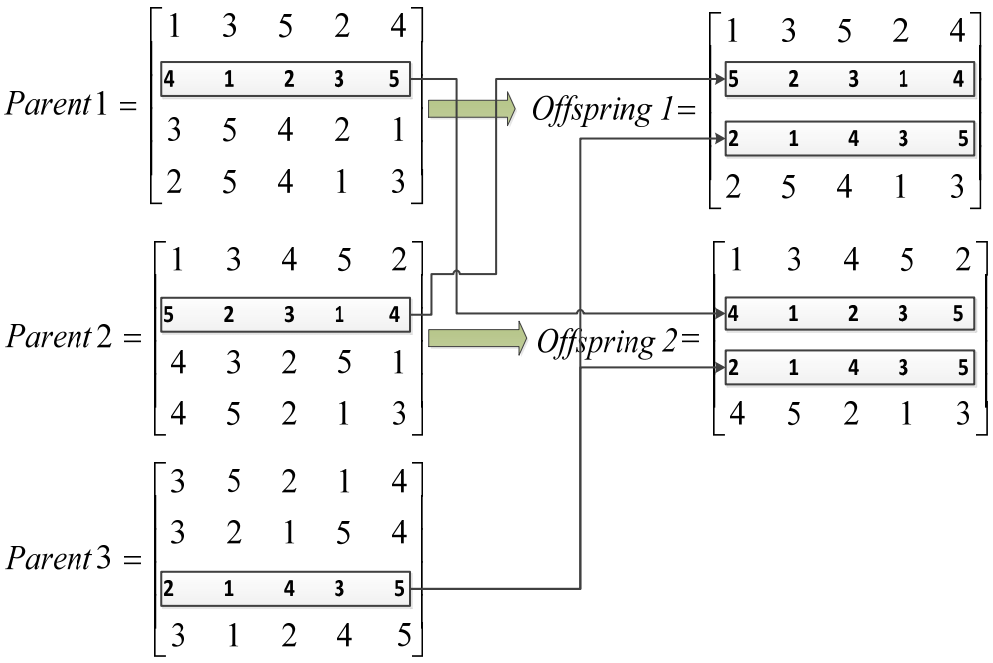

Figure 4-17. Three-parents / One-row Crossover

4.3.4.3 Mutation Procedures

Four types of mutation operator were introduced in this work. The working mechanism based on the matrix representation of these types of mutation is similar to the operation based representation.

1. ***Shift Mutation (Insertion Neighbourhood):*** One row is randomly selected from the matrix (individual), then one gene (job type) from the chosen row is also randomly selected and shifted to a different position in that row. All remaining jobs between these two positions in the row are pushed one position towards the selected job position in the same row. This procedure is depicted in Figure 4-18.

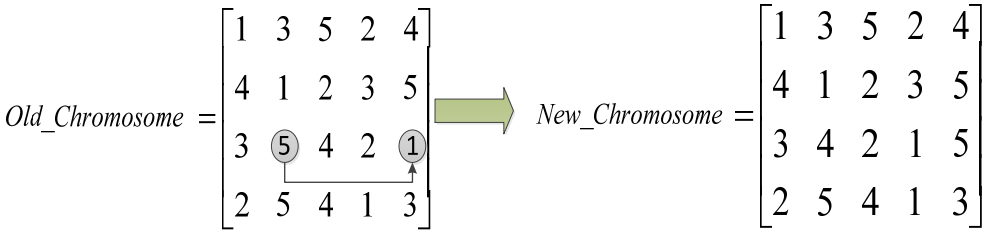

Figure 4-18. Shift Mutation (Insertion Neighbourhood)

2. ***Pairwise Interchange Mutation (Swap Neighbourhood):*** One row is randomly selected from the matrix (individual) then two genes (jobs) from the chosen row are randomly selected and swap their position in that row. This procedure is depicted in Figure 4-19.

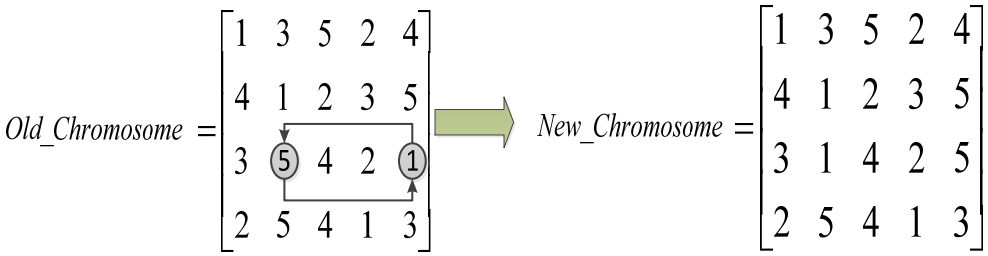

Figure 4-19. Pairwise Interchange Mutation (Swap Neighbourhood)

3. ***Inversion Mutation:*** one row is randomly selected from the matrix (individual) then two points from the chosen row are randomly identified and the genes (jobs) in that segment are reinserted in the reverse order. These procedures are shown in Figure 4-20.

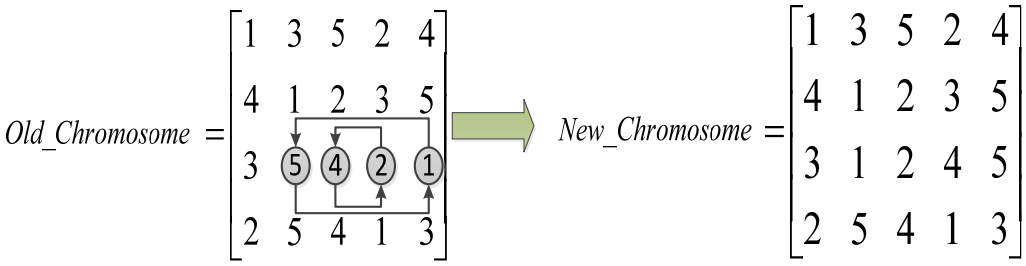

Figure 4-20. Inversion Mutation

4.3.5 Solving Scheduling Problems with Machine Setup Time

To solve JSSPs with machine dependent, job sequence independent, and item availability with anticipatory setup times, complex numbers have been used in the machine matrix to represent the machine number and setup group for each operation on that machine. For example if operation q of job a on machine k belongs to setup group G then it will appear in the machine matrix as $(k+Gi)$. The real part (k) of the complex number refers to the machine required to perform operation q, and the imaginary part (G) of the complex number refers to the setup group for the operation q of job a on machine k. For more illustration assume that; 3 different jobs need to be processed on 3 different machines. Operation 1 of job 1 and operation 2 of job 2 belong to the same setup group on machine 1 (setup group 1) with a setup time equal to 13, while operation 1 of job 3 belongs to a different setup group on machine 1 (setup group 2) with a setup time equal to 15. For machine 2; operation 1 of job 2 belongs to setup group 1 with a setup time equal to 20, while operation 2 of job 1 and operation 3 of job 3 belong to setup group 2 with a setup time equal to 27. In machine 3, 3 jobs have the same setup group with setup time equal to 11. The machine matrix including setup group and setup time for each group in each machine will appear as follows:

$$Mach = \begin{bmatrix} 1+1i & 2+2i & 3+1i \\ 2+1i & 1+1i & 3+1i \\ 1+2i & 3+1i & 2+2i \end{bmatrix} \qquad STime = \begin{bmatrix} 13 & 15 \\ 20 & 27 \\ 11 & 0 \end{bmatrix}$$

The evaluation function takes into account the machine setup times when they are required. For instance, from the machine matrix ($Mach$), if operation 2 of job 2 is processed immediately after operation 1 of job 1 on machine 1, then there is no machine setup time required between these two operations since they belong to the same setup group in machine 1. However, when operation 2 of job 2 is processed after operation 1 of job 3 on machine 1, then the setup time must be added between these two operations as they belong to the different groups. Machine setup time is also required when a job is processed on a machine for the first time. Furthermore, the setup can be started before the corresponding job is available on the machine, which is known as anticipatory setup time.

4.3.6 Solving Scheduling Problems with Alternative Machines

As mentioned in chapter 3, testing all possible alternative routes for each chromosome or solution would require an excessive computational effort. Therefore, having some heuristic procedure to allocate a machine from a given set of alternative machines, to process a specific type of operation can be very beneficial. In this research, whenever a job needs to be processed on one machine from a set of alternative machines, a roulette wheel selection is used to select the required machine. The probability of a machine being selected is proportional to the machine processing time to process the job. Thus, the

probability $Pk(k)$ of selecting machine k to process operation j of job i from a set of alternative machines AM_{ij} is given as follows:

$$Pk(k) = \frac{1/p_{ijk}}{\sum_{m \in AM_{ij}} (1/p_{ijm})}$$

For more illustration assume that, three alternative machines (M1, M2 and M3) can perform operation j of job i with different processing times. M1 can perform operation j in 30 mins, M2 can perform operation j in 35 mins and M3 can perform operation j in 26 mins. The probabilities of selection of M1, M2, and M3 are 33.2%, 28.5% and 38.3% respectively as shown in Figure 4-21.

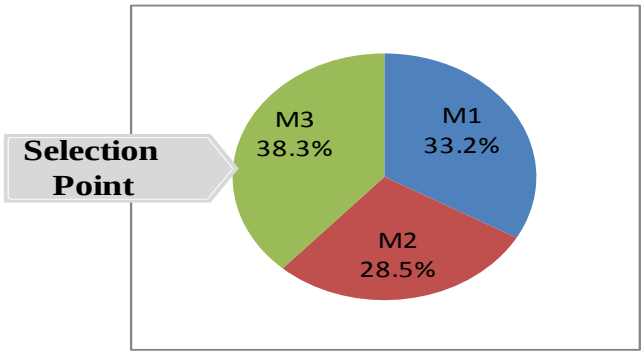

Figure 4-21. Machine Selection Based on the Processing Time

In this selection procedure, the machine with the least processing time to perform the operation would have a higher probability to be selected. After the machine is selected, all other required information such as machine processing time and setup time are extracted and used in the evaluation function. This information is maintained with the chromosome that used it. Different chromosomes can have different routes based on the machines that have been selected.

4.3.7 Representation of Processing Time Uncertainty

In this research an interval numbers theory is used to represent uncertainty of the processing time. Unlike fuzzy numbers theory and stochastic theory, using interval numbers theory only requires obtaining the lower and upper bound of intervals to indicate time uncertainty and does not require probability distribution or membership function. Here, a set of numbers between the lower bound or the most optimistic value of the processing time and upper bound or the most pessimistic value of the processing time, are generated randomly for each task. Two different scenarios are considered in this research; in the first scenario, the ratios of change in the processing time for all operations are considered to be the same. In the second scenario the ratios of change vary from one operation to another. In both scenarios two different cases are considered and compared with each other. In the first case, the genetic optimisation process is applied to find an optimal solution for some benchmark of JSSPs. The optimal sequence is then evaluated by applying the ratios of change in the processing time to find the deviation from the optimal solution and compare the results with the second case, in which the ratios of change in the processing time are applied at the initial stage to find the optimal sequence with the optimal makespan.

4.4 Summary

GA has been applied to solve many JSSPs and has proven to give very good results. In this chapter, the concept of GA and its procedure for solving JSSPs was introduced. The detail of the proposed system for solving MO-JSSPs, which utilizes GA and ENS was also provided. GA was applied to lead the search towards the Pareto optimality, while ENS was employed within the GA to

determine the front to which each solution belongs to, thus, evaluating the merit of each solution. In the GA, a new representation method for the solution called operation based matrix representation was introduced. The new representation method for the solution can preserve features of the parent after the crossover operator without repairing the solution. In the ENS a new strategy for the comparison called BPSS was proposed. In the proposed BPSS the comparison starts with the last created front so far and ends up where it finds its dominant solution. ENS-BPSS can reduce the number of comparisons to $O(M(N-1))$ in the best case. The proposed system also takes into consideration the release date, alternative machines and setup time. In the next chapter, experimental results and discussion on the research results with key observations in the research methodology are given.

Chapter 5. Computational Results and Discussion

5.1 Introduction

In order to validate the adopted methodology, this chapter describes the implementation of the proposed system and provides results of the various computations conducted in this study. The system was implemented in MATLAB R2014a on a PC with an AMD A4-5300B APU 3.4 GHz processor and 4GB of RAM. Although the proposed system was developed for a multi-objective case, the system can also be used for single objective case. At the beginning, the proposed system was tested with the classical JSSP using a number of published benchmark problems taken from the OR-Library (Beasley 1990) web site (URL: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/jobshop1.txt). Some of these benchmark JSSPs were then modified in order to suit JSSPs with release dates, setup times, alternative machines, and multi objective optimization. More details are given in the following sections.

5.2 Computational Results for Classical JSSP

In this section the proposed system was tested using 34 classical benchmark JSSPs. Benchmark problems provide a common standard to test and compare the developed system with other developed systems. Here, the aim of using these benchmark problems was to clearly demonstrate the feasibility and practicability of the new representation of the chromosome as well as the effectiveness of the algorithm for solving JSSPs. The computational experiments were performed with 200 for the population size, 0.7 for the crossover rate and 0.3 for the mutation rate. The termination condition varies in different instances. For instance the termination condition set is to be 3000 in FT10 as the problem is hard to solve to optimality, while in LA1 it is set to be

100 as it is easy to solve to optimality. These parameters were decided after a pilot test.

Table 5-1 shows the obtained results from the conducted experiments.

Table 5-1. Obtained Results for Benchmark JSSPs Using Operation Based Matrix Representation

Instance	Size $(n \times m)$	Best Achieved Makespan	Max Number of Generations
FT6	6×6	55	50
FT10	10×10	930	3000
ABZ5	10×10	1234	3000
ABZ6	10×10	943	3000
LA1	10×5	666	100
LA2	10×5	655	300
LA3	10×5	597	300
LA4	10×5	590	300
LA5	10×5	593	300
LA6	15×5	926	300
LA7	15×5	890	300
LA8	15×5	863	400
LA9	15×5	951	200
LA10	15×5	958	400
LA11	20×5	1222	200
LA12	20×5	1039	200
LA13	20×5	1150	200
LA14	20×5	1292	200
LA15	20×5	1207	100
LA16	10×10	945	100
LA17	10×10	784	100
LA18	10×10	848	150
LA19	10×10	842	150
LA20	10×10	902	1000
LA31	30×10	1784	1000
LA32	30×10	1850	1000
LA33	30×10	1719	600
LA34	30×10	1721	2000
LA35	30×10	1888	2000
SWV16	50×10	2924	600
SWV17	50×10	2794	600
SWV18	50×10	2852	600
SWV19	50×10	2843	600
SWV20	50×10	2823	600

The results show that, in all 34 tested benchmark JSSPs the developed system was able to find the optimal solution using operation based matrix representation. As these tested problems have different sizes and grades of difficulty, the times needed to reach optimal solutions vary from one problem to another. For instance the optimal solutions for some of these problems such as FT6, LA1 and LA2 were found in relatively small times while in some other problems such as FT10, ABZ5 and ABZ6, they were found in relatively high times. However, the optimal solution for some problems with greater size, such as SWV16 and SWV 20 were found in less time compared with FT10 and ABZ6.

5.3 Experimental Plan

After the proposed system was tested in different types of benchmark problems for the classical JSSP, some of these benchmark problems were modified to suit the JSSPs with included factors that were considered in this research. Initially, LA1 was used in each section to demonstrate the interpretation for each considered factor, starting from release time and then gradually building other factors upon it.

5.3.1 Incorporating Job / Machine Release Time

In this section job and machine release times were incorporated in the system by considering different arrival dates for the jobs and different initial ready times for the machines. Table 5-2 presents the Job and machine release times that were used to modify the selected benchmark problems. The same release time of each job and each machine was used for all selected benchmark problems in this work. Note that machines from M11 to M20 were used as alternative

machines and M21 was added later to be used by some jobs to show the case of jobs with different numbers of operations as well as jobs with recirculation. The job processing time of alternative machines were calculated as a percentage of machine time in the main benchmark problems as shown in Table 5-4.

Table 5-2. Releas Times for Jobs and Machines

Job ID	Job Release Time	Machine ID	Machine Release Time
J1	10	*M1*	11
J2	0	*M2*	0
J3	17	*M3*	25
J4	23	*M4*	0
J5	0	*M5*	15
J6	39	*M6*	13
J7	13	*M7*	7
J8	15	*M8*	14
J9	7	*M9*	2
J10	12	*M10*	0
J11	30	*M11*	4
J12	12	*M12*	25
J13	32	*M13*	16
J14	22	*M14*	0
J15	41	*M15*	13
J16	12	*M16*	12
J17	3	*M17*	7
J18	0	*M18*	19
J19	0	*M19*	26
J20	10	*M20*	30

To demonstrate how these factors influence the scheduling system, LA1 was used here to give some details. Figure 5-1 shows the Gantt chart of LA1 with job and machine release dates. As it has been shown in the Gantt chart, jobs cannot be processed in a particular machine before the machine release date. Similarly, a job cannot be processed before it becomes available in the manufacturing shop floor. These two constraints make the system more dynamic in the way that, in real manufacturing practice not all the jobs and machines are available simultaneously. Although these two factors are

commonly uncertain, the decision maker needs to make a decision based on the best available information. Therefore considering these two factors will make the scheduling system more accurate and reliable.

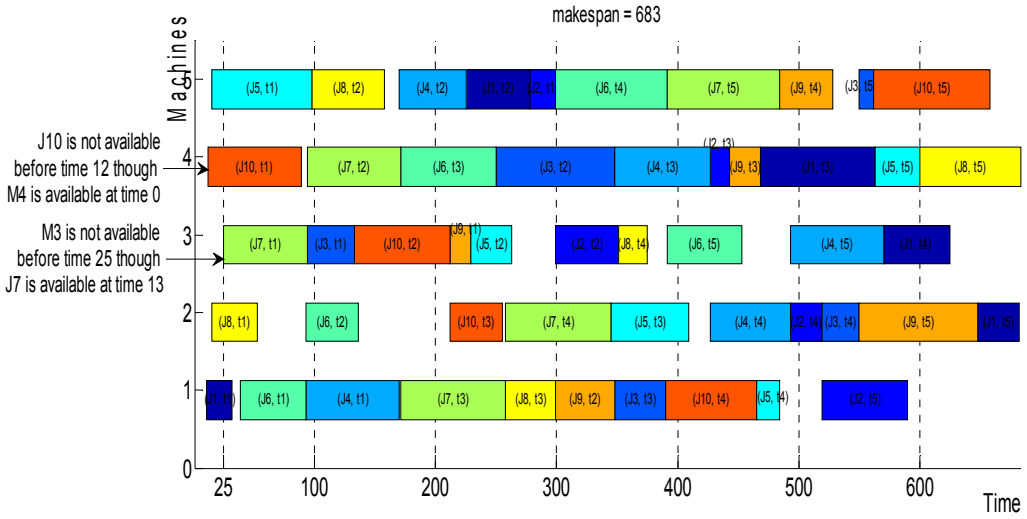

Figure 5-1. LA1 with Job and Machine Release Date

5.3.2 Incorporating Machine Setup Time

This section demonstrates how the machine setup times have been incorporated into the scheduling system. As mentioned earlier, this work deals with machine dependent setup times of type job sequence independent setup times. Table 5-3 presents setup groups and setup times in each machine. Jobs have been divided into different setup groups in each machine and setup time has been assigned to each setup group. The same setup groups and setup times in each machine were used for all tested benchmark problems in this research. Complex numbers have been used in order to incorporate the operations setup groups in the machine matrix. The real part of the complex number in the machine matrix represents the machine number for the particular operation of the particular job, while the imaginary part of the complex number in the machine matrix represents the setup group of that operation on that machine.

Table 5-3. Jobs Setup Groups and Setup Times in Each Machine.

