OPTIMIZING THE FLEXIBLE JOB-SHOP SCHEDULING PROBLEM USING HYBRIDIZED GENETIC ALGORITHMS

by

Nasr H. Al-Hinai

A thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

Department of Mechanical and Manufacturing Engineering

University of Manitoba

Winnipeg, Manitoba, Canada

Abstract

Flexible job-shop scheduling problem (FJSP) is a generalization of the classical job-shop scheduling problem (JSP). It takes shape when alternative production routing is allowed in the classical job-shop. However, production scheduling becomes very complex as the number of jobs, operations, parts and machines increases. Until recently, scheduling problems were studied assuming that all of the problem parameters are known beforehand. However, such assumption does not reflect the reality as accidents and unforeseen incidents happen in real manufacturing systems. Thus, an optimal schedule that is produced based on deterministic measures may result in a degraded system performance when released to the job-shop. For this reason more emphasis is put towards producing schedules that can handle uncertainties caused by random disruptions. The current research work addresses solving the deterministic FJSP using evolutionary algorithm and then modifying that method so that robust and/or stable schedules for the FJSP with the presence of disruptions are obtained.

Evolutionary computation is used to develop a hybridized genetic algorithm (hGA) specifically designed for the deterministic FJSP. Its performance is evaluated by comparison to performances of previous approaches with the aid of an extensive computational study on 184 benchmark problems with the objective of minimizing the makespan.

After that, the previously developed hGA is modified to find schedules that are quality robust and/or stable in face of random machine breakdowns. Consequently, a two-stage hGA is proposed to generate the predictive schedule. Furthermore, the effectiveness of

the proposed method is compared against three other methods; two are taken from literature and the third is a combination of the former two methods.

Subsequently, the hGA is modified to consider FJSP when processing times of some operations are represented by or subjected to small-to-medium uncertainty. The work compares two genetic approaches to obtain predictive schedule, an approach based on expected processing times and an approach based on sampling technique. To determine the performance of the predictive schedules obtained by both approaches with respect to two types of robustness, an experimental study and Analysis of Variance (ANOVA) are conducted on a number of benchmark problems.

Acknowledgment

In writing this thesis I have made much use of the work of a generation of scientists, researchers, and engineers. To all those to whose work I have made reference I am greatly beholden. I am particularly indebted to those sessions of discussions filled with guidance, support, patience and encouragement with my supervisor, Dr. Tarek ElMekkawy, without which this work would not be possible. I would also like to acknowledge Dr. Youssef Loukili and Mr. Fouad Alallah for their great help and valuable advice in writing the C++ code used in this work. I extend my gratitude to Dr. Qingjin Peng and Dr. Jun Cai, my advisory committee members, as well as Dr. Tariq Gulam for their revisions and valuable suggestions. Also, I would like to express my great appreciation to Sultan Qaboos University (SQU) for giving me this chance to continuing my PhD study at the University of Manitoba (U *of* M) and providing me with the financial assistance throughout these years.

To my respectful Father, my caring Mother,

my compassionate Wife,

and my lovely Daughters.

Table of Contents

List of Tables

List of Figures

Acronyms

AC	Ant Colony
$Ave.Abs\ RMS_{\Delta}$	Average Absolute Relative Makespan Deviation
$Ave.RDev$	Average Relative Deviation
AGV	Automated Guided Vehicle
AMS_RI	Average Realized Makespan Improvement Percentage
ANOVA	Analysis of Variance
ASTBI	Average Stability Improvement Percentage
BD	Breakdown Type
CDR-PopGen	Composite Dispatching Rules Population Generation
CIM	Computer Integrated Manufacturing
CNC	Computer Numerically Controlled machines
DMS	Makespan of Schedules Obtained Using Deterministic Method
DSTB	Stability of Schedules Obtained Using Deterministic Method
DT	Disturbance Type
FJS	Flexible Job-shop
FJSP	Flexible Job-shop Scheduling Problem
FMS	Flexible Manufacturing System
GA	Genetic Algorithm
GENACE	Cultural Evolution Genetic Algorithm
hGA	Hybridized Genetic Algorithm
IT	Information Technology
Ini-PopGen	Initial Population Generation

JIT	Just-in-Time
JSP	Job-shop Scheduling Problem
JSSANT	Job-Shop Scheduling Using Ant Colony
JSSGA	Job-Shop Scheduling Using Genetic Algorithm
LB	Lower Bound
LEGA	LEarnable Genetic Architecture
LS	Local Search
MBT	Machine Busy Time
MBM	Machine Based Mutation
MRP I	Material Requirement Planning
MRP II	Manufacturing Resource Planning
MS	Makespan
P_c	Crossover Probability
P_m	Mutation Probability
P-FJSP	Partial Flexible Job-Shop Scheduling Problem
PBM	Position Based Mutation
POX	Precedence Preserving Order-Based Crossover
RE	Relative Error
RMS	Makespan of Schedules Obtained Using Robust Method
RQULT	Relative Solution Quality Measure
RSTB	Stability of Schedules Obtained Using Robust Method
SA	Simulated Annealing
T-FJSP	Total Flexible Job-Shop Scheduling Problem

TS	Tabu Search
TSP	Travelling Salesman Problem
tot_noper	Total Number of Operations

CHAPTER 1
INTRODUCTION

1.1 Background

The main driving factors to improve the performance of manufacturing systems are the continuous changing demands of customers and the need to deliver better quality and low-priced products. To achieve these requirements the manufacturing system has to have flexibility that enables it to quickly respond to this changing environment.

During the past three decades, manufacturing systems have gone through tremendous changes by introducing and implementing different managerial concepts and tools that controls the production procedure related to mass production or mass customization production such as Just-in-Time (JIT), Kanban, Material Requirement Planning (MRP I), Manufacturing Resource Planning (MRP II), etc, as well as coordinating the communications between different levels of the organization or factory using information technology (IT). Using IT tools, the computer integrated manufacturing (CIM) concept attempted to integrate all activities within a manufacturing facility. The CIM concept was expected (at its early stages in 1980s and early 1990s) to be able to deliver the best solution for all the manufacturing problems. However, CIM implementations resulted in rigid centralized systems that are incapable to deliver the expected flexibility in response to changes of any nature (Babiceanu and Chen, 2006). For that reason, the conventional centralized manufacturing systems with their traditional planning, scheduling and control mechanism are concluded to be insufficiently flexible to respond to restructuring the manufacturing system production style caused by the highly dynamic nature of

customers' demands and global competition. Hence, the need for a new manufacturing paradigm has become more demanding than ever.

As a result, advanced equipment like Computer Numerically Controlled machines (CNC), Automated Guided Vehicles (AGV), and material handling systems like robots were integrated with computers and assimilated within a manufacturing system forming what is called Flexible Manufacturing System (FMS). These systems provided a balance between mass production, capacity, and high flexibility, wide variety. Consequently, FMSs have better adaptability to variable production plans and goals as well as producing a higher quality and wider variety of products, with minimal machines' setups.

However, FMS usually exhibit a high degree of resource sharing. Thus, a number of challenges related to the constraints of different jobs' processing plans, machines and other resources availability during the scheduling horizon were raised. Among them is the scheduling task, which is concerned with allocating a number of tasks to limited resources in order to optimize a certain performance criteria. Depending on the problem size, represented by the number of tasks and resources, the scheduling task may turn out to be the most time consuming and challenging activity within an enterprise. Therefore, the performance of an FMS not properly supported by an efficient scheduling may be significantly limited and the advantages derived from its flexibility may suffer a sharp decrease. For example, an inefficient scheduling results in poor utilization of resources, over-loaded/idle capacity, long production lead time and unreliable due date commitments, which in turn increases production costs and reduces competitiveness in the market place. Furthermore, ineffective scheduling often delays orders and results in unsatisfied customers and may subject the firm to penalties.

1.1.1 The flexible job-shop scheduling problem

Since flexible job-shops (FJS) provide great flexibility on the shop floor and the efficiency of large volume production, the production scheduling and control in FJS becomes very complex as the number of jobs, operations, parts and machines increases. Setting different processes in a FJS is subjected to various constraints such as processing time of an operation, capacity, tooling and processing sequence (sometimes called the technological constraints). In those systems, each entity has to share efficiently the available resources in order to optimize the tasks scheduling. The tasks scheduling consist of defining a schedule that can meet all timing and logical constraints of the jobs' operations, and in general, it has been classified as an NP-hard problem. In literature, this scheduling problem is called the flexible job-shop scheduling problem (FJSP).

In order to visualize a final schedule, A Gantt-chart is usually used. Figure 1.1 shows an example of a Gantt-chart describing a schedule for an FJS consisting of four jobs and three machines. In the Gantt-chart, each row represents a machine whereas boxes represent operations. These boxes are tagged with two-digit numbers. The first digit refers to the job number and the second digit refers to the operation's number of that job. The time-axis shows the timetabling information of each operation.

The objective of the scheduling task is to optimize a certain criterion[1]. This criterion is usually referred to as the performance measure. Number of time-based performance measures exist in literature. These measures are:

- Makespan: the objective is to minimize the maximum completion time of the schedule.

[1] More in depth definitions are given in Section 1.2

- Mean flow time: the objective is to minimize the average time spent by a job in the system. Flow time is defined as the elapsed time since the job is ready to be processed until it has finished
- Total tardiness: the objective is to minimize the summed lateness of all jobs in the system. Lateness is defined as how much later a job has finished after its deadline.
- Total earliness: the objective is to minimize the summed earliness of all jobs in the system. Earliness is defined as how much earlier a job has finished before its deadline.

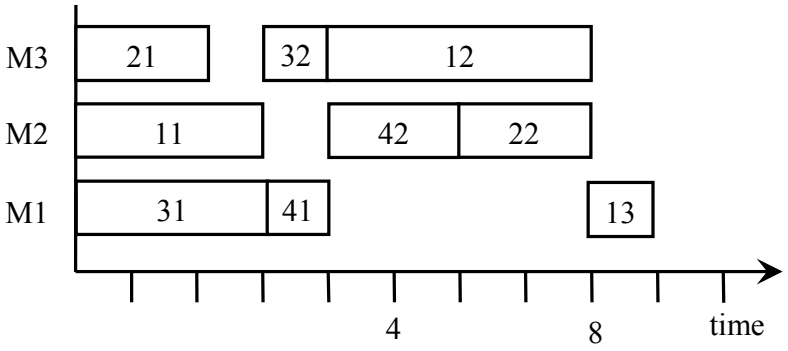


Figure 1.1: An example of Gantt-chart.

At this stage, it is worth pointing out that schedules are divided into two classes, feasible schedule and unfeasible schedules. The unfeasible schedules are those schedules that violates some or all of the timing and technological constraints. Feasible schedules are categorized in the following classes (French, 1982; and Pinedo, 2002):

1- Semi-active schedule wherein no operation can be started earlier without changing the process order or violating the technological constraints.

2- Active schedule wherein no operation can be started earlier without delaying at least one other operation or violating the technological constraints.