Machines	**Setup Groups & Times**					
	g1	St_{1k}	g2	St_{2k}	g3	St_{3k}
M1	J1, J2, J3, J11, J15, J18	20	J4, J6, J7, J13, J16, J17, 19J	15	J5, J8, J9, J10, J12, J14, J20	9
M2	J6, J7, J10, J12, J14, J16	18	J3, J5, J8, J9, J11, J15, J17	13	J1, J2, J4, J13, J18, J19, J20	15
M3	J4, J7, J8, J13, J15, J16, J20	9	J1, J3, J6, J10, J11, J17	11	J2, J5, J9, J12, J14, J18, J19	16
M4	J1, J2, J4, J8, J11, J16, J20	14	J3, J5, J6, J7, J12, J14, J17	14	J9, J10, J13, J15, J18, J19	11
M5	J1, J2, J3, J4, J19, J20	17	J5, J6, J7, J11, J12, J17, J18	23	J8, J9, J10, J13, J14, J15, J16	12
M6	J3, J6, J8, J11, J15, J17, J20	13	J1, J5, J10, J13, J14, J19	21	J2, J4, J7, J9, J12, J16, J18	17
M7	J4, J7, J9, J12, J14, J18	17	J3, J6, J8, J10, J13, J15, J20	22	J1, J2, J5, J11, J16, J17, J19	15
M8	J1, J3, J5, J7, J11, J14, J15	12	J2, J4, J6, J12, J13, J17, J19	14	J8, J9, J10, J16, J18, J20	14
M9	J2, J9, J10, J11, J12, J16	19	J1, J3, J4, J6, J13, J19, J20	15	J5, J7, J8, J14, J15, J17, J18	11
M10	J1, J4, J8, J10, J17, J18, J19	24	J2, J3, J5, J6, J11, J13, J16	12	J7, J9, J12, J14, J15, J20	15
M11	J1, J2, J3, J11, J15, J18	24	J4, J6, J7, J13, J16, J17, 19J	18	J5, J8, J9, J10, J12, J14, J20	11
M12	J6, J7, J10, J12, J14, J16	24	J3, J5, J8, J9, J11, J15, J17	17	J1, J2, J4, J13, J18, J19, J20	20
M13	J1, J2, J4, J8, J11, J16, J20	13	J3, J5, J6, J7, J12, J14, J17	13	J9, J10, J13, J15, J18, J19	10
M14	J1, J2, J4, J8, J11, J16, J20	13	J3, J5, J6, J7, J12, J14, J17	13	J9, J10, J13, J15, J18, J19	10
M15	J1, J2, J4, J8, J11, J16, J20	14	J3, J5, J6, J7, J12, J14, J17	14	J9, J10, J13, J15, J18, J19	11
M16	J1, J2, J3, J4, J19, J20	15	J5, J6, J7, J11, J12, J17, J18	20	J8, J9, J10, J13, J14, J15, J16	11
M17	J3, J6, J8, J11, J15, J17, J20	10	J1, J5, J10, J13, J14, J19	20	J2, J4, J7, J9, J12, J16, J18	15
M18	J1, J3, J5, J7, J11, J14, J15	10	J2, J4, J6, J12, J13, J17, J19	11	J8, J9, J10, J16, J18, J20	13
M19	J1, J3, J5, J7, J11, J14, J15	10	J2, J4, J6, J12, J13, J17, J19	11	J8, J9, J10, J16, J18, J20	13
M20	J1, J4, J8, J10, J17, J18, J19	24	J2, J3, J5, J6, J11, J13, J16	12	J7, J9, J12, J14, J15, J20	15

LA1 is used here to give some details of how these representations appear in the machine matrix and how setup time factors affect the scheduling problem. The matrices below show the representation of machines, with setup groups, $Mach$, operation processing times, $PTime$, and machine setup times, $STime$, for LA1.

$$Mach=\begin{bmatrix} 1+1i & 5+1i & 4+1i & 3+2i & 2+3i \\ 5+1i & 3+3i & 4+1i & 2+3i & 1+1i \\ 3+2i & 4+2i & 1+1i & 2+2i & 5+1i \\ 1+2i & 5+1i & 4+1i & 2+3i & 3+1i \\ 5+2i & 3+3i & 2+2i & 1+3i & 4+2i \\ 1+2i & 2+1i & 4+2i & 5+2i & 3+2i \\ 3+1i & 4+2i & 1+2i & 2+1i & 5+2i \\ 2+2i & 5+3i & 1+3i & 3+1i & 4+1i \\ 3+3i & 1+3i & 4+3i & 5+3i & 2+2i \\ 4+3i & 3+2i & 2+1i & 1+3i & 5+3i \end{bmatrix} PTime=\begin{bmatrix} 21 & 53 & 95 & 55 & 34 \\ 21 & 52 & 16 & 26 & 71 \\ 39 & 98 & 42 & 31 & 12 \\ 77 & 55 & 79 & 66 & 77 \\ 83 & 34 & 64 & 19 & 37 \\ 54 & 43 & 79 & 92 & 62 \\ 69 & 77 & 87 & 87 & 93 \\ 38 & 60 & 41 & 24 & 83 \\ 17 & 49 & 25 & 44 & 98 \\ 77 & 79 & 43 & 75 & 96 \end{bmatrix} STime=\begin{bmatrix} 20 & 15 & 9 \\ 18 & 13 & 15 \\ 9 & 11 & 16 \\ 14 & 14 & 11 \\ 17 & 23 & 12 \end{bmatrix}$$

Figure 5-2 shows the Gantt chart with the optimal makespan for LA1 after including the machine setup times and release times of jobs and machines.

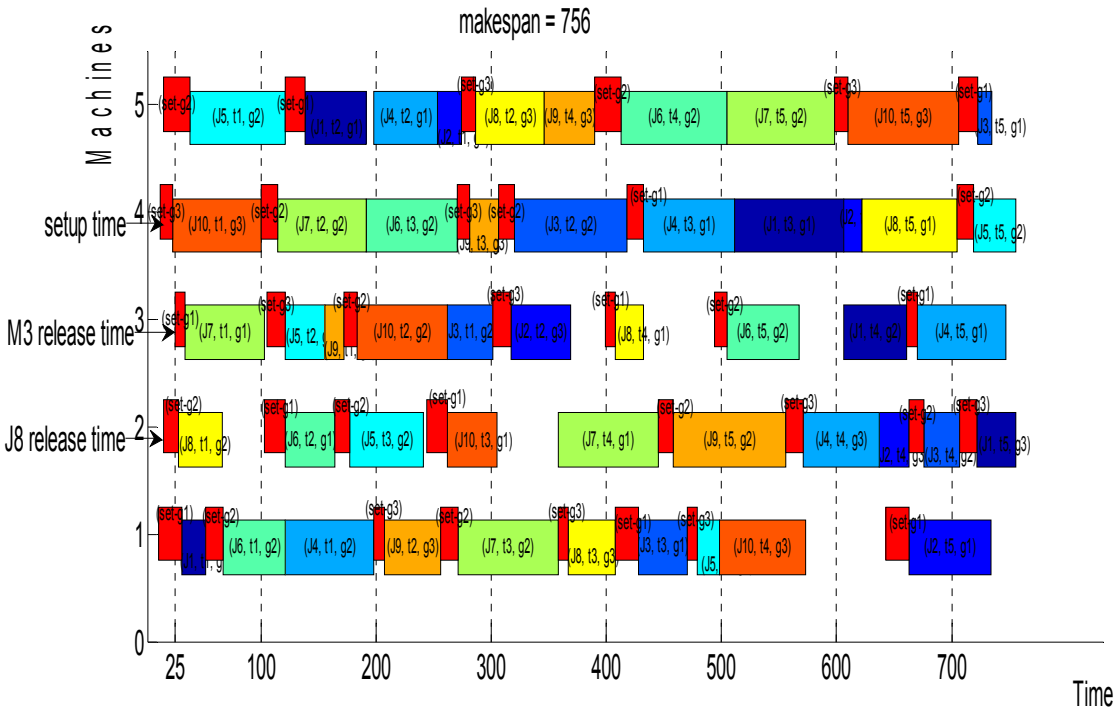

Figure 5-2. LA1 with Machine Setup times and Release time

For more illustration, the first operation of J3 is on M3 with a processing time equal to 39. This operation belongs to setup group 2 with a setup time equal to 11. If operation 2 of J10 is the immediate successor of operation 1 of J3 on M3, then there is no machine setup time required between these two operations. However, if the immediate successor of operation 1of J3 on M3 was operation 1 of J7, then machine setup will take place between these two operations because they belong to different setup groups. This procedure is continually applied between any two immediate successors for any operations on the same machine. The setup can also be started before the corresponding job is available on the machine. For instance the setup between J10 and J2 on M1 was finished before J2 became available on M1.

A comparison can be made here between two cases. The case when machine setup times were not included with the case when machine setup times were incorporated in the scheduling system. The total completion time has been remarkably increased from 683 time units to 756 time units after the setup time was included. This increase in time of 10.69% indicates that the addition of setup times gives significant improvements in reliability and accuracy of the overall scheduling system.

5.3.3 Incorporating Alternative Machines

In JSSPs with alterative machines, an operation can be processed by any machine from a given set of alternative machines. To extend the selected benchmark problems in order to be used in partially flexible JSSPs, alternative machines have been introduced for some operations. Table 5-4 represents the alternative machines and the ratio of job processing time that was used in each machine.

Table 5-4. The Alternative Machines Processing Times

Machine	Alternative Machine	Job Processing Time on Alternative Machine
M1	*M11*	$M11 = \lceil 0.9M1 \rceil$
M2	*M12*	$M12 = \lceil 1.2M2 \rceil$
M3	-	-
M4	*M13* *M14* *M15*	$M13 = \lceil 1.1M4 \rceil$ $M14 = \lceil 0.95M4 \rceil$ $M15 = M4$
M5	*M16*	$M16 = \lceil 0.85M5 \rceil$
M6	*M17*	$M17 = \lceil 1.4M6 \rceil$
M7	-	-
M8	*M18* *M19*	$M18 = \lceil 1.2M8 \rceil$ $M19 = \lceil 1.2M8 \rceil$
M9	-	-
M10	*M20*	*M20=M10*

Figure 5-3 shows the Gantt chart with the optimal makespan for LA1 after including the alternative machines with machine independent setup times and the release times of the jobs and machines.

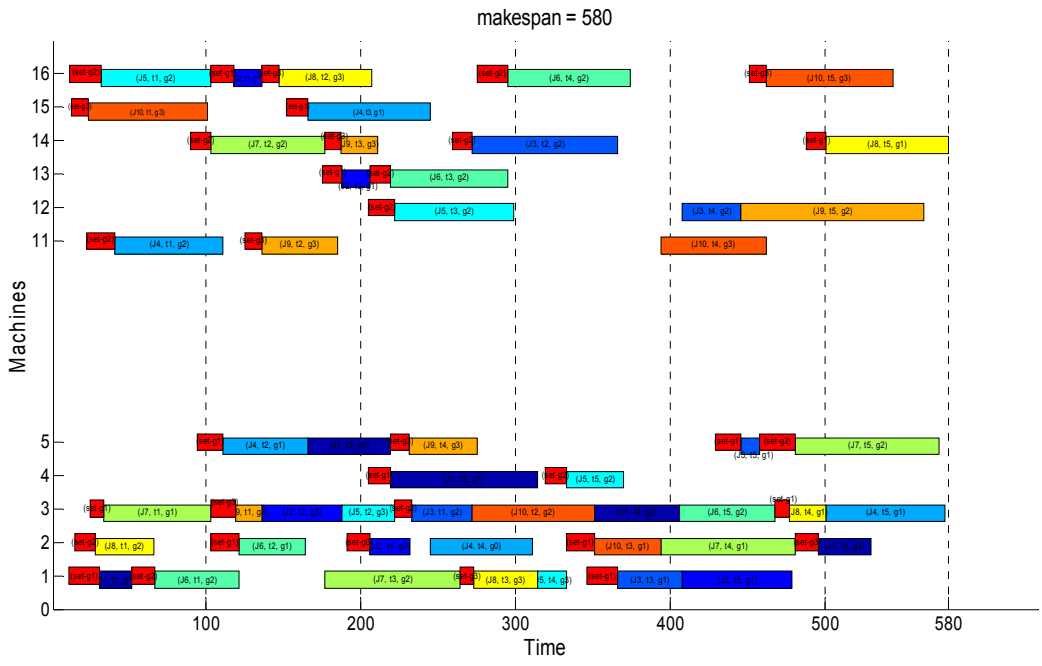

Figure 5-3. Gantt chart LA1 with Alternative Machines

Compared to the previous result, the makespan has been reduced from 756 time units to 589 time units. This reduction in time by 22.1% shows how the alternative machines can have a significant benefit to the overall system performance. The alternative machines are useful when many operations have to wait for a long time on the same machine in order to be processed, so that some of these operations can be processed on alternative machines. As a result, operation waiting time and job lead time is reduced and overall machine utilization is improved. However, in the partially flexible system, the resulting schedule mostly depends on those machines that have no alternative machines. These machines can be very crucial to the system. For instance the optimal schedule in Figure **5-3** is mainly dependent upon machine M3.

5.3.4 Incorporating Jobs with Recirculation and Various Numbers of Operations

One other case that usually exists in the job shop manufacturing environment is jobs with different numbers of operations and jobs with recirculation. In this case a job does not necessarily visit every machine and it may visit a machine more than once. In this research machine M21 is introduced to replace machine M3 for jobs J1, J7, J8 and J17. The processing times, machine release times, setup times and setup groups of M3 for J1, J7, J8 and J17 are used in M21. Another extra operation is added at the end of J1 and J8 with the same processing time, but the setup groups have been swapped so that J1 at the second visit to M21 requires setup group 1 and J8 requires setup group 2. Figure 5-4 shows the Gantt chart with the optimal makespan for the modified LA1 after including jobs with recirculation, different numbers of operations, alternative machines with machine independent setup times and release times of jobs and machines.

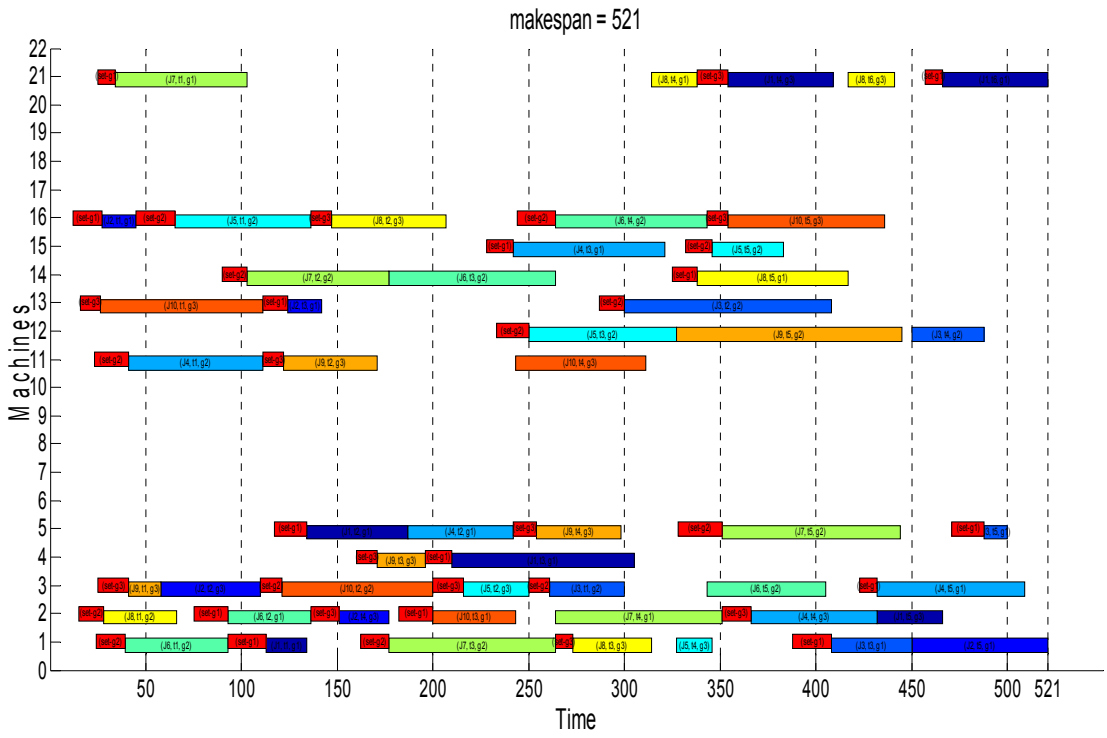

Figure 5-4. Jobs with Recirculation and Different Numbers of Operations for LA1

After showing how each factor appears and influences the scheduling system, the final results including these factors and using different modified benchmark JSSPs are presented in the next section, with consideration given to multi-objective optimization.

5.3.5 Scheduling System with Multi-Objective Optimization

To illustrate the effectiveness and performance of the proposed system (GA & ENS-BPSS) for solving MO-JSSPs with incorporation of release date, setup times and alternative machines, the system was tested using different modified benchmark problems. Since this work aims to minimize three objectives simultaneously; the makespan $(C_{\max})$, the maximum tardiness $(T_{\max})$ and the weighted number of tardy jobs $(\sum_{i=1}^{n} w_i U_i)$, the due date (d_i) for each job is needed and was set to be equal to the release date (r_i) plus the sum of the

processing times $(\sum_{j=1}^{J_i} p_{ij})$ of its operations multiplied by a due date tightness factor (tf) (Eilon and Chowdhury 1976).

$$d_i = r_i + \left\lceil tf * \sum_{j=1}^{J_i} p_{ij} \right\rceil$$

In this research the values of tf set as follows:

- $tf = 1.7$ for LA1, LA2, LA3, LA4 & LA5.
- $tf = 2.1$ for LA6, LA7, LA8, LA09 & LA10.
- $tf = 2.5$ for LA11, LA12, La13, LA14 & LA15.
- $tf = 1.3$ for LA16, LA17, LA18 LA19 & LA20.

These values were decided after a series of pilot tests. The ratio of the number of jobs to the number of machines in the system as well as avoiding a single nondominated solution was considered to decide these values. In the case of alternative machines, the machine with maximum operational processing time is selected in job i to find the due date.

Regarding to the job weight (w_i); the following weights were decided randomly to show the importance of different customers.

- $w_i = 2$ for J1, J3, J9 and J12.
- $w_i = 1$ for J2, J5, and J11.
- $w_i = 4$ for J4, J6, J14, J16 and J18.
- $w_i = 3$ for J7, J10, J15 and J19.

- $w_i = 5$ for J8, J13, J17 and J20.

The computational experiments were performed with the following parameters.

- Population size: N = 250
- Crossover rate: P_c = 0.7
- Mutation rate: P_m = 0.3
- Termination condition: 2000 generation

These parameters were also decided after a series of pilot tests. In these tests, the values of these parameters were identified to balance between computational time, speeds of convergence, avoiding saturation around one solution, stability and convergence to a global optimal

The final obtained results of the multi-objective optimization for 20 tested benchmark problems of JSSP's after they have been modified are depicted in Figure 5-5 to Figure 5-24. (Note that M_LA1 means a modified LA1 and so on).

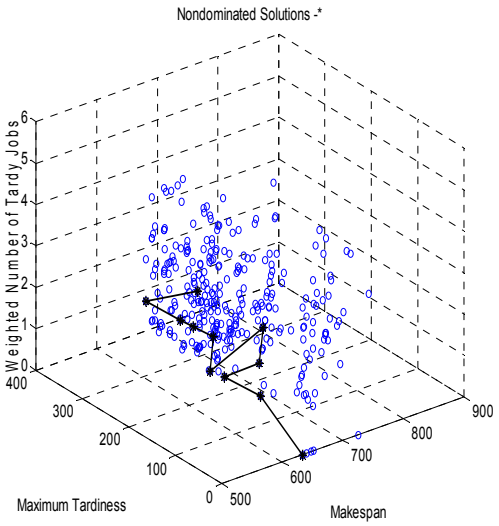

Figure 5-5. Pareto Front for M_LA1

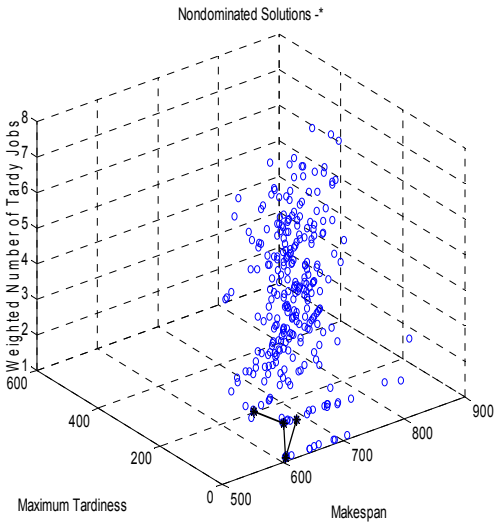

Figure 5-6. Pareto Front for M_LA2

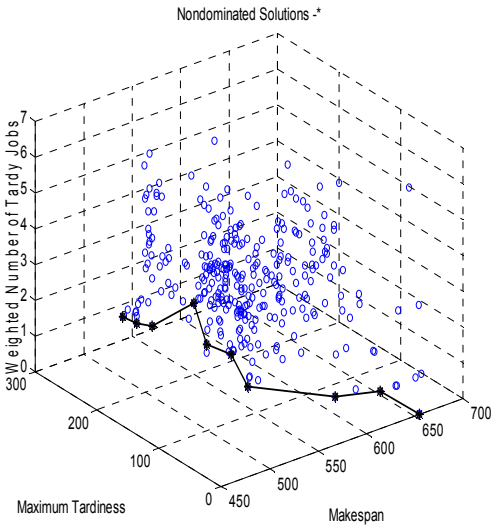

Figure 5-7. Pareto Front for M_LA3

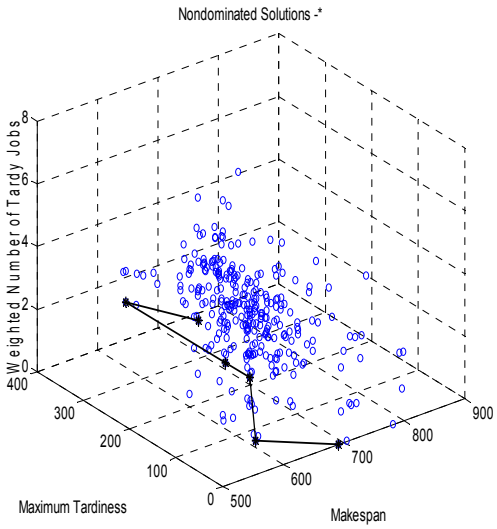

Figure 5-8. Pareto Front for M_LA4

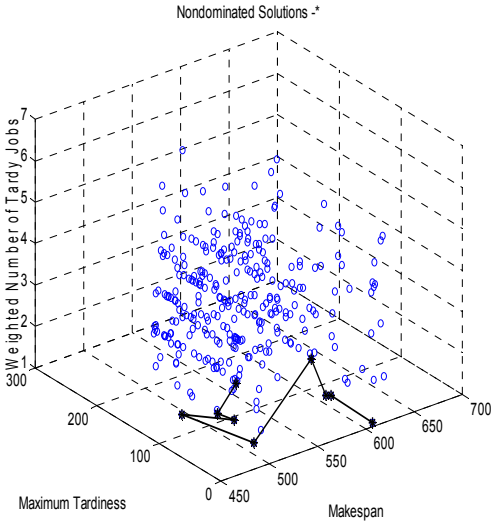

Figure 5-9. Pareto Front for M_LA5

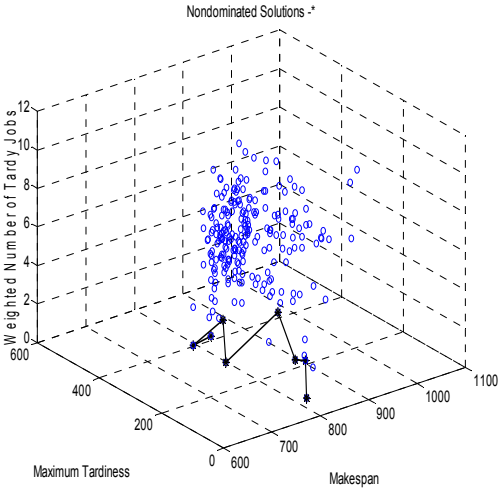

Figure 5-10. Pareto Front for M_LA6

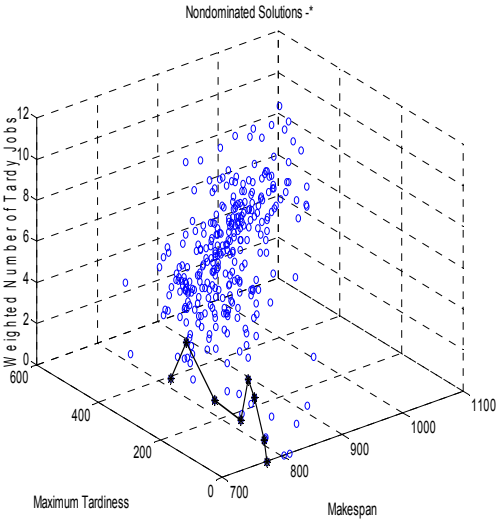

Figure 5-11. Pareto Front for M_LA7

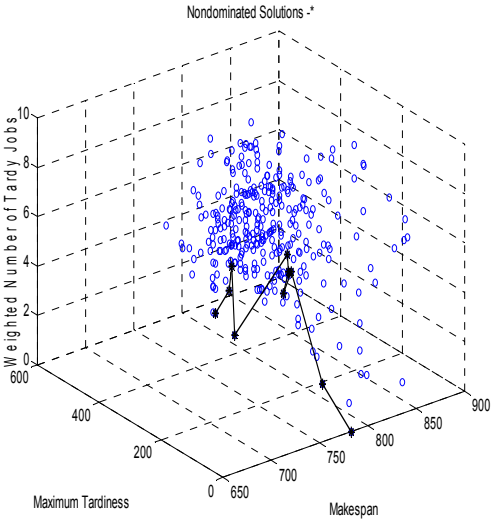

Figure 5-12. Pareto Front for M_LA8

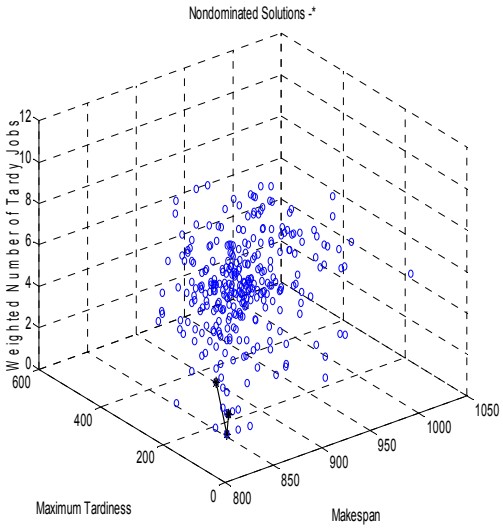

Figure 5-13. Pareto Front for M_LA9

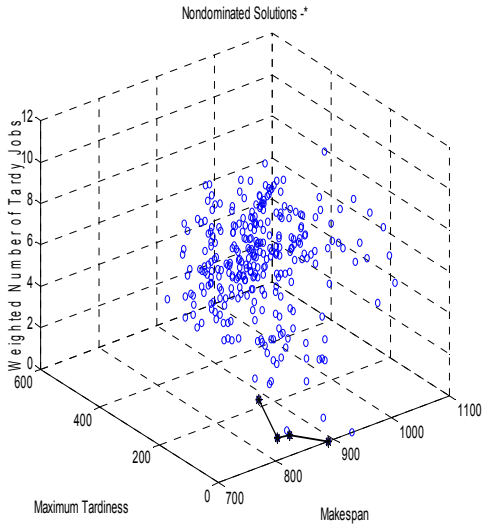

Figure 5-14. Pareto Front for M_LA10

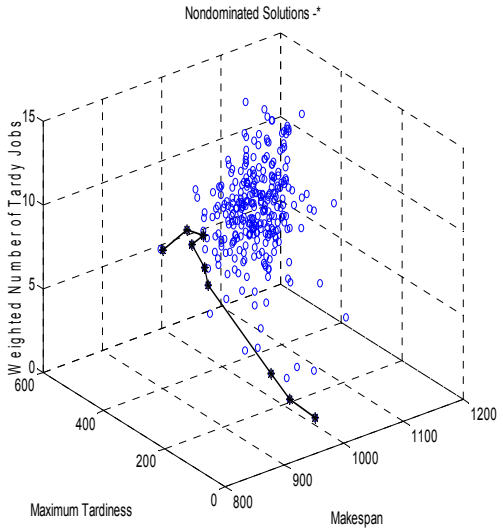

Figure 5-15. Pareto Front for M_LA11

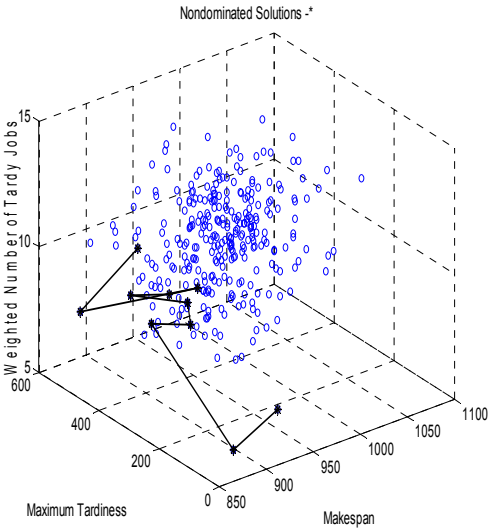

Figure 5-16. Pareto Front for M_LA12

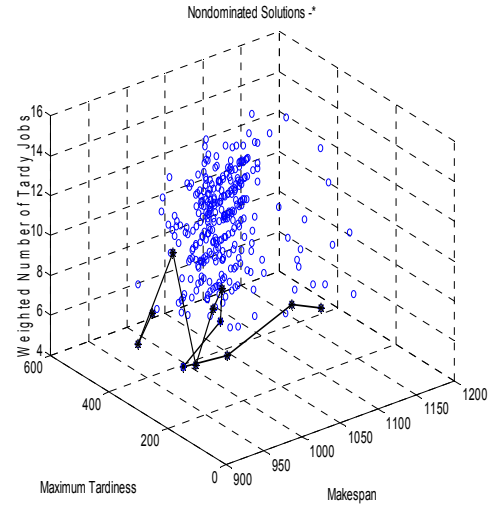

Figure 5-17. Pareto Front for M_LA13

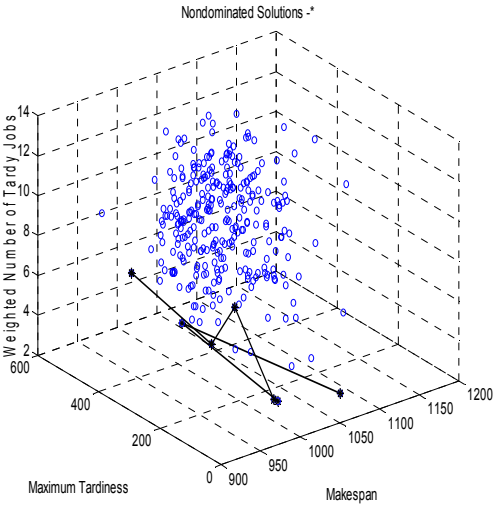

Figure 5-18. Pareto Front for M_LA14

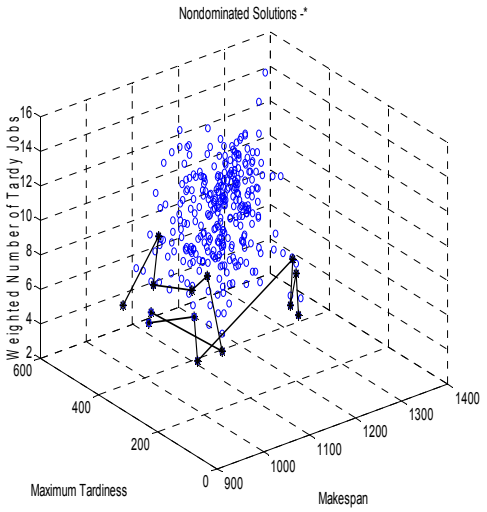

Figure 5-19. Pareto Front for M_LA15

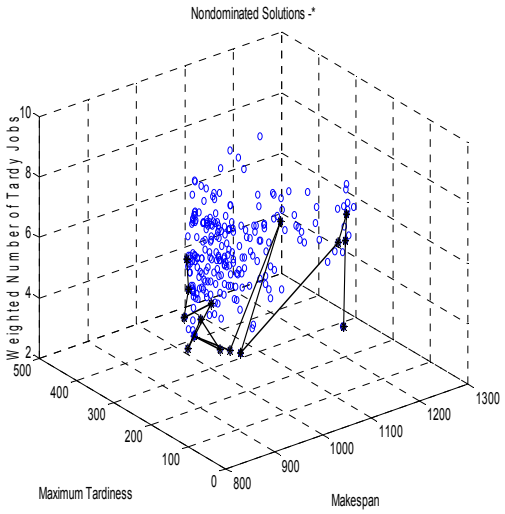

Figure 5-20. Pareto Front for M_LA16

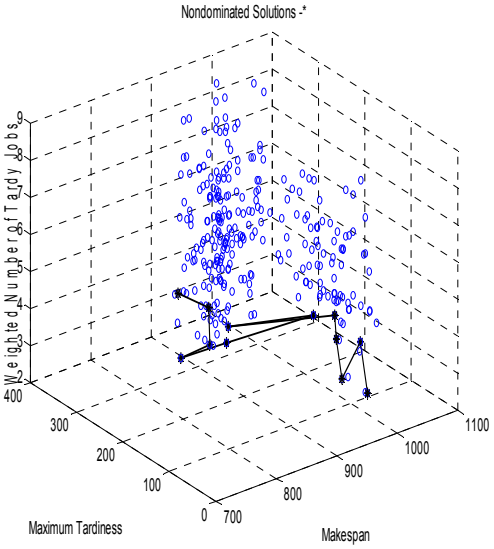

Figure 5-21. Pareto Front for M_LA17

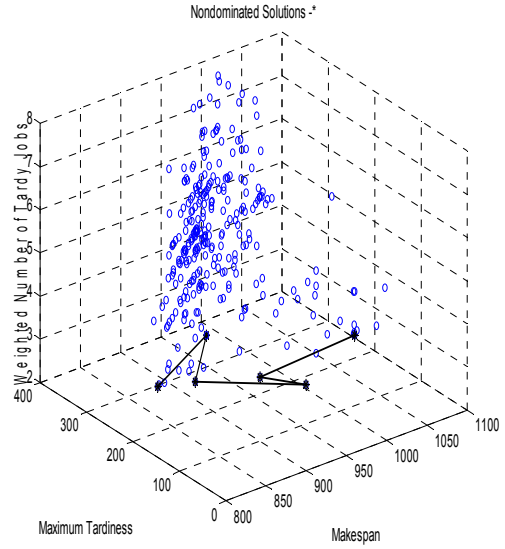

Figure 5-22. Pareto Front for M_LA18

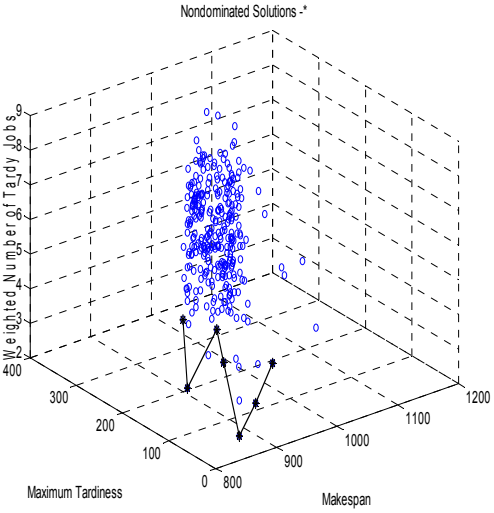

Figure 5-23. Pareto Front for M_LA19

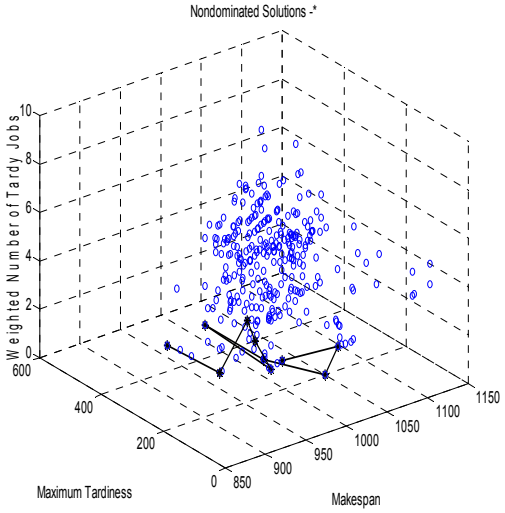

Figure 5-24. Pareto Front for M_LA20

5.3.5.1 Discussion

The 20 modified benchmark JSSPs were tested to validate the system. In each instance the developed GA started by generating 250 random solutions using an operation-based matrix representation to create the initial population. Then the evaluation function was used to obtain the required information, such as job completion time as well as to calculate the objective functions for each solution. To decide which machine to be selected when there are a set of alternative machines, a roulette wheel selection was performed in the evaluation function to select the machine with its associated information, such as machine processing time and setup time. This information was maintained with the chromosome that used it. Different chromosomes can have different routes based on the machines that have been selected.

The ENS-BPSS was applied after each generation and before the reproduction stage to determine the front to which each solution belongs, thus, evaluating the merit of each solution. This process starts with sorting all the solutions in ascending order based on the $C_{\max}$ values, as mentioned in chapter 4. After sorting all solutions, the BPSS was applied to assign solutions from the sorted population, starting from the first solution in the list and ending with the last one, one after another to their fronts. The comparison starts with the last created front so far and ends where it finds its dominant solution. The number of comparisons needed by any solution to be assigned to the front is inversely proportional to the difference between its index and its dominant solution index in the sorted list. The number of comparisons, the number of fronts, as well as the number of members in each front varies from one generation to another and

from one instance to another. Solutions in each front were also sorted in ascending order based on $C_{\max}$ then $T_{\max}$ then $\sum_{i=1}^{n} w_i U_i$.

After assigning each solution to the front that it belongs to, solutions were given fitness values based on their front number. For instance if there is 10 fronts in the population; the fitness values of the solutions in the first front is 10, while the fitness values of the solutions in the last front is 1. This ensures that chromosomes in the higher levels or with better Pareto fronts have a better chance of selection, whereas chromosomes in the same level or same front have an equal chance of selection. New generations of solutions were produced using two-parent / two-row crossover operators with 0.7 probability of crossover and pairwise interchange mutation and with 0.3 probability of mutation. All Solutions in the first front were maintained from the current generation to the next generation.

Evaluation, selection, crossover and mutation were performed repeatedly in each generation until the last generation was reached. The GA terminated after 2000 generations. The obtained results demonstrated the ability of the system to find a set of diverse solutions with or close to a Pareto optimal front. The results also show that the developed system was able to find an optimal point for a single objective in a Pareto front for most of the instances.

As previously stated, three objectives were considered in this research; the $C_{\max}$, the $T_{\max}$ and the $\sum_{i=1}^{n} w_i U_i$. Minimizing the $C_{\max}$ objective aims to increase resources utilization, while minimizing the $T_{\max}$ and the $\sum_{i=1}^{n} w_i U_i$ reflects factors of external cost based on due dates such as customer

satisfaction and they also give an indication of whether the job is completed ahead of, on, or behind its due date. The $T_{\max}$ represents the worst performance in the schedule while the $\sum_{i=1}^{n} w_i U_i$ represents ranks of customers that will not be satisfied. These two objectives are directly influenced by the due date assignment and machines workloads. There are several instances where the resulted schedule has no tardy jobs, thus $T_{\max} = \sum_{i=1}^{n} w_i U_i = 0$, however in all such cases the $C_{\max}$ of the other non-dominated solutions was better (lower). For instance in M_LA1 (Figure 5-5) one of the resulting schedules has the best values of both $T_{\max}$ (0) and $\sum_{i=1}^{n} w_i U_i$ (0) but the worst value of $C_{\max}$ (633) in the Pareto front. The other solutions in the Pareto front all have better (lower) values of $C_{\max}$ than this solution. If the decision maker only wanted to consider customer satisfaction, this schedule has a Utopia Point regarding to $T_{\max}$ and $\sum_{i=1}^{n} w_i U_i$. However when the machine utilization has to also be considered, the decision requires some compromise on the three objectives.