3- Non-delay schedule wherein no machine is kept idle if an operation is ready to be processed.

French (1982) and Pinedo (2002) demonstrated that the set of non-delay schedules is a sub-set of the active schedules which is a sub-set of the semi-active schedules. They also verified that for regular performance measures[2] an optimal solution schedule exists within the set of active schedules.

1.2 Optimization of Scheduling Problems

Before going further in describing modeling tools and methods used in FJSP optimization problems, the word optimization and other related definitions in scheduling have to be introduced. An optimization problem is the task of minimizing or maximizing an objective function under a set of constraints. Many researchers classified scheduling optimization problems as combinatorial problems, which in the simplest case mean that some or all of the decision variables take only discrete values. For a given optimization problem, an assignment of values to the variables such that all constraint equations hold is called feasible solution. The value of the objective function for a given variable assignment is called the objective value. A solution for which it is not possible to find another solution with a better objective value is called optimal. Therefore, for an

[2] Regular performance measures are measures that can never be increased by finishing an operation earlier (French, 1982).

optimization problem, the solution with respect to the objective function does not imply best, but it is a value that is connected to the constraints.

A solution method or algorithm for an optimization problem is well-defined procedure for finding a feasible solution (as good as possible) to the problem. If the method guarantees to find an optimal solution for any problem given enough computing time and memory, then the method is called exact, otherwise it is called approximate or heuristic.

1.2.1 Performance modeling tools

Modeling and performance evaluation play vital role in the planning, design, and operation of an FMS. This is because of the fact that a decision making is involved in various stages of planning, design, and operation. The role of performance modeling is to aid this decision making in effective way. The high cost of FMS development motivates the need for modeling and analysis. It is critical that one be able to determine accurate estimates for system performance of a given system design or reconfiguration before implementation.

During the operation phase of an FMS, performance modeling can help in making decisions related to finding the best routes in the event of breakdowns, predicting the effect of adding or withdrawing resources and parts, obtaining optimal schedules in the event of machine failures, uncertainty of processing times of operations or sudden changes in part mix or demands, and in avoiding unusable situation, such as deadlocks (Wang, 1998).

Performance of the system can be evaluated at different levels of flexibility to enable managers to set the amount of flexibility in order to get a reasonable performance. In the

literature, a variety of approaches have been proposed to study FMS problems at the different levels. General classifications of these approaches are analytical, simulation, heuristic, and meta-heuristic approaches.

1.2.2 Comparison of analytical, simulation, and meta-heuristic approaches

Several mathematical techniques are reported for scheduling project activities. Unfortunately, the mathematical formulations are extremely cumbersome because of the complexity inherent in large projects. One difficulty that most mathematical scheduling method encountered is the high computational complexity involved in finding an optimum solution, especially in systems with high processing flexibility. This is because high processing flexibility implies that there exist a large number of alternative solutions. Consequently the search space of these problems tends to be very large (Chen and Luh, 1993; and Reddy et al., 2001).

Recently, there has been an increased interest in the use of simulation for real-time planning, scheduling, and control. Traditionally, simulation has been applied to long-term planning and design of manufacturing systems. The use of simulation has appeared favorable to purely analytical methods which often fail to capture complex interactions of a particular FMS (Wu and Wysk, 1989). Applying simulation as a real-time tool requires insight into the responsibilities of the simulation model and its role within the FMS.

In the case of simulation techniques, a system's model is generated based on assumptions and observations. It is defined as the representation of the dynamic behavior of the modeled system. It can capture the most details of the manufacturing systems and can be

built as accurate as one desire, limited only by cost and time. However, for complex systems simulation run can be lengthy and expensive, hence preventing the analyst from trying wide range of parameters values. Moreover, the model validation will become quite difficult.

It is clear from what stated above that scheduling is one of the most important issues in the operations of FJSP. However, scheduling problems are known to be NP-hard combinatorial problems. That is, the optimal schedule for a large system is hard to obtain within reasonable computation time and time grows exponentially with the problem size. Hence, a practical approach to scheduling is through the application of heuristics. The heuristics solve complex problems by reducing the number of evaluations and obtaining solutions within reasonable time, but it is difficult to evaluate the performance of the schedules (Chen and Jeng, 1995; and Reddy et al., 2001). Furthermore, heuristics, which can be efficient for some problem instances do not have a guaranteed distance from the optimal solution.

All the above-mentioned factors and alternative methods make performing the scheduling optimization for a FJS a complex task. Hence, the need arises for more powerful search techniques. Recently, using a high-level strategy to guide other heuristics, better known as meta-heuristics, led to better and more appreciated results in a relatively short period. A number of meta-heuristics were proposed in literature for the past few decades to deal with FJSP such as simulated annealing (SA), tabu search (TS) and genetic algorithm (GA).

1.2.3 An overview of genetic algorithms

The concept of GA is an abstraction of the process of information transfer in natural organisms through a sequence of genes called chromosome. The success of this transfer of information allows a certain organism to excel above other competing organisms and survive. Thus, those organisms that survive have the opportunity to reproduce. This allows dominate characteristics to survive. This survival process can be very instrumental in the development of techniques for solving combinatorial optimization problems such as job-shop scheduling (Gilkinson et al., 1995; Chiu and Fu, 1997).

The main advantage of using GA in scheduling problems is its ability to find optimal or near-optimal scheduling solutions in a relatively short period of time. Therefore, genetic algorithms have gained an increasing popularity as a search tool used for global optimization in a complex search space. The assumption underlying the use of GA's for scheduling is that the optimal solution will be found in the neighborhood of good solutions. In general, GA consists of:

1. A chromosome representation of the nodes in the search space.
2. A set of simple operations that takes the current population into consideration and generates the successive imposed population.
3. A fitness function to evaluate the search nodes which represent the chosen performance measure.
4. A set of stochastic assignments to control the genetic operations.

The first and very important step in a successful implementation of GA is the selection of string format or formally, the chromosome encoding or representation. Holland (1975) as

well as early researchers used binary digits to represent a chromosome. While this kind of representation has its advantages, sometimes it is meaningless to be used in many problems such as the travelling salesman problem (TSP) and FJSP in which permutation of cities or operations is more appropriate. Usually, the initial population is performed randomly to ensure high degree of diversity in it. Once an appropriate syntax is chosen, it is required to convert the chromosome to a solution by an interpretation method. That interpretation is used in the evaluation of the chromosome according to the selected fitness function.

The second step is the selection mechanism that determines which individuals in the population will undergo the mating pool. The selected individuals or *parents* will be subjected to a crossover operator which is regarded as the main genetic operator. It is worth mentioning here that the performance of genetic algorithms depends to a great extent on the performance of the crossover operator used (Gen and Cheng, 2000). Crossovers operate on two chromosomes, namely parent 1 (or donor) and parent 2 (or receiver), and combine the genetic features of both to produce an offspring. Following this, a single selected chromosome is forced to undergo mutation. Mutations affect one or more genes within that chromosome and change its characteristics to reproduce a new chromosome, which is put into the population or replaces the parent chromosome depending upon the implementation. However, it should be emphasized that crossovers and mutations may produce an invalid chromosome representation that may violate some of the constraint. Hence, these invalid chromosomes have to be identified and either repaired or replaced by some other valid chromosomes.

In GA, different termination criteria are used. However, the most common one is the maximum number of generations or maximum number of function evaluations. Other can be related to maximum stall generation, which is the maximum allowed number of generations with no improvement in the fitness function.

1.3 Motivation and Thesis Research Objectives

This thesis focuses on solving the flexible job-shop scheduling problem. In general, scheduling problems are classified into two main classes, offline scheduling, and online dynamic scheduling (sometimes called complete reactive scheduling). In the former, a detailed schedule is prepared beforehand and is then released to the shop floor for implementation. On the other hand, in dynamic scheduling no schedule is prepared in advance, but decisions related to which operation to be processed next are done in real time using priority dispatching rules. Each of the previously mentioned scheduling types has its own advantages and disadvantages. Briefly, offline scheduling enables a firm or a company to have a clearer future expectations related to when to release materials, material handling, set-up times of machines, possible satisfactions of due dates, etc. However, such scheduling procedure leads to shop floor that is prone to a decreased efficiency in face of any unexpected disruptions like orders cancelation, new orders, machine breakdowns, processing time variation, etc. In contrast, dynamic online scheduling can cope with such disruptions, but may lead to a poor utilization of resources due to the greedy nature of the heuristics used in this kind of scheduling. Therefore, implementing static offline scheduling procedure or dynamic online scheduling procedure is intensely problem dependent. In a relevant work, Wu et al. (1999) concluded

that dynamic online scheduling performs best when levels of disturbances and uncertainties are high.

Offline scheduling takes two paths. The first path assumes that all of the problem parameters are deterministic and known beforehand. However, such assumption does not reflect the reality as unforeseen events happen in real manufacturing systems. This makes the optimization a process of sustained pursuit, trying to follow an optimal solution that changes over time. The second path is about producing a predetermined schedule that is implemented until some unforeseen disruptions occurs in the system. After occurrence of disruption, the system uses some control strategies that aid the system to recover from those disruptions.

When considering control strategies in face of disruptions, two main issues will rise. The first is concerned about how to perform the rescheduling when a disruption occurs, and most importantly how to generate the predictive schedule. The importance of handling those issues on the best possible way determines the overall performance of the system. For example, if it was decided that whenever a disruption occurs, then the system must undergo a total rescheduling procedure may result in a very high schedule nervousness causing the manufacturing system to be unstable as this may affect the material handling, material ordering, disruptions to workers as they may need to move from one job to another one, etc. Furthermore, how fast the rescheduling algorithm is as well as determining the solution space that has to be searched and how to search it are very important concerns in such cases, as the entire shop floor may be idle waiting for the new schedule.

In light of the above, this research work has the following objectives. The first objective is to develop a meta-heuristic approach to provide an optimal or near optimal solutions for the deterministic FJSP. The approach to solve the deterministic FJSP is based on hybridizing GA with an initial population (or schedule) generation heuristic and then combing it with a local search method. The second is to modify the deterministic hGA to find solutions or schedules that are quality robust and/or stable in face of machine breakdown disruptions, as well as determining appropriate robustness and stability measures and when they should be applied. This objective has to satisfy two conflicting objectives where it first has to efficiently utilize the resources, and second, it has to allow sufficient flexibility for changes. Furthermore, this research work is extended by considering FJSP when processing times of some operations are represented by or subjected to a low-to-medium uncertainty level. The target is to compare the robustness of the predictive schedules obtained by two possible hGA methods. Moreover, experimental studies and Analysis of Variance (ANOVA) are conducted on number of benchmark problems.

Specifically, the methodology followed in this thesis can be detailed as follows:

1. Development of a heuristic method capable of solving deterministic FJSP and producing optimal or near optimal schedules for small and medium systems. This heuristic is then used as the initial stage to solve larger and/or more complicated systems.
2. Development of a meta-heuristic approach, genetic algorithm method, capable of solving the scheduling problems that cannot be solved optimally or nearly optimally due to computational limitations. Nevertheless, this GA is designed

such that it can easily be integrated with other heuristics as well as easily modified to accommodate different objectives.