The number of non-dominated solutions that have been found varied in different instances. For instance in M-LA1, 11 non-dominated solutions have been found, while only 4 non-dominated solutions have been found in M-LA2. As the number of non-dominated solutions increase in the Pareto front, so does the level of complexity of the decision itself. The results also indicate that the progress towards the exact Pareto front required a higher number of iterations when the problem size increased. For instance the obtained results of non-dominated solutions for M-LA1 to M-LA5 were found before reaching 65% of the

total number of iterations, whereas in M-LA11 to M-LA15 the non-dominated solutions were found after reaching 75% of the total number of iterations.

Finally, although was not possible to compare the results for MO-JSSP's in this research with any previous results, as the problem has some unique features, it was possible to demonstrate that the proposed system is computationally efficient and has less time complexity for sorting the non-dominated solutions than the current state-of-the-art methods. Even though the developed system achieved encouraging and promising results, there is still room for improvement such as testing different types of selection methods for finding an exact Pareto optimal front. Also machine selection, in the case of alternative machines, has an important role in finding an exact Pareto front; therefore a good heuristic selection method needs to be developed.

5.4 The Effect of Uncertainty on the Optimal Makespan

In a real manufacturing environment, the processing time of each job is usually subject to small changes due to unexpected events. Thus, in most cases, the processing time can be estimated within a certain Interval, which makes the system more realistic. However, these small changes of the processing time can have a significant impact on the optimal schedule. Therefore, in this section the effect of uncertainty on the optimal makespan was studied using two different scenarios. In the first scenario, the ratios of change of the processing times for all operations are considered to be the same. The purpose of this scenario is to show how the optimal sequence can be affected when the processing times change with the same ratio and to identify the ultimate optimistic and the pessimistic value of the makespan within a certain Interval. In

the second scenario the ratio of change was varied from one operation to another, which better reflects more to the real manufacturing environment. In each scenario two cases were considered, one by applying the deviated processing time to the optimal sequence that was found by using the benchmark data set and the other by finding the optimal sequence from the deviated processing time of the benchmark data set. In all cases the lower and upper bounds (or limits) of the change in the processing time were assumed to be $\pm 30\%$ of the given processing time. More details of these scenarios are given below.

5.4.1 Change of Processing Times with the Same Ratio

Two cases were considered for this scenario. In the first case the genetic optimisation process was applied to find the optimal sequence of the optimal makespan for different benchmark problems. The processing times for each operation were then decreased by $\delta_1 = 30\%$ in one set and increased by $\delta_2 = 30\%$ in another set and applied to the optimal sequence to find the deviation from the optimal makespan in both sets. In the second case each set of processing times, after they had been increased or decreased, was used to find the optimal sequences for the deviated processing times. The results from both cases were then compared to each other to show how the optimal sequence can be affected when the processing times of all operations change with the same ratios. Table 5-5 shows the results from 19 benchmark JSSPs with two different cases for the first scenario.

Table 5-5. Change of the Processing Times with the Same Ratio

Instance	Optimal makespan	First case		Second case	
		Deviation by -30%	Deviation by +30%	Deviation by -30%	Deviation by +30%
FT06	55	38.5	71.5	38.5	71.5
FT10	930	651	1209	651	1209
LA1	666	466.2	865.8	466.2	865.8
LA2	655	458.5	851.5	458.5	851.5
LA3	597	417.9	776.1	417.9	776.1
LA4	590	413	767	413	767
LA5	593	415.1	770.9	415.1	770.9
LA6	926	648.2	1203.8	648.2	1203.8
LA7	890	623	1157	623	1157
LA8	863	604.1	1121.9	604.1	1121.9
LA9	951	665.7	1236.3	665.7	1236.3
LA10	958	670.6	1245.4	670.6	1245.4
LA11	1222	855.4	1588.6	855.4	1588.6
LA12	1039	727.3	1350.7	727.3	1350.7
LA13	1150	805	1495	805	1495
LA14	1292	904.4	1679.6	904.4	1679.6
LA15	1207	844.9	1569.1	844.9	1569.1
ABZ5	1234	863.8	1604.2	863.8	1604.2
ABZ6	943	660.1	1225.9	660.1	1225.9

The results show that when the processing times of all operations increased or decreased by the same ratio, ($\pm 30\%$ in this case), the optimal sequence for the optimal makespan in both cases remained the same. The resulting values of the optimal makespan for the deviated processing times in all cases are equal to multiplying $1 \pm 30\%$ by the optimal makespan before deviating the processing times. It has also been noted that; there are a number of different sequences that can lead to the optimal value of makespan in the first case, but not all of these optimal sequences lead to the optimal makespan in the second case.

5.4.2 Change of Processing Times with Different Ratios

Similarly as in the first scenario, the genetic optimisation process was applied to find the optimal sequence of the optimal makespan for different benchmark problems. Then a set of combinations of the processing times was generated

randomly from the $[\delta_1 p_{ij}, \delta_2 p_{ij}]$ interval number; where p_{ij} is the value of the processing time in the benchmark data set, $\delta_1 = 0.70$ and $\delta_2 = 1.3$. In this case the processing time for any operation can be decreased or increased with a random ratio between $-30\% \& +30\%$ of p_{ij}. This randomly generated set was then used in the optimal sequence to find the deviation from the optimal makespan. In the second case, the same set that was generated randomly was used at the start to find the optimal sequences for the deviated processing times. A number of experiments were conducted using several sets to compare between these two cases. Table 5-6 shows the results of 19 benchmark JSSPs using two different sets for both cases in the second scenario.

Table 5-6. Change of the Processing Times with Different Ratios

Instance	Optimal Makespan	First Set			Second Set		
		First Case	Second Case	Ratio of Change	First Case	Second Case	Ratio of Change
FT06	55	60.18	52.733	12.37%	60.57	60.02	0.91%
FT10	930	990.64	981.6	0.91%	0.77	982.82	0.77%
LA1	666	713.18	631.78	11.41%	721.11	688.75	4.48%
LA2	655	676.3	668.17	1.20%	663.62	645.71	2.69%
LA3	597	606.1	584.31	3.59%	597.09	582.29	2.47%
LA4	590	630.22	629.52	0.11%	614.26	614.26	0%
LA5	593	580.81	580.81	0%	613.99	568.63	7.38 %
LA6	926	959.26	897.94	6.39 %	959.56	947.9	1.21%
LA7	890	881.94	833.68	5.47%	870.29	849.59	2.37%
LA8	863	858.66	825.27	3.88%	882.69	860.73	2.48%
LA9	951	1015.2	970.05	4.45%	949.27	906.28	4.53%
LA10	958	915.95	904.55	1.24%	1030.9	961.26	6.75%
LA11	1222	1230.1	1219.7	0.84%	1314.6	1158.6	11.86%
LA12	1039	1147.5	1005.8	12.34%	1087.9	1047.5	3.71%
LA13	1150	1204.7	1189.8	1.23%	1248	1214.8	2.66%
LA14	1292	1308.7	1269.5	2.99%	1343	1258.9	6.26%
LA15	1207	1280.6	1217.9	4.89%	1338.6	1247.8	6.78%
ABZ5	1234	1309.5	1246.4	4.82%	1296.8	1244.2	4.05%
ABZ6	943	1034.1	987.72	4.48%	968.28	944.47	2.45%

The results show that when the processing times of all or some operations change but with different ratios, the optimal sequence also changes in most instances. The ratio of change of the objective function between the first and second cases varies for different combinations of processing times. These values of the makespan for all sets of each instance in the second scenario are still within the range of higher and minimum values of makespan that were found in the first scenario. The ratio of change can be very small, thus its impact on the optimal sequence is insignificant and can be neglected. For instance the ratio of change between the first case and second case in LA11, after applying the processing time uncertainty in the first set of combinations, was 0.84%, which is very small. However, the ratio of change can be high, thus it has a significant impact on the resulting schedule. For instance the ratio of change between the first case and second case in the same instance (LA11), after applying the processing time uncertainty, in the second set of combinations, was 11.86%, which is relatively high. Two main factors are considered to be important here; the ratio of change and the slack time for each operation. In the next section more details are given for time uncertainty with slack time.

5.4.3 Time Uncertainty with Slack Time

The effect of time uncertainty on different operations will have different consequences on the objective function. For any specific schedule, when the time of any operation, that has no slack time, increases or decreases by δ, it will shift all successive operations from the same job, all successive operations in the same machine and their successive operations from the same jobs, and successive of successive and so on, with the same amount of time, δ, unless there is a slack time in the pathway. When an operation has slack time, it will

recompense the delay or the increase of the processing time by as much as the amount of that delay or the increase is not more than the operation slack time. More generally, the amount of change in the objective function for any schedule with respect to time uncertainty will depend on two main factors; time uncertainty factor and slack time for each operation. For any given schedule, the operation slack time can be found from the following equation:

$$F_{ijk} = \min(S_{pqk}, S_{i,j+1}) - C_{ijk}$$

Where: F_{ijk} is the float or slack time for operation j of job i on machine k, S_{pqk} is the start time for the direct successor of operation j of job i on the same machine k, $S_{i,j+1}$ is the start time for the direct successor of operation j from the same job i and C_{ijk} is the completion time for operation j of job i on machine k. To give more illustration let's use La01 from the benchmark dataset. Machine sequence matrix (M) and Time Matrix Machines (T) are given below.

$$M = \begin{bmatrix} 1 & 5 & 4 & 3 & 2 \\ 5 & 3 & 4 & 2 & 1 \\ 3 & 4 & 1 & 2 & 5 \\ 1 & 5 & 4 & 2 & 3 \\ 5 & 3 & 2 & 1 & 4 \\ 1 & 2 & 4 & 5 & 3 \\ 3 & 4 & 1 & 2 & 5 \\ 2 & 5 & 1 & 3 & 4 \\ 3 & 1 & 4 & 5 & 2 \\ 4 & 3 & 2 & 1 & 5 \end{bmatrix} \quad T = \begin{bmatrix} 21 & 53 & 95 & 55 & 34 \\ 21 & 52 & 16 & 26 & 71 \\ 39 & 98 & 42 & 31 & 12 \\ 77 & 55 & 79 & 66 & 77 \\ 83 & 34 & 64 & 19 & 37 \\ 54 & 43 & 79 & 92 & 62 \\ 69 & 77 & 87 & 87 & 93 \\ 38 & 60 & 41 & 24 & 83 \\ 17 & 49 & 25 & 44 & 98 \\ 77 & 79 & 43 & 75 & 96 \end{bmatrix}$$

The optimal makespan for LA01 is depicted in the Gantt chart shown in Figure 5-25.

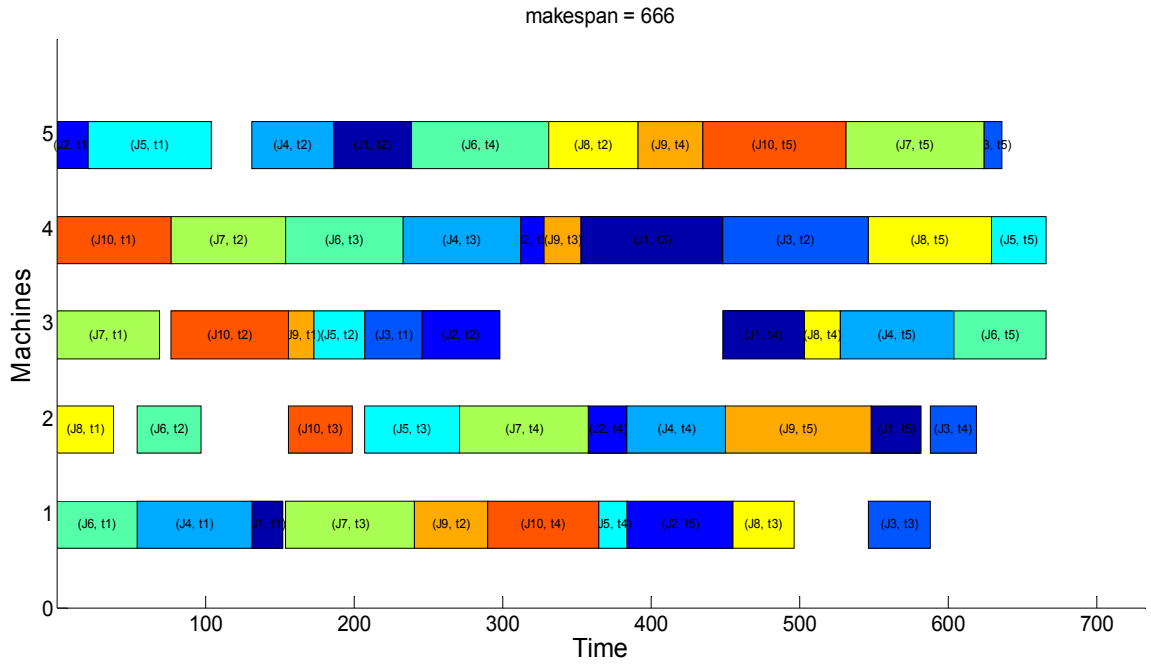

Figure 5-25. Gant Chart of the Optimal Makespan for LA01

By using the above equation for F_{ijk}, the float or resulting slack time for each operation of each job is shown in the matrix below:

$$Ft = \begin{bmatrix} 2 & 0 & 0 & 0 & 6 \\ 0 & 14 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 30 \\ 0 & 0 & 0 & 0 & 0 \\ 27 & 0 & 0 & 0 & 0 \\ 0 & 57 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 \\ 16 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 \end{bmatrix}$$

The F_t matrix shows the tolerance that each operation can have with time uncertainty. For example the second operation of J2 (J2, t2) has a tolerance of time uncertainty equal to 14 time units. This margin of tolerance for time uncertainty will recompense any delay or increase in the processing time with no more than 14 time units. On the other hand, any increase or delay for any operation in machine 4 by δ will increase the makespan by δ. It has been

noted that, some machine idle time is not considered as slack time for the predecessor operation. For example the time from 312 to 448 in machine 3 is idle time but using this time for (J2, t2) by δ will cause of delay on (J2, t3) and consequently increase the makespan by the same amount of time δ. Yet, rearranging (J2, t3) and (J9, t3) on machine 4 so that (J9, t3) is processed before (J2, t3), the slack time of (J2, t2) will increase by 25 time units so that the slack time of (J2, t2) will be equal to 39 time units. Considering such an issue can lead to a more flexible scheduling system. More attention will be given to these issues in the recommendations for future work section in chapter 6.

5.5 Summary

To evaluate and testify the validity of the system several experiments were conducted in this chapter. First, the representation method was tested using 34 published benchmark problems. The results show that the developed system was able to find the optimal solutions for these benchmark problems using operation based matrix representation. 20 instances of these benchmark problems (LA1 to LA20) were then modified to incorporate release dates, setup times, alternative machines and multi-objective to evaluate the practicability and effectiveness of the proposed system for solving the MO-JSSPs with incorporated factors. In all instances ENSGA-BPSS was able to find several non-dominated solutions. The effect of processing time uncertainty on the optimal makespan was also tested in this chapter. Two main factors were found to play an important rule on the optimal solution which are; the ratio of change in the processing time and the slack time for each operation. In the next chapter a summary and conclusions of this research are presented and directions for future research are proposed.

Chapter 6. Conclusions, Contributions to knowledge and Future Work

6.1 Introduction

In this chapter a summary of this PhD thesis, conclusions and contributions to knowledge are given and directions for future research are proposed. In the next section, the research work including the knowledge gaps, methodology, and the outcome of the developed system are summarized. Conclusions of this thesis are given in section 6.3 and contributions to knowledge are stated in section 6.4. The future research directions based on the findings of this research are proposed in the last section 6.5.

6.2 Summary of the Research work

The Job Shop Scheduling Problem (JSSP) is one of the most difficult optimization problems in the area of operation research and scheduling. Over the past decades, a large number of methods have been proposed to solve the problem optimally, yet there is still no efficient method which can guarantee an optimal solution consistently, even for a single criterion, and there is no work to show that any of these methods outperform each other with regards to all problem aspects. Even though the JSSP with a single-objective has been widely studied, the research on Multi-Objectives Job Shop Scheduling Problems (MO-JSSPs) is still relatively limited. Obviously, solving MO-JSSPs is considered to be more complex than solving JSSPs with single-objectives because the objectives are often conflicting or even contradictory. Due to the great difficulty but also necessity, a combination of Genetic Algorithm (GA) and a modified version of a very recent and computationally efficient approach to non-dominated sorting, called Efficient Non-dominated Sort (ENS), was proposed in this research to solve MO-JSSPs. GA was used to lead the search

towards the Pareto optimality, while ENS was applied to determine the front to which each solution belongs to, thus, evaluating the merit of each solution.

In the GA, a new solution representation approach was proposed to represent the solution. This representation method is an adapted version of operation-based representation but instead of using a vector to represent the solution, the solution or chromosome was presented in matrix form and this was termed as operation-based matrix representation. Relating to the solution representation method, three forms of crossover and mutation operators were presented. The proposed method can preserves the features of the parents after the crossover operation without repairing the algorithm. Thirty-four benchmark instances were used to demonstrate the feasibility and practicability of the developed GA with the proposed new representation and to evaluate the performance of the system. The results show that, in all Thirty-four tested benchmark problems of the classical JSSP; the developed GA was able to find the optimal makespan using operation based matrix representation.

In the ENS, a solution in the sorted list that is to be assigned to a front, only needs to be compared with those solutions that have already been assigned to a front, thereby avoiding many unnecessary comparisons. Instead of starting the comparison with the first front, a new strategy called Backward Pass Sequential Strategy (BPSS) was adopted in this research. In BPSS, a solution to be assigned to the front starts the comparison with the last created front so far and ends up where it finds its dominant solution. As with ENS, Efficient Non-dominated Sorting using the Backward Pass Sequential Strategy (ENS-BPSS) has a time complexity of $O(MN^2)$ in the worst case scenario when all N

solutions belong to the same front. The number of comparisons needed for N solutions with M objectives, when there are N fronts and there exists only one solution in each front, was reduced in ENS-BPSS to $O(M(N-1))$. This is because, in this case, each solution in the sorted list is dominated by the direct preceding solution as well as the other preceding solutions. Thus, each solution will be compared only with its direct preceding solution. The obtained results from different instances demonstrate that the proposed method was able to find a set of diverse solutions with or close to the Pareto optimal front.

In order to identify the limitations of the previously developed system for solving MO-JSSPs, a comprehensive literature review was conducted in this research. The review of MO-JSSPs was based on four main factors; the release times, the setup times, the alternative machines and unfixed processing times. The review showed that; amongst fifty three journal articles that were found in the literature, thirty one journal articles (57%) considered alternative machines, eleven journal articles (20%) considered job release date, eight journal articles (15%) considered unfixed processing time and only four journal articles (8%) considered setup times. None of the articles incorporated all four factors or even three factors all together, as thirty eight articles (72%) considered only one factor, mostly the alternative machines, eight articles (15%) considered two factors and seven articles (13%) did not consider any of these factors. Therefore, to close the gap of knowledge, a scheduling system for solving MO-JSSPs that incorporated the release date of jobs and machines, setup times and alternative machines was developed in this research. The setup time that was considered is a type of machine dependent setup time with job sequence independent setup time. Although JSSPs with job sequence dependent setup

time are more complicated, there have still been no studies undertaken on MO-JSSPs with machine dependent setup time and job sequence independent setup time.

To demonstrate the feasibility and practicability of the proposed system, the system was tested with different instances. Although the benchmark sets for JSSPs that are available in the literature provide a common standard and some insight into the strength and performance of the proposed system, there are currently no benchmarks for JSSPs that consider all the aforementioned factors. Therefore, in this research twenty benchmark problems of classical JSSPs were used after they have been modified to suit the intentional problems. Three objectives were used to test the system and to find the Pareto front with non-dominated solutions. The obtained results show that the proposed system was effective and promising. Yet, there is still room to develop a better selection method for finding an exact Pareto front and for alternative machines.

The effect of uncertainty on the optimal makespan was also studied using two different scenarios; first by applying the same ratio of change for all the processing times and second using different ratios of change. The results show that when all processing times change with the same ratio the optimal sequence remains the same and is equal to multiplying $(1+\delta)$ by the optimal makespan before deviating the processing time. However, when the processing times change but with a different ratio, the optimal sequence often changes. Two main factors are considered to be important here; the ratio of change and slack time for each operation.

6.3 Conclusion

In this thesis, a new system that integrates GA and ENS was developed for solving MO-JSSPs. In this developed system, GA was used to lead the search towards the Pareto optimality whilst an ENS was used to determine the front to which each solution belongs. In the proposed GA a new solution representation method, called operation based matrix representation, was presented, which can preserve features of the parent after the crossover operator without repairing the solution. The evaluation function in the GA also takes into account machine setup times, alternative machines and release dates for jobs and machines. In the ENS, a new strategy called BPSS was adopted to determine the front, to which each solution belongs. The best case time complexity of ENS was reduced with BPSS to $O(M(N-1))$. To testify the validity of the system, Thirty-four classical benchmark JSSPs were used initially to clearly demonstrate the feasibility and practicability of the new chromosome representation method within the GA. In all instances, the optimal solutions were found by using operation based matrix representation. Twenty benchmark JSSPs were then modified to include setup times, alternative machines, release dates for jobs and machines and multi-objective. The experimental results show that the proposed system was able to find several nondominated solutions and was effective for solving the Mo-JSSPs with setup times, alternative machines and release dates. Finally, processing time uncertainty was studied in order to identify the most important parameters affecting the scheduling objectives. The study showed that operation slack time and ratio of change in the processing time can have a major impact on the objective functions in the case of processing time uncertainty.

6.4 Contributions

The uniqueness of this research can be summarized as follows.

1. Development of a scheduling system for solving MO-JSSPs that incorporates release dates of jobs and machines, setup times and alternative machines.
2. A modified version of an operation based representation for solving job shop scheduling problems with genetic algorithm was proposed. The proposed representation method uses an operation based representation in the matrix form which can preserve features of the parent after crossover operator without repairing the solution.
3. A combination of Genetic Algorithm (GA) and a modified version of a very recent and computationally efficient approach to non-dominated sort called Efficient Non-dominated Sorting (ENS) have been introduced to solve the Multi-Objective Job Shop Scheduling Problem (MO-JSSP).
4. A new strategy called Back Pass Sequential Strategy (BPSS) was adopted within ENS to determine the front, to which each solution belongs.
5. Using complex numbers in the machine matrix to represent the machine number and setup group for each operation on that machine.
6. The effect of the processing time uncertainty on the optimal makespan was also investigated.

6.5 Future Research Directions

The multi-objective optimization system that has been developed in this thesis is useful for solving MO-JSSPs with deterministic time parameters. In the future, the system can be further developed to solve stochastic MO-JSSPs, where some characteristics of the job are modelled as random variables with separate setup times and machines may be subject to random breakdowns. Also considering the availability of other resources, such as operators and tools, will benefit the overall scheduling system, and this will be taken into account in the future research.

In addition using real data from a job shop manufacturing company to test the developed system is more practical and reflecting to real world manufacturing practice, therefore acquiring such realistic data and using it in the developed system will be considered in the future.

The effect of time uncertainty on MO-JSSPs will also be studied in the future research. Alternative machine selection has been shown to be an important issue for MO-JSSPs as it has a considerable impact on finding an exact Pareto front; therefore, in the future research a more sophisticated method for alternative machine selection should be developed.

Appendices

Appendix 1. Previous research in MO-JSSP

Author(s) and reference	Factor of Describing the JSSP				Approach
	ST	*AM*	*RD*	$\tilde{P}$	
Sakawa and Mori (1999)	-	-	-	✓	On the basis of the agreement index of fuzzy completion time and fuzzy due date for each job GA was proposed after incorporating the fuzzy processing time and fuzzy due date to formulate the fuzzy JSSPs in order to solve fuzzy JSSPs.
Ponnambalam, Ramkumar et al. (2001)	-	-	-	-	MOGA was proposed where each Chromosome was represented based on PDRs using Giffler and Thompson (G&T) procedure. To lead the search in multi direction the weights for merging the objectives into a scalar fitness function were specified randomly in each evaluation and were not constant.
Baykasoğlu, özbakir et al. (2004)	-	✓	-	-	The problem was presented as a grammar and the productions in the grammar are defined as controls. Then MO-TS algorithm was employed using these controls and G&T priority rule-based heuristic to solve the problem.
Low, Wu et al. (2005)	✓	-	-	-	Integer programming model to optimise each single objective was developed and an acceptable trade-off schedule, which made use of multiple-decision-making technique, the global criterion method, was obtained by evaluating three objectives simultaneously.
Xia and Wu (2005)	-	✓	-	-	PSO was used to assign operations on machines and then SA was proposed to schedule operations on each machine in order to solve multi objective flexible JSSP.
Suresh and Mohanasundaram (2006)	-	-	-	-	SA was proposed based on the Pareto dominance or through the implementation of a simple probability function for searching on the non-dominated solution to solve MO-JSSP.
Lei (2008)	-	-	-	✓	JSSP has been converted to a continuous optimization problem in order to apply PSO. Then the proposed algorithm combined the global best position selection with the external Pareto archive set.
Xing, Chen et al. (2009)	-	✓	-	-	A simulation model was presented for optimizing the flexible JSSPs with multiple objectives.
Manikas and Chang (2009)	✓	-	-	-	GA was applied for solving weighted sum MO-JSSPs with sequence-dependent setup times.

Appendix 1. **Continued**

Author(s) and reference	Factor of Describing the JSSP				Approach
	ST	*AM*	*RD*	$\tilde{P}$	
Zhang, Shao et al. (2009)	-	✓	-	-	A hybrid PSO and TS was proposed for optimizing the flexible JSSP with multiple objectives.
Li and Huo (2009)	-	✓	-	-	GA was proposed to solve MO-JSSPs with considering the parallel machines with capacity and speed constraint, maintenance of machines as well as intermediate inventory restriction. The problem was formulated as MIP model to decide the flexible routes for every job and to optimize the sequence of jobs On the basis of the non-linear MIP.
Huang (2010)	✓	-	-	-	ACO was proposed and a heuristic algorithm was generated to rapidly solve the lot-splitting JSSPs with multi objective optimisation.
Adibi, Zandieh et al. (2010)	-	-	✓	-	VNS was proposed for solving multi objective JSSP with random job arrivals and machine breakdowns. To enhance the performance of VNS, weights obtained from ANN at any rescheduling point were used to calculate proper parameters for VNS.
Li, Pan et al. (2010)	-	✓	-	-	A hybrid TS algorithm with two adaptive neighbourhood structures, which builds better local search in the machine assignment component, was developed for solving the multi objective Flexible JSSP. In addition, VNS with three insert and swap neighbourhood structures was presented to perform local search in the operation scheduling part.
Wang, Gao et al. (2010)	-	✓	-	-	An improved GA based on immune and entropy principle was proposed to solve the multi objective flexible JSSPs. The applied fitness scheme was based on the Pareto optimality.
Sha and Lin (2010)	-	-	-	-	PSO was used after modifying the particle position representation, particle movement and particle velocity in order to solve MO-JSSP.
Moslehi and Mahnam (2011)	-	✓	✓	-	A Pareto approach and an integrated method based on a hybridization of PSO and local search algorithm was applied for solving multi-objective flexible JSSPs. PSO was employed to allow a wide search of solution space while the local search algorithm was employed to reschedule the results achieved by the PSO, to increase convergence speed.

Appendix 1. **Continued**

Author(s) and reference	Factor of Describing the JSSP				Approach
	ST	*AM*	*RD*	$\tilde{P}$	
Zheng, Li et al. (2011)	-	✓	-	✓	A multi objective swarm-based neighborhood search was proposed to solve fuzzy flexible JSSP. Two swaps and an insertion were applied to produce new solutions and simple weighted objective-based methods were used to update swarm and external archive in order to obtain a set of non-dominated solutions.
Kachitvichyanukul and Sitthitham (2011)	-	-	-	-	A two stage GA was proposed. In the first stage parallel GA was applied to find the best solution for each objective individually with migration among populations. In the second stage the populations were combined all together and the final schedule was identified based on the weighted aggregating objective function.
Li, Pan et al. (2011)	-	✓	-	-	A hybrid Pareto-based discrete ABC was presented to solve multi objective flexible JSSP. To record the non-dominated solutions, an external Pareto archive set was introduced. In addition, a fast Pareto set update function was developed to reduce the computational times.
Tavakkoli-Moghaddam, Azarkish et al. (2011)	✓	-	✓	-	A hybrid PSO and VNS based on Pareto archive was proposed and character of scatter search to select new swarm in each iteration was employed in order to find Pareto optimal solutions for bi-objective JSSP with sequence-dependent setup times.
Ramkumar, Tamilarasi et al. (2012)	-	-	-	✓	FL for solving multi objective fuzzy JSSP was proposed where a triangular fuzzy membership function was used to represent customer priority and due date with the aim of maximizing the minimum agreement index, maximizing the average agreement index and minimizing the maximum fuzzy completion time.
Wang, Zhou et al. (2012)	-	✓	-	-	An enhanced Pareto-based ABC algorithm was presented to solve the multi objective flexible JSSP.
Li, Pan et al. (2012)	-	✓	-	-	A hybrid Pareto-based local search algorithm was developed to solve multi criteria flexible JSSP. An external Pareto archive set was used to record the non-dominated solutions.
Li, Pan et al. (2012)	-	✓	-	-	A very recently method known as shuffled frog-leaping algorithm was proposed with two crossover operators for solving multi objective flexible JSSP.

Appendix 1. **Continued**

Author(s) and reference	Factor of Describing the JSSP				Approach
	ST	*AM*	*RD*	$\tilde{P}$	
Frutos and Tohmé (2012)	-	-	-	-	A multi objective memetic algorithm was introduced for the treatment of the JSSP combining a multi Objective evolutionary algorithm and multi objective SA.
Dalfard and Mohammadi (2012)	-	✓	✓	-	A hybrid GA and a SA algorithm were proposed to solve multi objective flexible JSSP with parallel machines and maintenance constraints.
Lei (2012)	-	-	-	✓	A multi objective ABC was proposed for solving interval JSSP with non-resumable jobs and flexible preventive maintenance.
Rahmati, Zandieh et al. (2012)	-	✓	-	-	Non-dominated sorting genetic algorithm and non-dominated ranking genetic algorithm were adapted for solving multi objective flexible JSSP. These two algorithms used new multi objective Pareto-based modules instead of multi-criteria concepts to guide their process.
Zhang, Gao et al. (2013)	-	-	✓	-	A rescheduling method based on the hybrid GA and TS was introduced to solve multi objective JSSP with random job arrivals and machine breakdowns.
Wang, Wang et al. (2013)	-	✓	-	-	A Pareto-optimality-based fitness evaluation was employed and a probability model with the Pareto superior population was designed to solve multi objective flexible JSSP.
Shahsavari-Pour and Ghasemishabankareh (2013)	-	✓	-	-	A new hybrid GA and SA that using Pareto optimal solution approach in its process was introduced to solve multi objective flexible JSSP.
Shao, Liu et al. (2013)	-	✓	-	-	A hybrid PSO and SA was proposed to solve the multi objective flexible JSSP. In the proposed method, Non-dominated solutions were stored by using best position of particles.
Qiu and Lau (2013)	-	-	✓	-	An artificial immune systems that dynamically select the most appropriate PDRs for the jobs waiting for an available machine was developed for solving multi objective dynamic online JSSP.
Niu, Ong et al. (2013)	-	-	-	-	A new meta-heuristic algorithm, namely the Intelligent Water Drops algorithm was employed for solving multi objective JSSP.
Gao, Suganthan et al. (2014)	-	✓	-	-	A Pareto-based grouping discrete harmony search algorithm was proposed to solve bicriteria flexible JSSPs.

Appendix 1. **Continued**

Author(s) and reference	Factor of Describing the JSSP				Approach
	ST	*AM*	*RD*	$\tilde{P}$	
Li, Pan et al. (2014)	-	✓	-	-	A hybrid ABC algorithm and TS based on the Pareto archive set to record the non-dominated solutions was proposed for solving the multi objective flexible JSSPs with preventive maintenance activities.
Zhao, Tang et al. (2014)	-	-	✓	-	An improved multi objective PSO with decline disturbance index was presented to solve MO-JSSP. The decline disturbance index was used to improve particles ability for exploring the local and global optimization solutions, as well as decreasing the probability of being trapped into the local optima.
Su, Mengjie et al. (2014)	-	-	✓	✓	NSGA II, SPEA2, and harmonic distance-based multi-objective evolutionary algorithm were employed to solve MO-JSSP. In addition, a new method called diversified multi objective cooperative evolution was also proposed.
Gao, Suganthan et al. (2014)	-	✓	-	-	A Pareto-based grouping discrete harmony search algorithm was proposed to solve bicriteria flexible JSSPs.
Jia and Hu (2014)	-	✓	-	-	A new path-relinking algorithm based on the TS algorithm with back-jump tracking was proposed for solving multi objective flexible JSSPs.
Hosseinabadi, Siar et al. (2014)	-	✓	✓	-	A new method called TIME_GELS that uses the gravitational emulation local search algorithm was proposed to solve the multi objective flexible dynamic JSSP.
Xue, Zhang et al. (2014)	-	✓	-	-	A quantum immune algorithm based on the quantum and immune principles was presented to solve multi objective flexible JSSP.
Karthikeyan, Asokan et al. (2014)	-	✓	-	-	A discrete firefly algorithm was adopted to solve multi objective flexible JSSP, in which the operation sequence and machine assignment are treated by building an appropriate conversion of the continuous functions as attractiveness, distance and movement, into new discrete functions. In addition, local search algorithm with neighbourhood structures was hybridised to improve the exploitation ability.
Pérez and Raupp (2014)	-	✓	-	-	A new hierarchical approach was proposed to solve the multi objective flexible JSSP. In this method each new iterated solution improves all the objective functions simultaneously.

Appendix 1. **Continued**

Author(s) and reference	Factor of Describing the JSSP				Approach
	ST	*AM*	*RD*	$\tilde{P}$	
Yang and Gu (2014)	-	-	-	-	GA and TS were incorporated in the frame of a new cultural algorithm to search for the Pareto-optimal solution in order to solve MO-JSSPs
Shen and Yao (2015)	-	✓	✓	-	A multi objective evolutionary algorithm-based proactive-reactive method was developed in order to solve the multi objective dynamic flexible JSSP, and provide different trade-offs between different objectives.
Huang and Süer (2015)	-	-	-	✓	A dispatching rule based genetic algorithm with fuzzy satisfaction levels was proposed to solve the multi objective JSSP.
Shivasankaran, Kumar et al. (2015)	-	✓	✓	-	Hybrid sorting immune SA technique was proposed for solving the multi objective flexible JSSP.
Zhao, Gao et al. (2015)	-	✓	-	-	A two generation ACO for solving the multi objective flexible JSSP with alternative process plans and unrelated parallel machines was proposed. The Pareto ACO built the applicable pheromone matrixes and heuristic information with respect to the flexible processing route decision and task sorting, then objectives and NSGAII was used for comparison.
Singh, Singh et al. (2015)	-	✓	-	-	A new PSO algorithm for solving multi-objective flexible JSSPs was proposed.
Zhang and Chiong (2016)	-	-	-	✓	A multi objective GA incorporated with two local improvement strategies was proposed to solve a MO-JSSP. These local improvement strategies aim to enhance the solution quality by utilizing the mathematical models of the two subproblems derived from the original problem.
Kaplanoğlu (2016)	-	✓	-	-	An object-oriented approach along with SA optimization algorithm was proposed for solving multi objective Flexible JSSP.

Appendix 2. Optimal sequence for C_{max} using operation based matrix representation

Problem	Optimal sequence
FT06	2 3 6 5 1 4 1 3 6 2 4 5 1 4 3 5 2 6 3 4 1 6 2 5 2 5 6 4 1 3 2 4 5 3 6 1
FT10	4 7 5 8 10 6 2 9 1 3 6 10 4 7 9 2 1 5 3 8 1 9 6 4 3 5 7 10 8 2 6 4 8 5 3 7 9 2 1 10 2 4 7 8 1 10 6 3 5 9 6 10 9 2 1 5 3 7 4 8 7 5 3 1 10 6 8 4 9 2 4 6 5 9 7 8 10 3 2 1 9 6 2 7 3 1 5 8 4 10 5 4 2 6 8 9 1 7 10 3
ABZ5	2 5 7 8 1 10 3 6 4 9 3 1 2 8 5 7 4 10 9 6 5 6 3 2 9 1 4 8 7 10 3 5 8 10 4 6 2 7 9 1 2 8 10 3 9 4 6 5 1 7 4 1 7 10 5 8 9 6 3 2 6 2 3 4 8 7 9 1 5 10 8 4 9 3 5 2 1 6 10 7 5 9 6 1 8 4 3 7 10 2 6 7 8 5 9 10 2 1 3 4
ABZ6	3 4 6 1 5 7 2 9 8 10 1 5 3 2 4 8 6 7 9 10 2 4 7 9 1 10 3 5 8 6 5 6 9 1 10 2 8 4 3 7 7 3 6 1 8 10 4 2 5 9 5 2 3 6 4 8 1 9 10 7 1 5 6 2 4 9 10 7 3 8 2 8 3 10 7 5 9 4 1 6 1 8 2 3 5 4 9 6 10 7 8 1 5 3 6 7 4 2 10 9
La01	2 10 7 9 6 8 4 1 5 3 6 4 3 8 5 10 9 2 1 7 1 4 5 7 2 8 6 3 9 10 6 7 1 10 2 5 9 8 3 4 5 2 8 9 6 7 10 4 1 3
La02	1 5 2 6 3 7 10 4 8 9 5 7 8 2 10 1 3 4 9 6 1 3 2 4 7 9 8 10 5 6 2 3 5 9 1 10 7 6 8 4 2 9 3 5 8 6 1 4 7 10
La03	1 7 2 4 5 3 9 6 8 10 2 4 1 10 6 9 5 8 7 3 2 3 10 7 6 9 1 4 5 8 4 7 8 9 2 5 1 3 6 10 3 8 6 4 7 2 9 5 1 10
La04	2 3 5 9 10 1 8 4 6 7 1 9 10 8 5 3 6 7 2 4 2 9 3 6 10 5 4 1 7 8 2 5 6 4 7 3 9 10 8 1 3 6 8 9 10 4 5 7 1 2
La05	2 3 1 4 6 5 8 7 9 10 2 4 5 6 3 9 10 1 7 8 1 2 5 3 7 4 8 10 6 9 4 5 1 2 6 7 3 9 8 10 3 4 2 7 5 6 10 9 1 8
La06	1 2 10 3 5 4 8 9 11 14 6 15 12 7 13 1 2 4 5 6 9 10 8 13 14 12 7 11 15 3 5 1 2 3 8 14 4 6 7 11 9 10 13 15 12 2 4 3 7 1 8 14 10 15 13 12 11 6 9 5 3 4 5 1 7 2 15 11 12 13 14 8 10 9 6