3. Development of a local search procedure that can improve a given schedule by the exploitation of neighboring solutions.
4. Integration of the three previously approaches to form a hybridized genetic algorithm (hGA) and verify the quality of obtained schedules using benchmarks.
5. Modification of the hGA to obtain schedules for the FJSP that are robust and/or stable in face of random machine breakdowns.
6. Modification of the hGA to obtain predictive schedules for the FJSP when processing times of some operations are represented by or subjected to a low-to-medium uncertainty level.
7. Conducting an experimental study and Analysis of Variance to study the effect of different proposed robustness and stability measures on the performance of schedules obtained using different methods.

1.4 List of Publications

During the advancement of this research work and achieving the set of previously stated objectives, results were presented in the form of papers that have been published or submitted to be published. For example, objective #2 was first introduced in Al-Hinai and ElMekkawy (2009) and the details of the achieved objectives #1 through #4 were published in Al-Hinai and ElMekkawy (2011a). Similarly, after the success in achieving objective #5 and its relevant part from objective #7, the research work results were published in Al-Hinai and ElMekkawy (2011b). Furthermore, the details for achieving

objective #6 and its relevant part from objective #7 are presented in Al-Hinai and ElMekkawy (2011c) and submitted to the Journal of Manufacturing System.

- Al-Hinai, N. and ElMekkawy, T. (10-12 Feb 2009) 'A robust genetic algorithm approach for the flexible job-shop scheduling problem', *PEDAC'2009*, Alexandria, Egypt.
- Al-Hinai, N. and ElMekkawy, T. (2011a) 'An efficient hybridized genetic algorithm architecture for the flexible job-shop scheduling problem', *Flexible Services and Manufacturing Journal*, Vol. 23, pp. 64-85, doi: 10.1007/s10696-010-9067-y.
- Al-Hinai, N. And ElMekkawy, T.Y (2011b) 'Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm', *International Journal of Production Economics*, Vol. 132, pp. 279-291, doi: 10.1016/j.ijpe.2011.04.020.
- Al-Hinai, N. And ElMekkawy, T.Y (2011c) 'Robust scheduling of flexible job shop with processing time uncertainty: A comparison study', *Manuscript Submitted to the Journal of Manufacturing Systems*.

1.5 Thesis Outline

The outline of this thesis is as follows. Chapter 2 introduces the definition of deterministic FJSP and related literature with emphasis on evolutionary algorithms used to solve FJSP. Moreover, it covers the formulation of a heuristic approach used to generate initial schedules, the GA architecture and the local search procedure. Chapter 3 presents the basic concepts and definitions of robustness and stability of schedules subject to disturbances. The chapter gives more focus to disturbances that are in the form of random machine breakdowns. Furthermore, it discusses the idea of how to integrate the knowledge of the expected machine breakdowns in the determination of predictive

schedules. An effectiveness of the proposed approach is evaluated via a comparative analysis against three other methods. In Chapter 4, a comparative study is presented for the FJSP when processing times of some operations are represented by or subjected to a low-to-medium uncertainty level. Two hybridized genetic algorithm approaches used to obtain predictive schedules are compared using experimental study and ANOVA. Finally, the conclusions of this research and recommended directions for future work are covered in Chapter 5.

CHAPTER 2
AN EFFICIENT HYBRIDIZED GENETIC ALGORITHM ARCHITECTURE FOR THE FLEXIBLE JOB SHOP SCHEDULING PROBLEM

This chapter provides the definition for the deterministic flexible job-shop scheduling problem (FJSP) and highlights the most relevant literature review. Also, it proposes new hybridized genetic algorithm architecture for the FJSP. The efficiency of the genetic algorithm is enhanced by integrating it with an initial population generation algorithm and a local search method. The usefulness of the proposed methodology is illustrated with the aid of an extensive computational study on 184 benchmark problems with the objective of minimizing the makespan.

2.1 Introduction

Scheduling is concerned with allocating number of tasks to limited resources in order to optimize a certain performance criteria. Depending on the problem size, represented by the number of tasks and resources, the scheduling task may turn out to be the most time consuming and challenging activity within an enterprise.

Flexible job-shop scheduling problem (FJSP) is a generalization of the classical job-shop scheduling problem (JSP). It takes shape when alternative production routing is allowed in the classical job-shop. Here, the scheduling problem becomes more difficult to deal with as it introduces a further machine assignment decision level beside the operations' sequencing level. In manufacturing systems, most scheduling problems are very complex in nature and very complicated to be solved by conventional optimization techniques to obtain a global optimal schedule. For example, determining an efficient schedule for a flexible job-shop (FJS) with n jobs and m machines will have $(n!)^m$ possible sequences (Mellor, 1966). Hence, scheduling problems are considered as combinatorial optimization problems and classified as NP-hard problems (Garey, Johnson and Sethi, 1976). Nevertheless, modeling and solving the more complex FJSP is increasingly attracting the interest of many researchers. This can be recognized by the increase in the number of research papers addressing this problem.

As highlighted in Chapter 1, using a pure mathematical optimization approach to determine an optimal solution may not be efficient in practice (due to the NP-hard nature of the FJSP). Similarly, heuristics, which can be efficient for some problem instances do not have a guaranteed distance from the optimal solution. Recently, using a high-level strategy to guide other heuristics, known as *meta-heuristics*, led to better and more appreciated results in a relatively short period. Therefore, a number of meta-heuristics were proposed in literature for the past two decades to deal with FJSP such as simulated annealing (SA), ant colony (AC), genetic algorithm (GA), etc.

The main advantage of using GA approaches in contrast of local search techniques, is the fact that GA utilizes a population of solutions in its search, giving it more resistance to

premature convergence on local minima. The proposed approach to solve the FJSP is based on hybridizing GA with an initial schedule generation heuristic and then combing it with a local search method. The current proposed method modifies some of the already known techniques in literature and combines them to produce efficient hybridized GA architecture.

The remainder of this Chapter is organized as follows: Section 2.2 describes the FJSP definition. Previous research work in this area is summarized in Section 2.3. Section 2.4 introduces the proposed GA architecture and the local search method. The computational results are presented and discussed in Section 2.5. Finally, the research summary is covered in Section 2.6.

2.2 Problem Definition

FJSP is strongly NP-hard due to a) assignment decisions of operations to a subset of machines and b) sequencing decisions of operations on each machine (Tay and Wibowo, 2004). The FJSP can be formulated as follows:

- There are n independent jobs of each other and indexed by i.
- All jobs are ready to start at time zero.
- Each job i has Q_i operations and the operations' sequence is given by O_{ij} for $j = 1, \ldots, Q_i$.
- There are m machines indexed by k.
- Machines never breakdown and are always available.

- For each operation O_{ij}, there is a set of machines capable of performing it represented by $M_{kij}, M_{kij} \subseteq \{1,...,m\}$.
- The processing time of an operation O_{ij} on machine k is predefined and given by t_{ijk}.
- The setup time of any operation is independent of the schedule, fixed, and included in the corresponding processing time.
- A started operation cannot be interrupted (non-preemption condition).
- Each machine can process at most one operation at any time (resource constraints).
- The precedence constraints of the operations in a job can be defined for any pair of operations.

The objective is to find a schedule that has lowest possible value of makespan, where the makespan is the time required for all jobs to be processed according to a given schedule.

2.3 Literature Review

Generally, used approaches to solve the complex FJSP can be categorized into two main basic approaches: concurrent approaches and hierarchical approaches. Hierarchical approaches are based on decomposing the problem to reduce its complexity. This complexity reduction is achieved by separating the assignment decisions level (or flexible routing) from the sequencing decisions of operations into two sub-problems. It is worth mentioning here that the sequencing decisions sub-problem is in fact a classical job-shop

scheduling problem. Thus, this approach had gained the interest of researchers at earlier stages like Brandimarte (1993) and Paulli (1995) both of whom solved the assignment problem using some heuristic dispatching rules and then used tabu search to solve the remaining sequencing problem. Similarly, Bona et al. (1990) followed a similar heuristic approach to handle the machine assignment part and then used simulated annealing heuristic.

On the other hand, concurrent approaches (also known as integrated approaches) integrate both problem levels and solve them simultaneously. Although this approach was attempted at early stages like in Lee and Mirchandani (1988) and in Mirchandani et al. (1988), hierarchical approaches were favored over it for sometime due to the simplifications associated with the later. However, the high quality results obtained using concurrent approach shifted the interest towards this approach to deal with the FJSP. For example, the concurrent approaches developed using tabu search methodologies as in Hurink et al. (1994); Brucker and Neyer (1998); Dauzére-Pérés and Paulli (1997); and Mastrolilli and Gambardella (2000); and simulated annealing as in Najid et al. (2002).

Among different meta-heuristic algorithms, GA is considered to be a very successful to tackle the FJSP and this can be noticed by the growing number of papers discussing this topic. Falkenauer and Bouffouix (1991) were among the first to propose a machine parallel representation to solve JSP. They encoded the chromosomes using list of machines operating in parallel. Their work was then extended by other researchers like Mesghouni et al. (1997) who proposed parallel job representation and Chen et al. (1999) who divided the chromosome into two strings; *A* and *B*; where the first is defining the routing policy and the second defines the sequence of operations on each machine.

Similarly, Ho and Tay (2004) proposed a new GA structure called GENACE. Their chromosome consists of two parts, one dedicated to define the operation order and the second for machine selection. GENACE was then combined with a learning knowledge procedure and converted to another GA structure called LEGA by Ho et al. (2007). A similar chromosome representation was adapted by Gao et al. (2008) who developed a GA hybridized with variable neighborhood descent local search procedures for the flexible job-shop problem.

Kacem et al. (2002a, 2002b) suggested using an assignment table representation of chromosomes. In the assignment table, a machine is mapped to a consequent operation. This work was recently modified by Pezzella et al. (2008) by integrating more strategies in the genetic framework that led to better results. Kacem (2003) used a task sequencing representation of chromosomes where each cell represents an operation on a machine. Giovanni and Pezzella (2010) proposed an improved genetic algorithm for the distributed and flexible job-shop scheduling problem. Wei and Qiaoyun (2009) offered an adaptive genetic algorithm that considers the processing time, the completion time of previous operation and the idle time of current machine to select a suitable machine in the decoding process when solving the FJSP.

Hussain and Joshi (1998) proposed a two pass GA for the job-shop scheduling problem with alternate routing. Yang (2001) offered GA-based discrete dynamic programming approach. Jia et al. (2003) proposed a modified GA to solve distributed scheduling problems. Chan et al. (2006) introduced the idea of dominant genes, which is based on the principle of Automata introduced by White and Oppacher (1994). Zribi et al. (2007) used a hierarchical GA approach to solve the machine assignment and scheduling of

FJSP. Girish and Jawahar (2008) proposed two concurrent meta-heuristic approaches, genetic algorithm (JSSGA) and ant colony (JSSANT) to solve job-shops with multiple routings. Xu et al. (2009), Xing et al. (2010) and Ling et al. (2010) presented an ant colony optimization algorithm for the FJSP.