***Appendix 2*. Continued**

Problem	Optimal sequence
La07	3 2 12 4 1 6 10 7 9 13 11 8 14 15 5 1 2 5 3 4 6 7 8 12 10 14 15 9 13 11 1 3 4 2 8 9 7 6 13 15 10 11 14 12 5 1 14 2 4 10 13 3 5 9 6 7 12 11 8 15 3 9 1 7 11 14 12 10 4 15 13 5 6 2 8
La08	6 7 4 5 9 12 1 13 15 8 11 2 14 10 3 1 9 5 7 4 11 15 10 6 12 14 13 8 3 2 4 2 3 11 1 5 8 10 15 13 9 14 6 7 12 2 8 1 4 9 7 5 14 6 12 3 15 13 11 10 3 5 12 4 13 7 9 1 10 14 15 8 11 2 6
La09	1 5 8 6 9 2 3 13 4 7 15 11 12 10 14 2 1 9 15 3 6 7 5 10 12 13 8 14 11 4 3 4 1 2 5 6 8 10 9 11 7 13 14 12 15 2 5 12 1 10 11 14 6 15 3 13 7 4 8 9 4 3 6 15 11 1 2 7 5 9 12 14 8 13 10
La10	2 5 6 1 3 9 11 10 12 7 13 15 8 4 14 1 5 6 2 3 4 12 8 9 14 7 11 10 13 15 3 6 1 5 2 4 7 11 8 9 10 12 13 15 14 2 4 1 6 5 13 8 14 10 15 3 7 9 11 12 3 5 4 1 6 9 7 12 8 11 14 2 15 10 13
La11	1 3 4 2 5 8 9 6 14 17 19 7 20 12 15 18 11 16 13 10 1 2 3 7 6 9 8 11 12 14 20 13 18 19 4 15 10 16 5 17 3 6 1 2 13 5 7 10 8 18 19 16 14 11 15 20 4 12 17 9 2 5 4 6 18 7 3 9 11 12 15 8 19 10 17 16 14 13 20 1 2 1 5 6 8 3 7 11 13 15 17 20 9 14 16 18 12 19 10 4
La12	7 2 9 5 11 3 1 14 10 8 17 19 12 20 16 18 4 15 13 6 2 3 16 1 6 7 8 9 10 13 12 11 14 17 19 5 18 20 15 4 1 3 4 8 11 14 19 2 5 7 15 16 9 20 18 10 6 12 13 17 2 3 7 4 17 1 10 16 15 19 13 9 11 5 20 14 6 18 8 12 4 11 1 2 16 3 8 5 14 18 7 6 12 13 20 10 19 17 15 9
La13	1 5 6 2 4 7 3 8 13 18 14 19 16 12 20 10 15 9 17 11 2 6 5 3 4 14 9 7 16 12 15 18 8 11 17 13 1 19 20 10 3 8 1 5 4 2 10 12 6 16 17 14 20 18 13 19 7 11 15 9 5 10 4 7 6 8 15 2 16 3 13 17 9 1 11 20 14 19 12 18 4 3 1 6 8 13 19 11 5 9 20 10 14 12 15 17 2 18 16 7
La14	1 2 9 6 8 11 18 12 17 5 19 16 3 20 14 15 10 13 7 4 1 2 8 17 16 5 15 14 4 13 11 9 18 6 10 20 12 19 3 7 1 6 18 13 3 14 7 20 11 16 5 4 2 12 19 17 15 8 10 9 3 5 7 10 12 14 9 17 20 18 13 6 11 15 8 16 4 19 2 1 1 14 2 11 3 20 4 13 9 8 6 15 19 16 10 5 18 12 17 7
La15	11 15 14 17 13 2 9 16 12 19 20 1 7 6 8 5 4 18 3 10 2 4 12 5 13 14 19 9 11 20 3 16 8 17 18 15 10 1 7 6 3 12 10 15 11 4 7 20 18 5 8 19 13 1 2 16 6 9 14 17 1 3 9 19 15 12 10 8 13 16 2 11 17 14 18 6 7 4 5 20 3 16 1 18 19 15 11 5 10 17 12 4 2 14 7 20 13 9 8 6
La16	1 2 3 4 7 10 5 9 8 6 3 6 7 2 9 10 1 8 4 5 1 2 3 5 7 6 10 4 9 8 6 10 1 2 8 4 5 3 9 7 6 3 4 8 7 5 9 1 2 10 8 10 7 1 2 3 5 9 6 4 2 8 10 5 3 1 4 6 7 9 4 2 5 8 6 1 7 9 10 3 3 1 7 10 5 4 6 2 8 9 1 6 7 4 3 8 9 10 5 2
La17	1 4 2 6 8 3 9 5 10 7 2 8 3 4 5 7 6 1 9 10 1 4 7 8 3 2 9 10 5 6 1 5 7 6 8 10 2 3 9 4 3 4 5 6 1 10 7 9 2 8 3 8 4 5 7 2 1 9 6 10 2 7 5 4 9 1 3 8 6 10 6 2 8 10 4 9 1 7 5 3 1 7 9 2 3 6 10 4 8 5 5 6 7 9 1 2 3 8 4 10

***Appendix 2*. Continued**

Problem	Optimal sequence
La18	1 5 3 8 7 10 2 9 6 4 3 4 6 8 7 9 5 1 2 10 5 10 7 3 4 1 6 8 2 9 9 1 3 5 10 2 6 4 7 8 7 3 4 8 6 5 2 1 10 9 5 7 6 2 1 3 9 8 4 10 5 2 9 10 1 3 6 8 4 7 9 2 5 3 6 7 4 10 8 1 1 10 3 2 7 5 4 6 9 8 6 2 8 7 1 3 9 10 5 4
La19	4 8 9 1 3 5 7 10 2 6 2 4 9 7 10 1 3 5 6 8 5 8 4 10 3 9 6 7 1 2 1 4 8 9 2 3 5 10 7 6 1 2 3 5 8 6 7 4 9 10 2 5 6 10 9 4 7 3 1 8 2 10 6 1 7 8 9 4 5 3 6 1 8 3 4 10 5 9 7 2 8 7 10 3 4 9 2 1 5 6 9 6 4 7 10 1 2 8 5 3
La20	7 10 1 2 6 9 4 8 3 5 6 5 9 10 1 4 2 7 8 3 10 1 8 5 3 7 2 6 4 9 1 4 3 5 7 9 8 2 10 6 5 7 2 6 9 1 4 10 8 3 6 2 5 8 10 1 4 3 9 7 6 10 4 2 9 7 1 5 3 8 2 6 7 4 8 1 9 3 5 10 3 1 8 10 5 7 9 4 2 6 3 1 9 5 10 2 6 8 4 7
La31	Row1 4 7 15 19 9 12 18 17 1 28 14 5 24 6 21 29 20 25 27 22 26 3 13 8 10 16 23 2 11 30 Row2 2 5 8 16 1 17 22 9 14 18 27 11 4 20 12 25 7 23 24 10 28 29 15 21 19 30 26 13 3 6 Row3 11 19 4 2 15 5 18 3 22 8 26 21 9 12 30 23 28 25 20 24 14 10 16 29 27 17 6 13 7 1 Row4 15 24 9 8 22 13 23 21 12 5 29 2 30 10 18 27 1 14 19 4 11 28 3 17 7 26 20 6 25 16 Row5 4 5 17 21 8 6 11 7 2 15 10 22 23 28 3 9 14 19 26 16 18 13 25 20 1 29 27 30 12 24 Row6 2 18 4 9 1 28 14 15 20 7 24 5 26 25 27 11 21 13 12 19 17 23 8 29 22 10 30 16 3 6 Row7 20 23 1 2 8 6 13 12 3 4 16 29 18 9 19 21 17 30 7 22 26 27 24 14 11 28 25 5 15 10 Row8 3 6 29 13 10 20 28 16 8 19 7 12 14 22 2 9 26 5 25 15 21 23 18 30 11 4 17 27 1 24 Row9 1 3 7 15 6 4 13 19 17 8 23 2 25 24 10 28 16 18 21 30 27 14 12 9 11 29 20 5 26 22 Row10 1 8 21 29 2 5 7 6 19 9 3 11 14 4 16 30 10 18 20 24 22 26 13 28 15 25 23 27 12 17

***Appendix 2*. Continued**

Problem	Optimal sequence									
La32	Row1	2	3	9	14	29	25	19	28	7	11
		5	30	15	12	24	18	10	23	6	22
		26	13	1	20	16	21	27	17	4	8
	Row2	15	7	10	6	26	2	1	11	12	14
		17	24	18	23	29	9	20	4	28	8
		21	13	30	19	27	5	25	3	16	22
	Row3	26	4	7	16	19	15	18	20	2	6
		9	1	22	25	21	27	8	29	11	30
		23	3	5	10	14	24	13	12	17	28
	Row4	6	7	15	27	5	12	9	10	13	3
		25	23	1	22	18	16	19	21	2	14
		28	17	4	29	30	11	24	26	8	20
	Row5	8	27	15	12	14	21	23	29	5	16
		1	24	17	4	7	22	3	9	18	10
		20	2	11	28	26	13	6	25	30	19
	Row6	1	13	17	22	4	10	3	2	26	7
		8	28	27	21	9	14	5	11	30	6
		15	19	23	25	24	29	12	16	20	18
	Row7	10	24	3	15	11	17	19	1	16	27
		20	21	25	14	2	28	6	29	12	22
		7	30	13	18	9	23	26	5	8	4
	Row8	3	22	12	28	29	16	27	8	18	6
		7	11	2	19	13	4	15	30	26	17
		5	25	23	1	21	10	9	20	14	24
	Row9	19	26	5	1	27	3	30	15	25	14
		6	24	11	2	16	10	28	18	23	9
		17	20	29	12	7	22	8	21	4	13
	Row10	7	5	14	11	18	25	6	9	20	22
		21	30	23	27	8	1	4	13	16	26
		10	24	15	3	28	12	2	29	17	19
La33	Row1	14	24	25	8	11	20	4	5	18	29
		26	2	13	30	1	9	22	3	10	7
		21	19	16	17	23	15	12	27	6	28
	Row2	18	1	8	6	23	10	28	4	3	30
		7	13	17	15	22	19	9	24	16	14
		11	29	21	20	2	26	25	5	27	12
	Row3	4	5	14	16	1	19	22	26	9	7
		25	15	6	18	11	12	13	20	23	27
		30	2	10	17	28	21	8	3	29	24
	Row4	9	10	12	13	19	16	17	15	20	28
		3	8	4	23	18	24	7	6	29	14
		5	25	30	1	2	11	21	26	22	27
	Row5	1	2	18	4	23	8	3	9	16	12
		27	20	14	29	30	19	6	26	15	13
		7	25	24	11	21	28	22	17	5	10
	Row6	5	12	19	22	9	1	7	2	13	21
		11	8	24	27	3	25	6	10	20	26
		28	4	23	30	14	16	29	15	18	17
	Row7	2	6	8	12	9	22	4	10	13	15
		17	21	1	16	14	26	19	7	28	30
		3	29	18	11	27	5	24	20	23	25
	Row8	12	17	25	3	21	23	30	14	5	2
		20	11	22	24	9	15	16	27	28	1
		10	29	6	26	7	19	4	13	8	18
	Row9	3	22	10	11	15	25	8	12	23	19
		27	24	7	17	26	18	30	29	13	20
		6	4	9	16	5	2	28	14	21	1
	Row10	3	7	4	8	18	10	11	23	1	12
		27	28	21	5	14	25	30	22	9	6
		17	13	19	20	24	29	26	16	2	15

***Appendix 2*. Continued**

Problem	Optimal sequence										
La34	Row1	7	9	19	3	6	20	4	1	8	5
		10	12	11	23	30	25	14	13	24	21
		16	18	2	15	17	26	22	27	28	29
	Row2	1	4	22	23	27	5	3	12	29	8
		6	10	11	7	17	15	2	19	26	14
		28	25	30	18	9	16	20	21	24	13
	Row3	2	8	16	20	7	12	14	25	3	4
		13	18	17	21	5	24	6	23	15	19
		11	29	22	30	27	1	9	10	26	28
	Row4	3	9	17	8	16	13	21	28	1	6
		14	27	15	5	7	10	18	4	12	23
		24	22	20	26	25	29	11	2	30	19
	Row5	6	2	10	5	1	7	26	12	13	14
		17	4	20	25	28	9	21	16	23	30
		8	29	11	19	3	15	18	22	24	27
	Row6	17	7	4	21	8	22	10	9	25	26
		29	3	6	13	19	20	16	28	30	15
		18	5	24	23	2	12	11	14	27	1
	Row7	7	14	21	1	29	2	15	16	20	3
		12	24	10	8	4	26	30	5	9	6
		19	13	22	27	11	23	17	25	28	18
	Row8	5	9	17	2	11	13	18	23	25	27
		7	21	15	24	16	26	28	4	19	8
		6	1	30	12	10	3	14	20	29	22
	Row9	7	1	9	29	6	2	8	3	10	14
		25	19	17	18	20	4	11	22	16	23
		24	5	15	21	28	12	26	30	27	13
	Row10	12	2	11	28	3	22	1	7	17	15
		6	13	14	4	21	9	10	18	25	23
		5	27	20	16	30	19	24	29	8	26
La35	Row1	2	7	13	12	18	19	8	27	29	20
		21	22	14	17	24	30	28	11	5	10
		4	3	23	1	6	9	15	16	26	25
	Row2	5	28	1	19	6	8	20	21	2	11
		22	10	3	27	14	25	16	7	9	15
		12	17	18	24	23	29	30	26	13	4
	Row3	3	5	8	9	11	7	21	1	19	23
		16	17	12	24	10	29	15	25	27	2
		6	13	30	4	22	18	26	20	28	14
	Row4	1	8	3	10	5	12	11	6	24	21
		19	7	13	16	18	28	22	14	4	17
		20	26	2	23	9	30	15	25	27	29
	Row5	3	8	9	11	20	10	16	12	13	17
		29	14	18	27	2	5	19	30	4	22
		6	24	26	21	15	25	1	7	28	23
	Row6	3	13	1	16	8	9	22	10	4	24
		14	11	21	23	15	7	6	17	18	26
		30	20	2	19	27	25	12	28	29	5
	Row7	4	5	11	24	12	7	15	25	26	29
		2	28	3	22	6	8	9	10	13	16
		14	19	20	30	1	17	23	21	27	18
	Row8	5	6	24	3	9	7	14	18	21	23
		20	22	27	11	28	30	29	2	8	13
		15	26	16	10	17	19	25	4	12	1
	Row9	13	1	10	5	6	8	14	16	12	20
		29	15	27	4	2	9	11	17	28	19
		23	7	18	22	24	3	30	21	26	25
	Row10	4	10	18	14	16	12	7	27	15	23
		11	17	20	19	21	28	22	2	6	3
		5	8	26	13	25	24	1	9	30	29

***Appendix 2*. Continued**

Problem	Optimal sequence									
swv16	Row1	20	22	21	30	5	6	26	45	34	18
		15	43	16	31	23	28	35	41	32	3
		46	9	19	33	44	49	13	8	42	36
		37	11	25	27	14	29	10	4	12	2
		24	39	50	38	48	47	7	1	40	17
	Row2	1	9	7	33	38	44	26	13	11	46
		19	37	28	14	35	30	40	10	18	23
		12	39	8	6	25	16	2	48	15	32
		31	21	22	5	20	45	29	34	47	27
		17	4	3	49	36	42	24	41	43	50
	Row3	11	20	21	12	23	34	6	29	30	43
		24	19	26	15	10	46	9	49	40	45
		42	47	16	17	31	36	5	33	44	48
		8	4	2	35	32	1	27	38	50	3
		39	37	22	28	7	41	14	25	13	18
	Row4	3	4	14	12	26	30	21	37	50	17
		1	18	19	33	16	25	11	29	9	23
		41	40	42	39	32	49	2	13	47	46
		10	31	28	27	34	22	7	15	48	43
		44	38	36	5	20	24	35	8	6	45
	Row 5	29	1	17	4	39	13	49	14	23	47
		28	44	40	32	3	26	43	30	18	16
		10	35	38	20	11	46	42	50	22	7
		24	33	37	48	12	27	21	2	15	9
		45	31	41	19	6	34	8	36	25	5
	Row6	3	20	6	48	12	10	46	47	16	22
		31	26	39	29	4	1	28	36	9	42
		33	38	23	24	2	44	13	17	30	5
		7	35	27	18	43	49	50	45	34	32
		21	11	14	37	40	15	19	41	25	8
	Row7	3	10	5	16	1	22	23	43	29	31
		25	17	32	33	38	42	48	13	8	19
		34	36	2	50	39	40	7	21	18	15
		11	4	9	44	46	49	24	26	37	30
		27	45	47	35	20	28	41	12	6	14
	Row8	11	6	19	9	14	43	28	18	47	41
		27	29	24	50	31	16	20	17	39	32
		23	8	37	26	12	34	33	1	45	5
		22	2	40	48	10	49	44	7	25	13
		42	3	15	46	36	4	30	35	38	21
	Row9	1	25	6	9	43	44	34	32	31	38
		18	35	40	17	23	13	11	49	42	27
		30	12	41	2	50	19	4	36	48	28
		47	3	14	45	7	39	5	8	20	29
		22	46	24	10	26	15	33	16	37	21
	Row10	19	5	20	10	28	24	26	47	1	2
		7	35	49	11	6	39	31	32	13	17
		21	46	25	38	8	29	16	41	27	34
		9	18	37	14	3	43	48	22	42	40
		12	30	44	15	33	36	23	50	45	4

***Appendix 2*. Continued**

Problem	Optimal sequence									
swv17	Row1	3	7	14	29	8	24	33	25	22	16
		27	32	43	49	12	17	30	1	38	34
		35	31	44	15	37	41	20	19	6	28
		45	40	21	13	2	9	39	48	18	5
		46	36	47	11	26	50	23	42	10	4
	Row2	5	8	12	15	14	25	20	38	43	21
		13	29	26	37	46	45	18	19	31	50
		30	33	48	32	17	23	49	34	3	10
		24	41	35	16	4	22	6	47	39	28
		7	11	40	36	2	1	9	44	27	42
	Row3	16	9	46	12	44	15	47	1	19	21
		18	35	28	7	22	34	38	17	4	14
		49	5	26	50	10	45	25	30	32	27
		24	11	36	40	3	13	41	2	31	23
		43	29	20	8	33	39	6	42	48	37
	Row4	1	21	36	18	13	29	31	39	19	43
		27	22	44	4	38	23	33	46	45	15
		41	47	30	16	49	40	12	10	20	25
		48	42	28	50	32	17	34	9	5	35
		6	8	14	37	2	24	11	26	3	7
	Row5	13	3	11	22	9	30	23	34	32	8
		36	25	14	40	31	47	16	24	33	46
		37	10	29	42	20	26	35	44	28	6
		41	39	1	27	19	49	43	15	17	5
		38	18	50	2	7	48	45	4	21	12
	Row6	8	4	2	15	16	23	3	18	36	45
		19	30	24	39	48	6	13	32	50	27
		46	21	10	41	47	42	37	33	38	28
		7	31	9	12	14	34	35	22	44	25
		49	20	40	17	43	29	5	1	26	11
	Row7	15	41	3	26	2	6	32	31	46	28
		5	4	12	1	45	29	10	20	22	21
		9	36	23	40	25	38	44	48	47	39
		43	42	14	35	33	30	49	11	34	37
		16	27	19	13	24	50	8	17	7	18
	Row8	7	15	30	3	11	1	5	33	6	14
		38	49	32	29	18	21	25	45	35	47
		28	24	9	50	19	31	41	26	27	17
		48	46	20	37	44	12	40	34	43	8
		4	2	39	22	23	42	10	36	16	13
	Row9	3	16	13	23	2	30	14	41	8	1
		48	29	33	6	26	39	44	20	24	19
		18	4	7	34	5	38	17	49	21	37
		50	45	46	22	15	36	27	40	11	12
		42	47	32	35	10	9	28	31	43	25
	Row10	18	10	2	16	7	22	40	33	25	24
		23	37	45	26	4	5	36	15	3	27
		47	14	8	28	48	34	50	32	17	44
		29	38	19	41	1	35	42	9	46	31
		39	43	21	11	13	12	30	6	20	49

***Appendix 2*. Continued**

Problem	Optimal sequence									
Swv18	Row1	15	1	16	21	12	44	25	3	45	7
		50	22	26	31	36	37	19	9	40	48
		17	23	10	27	11	34	5	33	38	46
		49	13	20	8	18	28	32	47	14	6
		2	39	43	35	4	30	41	42	29	24
	Row2	13	17	32	43	29	47	30	49	25	1
		46	27	23	28	48	6	12	15	41	42
		37	33	19	24	45	3	40	8	22	21
		4	35	36	18	10	14	26	7	11	50
		44	31	5	2	16	39	38	34	20	9
	Row3	19	17	18	2	27	22	33	25	29	50
		39	42	36	23	45	38	24	35	21	9
		48	40	10	26	3	15	12	47	14	1
		7	16	30	34	4	11	32	6	44	37
		49	41	20	31	5	13	8	43	28	46
	Row4	12	26	32	33	47	17	11	5	19	45
		31	49	41	30	25	35	28	50	16	39
		13	48	36	22	34	14	29	43	6	2
		38	18	8	1	3	21	20	42	15	44
		24	4	23	10	7	46	40	9	27	37
	Row5	25	13	3	8	45	4	5	27	6	50
		43	11	37	35	14	26	39	36	29	31
		18	2	20	42	40	41	49	21	24	48
		17	44	12	38	28	15	19	30	9	46
		22	10	16	23	32	33	1	34	7	47
	Row6	32	5	30	17	13	10	22	46	47	36
		43	40	2	3	14	31	34	19	11	33
		42	27	16	8	28	7	39	37	21	38
		26	23	49	4	35	44	9	41	29	25
		45	20	18	50	24	6	15	1	12	48
	Row7	12	11	42	45	29	3	15	36	23	19
		35	47	32	48	44	25	21	18	24	13
		38	31	30	50	2	1	14	40	34	41
		10	5	8	37	39	27	20	16	43	49
		26	9	46	33	22	17	6	28	7	4
	Row8	7	14	12	20	25	47	35	10	8	30
		34	27	16	23	38	13	43	3	45	44
		17	26	5	18	29	48	19	21	6	1
		40	39	28	36	50	15	49	46	22	9
		2	24	11	42	31	4	37	41	33	32
	Row9	2	12	16	9	26	23	35	46	29	32
		49	22	10	13	50	27	39	8	17	34
		48	6	11	25	37	24	44	41	31	36
		14	7	33	3	42	30	5	28	19	15
		40	18	1	20	4	45	43	38	47	21
	Row10	5	19	2	42	24	7	18	8	49	13
		40	35	47	30	17	16	45	34	29	31
		27	41	37	25	10	11	21	50	32	39
		33	38	1	46	20	12	23	48	43	28
		26	36	22	15	44	14	6	9	3	4

***Appendix 2*. Continued**

Problem	Optimal sequence									
Swv19	Row1	4	14	27	6	49	5	30	10	21	29
		39	8	28	36	42	9	43	35	24	47
		46	17	50	38	45	13	23	19	40	12
		41	1	26	15	48	32	31	2	22	25
		34	3	18	33	20	7	44	37	11	16
	Row2	20	21	5	2	10	38	39	26	1	47
		45	28	32	17	6	8	14	15	33	7
		50	27	16	13	11	48	49	34	24	3
		29	36	25	4	37	18	9	35	19	23
		42	44	30	40	41	22	46	31	12	43
	Row3	5	11	36	16	33	27	42	14	40	29
		43	2	20	26	44	25	6	10	37	9
		30	32	3	8	18	12	4	17	1	45
		22	24	28	35	13	21	34	50	38	31
		23	48	15	41	19	47	39	46	7	49
	Row4	3	2	6	21	23	39	16	33	50	12
		11	26	28	46	8	7	5	15	18	32
		38	22	27	44	24	9	45	49	43	4
		19	29	20	30	1	42	25	40	36	31
		37	17	13	48	10	41	34	35	14	47
	Row5	44	3	24	46	34	23	41	1	9	43
		10	2	50	40	4	22	29	20	19	48
		12	13	33	32	38	18	11	5	47	15
		27	8	16	7	26	21	42	31	45	39
		6	36	49	30	17	25	28	14	37	35
	Row6	26	39	23	4	36	49	40	22	50	44
		18	12	15	34	11	14	3	27	41	32
		1	6	38	31	35	42	21	46	17	33
		45	5	13	29	20	24	30	37	43	47
		16	2	19	9	48	8	10	7	25	28
	Row7	13	22	4	16	8	26	23	47	49	19
		33	50	28	18	5	30	45	21	34	6
		15	44	14	25	17	38	37	42	9	39
		24	1	41	7	12	31	48	11	3	46
		40	35	32	27	43	20	2	36	29	10
	Row8	2	5	9	29	15	24	17	33	39	20
		6	23	41	35	4	50	26	46	49	7
		16	19	28	40	43	44	48	3	42	31
		27	45	12	36	34	14	11	13	8	22
		37	21	1	10	38	47	32	25	18	30
	Row9	9	12	29	26	15	13	33	35	43	34
		4	19	39	20	11	44	48	41	2	3
		21	24	36	16	6	40	17	27	37	14
		45	7	23	30	28	49	46	18	1	47
		10	42	50	22	5	31	25	8	32	38
	Row10	6	10	14	17	4	18	5	26	30	9
		36	37	46	21	1	24	43	42	15	31
		40	11	34	23	8	45	33	25	28	47
		41	50	39	32	38	7	27	16	12	48
		13	22	44	29	3	35	19	20	2	49

***Appendix 2*. Continued**

Problem	Optimal sequence									
Swv20	Row1	4	28	19	10	21	16	38	27	44	48
		33	34	46	8	12	45	1	6	14	23
		17	24	42	18	39	2	29	9	37	41
		7	11	20	15	49	31	47	25	30	13
		22	5	36	35	40	32	3	26	43	50
	Row2	5	23	7	44	3	4	47	14	17	38
		41	8	9	29	45	16	18	21	2	42
		30	6	34	15	37	31	20	27	33	10
		22	28	43	46	25	39	13	11	36	49
		40	32	19	1	48	35	26	50	24	12
	Row3	15	24	19	29	30	31	3	25	43	41
		12	7	39	49	17	22	40	9	4	1
		20	26	23	50	6	35	34	47	2	18
		8	37	5	44	46	21	38	11	36	45
		28	33	32	13	10	48	16	14	42	27
	Row4	3	6	1	4	22	18	10	34	44	47
		2	40	5	23	36	26	7	13	20	39
		38	41	21	30	25	28	24	33	17	27
		46	48	49	12	14	42	32	15	43	29
		50	16	8	45	11	31	35	19	9	37
	Row5	18	6	45	28	35	2	13	12	3	25
		39	47	9	20	23	19	16	32	48	37
		4	5	49	50	26	36	40	41	44	27
		8	42	11	22	43	7	30	46	21	34
		38	10	17	31	29	14	24	33	15	1
	Row6	11	41	5	7	9	3	25	21	34	50
		37	38	23	15	14	28	48	30	39	29
		13	31	44	1	27	42	43	8	20	6
		19	17	16	32	26	36	4	49	46	2
		12	35	22	40	10	24	18	47	45	33
	Row7	29	32	20	5	7	1	28	14	15	8
		36	22	19	11	34	4	27	49	37	16
		44	18	39	33	24	17	40	21	9	12
		13	30	23	2	42	10	3	26	43	47
		31	50	38	6	48	41	35	46	45	25
	Row8	32	5	12	37	14	42	20	45	47	28
		22	4	33	21	50	24	6	29	26	10
		38	30	3	44	46	17	48	16	1	15
		19	11	18	25	31	36	8	7	41	34
		35	39	2	13	43	9	27	23	49	40
	Row9	9	8	10	23	15	17	35	24	21	25
		26	13	50	38	11	20	31	22	27	33
		34	3	42	41	30	6	36	1	48	32
		2	44	7	45	40	49	28	14	18	16
		37	46	29	43	5	19	4	12	39	47
	Row10	33	15	22	35	8	24	28	2	44	46
		12	30	16	40	27	21	3	6	36	11
		50	25	43	17	32	1	19	26	31	9
		47	38	42	4	39	5	34	7	48	45
		14	10	18	49	41	23	20	37	13	29

Appendix 3. Non-dominated solutions for M-La01 to M-La20

Non-dominated Solutions											
C_{max}	T_{max}	$\sum_{i=1}^{n} wU_i$	C_{max}	T_{max}	$\sum_{i=1}^{n} wU_i$	C_{max}	T_{max}	$\sum_{i=1}^{n} wU_i$	C_{max}	T_{max}	$\sum_{i=1}^{n} wU_i$
M-La01			M-La07			M-La13			M-La17		
521	79	4	710	186	3	934	342	8	757	156	6
521	191	3	718	151	5	934	391	6	766	102	6
523	120	3	719	61	3	952	317	11	808	153	5
526	96	3	748	33	2	954	245	6	882	219	4
530	58	3	755	23	4	963	210	9	923	208	4
539	76	2	766	24	3	971	199	10	945	63	5
597	38	3	773	6	1	979	226	8	970	86	4
603	54	2	774	0	0	994	392	4	1033	112	2
614	143	1	M-La08			1019	307	5	M-La18		
620	73	1	689	146	5	1034	125	9	834	206	3
633	0	0	696	124	6	1080	144	8	882	184	4
M-La02			697	117	7	M-La14			922	278	2
596	86	2	705	133	4	925	359	8	978	235	2
607	11	2	746	91	7	947	153	6	998	69	4
610	11	1	755	116	6	961	114	8	998	149	3
626	6	2	755	132	5	976	25	4	M-La19		
M-La03			757	113	6	977	14	4	850	126	3
451	161	3	762	28	2	1030	467	3	872	77	4
454	143	3	783	0	0	1063	34	3	882	56	3
461	129	3	M-La9			M-La15			1000	212	2
475	83	4	809	17	3	978	441	6	La20		
477	65	3	809	23	2	989	338	11	870	240	3
482	34	3	879	276	1	989	354	8	875	84	4
497	30	2		La10		991	228	9	875	175	3
581	20	1	781	21	3	1006	199	10	879	88	3
616	2	1	806	8	1	1014	160	6	902	66	5
655	0	0	824	2	1	1039	441	5	920	36	4
M-La04			890	0	0	1049	465	4	958	347	2
543	108	4	M-La11			1088	371	5	963	169	2
552	276	3	879	360	9	1108	391	2	1018	34	3
553	63	3	911	343	10	1122	89	11	1079	228	1
554	12	3	919	304	10	1132	113	8			
558	4	1	919	343	9	1133	96	10			
691	0	0	923	308	8	1159	129	7			
M-La05			925	301	7	M-La16					
476	16	3	927	97	4	879	203	6			
476	46	2	932	46	3	881	195	7			
480	26	2	956	9	2	881	214	5			
496	135	1	M-La12			895	176	6			
513	45	1	862	308	12	895	191	5			
551	12	3	862	501	8	897	148	7			
566	12	2	887	187	11	915	154	6			
569	8	2	887	282	10	922	173	4			
611	7	1	887	411	9	987	327	3			
M-La06			891	234	10	998	246	3			
664	140	4	896	239	9	1017	134	8			
664	198	3	897	372	8	1072	313	2			
673	115	5	898	104	5	1112	103	7			
755	235	1	945	102	6	1117	88	8			
763	78	5				1122	144	6			
764	24	3				1125	101	7			
776	10	3									
779	11	1									

REFERENCES

(1988). "OPERATIONS RESEARCH: THE NEXT DECADE." Operations Research **36**(4): 619.

Abdullah, S. and M. Abdolrazzagh-Nezhad (2014). "Fuzzy job-shop scheduling problems: A review." Information Sciences **278**(0): 380-407.

Adams, J., E. Balas and D. Zawack (1988). "The shifting bottleneck procedure for job shop scheduling." Manage. Sci. **34**(3): 391-401.

Adibi, M. A., M. Zandieh and M. Amiri (2010). "Multi-objective scheduling of dynamic job shop using variable neighborhood search." Expert Systems with Applications **37**(1): 282-287.

Allahverdi, A. (2015). "The third comprehensive survey on scheduling problems with setup times/costs." European Journal of Operational Research **246**(2): 345-378.

Allahverdi, A., J. N. D. Gupta and T. Aldowaisan (1999). "A review of scheduling research involving setup considerations." Omega **27**(2): 219-239.

Allahverdi, A., C. T. Ng, T. C. E. Cheng and M. Y. Kovalyov (2008). "A survey of scheduling problems with setup times or costs." European Journal of Operational Research **187**(3): 985-1032.

Balas, E. (1965). "AN ADDITIVE ALGORITHM FOR SOLVING LINEAR PROGRAMS WITH ZERO-ONE VARIABLES." Operations Research **13**(4): 517.

Balas, E. (1967). "DISCRETE PROGRAMMING BY THE FILTER METHOD." Operations Research **15**(5): 915.

Bandyopadhyay, S. and S. Saha (2013). Unsupervised Classification, Similarity Measures, Classical and Metaheuristic Approaches, and Applications, Springer London.

Bayındır, Z., P. (2005). "Production and Distribution Systems class notes."

Baykasoğlu, A., L. özbakir and A. Sönmez (2004). "Using multiple objective tabu search and grammars to model and solve multi-objective flexible job shop scheduling problems." Journal of Intelligent Manufacturing **15**(6): 777-785.

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

Beasley, J. E. (1990). "OR-Library: Distributing Test Problems by Electronic Mail." J Oper Res Soc **41**(11): 1069-1072.

Bertsimas, D. and J. Tsitsiklis (1993). "Simulated Annealing." Statistical Science **8**(1): 10-15.

Bilkay, O., O. Anlagan and S. E. Kilic (2004). "Job shop scheduling using fuzzy logic." The International Journal of Advanced Manufacturing Technology **23**(7): 606-619.

Bowman, E. H. (1959). "THE SCHEDULE-SEQUENCING PROBLEM." Operations Research **7**(5): 621.

Browne, J., M. E. J. O'Kelly and B. J. Davies (1982). "Scheduling in a batch or job shop production environment." Engineering Management International **1**(3): 173-184.

Çaliş, B. and S. Bulkan (2013). "A research survey: review of AI solution strategies of job shop scheduling problem." Journal of Intelligent Manufacturing: 1-13.

Çaliş, B. and S. Bulkan (2015). "A research survey: review of AI solution strategies of job shop scheduling problem." Journal of Intelligent Manufacturing **26**(5): 961-973.

Caramia, M. and P. Dell'Olmo (2008). Multi-objective Management in Freight Logistics: Increasing Capacity, Service Level and Safety with Optimization Algorithms, Springer London.

Chakraborty, U. K. (2009). Computational Intelligence in Flow Shop and Job Shop Scheduling, Springer Publishing Company, Incorporated.

Chankong, V. and Y. Y. Haimes (1983). "Multiobjective Decision Making: Theory and Methodology. Elsevier Science Publishing, New York."

Charnes, A. and W. W. Cooper (1961). "Management models and industrial applications of linear programming." Journal of the Franklin Institute **272**(4): 334.

Charnes, A., W. W. Cooper and R. O. Ferguson (1955). "OPTIMAL ESTIMATION OF EXECUTIVE COMPENSATION BY LINEAR PROGRAMMING." Management Science **1**(2): 138-151.

Chaudhry, I. A. (2012). "Job shop scheduling problem with alternative machines using genetic algorithms." Journal of Central South University **19**(5): 1322-1333.

Chaudhry, I. A. and A. A. Khan (2015). "A research survey: review of flexible job shop scheduling techniques." International Transactions in Operational Research: n/a-n/a.

Cheng, R., M. Gen and Y. Tsujimura (1996). "A tutorial survey of job-shop scheduling problems using genetic algorithms—I. representation." Computers & Industrial Engineering **30**(4): 983-997.

Chin Soon, C., M. Y. H. Low, A. I. Sivakumar and G. Kheng Leng (2006). A Bee Colony Optimization Algorithm to Job Shop Scheduling. Simulation Conference, 2006. WSC 06. Proceedings of the Winter.

Coello Coello, C. A., G. B. Lamont and D. A. Van Veldhuisen. (2007). "Evolutionary algorithms for solving multi-objective problems." from http://dx.doi.org/10.1007/978-0-387-36797-2.

Collier, D. A. and J. R. Evans (2009). OM, an Innovative Approach to Teaching Operation Management Mason, OH, US, Cengage

Colorni, A., M. Dorigo and V. Maniezzo (1991). "Distributed optimization by ant colonies." Proceedings of the first European conference on artificial life **142**: 134-142.

Dalfard, V. M. and G. Mohammadi (2012). "Two meta-heuristic algorithms for solving multi-objective flexible job-shop scheduling with parallel machine and maintenance constraints." Computers & Mathematics with Applications **64**(6): 2111-2117.

Darwin, C. (1859). On the Origin of Species. . John Murray, London.

Davis, L. (1985). Applying adaptive algorithms to epistatic domains. Proceedings of the 9th international joint conference on Artificial intelligence - Volume 1. Los Angeles, California, Morgan Kaufmann Publishers Inc.**:** 162-164.

Davis, L. (1985). Job Shop Scheduling with Genetic Algorithms. Proceedings of the 1st International Conference on Genetic Algorithms, L. Erlbaum Associates Inc.**:** 136-140.

Deb, K., A. Pratap, S. Agarwal and T. Meyarivan (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II." Evolutionary Computation, IEEE Transactions on **6**(2): 182-197.

Dorndorf, U. and E. Pesch (1995). "Evolution based learning in a job shop scheduling environment." Computers & Operations Research **22**(1): 25-40.

Ebadian, M., M. Rabbani, S. A. Torabi and F. Jolai (2009). "Hierarchical production planning and scheduling in make-to-order environments: reaching short and reliable delivery dates." International Journal of Production Research **47**(20): 5761-5789.

Eilon, S. and I. G. Chowdhury (1976). "Due dates in job shop scheduling." International Journal of Production Research **14**(2): 223-237.

Fang, H. L., P. Ross and D. Corne (1993). A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems. Proceedings of the Fifth Annual Conference on Genetic Algorithms, Morgan Kaufmann**:** 375-382.

Feo, T. A. and M. G. C. Resende (1989). "A probabilistic heuristic for a computationally difficult set covering problem." Oper. Res. Lett. **8**(2): 67-71.

Fera, M., F. Fruggiero, A. Lambiase, G. Martino and M. E. Nenni (2013). Production Scheduling Approaches for Operations Management.

Fishburn, P. C. (1967). ADDITIVE UTILITIES WITH INCOMPLETE PRODUCT SETS: APPLICATION TO PRIORITIES AND ASSIGNMENTS, INFORMS: Institute for Operations Research. **15:** 537-542.

Fonseca, C. M. and P. J. Fleming (1993). Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization. Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc.**:** 416-423.

Fox, B., W. Xiang and H. Lee (2007). "Industrial applications of the ant colony optimization algorithm." The International Journal of Advanced Manufacturing Technology **31**(7-8): 805-814.

Framiñán Torres, J. M., R. Leisten and R. Ruiz García (2014). Manufacturing scheduling systems : an integrated view on models, methods and tools. London, Springer.

Frutos, M. and F. Tohmé (2012). "A Multi-objective Memetic Algorithm for the Job-Shop Scheduling Problem." Operational Research **13**(2): 233-250.

Gao, K. Z., P. N. Suganthan, Q. K. Pan, T. J. Chua, T. X. Cai and C. S. Chong (2014). "Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling." Information Sciences **289**: 76-90.

Gembicki, F. and Y. Y. Haimes (1975). "Approach to performance and sensitivity multiobjective optimization: The goal attainment method." Automatic Control, IEEE Transactions on **20**(6): 769-771.

Gen, M. and R. Cheng (2000). Genetic Algorithms and Engineering Optimization, Wiley.

Gen, M., Y. Tsujimura and E. Kubota (1994). Solving job-shop scheduling problems by genetic algorithm. Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on.