Xia and Wu (2005) combined particle swarm optimization algorithm with simulated annealing to form hybrid approach for the multi-objective FJSP. Girish and Jawahar (2009) offered particle swarm algorithm to solve FJSP. Also, Zhang et al. (2009) proposed a hybridized particle swarm optimization algorithm with tabu search to solve multi-objective FJSP. Liu et al. (2009) formulated a multi-particle swarm approach to solve multi-objective FJSP. Xing et al. (2009) addressed a local search method based on empirical knowledge for the multi-objective FJSP. Another local search method that adapts the concept of climbing discrepancy search method is proposed by Hmida et al. (2010).

2.4 GA Structure

2.4.1 Chromosome coding

A proper chromosome representation has a great impact on the success of the used GA. Cheng et al. (1996) gave a detailed tutorial survey on papers using different GA chromosome representations to solve classical JSP. It can be concluded from their work (and others like Ho et al., 2007; Mattfeld, 1996; and Tay and Wibowo, 2004) that the search space of an operation-based representation covers the whole solution space and any permutation of operators can correspond to a feasible schedule. For a recent overview

discussing these aspects readers are referred to Hart et al. (2005). In light of the above, the current work uses a similar effective permutation-based chromosome representation used in Kacem et al. (2002a and 2002b), Kacem (2003) and Chan et al. (2006). The used representation composes of a string consisting of triples (*k*,*i*,*j*) for each operation forms the chromosome, where

- *k* is machine assigned to the operation;
- *i* is current job number;
- *j* is the progressive number of that operation within job *i*.

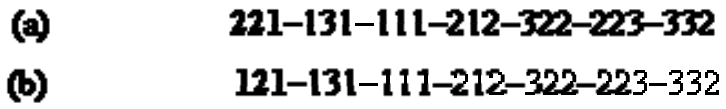


Figure 2.1: (a) Chromosome coding (b) Alternative routing

This chromosome representation integrates the machine assignment decision level and sequence decision level in one simple gene representation. Such a representation reduces the memory usage and permits more efficient genetic operators to be considered (see Subsection 2.4.5).

Another advantage of this representation is its ability to model alternative routing of the problem by changing the index *k*. The length of the chromosome is equal to the total number of operations (*tot_noper*) to be scheduled. Figure 2.1 shows a sample encoding of a chromosome for FJSP with three jobs and three machines according to the processing times given in Table 2.1. In Figure 2.1(a), the first gene (221) characterizes that operation

1 of job 2 is assigned to machine 2. Since this operation can be carried out on another machine, machine 1, then the gene can be modified to (121) as shown in Figure 2.1(b). The scheduling priority of operations is set to be from left to right. Based on this priority, a chromosome is decoded to produce an active schedule (see Subsection 2.4.2). Furthermore, both FJSP sub-problems known as total FJSP (T-FJSP) and partial FJSP (P-FJSP) can be represented with this representation without any modification (T-FJSP and P-FJSP are addressed in Section 2.5).

Table 2.1: Processing times for a FJSP with 3 jobs and 3 machines

J	O	M1	M2	M3
J_1	O_{11}	2	-	4
	O_{12}	-	3	-
J_2	O_{21}	2	3	-
	O_{22}	4	3	5
	O_{23}	-	1	-
J_3	O_{31}	2	3	1
	O_{32}	1	2	3

2.4.2 Chromosome decoding

Even though the used chromosome representation or *genotype* corresponds to feasible solution (schedule), mapping it to a proper *phenotype* is essential to reduce the very large feasible solution-space. Feasible schedules are categorized into semi-active schedules, active schedules and non-delay schedules. French (1982) and Pinedo (2002) demonstrated that the set of non-delay schedules is a sub-set of the active schedules

which is a sub-set of the semi-active schedules. They also verified that for regular performance measures an optimal solution exists within the set of active schedules. Accordingly, the proposed decoding algorithm minimizes the solution-space by constructing active schedules, while still ensuring that an optimal solution can be found. The decoding algorithm to produce active schedule is shown in *Algorithm Decode*.

Algorithm Decode (*decoding a chromosome to an active schedule*)

1. Initialize Gantt-chart structure
2. **For** each gene reading from left to right **do**
3. Identify the operation $O_{i,j}$
4. Identify the machine M_k which processes $O_{i,j}$ from the first gene's digit and its processing time $t_{i,j,k}$
5. **If** $O_{i,j}$ is the first operation **Then**

   Set t_0 to 0
6. **Else**

   Set t_0 to be stop time of the predecessor operation $O_{i,j-1}$
7. **End If**
8. **If** M_k did not process any operation **Then**

   Set t_1 to 0
9. **Else**

   Set t_1 to be the stop time of the last operation on M_k
10. **End If**

11. **If** $t_1 \leq t_0$ **Then**

    Add $O_{i,j}$ to M_k starting at t_0

12. **Else If** it exist between t_0 to t_1 (time interval between two consecutive operations on M_k) $\geq t_{i,j,k}$ **Then**

    Add $O_{i,j}$ to M_k starting at the end of the finished processing time of the operation to the left

13. **Else**

    Add $O_{i,j}$ to M_k starting at t_1

14. **End if**

15. **End for**

2.4.3 Initial population

The efficiency of meta-heuristic algorithms that use initial solutions as a starting point can be improved by generating proper solutions at promising points on the solution-space. However, if all initial solutions are generated and tuned to satisfy the objective function, then the risk of the algorithm being trapped at local minima increases. To satisfy these two conflicting requirements, the genetic search process adopts two tactics to generate the initial population.

The first procedure is to randomly generate a certain percentage of chromosomes. Each chromosome's gene is generated first by randomly selecting a progressive operation number of an unscheduled operation and then by randomly assigning that operation to an appropriate machine. This procedure is repeated until all operations are coded.

The second procedure uses a heuristic approach to construct a Gantt-chart which is used to generate a chromosome. The procedure considers the processing time as well as the work load on the machine while assigning an operation. Thus, an operation may not necessarily be assigned to the machine with minimum processing time, but it will be assigned to the machine that will finish it sooner than other appropriate machines. The pseudo-code of the developed heuristic is shown in *Algorithm Ini-PopGen*. It should be emphasized that this algorithm generates active schedules.

Algorithm Ini-PopGen (*initial population generation to form an active schedule*)

1. Initialize Gantt-chart structure
2. Generate a random job order array J (e.g. $J_2\ J_1\ J_3$)
3. Set *oper_count* = 1and *oper* = 1
4. **While** (*oper_count* ≤ *tot_noper*) **do**
5. **For** jobs $i \in J$ **do**
6. Find all proper machines M that can process operation $O_{i,oper}$ (e.g. $M_1\ M_3$)
7. Generate an $O_{i,oper}$ stopping time array T of the same size of M
8. Initialize t_0 and t_1
9. **If** $O_{i,oper}$ is the first operation **Then**
10. Set t_0 to 0
11. **Else**
12. Set t_0 to be stop time of the predecessor operation $O_{i,oper-1}$
13. **End If**
14. **For** machines $k \in M$ **do**

15. Identify processing time $t_{i,oper,k}$ of $O_{i,oper}$ on machine M_k

16. **If** M_k did not process any operation **Then**

17. Set t_1 to 0

18. **Else**

19. Set t_1 to be the stop time of the last operation on M_k

20. **End If**

21. **If** $t_1 \leq t_0$ **Then**

22. Add $O_{i,oper}$ to M_k starting at t_0 and set $T_{oper,k} = t_0 + t_{i,oper,k}$

23. **Else If** it exist between t_0 to t_1 (time interval between two consecutive operations on M_k) $\geq t_{i,oper,k}$ **Then**

24. Add $O_{i,oper}$ to M_k starting at the end of the finished processing time of the operation to the left and set $T_{oper,k}$ equal to this time + $t_{i,oper,k}$

25. **Else**

26. Add $O_{i,oper}$ to M_k starting at t_1 and set $T_{oper,k} = t_1 + t_{i,oper,k}$

27. **End If**

28. **End For**

29. **Find** the min. stopping time on T and assign $O_{i,oper}$ to that machine. **If** more than one machine satisfy this, **Then** assign $O_{i,oper}$ to the first machine with the min. stopping time on T

30. **Encode** the selected machine M, job J_i, and operation *oper* into the chromosome

31. *oper_count* ++

32. **End For**

33. *oper* ++

34. **End While**

Figure 2.2 demonstrates how Ini-PopGen algorithm creates an active schedule for the data given in Table 2.1. Suppose the randomly created jobs priority is [3, 1, 2] (step 2). Therefore, the first operations of jobs 3, 1, and then 2 will be scheduled in the Gantt-chart as shown in Figure 2.2 (a), respectively (steps 4 to 34). Figure 2.2(b) shows the scheduling of the second operations of the jobs. If we consider for example O_{22}, then it can be noted that the operation was assigned to M_1 even though the processing time on machine M_2 is the smallest among the other machines. However, M_1 is selected as it is the machine that can finish processing this operation before others (Step 29). The final Gantt-chart is shown in Figure 2.2(c), which indicates that it is the optimal schedule for this machines' assignment as the operations of job 2 falls on the critical path of this schedule and they cannot be finished sooner without violating the precedence constraints. This emphasizes the strength of this algorithm in obtaining good initial schedules or chromosomes.