Glover, F. (1989,1990). "Tabu Search-- Part I." ORSA Journal on Computing **1**(3): 190.

Glover, F. (1990). "Tabu Search: A Tutorial." Interfaces **20**(4): 74-94.

Goldberg, D. E. and J. Robert Lingle (1985). AllelesLociand the Traveling Salesman Problem. Proceedings of the 1st International Conference on Genetic Algorithms, L. Erlbaum Associates Inc.**:** 154-159.

Holland, J. H. (1975). Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence. Ann Arbor, University of Michigan Press.

Hollier, R. H. (1975). "A review of: "Introduction to Sequencing and Scheduling." By K. R. BAKER. (New York : Wiley, 1974.) [Pp. 305] Price £8-50." International Journal of Production Research **13**(6): 654-654.

Horn, J., N. Nafpliotis and D. E. Goldberg (1994). A niched Pareto genetic algorithm for multiobjective optimization. Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on.

Hosseinabadi, A. A. R., H. Siar, S. Shamshirband, M. Shojafar and M. H. N. M. Nasir (2014). "Using the gravitational emulation local search algorithm to solve the multi-objective flexible dynamic job shop scheduling problem in Small and Medium Enterprises." Annals of Operations Research **229**(1): 451-474.

Hsu, T. C. (2006). "New expression of scheduling performance measures." International Journal of Production Research **44**(15): 3147-3158.

Huang, J. and G. A. Süer (2015). "A dispatching rule-based genetic algorithm for multi-objective job shop scheduling using fuzzy satisfaction levels." Computers & Industrial Engineering **86**: 29-42.

Huang, R.-H. (2010). "Multi-objective job-shop scheduling with lot-splitting production." International Journal of Production Economics **124**(1): 206-213.

Jain, A. S. and S. Meeran (1999). "Deterministic job-shop scheduling: Past, present and future." European Journal of Operational Research **113**(2): 390-434.

Jia, S. and Z.-H. Hu (2014). "Path-relinking Tabu search for the multi-objective flexible job shop scheduling problem." Computers & Operations Research **47**: 11-26.

Jin, Y. (2003). Advanced Fuzzy Systems Design and Applications, Physica-Verlag.

Jones, A., L. C. Rabelo and A. T. Sharawi (2001). Survey of Job Shop Scheduling Techniques. Wiley Encyclopedia of Electrical and Electronics Engineering, John Wiley & Sons, Inc.

Kacem, I., S. Hammadi and P. Borne (2002). "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems." Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on **32**(1): 1-13.

Kachitvichyanukul, V. and S. Sitthitham (2011). "A two-stage genetic algorithm for multi-objective job shop scheduling problems." Journal of Intelligent Manufacturing **22**(3): 355-365.

Kaplanoğlu, V. (2016). "An object-oriented approach for multi-objective flexible job-shop scheduling problem." Expert Systems with Applications **45**: 71-84.

Karthikeyan, S., P. Asokan and S. Nickolas (2014). "A hybrid discrete firefly algorithm for multi-objective flexible job shop scheduling problem with limited resource constraints." The International Journal of Advanced Manufacturing Technology **72**(9): 1567-1579.

Kennedy, J. and R. Eberhart (1995). Particle swarm optimization. Neural Networks, 1995. Proceedings., IEEE International Conference on.

Kim, S. C. and P. M. Bobrowski (1994). "Impact of sequence-dependent setup time on job shop scheduling performance." International Journal of Production Research **32**(7): 1503-1520.

Kirkpatrick, S., C. D. Gelatt and M. P. Vecchi (1983). "Optimization by Simulated Annealing." Science, Number 4598, 13 May 1983 **220, 4598**: 671-680.

Kobayashi, S., I. Ono and M. Yamamura (1995). An Efficient Genetic Algorithm for Job Shop Scheduling Problems. Proceedings of the 6th International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc.**:** 506-511.

Kolen, A. W. J., J. K. Lenstra, C. H. Papadimitriou and F. C. R. Spieksma (2007). "Interval scheduling: A survey." Naval Research Logistics (NRL) **54**(5): 530-543.

Korytkowski, P., S. Rymaszewski and T. Wiśniewski (2013). "Ant colony optimization for job shop scheduling using multi-attribute dispatching rules." The International Journal of Advanced Manufacturing Technology **67**(1-4): 231-241.

Lei, D. (2008). "A Pareto archive particle swarm optimization for multi-objective job shop scheduling." Computers & Industrial Engineering **54**(4): 960-971.

Lei, D. (2011). "Simplified multi-objective genetic algorithms for stochastic job shop scheduling." Applied Soft Computing **11**(8): 4991-4996.

Lei, D. (2012). "Interval job shop scheduling problems." The International Journal of Advanced Manufacturing Technology **60**(1-4): 291-301.

Lei, D. (2012). "Multi-objective artificial bee colony for interval job shop scheduling with flexible maintenance." The International Journal of Advanced Manufacturing Technology **66**(9): 1835-1843.

Li, J.-Q., Q.-K. Pan and J. Chen (2012). "A hybrid Pareto-based local search algorithm for multi-objective flexible job shop scheduling problems." International Journal of Production Research **50**(4): 1063-1078.

Li, J.-Q., Q.-K. Pan and K.-Z. Gao (2011). "Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems." The International Journal of Advanced Manufacturing Technology **55**(9-12): 1159-1169.

Li, J.-q., Q.-k. Pan and Y.-C. Liang (2010). "An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems." Computers & Industrial Engineering **59**(4): 647-662.

Li, J.-Q., Q.-K. Pan and M. F. Tasgetiren (2014). "A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities." Applied Mathematical Modelling **38**(3): 1111-1132.

Li, J., Q. Pan and S. Xie (2012). "An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems." Applied Mathematics and Computation **218**(18): 9353-9371.

Li, L. and J.-z. Huo (2009). "Multi-Objective Flexible Job-Shop Scheduling Problem in Steel Tubes Production." Systems Engineering - Theory & Practice **29**(8): 117-126.

Lin, S.-C., E. D. Goodman and W. F. Punch A Genetic Algorithm Approach to Dynamic Job Shop Scheduling Problems. Michigan State University, Genetic Algorithms Research and Applications Group.

Low, C., T.-H. Wu and C.-M. Hsu (2005). "Mathematical modelling of multi-objective job shop scheduling with dependent setups and re-entrant operations." The International Journal of Advanced Manufacturing Technology **27**(1-2): 181-189.

M'Hallah, R. and R. L. Bulfin (2007). "Minimizing the weighted number of tardy jobs on a single machine with release dates." European Journal of Operational Research **176**(2): 727-744.

Manikas, A. and Y.-L. Chang (2009). "Multi-criteria sequence-dependent job shop scheduling using genetic algorithms." Computers & Industrial Engineering **56**(1): 179-185.

Mladenovi, N. and P. Hansen (1997). "Variable neighborhood search." Comput. Oper. Res. **24**(11): 1097-1100.

Moreno Pérez, J. A., J. Marcos Moreno-Vega and I. Rodríguez Martín (2003). "Variable neighborhood tabu search and its application to the median cycle problem." European Journal of Operational Research **151**(2): 365-378.

Moslehi, G. and M. Mahnam (2011). "A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search." International Journal of Production Economics **129**(1): 14-22.

Nakano, R. and T. Yamada (1991). Conventional Genetic Algorithm for Job Shop Problems. Proceedings of the 4th International Conference on Genetic Algorithms. B. a. Booker, Morgan Kaufman.

Niu, S. H., S. K. Ong and A. Y. C. Nee (2013). "An improved intelligent water drops algorithm for solving multi-objective job shop scheduling." Engineering Applications of Artificial Intelligence **26**(10): 2431-2442.

Norman, B. (1995). Random keys genetic algorithm for scheduling unabridged version. Technical report, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor.

Oliver, I. M., D. J. Smith and J. R. C. Holland (1987). A study of permutation crossover operators on the traveling salesman problem. Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application. Cambridge, Massachusetts, USA, L. Erlbaum Associates Inc.**:** 224-230.

Ono, I., M. Yamamura and S. Kobayashi (1996). A genetic algorithm for job-shop scheduling problems using job-based order crossover. Evolutionary Computation, 1996., Proceedings of IEEE International Conference on.

Oyetunji, E. O. (2009). "Some Common Performance Measures in Scheduling Problems: Review Article." Research Journal of Applied Sciences, Engineering and Technology **1**(2): 6-9.

Pérez, M. A. F. and F. M. P. Raupp (2014). "A Newton-based heuristic algorithm for multi-objective flexible job-shop scheduling problem." Journal of Intelligent Manufacturing: 1-8.

Pham, D. T., A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim and M. Zaidi (2006). The Bees Algorithm, A Novel Tool for Complex Optimisation Problems. Proceedings of the 2nd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2006), Oxford, Elsevier.

Pham, D. T. and D. Karaboga (1998). Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks, Springer-Verlag New York, Inc.

Pinedo, L. (2005). Planning And Scheduling In Manufacturing And Services, Springer Science+Business Media.

Pinedo, M. (1995). "Scheduling-theory, algorithms and systems." Prentice-Hall, Englewood Cliffs, NJ.

Pinedo, M. and X. A. CHAO (1999). Operations scheduling with applications in manufacturing and services, McGraw-Hill Companies.

Ponnambalam, S. G., V. Ramkumar and N. Jawahar (2001). "A multiobjective genetic algorithm for job shop scheduling." Production Planning & Control **12**(8): 764-774.

Qiu, X. and H. Y. K. Lau (2013). "An AIS-based hybrid algorithm with PDRs for multi-objective dynamic online job shop scheduling problem." Applied Soft Computing **13**(3): 1340-1351.

Rahmati, S. H. A., M. Zandieh and M. Yazdani (2012). "Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem." The International Journal of Advanced Manufacturing Technology **64**(5): 915-932.

Ramkumar, R., A. Tamilarasi and T. Devi (2012). "A real time practical approach for multi objective job shop scheduling using fuzzy logic approach." J. Comput. Sci., 8: 606-612.

Rangaiah, G. P. and A. Bonilla-Petriciolet (2013). Multi-Objective Optimization in Chemical Engineering : Developments and Applications. Somerset, Wiley.

Rodammer, F. A. and K. P. White, Jr. (1988). "A recent survey of production scheduling." Systems, Man and Cybernetics, IEEE Transactions on **18**(6): 841-851.

Sabuncuoglu, I. and M. Bayiz (1999). "Job shop scheduling with beam search." European Journal of Operational Research **118**(2): 390-412.

Sakawa, M. and T. Mori (1999). "An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy duedate." Computers & Industrial Engineering **36**(2): 325-341.

Sawik, T. (2006). "Hierarchical approach to production scheduling in make-to-order assembly." International Journal of Production Research **44**(4): 801-830.

Schaffer, J. D. (1985). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. Proceedings of the 1st International Conference on Genetic Algorithms, L. Erlbaum Associates Inc.**:** 93-100.

Sha, D. Y. and C.-Y. Hsu (2006). "A hybrid particle swarm optimization for job shop scheduling problem." Computers & Industrial Engineering **51**(4): 791-808.

Sha, D. Y. and H.-H. Lin (2010). "A multi-objective PSO for job-shop scheduling problems." Expert Systems with Applications **37**(2): 1065-1070.

Shah, S. C. and A. Kusiak (2004). "Data mining and genetic algorithm based gene/SNP selection." Artificial Intelligence in Medicine **31**(3): 183-196.

Shahsavari-Pour, N. and B. Ghasemishabankareh (2013). "A novel hybrid meta-heuristic algorithm for solving multi objective flexible job shop scheduling." Journal of Manufacturing Systems **32**(4): 771-780.

Shao, X., W. Liu, Q. Liu and C. Zhang (2013). "Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem." The International Journal of Advanced Manufacturing Technology **67**(9): 2885-2901.

Sharma, P. and A. Jain (2015). "A review on job shop scheduling with setup times." Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture.

Shen, X.-N. and X. Yao (2015). "Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems." Information Sciences **298**: 198-224.

Shivasankaran, N., P. S. Kumar and K. V. Raja (2015). "Hybrid Sorting Immune Simulated Annealing Algorithm For Flexible Job Shop Scheduling." International Journal of Computational Intelligence Systems **8**(3): 455-466.

Singh, M., M. Singh, S. S. Mahapatra and N. Jagadev (2015). "Particle swarm optimization algorithm embedded with maximum deviation theory for solving multi-objective flexible job shop scheduling problem." The International Journal of Advanced Manufacturing Technology: 1-14.

SMITH, R. D. (1966). A SOLUTION TO ONE TYPE OF THE N-JOB, M-MACHINE SEQUENCING PROBLEM. MASTER OF SCIENCE Master's thesis, Texas Technological College.

Sotskov, J. N., T. Tautenhahn and F. Werner (1996). On the Application of Insertion Techniques for Job Shop Problems with Setup Times, Otto-von-Guericke-Univ., Fak. für Math.

Srinivas, N. and K. Deb (1994). "Muiltiobjective optimization using nondominated sorting in genetic algorithms." Evol. Comput. **2**(3): 221-248.

Srinivasan, V. (1971). "A hybrid algorithm for the one machine sequencing problem to minimize total tardiness." Naval Research Logistics Quarterly **18**(3): 317-327.

Su, N., Z. Mengjie, M. Johnston and T. Kay Chen (2014). "Automatic Design of Scheduling Policies for Dynamic Multi-objective Job Shop Scheduling via Cooperative Coevolution Genetic Programming." Evolutionary Computation, IEEE Transactions on **18**(2): 193-208.

Suresh, R. K. and K. M. Mohanasundaram (2006). "Pareto archived simulated annealing for job shop scheduling with multiple objectives." The International Journal of Advanced Manufacturing Technology **29**(1-2): 184-196.

Syswerda, G. (1990). Schedule optimization using genetic algorithms. New York, l Davis, edVan Nostrand Reinhold.

Tamaki, H. and Y. Nishikawa (1992). A Paralleled Genetic Algorithm Based on a Neighborhood Model and Its Application to the Jobshop Scheduling. In Proc. of the Second Int. Conf. on Parallel Problem Soloing from Nature, Elsevier

Science Publishers, North-Holland**:** 573-582.

Tavakkoli-Moghaddam, R., M. Azarkish and A. Sadeghnejad-Barkousaraie (2011). "A new hybrid multi-objective Pareto archive PSO algorithm for a bi-objective job shop scheduling problem." Expert Systems with Applications **38**(9): 10812-10821.

Teodorovic, D. and M. Dell'Orco (2005). Bee Colony Optimization -- A Cooperative Learning Approach to Complex Transportation Problems. 10th EWGT Meeting and 16th Mini-EURO Conference.

Tu, J. V. (1996). "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes." Journal of Clinical Epidemiology **49**(11): 1225-1231.

Van Veldhuizen, D. A. (1999). Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations. 9928483 Ph.D., Air Force Institute of Technology.

Vilcot, G. and J.-C. Billaut (2008). "A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem." European Journal of Operational Research **190**(2): 398-411.

Wang, L. (1984). On the solution of special sequencing problems, TU

Magdeburg.

Wang, L., S. Wang and M. Liu (2013). "A Pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem." International Journal of Production Research **51**(12): 3574-3592.

Wang, L., G. Zhou, Y. Xu and M. Liu (2012). "An enhanced Pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling." The International Journal of Advanced Manufacturing Technology **60**(9-12): 1111-1123.

Wang, X., L. Gao, C. Zhang and X. Shao (2010). "A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem." The International Journal of Advanced Manufacturing Technology **51**(5): 757-767.

Weckman, G., C. Ganduri and D. Koonce (2008). "A neural network job-shop scheduler." Journal of Intelligent Manufacturing **19**(2): 191-201.

Werner, F. (2011) "GENETIC ALGORITHMS FOR SHOP SCHEDULING PROBLEMS: A SURVEY." 1-66.

Werner, F. and A. Winkler (1995). "Insertion techniques for the heuristic solution of the job shop problem." Discrete Applied Mathematics **58**(2): 191-211.

Wilbrecht, J. K. and W. B. Prescott (1969). "THE INFLUENCE OF SETUP TIME ON JOB SHOP PERFORMANCE." Management Science **16**(4): B-274-B-280.

Wilhelm, W. E. and S. Hyun-Myung (1985). "Effectiveness of alternate operations in a flexible manufacturing system." International Journal of Production Research **23**(1): 65.

Xhafa, F. and A. Abraham (2008). Metaheuristics for Scheduling in Industrial and Manufacturing Applications, Springer Publishing Company, Incorporated.

Xia, W. and Z. Wu (2005). "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems." Computers & Industrial Engineering **48**(2): 409-425.

Xiao-Juan, W., Z. Chao-Yong, G. Liang and L. Pei-Gen (2008). A Survey and Future Trend of Study on Multi-Objective Scheduling. Natural Computation, 2008. ICNC '08. Fourth International Conference on.

Xiaoyan, Z. and W. E. Wilhelm (2006). "Scheduling and lot sizing with sequence-dependent setup: A literature review." IIE Transactions **38**(11): 987-1007.

Xing, L.-N., Y.-W. Chen, P. Wang, Q.-S. Zhao and J. Xiong (2010). "A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems." Applied Soft Computing **10**(3): 888-896.

Xing, L.-N., Y.-W. Chen and K.-W. Yang (2009). "Multi-objective flexible job shop schedule: Design and evaluation by simulation modeling." Applied Soft Computing **9**(1): 362-376.

Xingyi, Z., T. Ye, C. Ran and J. Yaochu (2015). "An Efficient Approach to Nondominated Sorting for Evolutionary Multiobjective Optimization." Evolutionary Computation, IEEE Transactions on **19**(2): 201-213.

Xue, H., P. Zhang, S. Wei and L. Yang (2014). An Improved Immune Algorithm for Multi-objective Flexible Job-shop Scheduling.

Yamada, T. and R. Nakano (1992). A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems. In Proc. of the Second Int. Conf. on Parallel Problem Solving from Nature, Elsevier**:** 283-292.

Yang, Y. and X. Gu (2014). "Cultural-Based Genetic Tabu Algorithm for Multiobjective Job Shop Scheduling." Mathematical Problems in Engineering **2014**: 14.

Zadeh, L. A. (1965). "Fuzzy sets." Information and Control **8**(3): 338-353.

Zhang, G., X. Shao, P. Li and L. Gao (2009). "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem." Computers & Industrial Engineering **56**(4): 1309-1318.

Zhang, L., L. Gao and X. Li (2013). "A hybrid genetic algorithm and tabu search for a multi-objective dynamic job shop scheduling problem." International Journal of Production Research **51**(12): 3516-3531.

Zhang, R. and R. Chiong (2016). "Solving the energy-efficient job shop scheduling problem: a multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption." Journal of Cleaner Production **112, Part 4**: 3361-3375.

Zhang, W. (1999). State-Space Search: Algorithms, Complexity, Extensions, and Applications, Springer New York.

Zhao, B., J. Gao, K. Chen and K. Guo (2015). "Two-generation Pareto ant colony algorithm for multi-objective job shop scheduling problem with alternative process plans and unrelated parallel machines." Journal of Intelligent Manufacturing: 1-16.

Zhao, F., J. Tang, J. Wang and Jonrinaldi (2014). "An improved particle swarm optimization with decline disturbance index (DDPSO) for multi-objective job-shop scheduling problem." Computers & Operations Research **45**: 38-50.

Zheng, Y.-l., Y.-x. Li and D.-m. Lei (2011). "Multi-objective swarm-based neighborhood search for fuzzy flexible job shop scheduling." The International Journal of Advanced Manufacturing Technology **60**(9): 1063-1069.

Zitzler, E., M. Laumanns and L. Thiele (2002). Improving the strength pareto evolutionary algorithm for multiobjective optimization. International Center for Numerical Methods in Engineering. Athens, Greece**:** 95--100.

Zitzler, E. and L. Thiele (1999). "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach." Trans. Evol. Comp **3**(4): 257-271.

Zydallis, J. B. (2003). Explicit building-block multiobjective genetic algorithms: Theory, analysis, and development. 3077559 Ph.D., Air Force Institute of Technology.