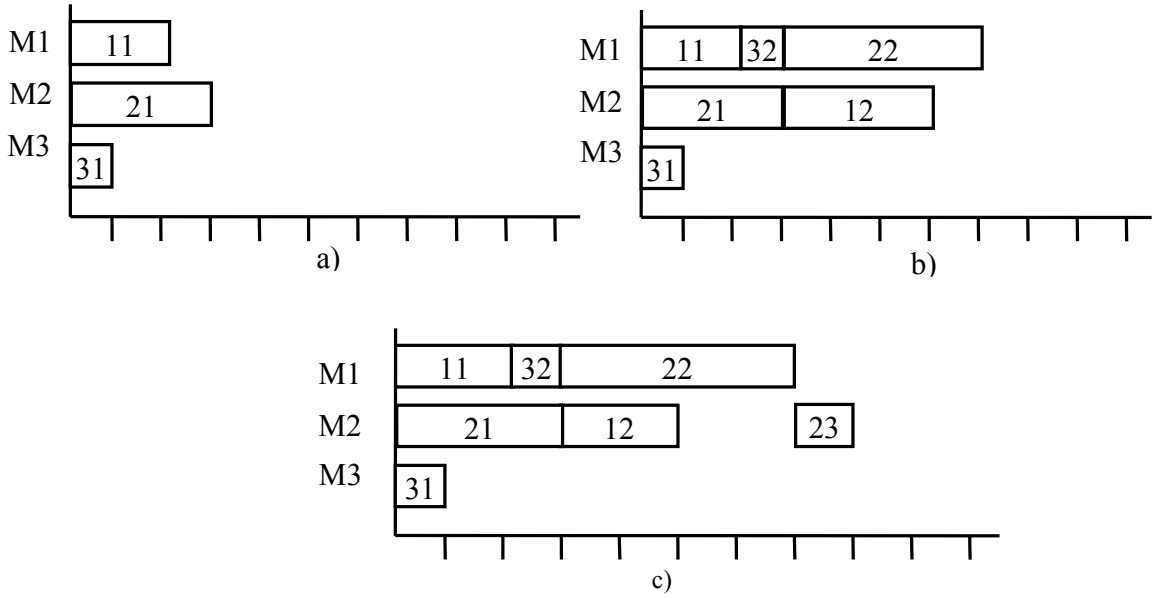


Figure 2.2: A few steps of the Ini-PopGen algorithm

2.4.4 Selection

Selection is an important element in GA. The task is to select individuals for reproduction by moving them into a mating pool. The selection method of individuals for mating is divided into two phases:

1. Phase 1 (forming the *donors' mating-pool*): Roulette wheel technique is chosen to be the main driving motor in selecting the donor chromosomes and form what is termed as, 'the donors' mating-pool'. The donors' mating-pool is formed at the beginning of each evolution process before the crossover stage starts. Two procedures were tested in this phase. The first procedure starts by first linearly ranking the individuals and sorting them according to their fitness values. Then, roulette wheel is used to select ranked individuals based on a selection probability given by the equation:

$$P_s = \frac{F_{ind}}{F_{tot}}, \qquad ind = 1, \dots, N \tag{2.1}$$

Where, P_s is the probability of choosing the ind^{th} individual; N is the population size; F_{ind} is the ind^{th} individual fitness; and F_{tot} is the total fitness of all individuals in the current generation.

The second procedure is to apply roulette wheel to form the donors' mating-pool using Eq. (2.1) without ranking the individuals. After comparing the two methods, results show that the second procedure performs better than the first one. This could be related to the population diversity that results from both procedures. When individuals are ranked according to their fitness values this gives individuals with better fitness values higher chances to be selected to form the mating pool and have higher chances to reproduce. Though such outcome is preferred to enhance the exploration around promising areas of the solution-space, at a certain stage of the evolution process, this causes the algorithm to converge prematurely to a local optimum. This is avoided in the second procedure by giving individuals with less fitness values the chance to be part of the evolution process and reduce the chances of the algorithm being trapped in local minima.

2. Phase 2 (forming the *receivers' mating-sub-pool*): After the donor's mating-pool has been formed and at the beginning of each crossover process, individual in the donors' mating-pool are subjected to a crossover probability P_c. If the individual satisfies this probability, then Phase 2 in the selection procedure starts by forming what is termed as 'receivers' mating sub-pool' using an *n*-Size tournament method, otherwise the donor will form a child. Phase 2 procedure starts by

selecting *n* different random number of chromosomes from the population and moves them to the receivers' mating-sub-pool. Following on, the individuals within the sub-pool are ranked according to the fitness and the best is chosen for reproduction. It is worth mentioning here that the donor is prevented from being reselected in this procedure.

2.4.5 Genetic operators

The performance of genetic algorithms to a great extent depends on the performance of the genetic operators used (Gen and Cheng, 2000). Therefore, designing the right genetic operators is very essential for the success of any GA. Furthermore, when applying genetic operators; crossover and/or mutation; there is a high chance of forming infeasible chromosomes by, for example, violating the precedence constraints among operations of the same job. In such cases, a repair or correction mechanism has to be implemented on the infeasible offspring. Such repair procedure is time-consuming. Therefore, it is more practical to design the operators to respect such precedence constraints. Ho et al. (2007) and Gao et al. (2008) satisfied this by dividing their chromosome into two parts where the first defines the operation order and the second defines the machine selection. Their genetic operators were separately applied to each part. Therefore, they were able to avoid producing infeasible chromosomes. However, such chromosome representation complicates the GA architecture and requires that the genetic operators to independently be applied to each part, which in turn increases the required computational time.

Since the proposed algorithm implements a chromosome representation that integrates both FJSP levels by adopting a gene of three triples (machine, job, operation), a genetic

operator which satisfies the previous requirement has to be carefully designed otherwise a repairing technique will be a necessity. In order to satisfy this, *Precedence Preserving Order-based Crossover* (POX) (Kacem et al., 2002a) and a modified *Position Based Mutation* (PBM) (Mattfeld, 1996) as well as *Machine Based Mutation* (MBM) are selected as the genetic operators.

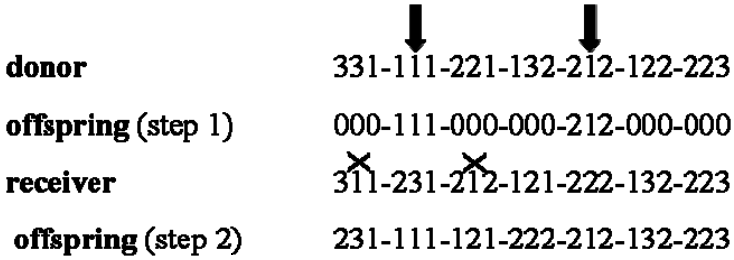


Figure 2.3: POX example

2.4.5.1 Crossover operator

The idea in POX is to transfer the selected operations in the donor to the same position in the offspring. Furthermore, we have modified the POX operator so that it does not treat the parents symmetrically. This modification assists the algorithm to control the spread of the genetic information and keep the *genetic drift* to minimum. The crossover is implemented in two steps. In step 1, a gene is randomly selected from the donor and all genes containing operations belonging to the job found in that gene are copied to the offspring. Meanwhile, the corresponding operations in the receiver are located and deleted. In step 2, the remaining offspring's genes are completed by moving the remaining operations from the receiver and inserting them in the offspring's empty genes in the same order. Figure 2.3 exemplifies this procedure for two randomly created

chromosomes representing the 3 × 3 problem given in Subsection 2.4.1. The selected genes from the donor have been pointed to by an arrow, and the corresponding deleted operations in the receiver are marked by (×). It can be observed from Figure 2.3 that POX includes an implicit mutation. For example, consider the relative position of operation 2 of job 2 to operation 2 of job 1. In both parents, operation 2 of job 1 precedes operation 2 of job 2. However, this is not the case in the offspring. Therefore, if both operations are to be processed on the same machine, then unlike the parents, operation 2 of job 2 will be scheduled before operation 2 of job 1 in the offspring. Moreover, when implementing POX the operations belonging to the same job are moved from the parent to the offspring in the same order. Consequently, the precedence constraints of jobs are not violated and hence no infeasible chromosome is produced. Thus, no repairing mechanism is required.

2.4.5.2 Mutation operator

After creating a number of children equal to the number of parents using crossover according to a crossover probability P_c, children are subjected to mutation according to a mutation probability P_m. This is done to maintain the diversity of the chromosomes and to introduce some extra variability into the population. Two strategies were followed during the mutation process:

1. The first strategy called *Machine Based Mutation* (MBM), where a random number of operations (denoted as *nrand*) are selected and reassigned to another machine. As introduced earlier, the purpose of mutation is to increase the diversity of the population. However, if a drastic change is imposed in the

chromosomes' structure, then the genetic search will turn to be a random search. Therefore, the resultant chromosome from mutation should not be significantly different from the original chromosome. Number of experiments were conducted to select the appropriate values of *nrand* that satisfy the previous condition. Hence, *nrand* is limited between [1, 3]. After *nrand* is uniformly selected from that range, *nrand* operations are randomly selected within the child's chromosome. The selected operations are then reassigned to another machine if applicable to form a *child'*. In order to reduce the computational time, a local memory is linked to each operation. This local memory contains information about the machines that can process this specific operation and their number. This mutation procedure is illustrated in Figure 2.4, where it is assumed that the generated *nrand* = 2 and the randomly selected genes are pointed to by an arrows. Figure 2.4(a) expresses that only one of the two operations can reassigned to another machine.

2. The second strategy called modified *Position Based Mutation* (PBM) (Mattfeld, 1996). PBM was originally designed for JSP using single triple permutation-based chromosomes representation. Thus, the PBM is modified so that no infeasible chromosomes are produced. This mutation starts by randomly selecting an operation from *child'* and reinserting it at another position in the new child. Then, the remaining operations are copied from *child'* to the new child taking care of the precedence constraints of the moved operation. Figure 2.4(b) demonstrates this mutation procedure, where the selected gene is underlined and the new position is pointed to by an arrow.

After crossover and mutation is implemented, the children are taken to form the new generation. The algorithm terminates after reaching a maximal number of generations.

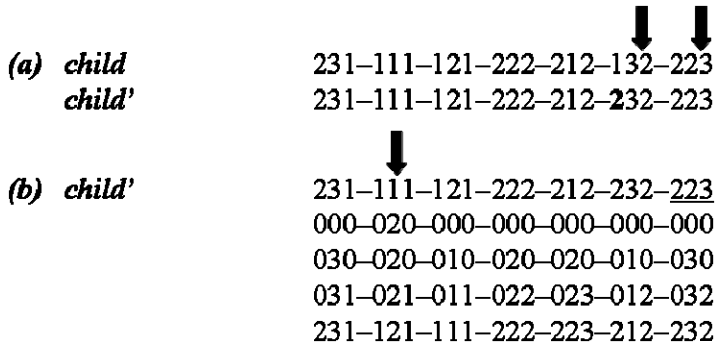


Figure 2.4: (a) Sample procedure of MBM (b) Sample procedure of modified PBM.

2.4.6 Local search

The use of local search techniques has been proven to be useful in solving combinatorial problems. Local search methods are applied to a neighborhood of a current solution. In the case of JSP, a neighborhood is achieved by moving and inserting an operation in a machine sequence. Therefore, when designing a local search method that works on neighborhood of solutions, two main issues have to be considered. The first issue is that the size of the neighborhood has to be limited. The second issue is the feasibility of solutions.

Generally, there are two scenarios to deal with the feasibility issue. The first scenario is to design a local search that only considers feasible moves as in the two neighborhood functions proposed by Mastrolilli and Gambardella (2000). The second scenario is to design a local search that accepts all moves and then either rejects the infeasible moves or

implement a repairing procedure like the proposed repairing technique presented by Murovec and Šuhel (2004).

Brandimarte (1993) suggested the following neighborhood structures:

1. Neighborhood $\mathcal{N}_1$ where a sample size is set to randomly sampling the potential exchanges within the neighborhood.
2. Neighborhood $\mathcal{N}_2$ where a job is randomly selected and it is exchanged on each machine with the adjacent jobs in each machine sequence.
3. Neighborhood $\mathcal{N}_3$ where a machine is randomly selected and the complete set of job exchanges on that machine is considered.
4. Neighborhood $\mathcal{N}_4$ where operations on the critical path are considered for the exchange.

The second and third neighborhood structures limit the search space by confining it to either focus on a specific arrangement, whereas the first gives some degree of random exploration. Several researchers have verified that when the makespan is the considered objective function, then neighborhood $\mathcal{N}_4$ is very useful since operations not laying on the critical path do not affect the makespan (see for example Brandimarte, 1993; Mastrolilli and Gambardella, 2000; Murovec and Šuhel, 2004). However, a local search that is solely devoted to work on critical path heads directly toward deterministically predictable local minima. On the other hand, a local search that is based on random search helps preserving the diversification of the population. Hence, a local search that combines both approaches by introducing randomness in a local search that is related to the critical path may be more desirable. Therefore, it is proposed using a local search method that combines neighborhoods $\mathcal{N}_1$ and $\mathcal{N}_3$ while defining hill climbing heuristic

that works on blocks of moves that contain a critical block as subset (for more details about critical blocks, readers are referred to Mattfeld, 1996).

The proposed local search starts with a feasible schedule $\mathcal{S}$ as an input. The input schedule is set to $\mathcal{S}_{best}$ which stands for the best found solution. Then, a neighborhood consisting of blocks containing feasible moves on each machine is constructed. A feasible move is defined by two consecutive operations on the same machine that can be interchanged without violating the precedence constraint. Therefore, each block will contain at least two consecutive operations which correspond to one move. To that, an ordinary hill climbing heuristic is applied to bring the new schedule $\mathcal{S}'$ to the local optimum with respect to the used neighborhood. The hill climbing heuristic works by interchanging or swapping two consecutive operations within one block at a time. Every time a move is done, the new makespan of $\mathcal{S}'$ is estimated. If the new $\mathcal{S}'$ is better (i.e. has a lower makespan) than $\mathcal{S}_{best}$, then $\mathcal{S}_{best}$ is replaced by $\mathcal{S}'$. The procedure is repeated until a maximal number of iterations (*loc_iter*) without improving moves is reached. The pseudo-code of the local search heuristic is shown in *Algorithm LS.*

Algorithm LS (*local search)*

35. Calculate the performance $C_{max}(\mathcal{S}_{best})$ of the current schedule $\mathcal{S}$

36. Set *count* to 1

37. **While** ((*count* ≤ *loc_iter*) && ($\mathcal{S}' \in \mathcal{N}_{hc,feasible}(\mathcal{S}_{best})$) **do**

38. **If** $C_{max}(\mathcal{S}') < C_{max}(\mathcal{S}_{best})$ **Then**

39. Update $\mathcal{S}_{best}$ by setting $\mathcal{S}_{best} = \mathcal{S}'$

40. Set *count* to 1

41. **Else**

42. *Count*++

43. **End If**

44. **End While**

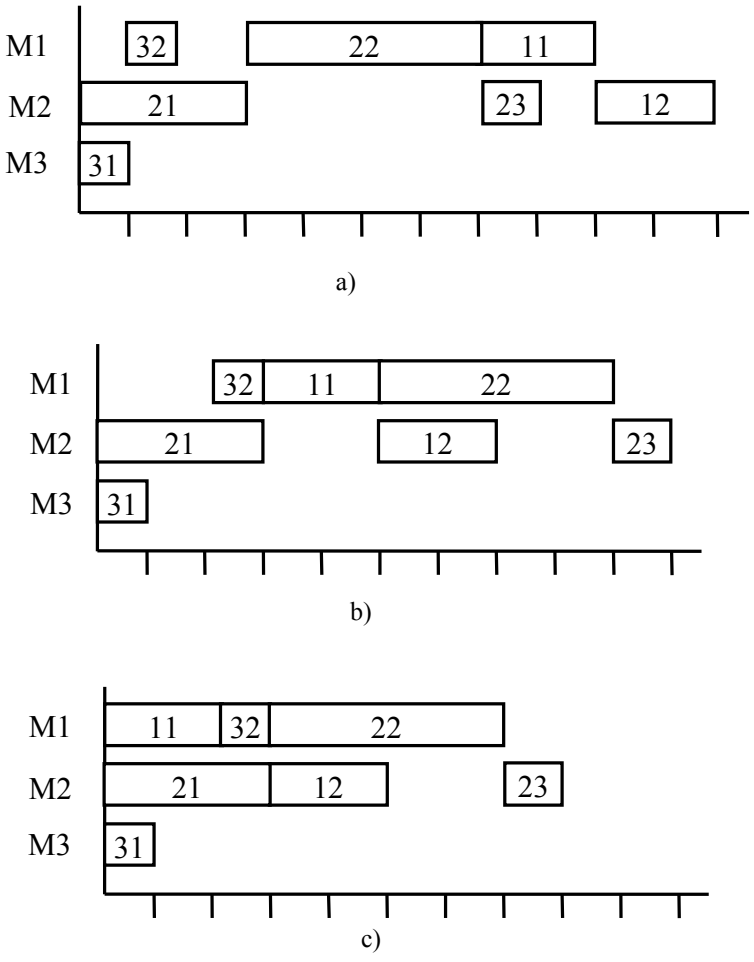


Figure 2.5: Gantt-charts of three neighborhoods.

To illustrate how the local search works, consider the Gantt-charts shown in Figure 2.5. Assuming that Figure 2.5(a) represents an input schedule $\mathcal{S}_{best}$ to the local search, blocks; which may or may not include critical operations; containing feasible moves are in machine 1 (O_{32}, O_{22}, O_{11}), and in machine 2 (O_{23}, O_{12}). It can be seen that O_{21} is not included in the second block as if it is to be swapped with O_{23}; it will result in an infeasible schedule. Since machine 3 does not process any operation other than O_{31}, then no block is formed. The first move is done by swapping O_{32} and O_{22}. However, since the proposed algorithm considers active scheduling this move does not change the schedule. The second move is done by interchanging O_{22} and O_{11}. This produces a better schedule $\mathcal{S}'$ as shown in Figure 2.5(b) and hence, $\mathcal{S}_{best}$ is replaced by $\mathcal{S}'$. In the same time, the blocks are automatically updated with the new changes in the schedule and the modified blocks become in machine 1 (O_{32}, O_{11}, O_{22}), and in machine 2 (O_{12}, O_{23}). Since an improvement was found in the previous move, the algorithm continues by moving operations on the second block (O_{12}, O_{23}). Again, since active schedules are considered, this move will result in no improvement of the makespan of the current schedule. Assuming that the maximum number of allowed moves without improvement is not reached, the algorithm continues the search by considering moves within block (O_{32}, O_{11}, O_{22}). Interchanging O_{32} and O_{11} results in an improved schedule as shown in Figure 2.5(c) and again $\mathcal{S}_{best}$ is replaced by $\mathcal{S}'$. The algorithm iterates until the termination criterion is met.

In order to reduce the computation time, *loc_iter* is limited and rather kept small. However, this is compensated for by combining the local search with the GA many times during the optimization process and every time the local search is called there is a

different starting solution. Hence, it is designed for exploitation purposes whereas GA takes care of the exploration of the solution space. Furthermore, unlike other memetic algorithms, the local search is only applied at every d^{th} generations of the evolution process. This is to allow the population to have some diversity and enhancing the chances of preventing the solution from being trapped on local minima.

2.4.7 Elitism strategy

To prevent the loss of the genetic information of the current best chromosome during the evolutions, a global memory containing the elitism is generated. During the evolution process (crossover or mutation) both best and worst chromosomes are identified and recorded. At the end of each generation, the best chromosome is compared with the elitism. If the best chromosome has a better fitness value than the elitism, then the global memory is updated by replacing the elitism with this new chromosome. On the other hand, the global memory does not interfere with the genetic evolution except if the fitness value of the best chromosome is worse than that of the elitism. In such case, the elitism is used to replace the worst chromosome in the population. However, this replacement procedure takes place every k^{th} generation. This strategy enables chromosomes to naturally evolve in almost all generations and hence, allowing an acceptable level of diversity on the population. Meanwhile, the knowledge gained from all generations is accumulated and only used to guide the evolution process when necessary.

2.5 Computational Results

FJSP are classified into two sub-problems known as total flexible job-shop problems (T-FJSP) and partial flexible job-shop problems (P-FJSP). In T-FJSP each operation can be processed on any machine of *M*. On the other hand, in P-FJSP each operation can be processed on at least one machine of a subset of *M*. Therefore, several sets of problem instances have been considered.

1. The first set (KMData) consists of 6 instances, 3 T-FJSP taken from Kacem et al. (2002b); 1 T-FJSP taken Mesghouni et al. (1997), 1 P-FJSP taken from Lee and DiCesare (1994); and 1 P-FJSP taken Kacem et al. (2002a).
2. The second set (BRData) consists of 10 P-FJSP instances taken from Brandimarte (1993).
3. The third set (BCData) consists of 21 P-FJSP instances taken from Barnes and Chambers (1996).
4. The fourth set (DPData) consists of 18 P-FJSP instances taken from Dauzére-Pérés and Paulli (1997).
5. The fifth set (HUData) consists of 129 P-FJSP instances created from 43 classical job-shop instances divided into three subsets, EData, RData and VData taken from Hurink, Jurisch and Thole (1994).

The performance of the different components of the proposed hybrid algorithm is analyzed using all sets.

For the hGA, experiments were conducted on an Intel® Pentium® D CPU 2.8 GHz with 2 GB RAM. For each test problem, a set of 5 runs were performed. A number of

experiments were conducted for the same test problem in order to determine the hGA parameters. In order to identifying the values for each parameter, the following procedure was adopted. First, from each of the five set instances number of test cases are randomly selected. For each selected problem the values of all hGA parameters are fixed and only the value of the parameter under consideration is varied. For example, to determine the value of the crossover probability, the parameters for the evolution were fixed to: mutation probability = 0.3, number of maximal generations = 1000; maximum number of moves without improvement in the local search *loc_iter* = *min* [*tot_noper*, 175]; the worst chromosome is replaced by the elite chromosome every k = 3 generations; number of parents in the receivers' mating sub-pool = 4; number of generations to perform local search (d value) 10, and the population size = 200. Then crossover probability of 0.9, 0.7, and 0.5 were investigated by solving the test problem 5 times. After solving all randomly selected test problems, the crossover probability that performed better in terms of solution quality is selected for the remaining instances. A similar procedure is followed in determining the values of remaining other parameters. In conclusion of the previous analyses, the following parameter values are common in all test cases: crossover probability 0.7; mutation probability 0.3; number of maximal generations 1000; maximum number of moves without improvement in the local search *loc_iter* = *min* [*tot_noper*, 175]; the worst chromosome is replace by the elite chromosome every k = 3 generations; number of parents in the receivers' mating sub-pool 4 for the KMData and BRData sets, and 6 for the rest; and number of generations to perform local search (d value) as well as the population size are reported in Table 2.2.

Table 2.2: *d* value and population size used in test sets

Test Set	Instance		*d* value	Population size
KMData	all		10	200
BRData	MK1/6/7/10		30	1200
	all remaining		10	200
BCData	all		30	1500
DPData	all		30	1200
HUData	EData	mt6/10	20	200
		mt20	30	800
		la01-la13	20	400
		la14-la25	20	500
		la26-la40	30	1200
	RData	mt6/10/20	20	300
		la01-la25	20	400
		la26-40	30	1000
	VData	mt6/10/20	10	200
		la01-la40	10	300

The first two sets are solved by using initial population generation heuristic (Ini-PopGen), initial population generation heuristic with local search procedure (Ini-

PopGen+LS), and finally using the full hybridized algorithm (hGA). In each instance of these sets, 60 instance schedules are created using the Ini-PopGen and another 60 schedules are created using Ini-PopGen+LS. The best and the average makespan of 5 runs are compared. Table 2.3 shows the results obtained for KMData set and compares them with these obtained by Ho et a. (2007) and Gao et al. (2008). The values in bold-face identify the best values obtained for each instance using all algorithms. The values between braces are the average results. It can be seen from Table 2.3 that the proposed algorithms are very efficient in solving all instances. Ho et al. (2007) solved these instances using two methods. Firstly they used composite dispatching rules population generation (CDR-PopGen), and in the second they used LEarnable Genetic Architecture (LEGA). The proposed Ini-PopGen is able to obtain the minimum known results for all instances except for Ex4. It even outperformed the best and the average results obtained by CDR-PopGen in Ex3 and Ex6; and the average results obtained by LEGA in Ex3 and Ex4. In Ex3 proposed by Mesghouni et al. (1997); Ini-PopGen was able to obtain the optimal results of 7 time units. The optimal result was also obtained by Mesghouni et al. (1997) using GA, but after 1500 generations. Furthermore, results in Table 2.3 indicate that Ini-PopGen can handle small and medium T-FJSP instances. The P-FJSP instance Ex5 consists of 5 jobs, 3 machines and 4 operations in each job with a lot size of 10. This problem was solved by Lee and DiCesare (1994) using Petri Nets combined with heuristic search and obtained an average makespan of 439; Kumar et al. (2003) using Ant Colony approach and obtained an average makespan of 420; and by Chan et al. (2006) using GA with dominant genes (DGA) and obtained minimum makespan of 360 with an average of 374. It can be noticed from Table 2.3 that the proposed algorithms

outperformed the previous results in both minimum and average makespans. Figure 2.6 shows the Gantt-chart for the production scheduling of this example. For Ex6 taken from Kacem et al. (2002a), the algorithms outperform that proposed by Ho et al. (2007) as the Ini-PopGen is able to obtain the best known minimum makespan. Moreover, the best results obtained by hGA for Ex3, Ex4 and Ex6 are the same as these obtained by Gao et al. (2008).

Table 2.3: Comparison with other meta-heuristics on KMData

Reference	Problem size		Gao et al. (2008)	Ho et al. (2007)		Proposed methods		
				CDR-PopGen	LEGA	Ini-PopGen	Ini-PopGen+LS	hGA
Kacem et al. (2002b)	**Ex1**	4 × 5	-	**11(11)**	**11(11)**	**11(11)**	**11(11)**	**11(11)**
	Ex2	10 × 7	-	**11(11)**	**11(11)**	**11(11)**	**11(11)**	**11(11)**
Mesghouni et al. (1997)	**Ex3**	10 × 10	**7(7)**	8(8)	**7**(7.56)	**7(7)**	**7(7)**	**7(7)**
Kacem et al. (2002b)	**Ex4**	15 × 10	**11(11)**	12(12.24)	12(12.04)	12(12)	12(12)	**11**(11.2)
L and DC (1994)	**Ex5**	5 × 3	-	-	-	**350**(352.5)	**350**(351)	**350(350)**
Kacem et al. (2002a)	**Ex6**	8 × 8	**14(14)**	16(16)	**14(14)**	**14**(14.5)	**14**(14.2)	**14(14)**

Values written in bold are the best values

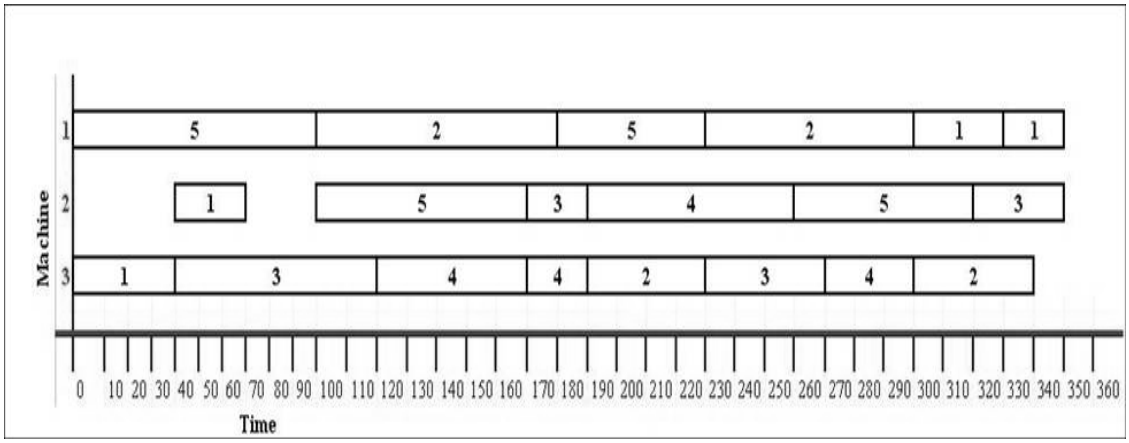


Figure 2.6: Diagram of Gantt-chart for the production scheduling of Ex5.

Table 2.4 compares the BRData set results obtained by the proposed algorithms with these obtained by Ho et al. (2007) using CDR-PopGen and LEGA; and these obtained by Girish and Jawahar (2008) who solved them using Ant Colony Optimization (JSSANT) and Genetic algorithms (JSSGA). Table 2.4 shows that the proposed hGA either outperforms other algorithms or obtain the same minimum makespan in all cases. Furthermore, Table 2.4 also highlights that the proposed Ini-PopGen has a very acceptable efficiency in solving P-FJSP as it is able to produce comparative results with these obtained by other best methods of JSSANT and LEGA. For example, Ini-PopGen is outperforming JSSGA in seven out of ten MK instances; and JSSANT in instances MK06, MK09 and MK10. Moreover, the noticeably close values of the obtained results between the minimum and the average makespan in both benchmark sets (T-FJSP, and P-FJSP) highlights the repeatability of the proposed hGA and its consistency in obtaining optimal or near optimal result.

Table 2.4: Comparison with other meta-heuristics on BRData

Problem size	Girish and Jawahar (2008)		Ho et al. (2007)		Proposed methods		
	JSSANT	JSSGA	CDR-PopGen	LEGA	Ini-PopGen	PopGen +LS	hGA
MK01 10 × 6	**40(40)**	**40**(41.2)	42(42)	**40**(41.5)	42(42)	42(42)	**40(40)**
MK02 10 × 6	**26**(26.8)	**26**(29.6)	30(30)	29(29.1)	28(28.88)	28(28.8)	**26**(27.10)
MK03 15 × 8	**204(204)**	212(215.2)	-	-	**204(204)**	**204(204)**	**204(204)**
MK04 15 × 8	66(66.8)	71(74)	68(68)	67(68.82)	74(74.75)	73(74)	**61(62.83)**
MK05 15 × 4	174(176.4)	188(191.2)	179(179.3)	176(178.1)	179(179.63)	177(179.4)	**173(174.67)**
MK06 10 × 15	77(78.4)	81(83)	69(69.2)	67(68.82)	70(70.63)	69(70)	**62(64.83)**
MK07 20 × 5	143(143.8)	152(154)	153(153.88)	147(152.9)	150(153.75)	150(153.4)	**141(143)**
MK08 20 × 10	**523(523)**	533(545)	527(528.44)	**523**(523.34)	544(544.25)	**523**(524.2)	**523(523)**
MK09 20 × 10	328(341.2)	378(382)	326(328.78)	320(327.74)	326(326)	319(323.2)	**307(307)**
MK10 20 × 15	247(254.8)	265(281.4)	234(236.12)	229(235.72)	233(234)	233(234)	**214(218.33)**

Values written in bold are the best values

When comparing the performance of Ini-PopGen algorithm with the performance of Ini-PopGen combined with LS (Ini-PopGen+LS) in Tables 2.3 and 2.4, it can be observed that no improvements are achieved in all T-FJSP and, in general, improvements in P-FJSP instances are small. This can be related to two main reasons. The first reason can be related to the possibility that the obtained solution using Ini-PopGen algorithm is the minimum or near minimum feasible solution. This reason can be supported by considering results obtained using different algorithms in Table 2.4, where different algorithms obtained the same minimum solutions. The second reason may be related to the nature of FJSP where it consists of two decision levels. Here, it is possible that solutions produced using Ini-PopGen are optimized to a near optimal local solution for

that machines' assignment decision level. In such case, expected improvements using any local search method that only works on the sequencing decision level, like the proposed LS, are small. This conclusion can be sustained by the obtained results of instances MK08 and MK09 where the LS has much improved the results obtained by Ini-PopGen and it even brought the solution of MK08 to optimality.

Table 2.5 compares the best results obtained by the proposed hGA to the best results of the particle swarm algorithm proposed by Girish and Jawahar (2009); and GA algorithms proposed by Zribi et al. (2007); Pezzella et al. (2008); and Gao et al. (2008), on the BRData set instances. The first column reports the instance name, the second reports the best-known lower bound and the third reports the currently best obtained results. The remaining columns report the best results of the three algorithms with the relative deviation with the respect to the proposed algorithm defined as follows:

$$dev = \left[\left(MS_{comp} - MS_{hGA}\right)/MS_{hGA}\right] \times 100 \quad (2.2)$$

Where, MS_{comp} is the makespan we compare to and MS_{hGA} is the makespan obtained by the proposed hGA algorithm. The results show that the obtained results by hGA are either outperforming other methods or of comparable quality.

Table 2.6 compares the relative error of the obtained computational results over the last four sets BRData, BCData, DPData, and HUData and the relative error of the results obtained by Zribi et al. (2007); Pezzella et al. (2008); and Gao et al. (2008); with respect to the best-known lower bound. The relative error (*RE*) is defined as:

$$RE = [(MS - LB)/LB] \times 100 \quad (2.3)$$

Where, MS is the best makespan obtained by the reported algorithm and LB is the best-known lower bound. The first column reports the data set, the second column reports the number of instances in each set, the third column reports the average number of alternative machines per operations. The last three columns report the RE of the results obtained by the proposed hGA algorithm; Zribi et al. (2007); Pezzella et al. (2008); and Gao et al. (2008); respectively. The results support the conclusion drawn from Table 2.3 above that the proposed hGA is stronger with high degree of flexibility. Also, it shows that the proposed hGA outperforms the GA proposed by Zribi et al. (2007) and by Pezzella et al. (2008); and produces a comparable quality to the algorithm proposed by Gao et al. (2008). It may worth to mention here that the proposed hGA has obtained these results with less number of chromosomes than the other two algorithms. For example, Pezzella et al. (2008) used 5000 chromosomes in all their experiments; and Gao et al. (2008) used on average +2000 chromosomes, however, the proposed hGA used on average 700 chromosomes to solve the test cases. Furthermore, the advantage of the hGA approach is that it can work for instances with different flexibility. Also, it can be adapted to deal with other criteria than the makespan. This is because the local search algorithm is designed to work in combined neighborhoods $\mathcal{N}_1$ and $\mathcal{N}_3$, rather than only considering neighborhood $\mathcal{N}_4$.

Table 2.5: Comparison with algorithms proposed by Zribi et al. (2007); Girish and Jawahar (2009); Pezzella et al. (2008); and Gao et al. (2008) on BRData

Name	LB	hGA	Z.K.K.	dev (%)	G.J.	dev (%)	P.M.C.	dev (%)	G.S.G.	dev (%)
MK01	36	40	41	+2.5	40	0	40	0	40	0
MK02	24	26	28	+7.69	27	+3.85	26	0	26	0
MK03	(204)	204	204	0	204	0	204	0	204	0
MK04	48	61	67	+9.84	62	+1.64	60	-1.64	60	-1.64
MK05	168	173	177	+2.31	178	+2.89	173	0	172	-0.58
MK06	33	62	61	-1.61	78	+25.81	63	+1.61	58	-6.45
MK07	133	141	154	+9.22	147	+4.26	139	-1.42	139	-1.42
MK08	(523)	523	523	0	523	0	523	0	523	0
MK09	299	307	321	+4.56	341	+11.07	311	+1.3	307	0
MK10	165	214	219	+2.34	252	+17.76	212	-0.93	197	-7.94

Table 2.6: Mean relative error over the best-known lower bound

Data Set	Num. of Ins	Alter.	hGA	Z.K.K.	P.M.C.	G.S.G.
BRData	10	2.59	17.58	21.62	17.53	14.92
BCData	21	1.18	24.74	-	29.56	22.61
DPData	18	2.49	6.83	8.27	7.63	2.12
HUData						
EData	43	1.15	3.92	-	6.00	2.51
RData	43	2	3.68	-	4.42	1.21
VData	43	4.31	0.80	-	2.04	0.09

2.6 Conclusion

In this Chapter, a hybridized genetic algorithm for the flexible job-shop scheduling problem is proposed. The hybrid architecture consists of three main parts, initial population generation heuristic, a local search method, and a genetic algorithm. The experimental results advocate the good performance of the proposed Ini-PopGen heuristic by outperforming some of the existing approaches in the literature. The performance of Ini-PopGen is improved when combining it with the proposed LS. This performance indicates that Ini-PopGen with LS can be used as a stand-alone tool for small to medium sized T-FJSP and P-FJSP. The proposed hGA has the ability to further improve the quality of results obtained by using Ini-PopGen with LS. Thus, the hGA structure is very effective and has a good potential of obtaining optimal or near optimal results. A very strong advantage of the proposed hGA architecture is that the genetic operators of

crossover and mutation do not require a repair process to obtain a feasible schedule. Furthermore, the current work maintains the diversity of population by implementing different techniques like the individuals selection method, the mutation techniques, the elitism strategy, and by applying the local search every *d* generations. This allows the GA to explore more solution space whereas Ini-PopGen and LS does the exploitation part.

CHAPTER 3

ROBUST AND STABLE FLEXIBLE JOB-SHOP SCHEDULING WITH RANDOM MACHINE BREAKDOWNS USING A HYBRIDIZED GENETIC ALGORITHM

This Chapter addresses the problem of finding robust and stable solutions for the flexible job-shop scheduling problem with random machine breakdowns. A number of bi-objective measures combining the robustness and stability of the predicted schedule are defined and compared while using the same rescheduling method. Consequently, a two-stage hybrid Genetic Algorithm (hGA) is proposed to generate the predictive schedule. The first stage optimizes the primary objective, minimizing makespan in this work, where all the data is considered to be deterministic with no expected disruptions. The second stage optimizes the bi-objective function and integrates machines assignments and operations sequencing with the expected machine breakdown in the decoding space. An experimental study and Analysis of Variance (ANOVA) is conducted to study the effect of different proposed measures on the performance of the obtained results. Results indicate that different measures have different significant effects on the relative performance of the proposed method. Furthermore, the effectiveness of the current

proposed method is compared against three other methods; two are taken from literature and the third is a combination of the former two methods.

3.1 Introduction

Job-shop scheduling problems are among the most intensive combinatorial problems studied in literature. Until recently, scheduling problems were studied assuming that all of the problem parameters are known beforehand. However, such an assumption does not reflect the reality as unforeseen incidents happen in real manufacturing systems. Thus, an optimal schedule that is produced based on deterministic measures may result in a degraded system performance when released to the shop floor (Leon et al., 1994). For this reason more emphasis is put towards producing schedules that can handle uncertainties caused by random disruptions.

Generally, job-shop scheduling can be classified into two main classes, *static* or *deterministic* (offline) scheduling, and *dynamic* (online) scheduling. Liu et al. (2007a) classified attempts to scheduling in the presence of disruptions into two groups. One group with completely reactive job dispatching scheduling, and the second group offers control strategies to aid the system recovery from disruptions with consideration of an initial schedule. The main difference between the two groups is that no schedule is generated in advance in the complete reactive scheduling, but decisions are made on real time using priority dispatching rules. On the other hand, the second group uses a predetermined schedule called *preschedule* or *predictive schedule* that optimizes a certain performance measure and is implemented until some unforeseen disruption occurs in the system. Based on the nature of the control strategy the system will respond to the

disruption by *rescheduling* the part of the predictive schedule that is not implemented into a new schedule to accommodate that disruption.

This Chapter studies the scheduling robustness and stability of flexible job-shops where disruptions in the form of machine breakdown are expected. It proposes an approach to obtain a predictive schedule that minimizes the effect of machine breakdowns in the overall performance such that the makespan is preserved and increases the schedule stability with respect to the sum of the absolute deviations of operation completion times between the realized schedule and the predictive schedule. The obtained predictive schedule is meant to satisfy two conflicting objectives where it first has to efficiently utilize the resources, and second, it has to allow sufficient flexibility for changes. This is achieved by considering the robustness and stability as a bi-objective, with a primary objective to minimize the schedule makespan. The approach adopted here was first introduced by Liu et al. (2007b) for single machine scheduling problem with random machine breakdowns and we extended it for the FJSP. Unlike the approach of Liu et al. (2007b), the current proposed approach is a non-idle time insertion method. It is modified to utilize the available flexibility of machine routing with the expected machine breakdowns to obtain the predictive schedule. Furthermore, a two-stage hybridized Genetic Algorithm (hGA) is proposed to generate the predictive schedule. The first stage optimizes the primary objective where all data is considered to be deterministic with no expected disruptions. The second stage uses the final population from the first stage as its initial population. This stage optimizes the bi-objective function and integrates the determination of the operations sequence with the expected machine breakdown in the decoding space. Furthermore, Analysis of Variance (ANOVA) is conducted to study the

effect of different proposed robustness and stability measures on the performance of the obtained results.

The remained of this Chapter is organized as follows. Section 3.2 presents the literature review. Section 3.3 describes the problem formulation of the flexible job-shop scheduling problem. In Section 3.4, the measures of robustness and stability, the generation of machine breakdown and the rescheduling methods used in the experiments are illustrated. Section 3.5 presents the analysis of using the different robustness and stability measures, and the comparison between the proposed methodology and a classical model that assumes deterministic data with no disruptions. Finally, a concluding summery is given in Section 3.6.

3.2 Literature Review

Most scheduling problems including FJSP have been considered as NP-hard. Hence, heuristic and meta-heuristic approaches have received much attention in literature. In the following, a brief survey of stochastic scheduling approaches is given.

Numerous papers addressed stochastic single machine with uncertain jobs processing times, such as the work of Daniels and Kouvelis (1995); Kouvelis et al. (2000); Möhring et al. (1985); Montemanni (2007); Pinedo (1982); Wu et al. (2009); and Xia et al. (2008). Al-Turki et al. (1996); Cai and Tu (1996); and Liu et al. (2007b) considered single machine shops subjected to machine breakdowns. Furthermore, Sevaux and Sörensen (2004) used a modified GA to find robust solution in single machine environment subjected to stochastic release dates of jobs.

Byeon et al. (1998); Kutanoglu and Wu (1998); Kutanoglu and Wu (2004); and Wu et al. (1999) used decomposition heuristic to divide the classical job-shop scheduling problem with uncertain processing times into a series of subproblems and iteratively update the problem parameters to analyze the effect of the processing time variation using *a priori* stochastic information. Shafaei and Brunn (1999); and Shafaei and Brunn (2000) used the rolling time approach to investigate the robustness of schedules. Cowling et al. (2004) used a previously proposed multi-agent architecture with two measures of stability and utility to produce a robust predictive/reactive schedule. Policella et al. (2004) and Policella et al. (2005) studied two-stage approach to generate a robust flexible partial order schedule for the Resource-Constraint Project Scheduling problem with minimum and maximum time lags.

Leon et al. (1994) proposed a slack-time based robustness measures to analyze the effects of machine breakdowns and processing-time variability on the quality of the classical job-shop schedules. The most promising robustness measure is found to be

$$Z_{r=1}(s) = MS_{min} - RD3(s) \tag{3.1}$$

where, MS_{min} is the makespan of schedule s, and $RD3(s)$ is the average operation slack in schedule s. Lawrence and Sewell (1997) studied the performance of simple dispatching heuristics versus algorithmic solution techniques in job-shops subjected to uncertain processing times. A similar study was done by Sabuncuoglu and Karabuk (1999) which showed that dispatching rules are more robust to interruptions than the optimum seeking off-line scheduling algorithms. Mehta and Uzsoy (1998) presented an algorithm based on disjunctive graph representation to integrate random breakdowns of machines and minimizing their effect by inserting idle time into the predictive schedule of a job-shop to

absorb the impact of breakdowns. Jensen (2001b) and Jensen (2003) tried to improve the robustness and flexibility of the job-shop schedules when minimizing maximum tardiness, summed tardiness, total flow-time and makespan measures. Both studies used GA (developed in Mattfeld, 1996) and considered two robustness measures, the neighborhood-based robustness measure and the lateness-based robustness measure. He defined schedule neighborhood $\mathcal{N}_1$ (s) robustness measure, where $\mathcal{N}_1$ (s) contains s and all feasible schedules that can be created from s by interchanging two consecutive operations on the same machine, as a weighted average of makespans of schedules in $N_1(s)$ and is given as follows

$$Z_{MS_{min}}(s) = \frac{1}{|N_1(s)|} \sum_{s' \in N_1(s)} MS_{min}(s') \qquad (3.2)$$

where, $MS_{min}(s')$ is the makespan of schedule s'. Laslo et al. (2008) considered the case of determining the machine booking schedule for a virtual job-shop problem to maximize the economic gain from outsourced rented machines. Authors assumed that operations processing times are normally distributed, and hence proposed a heuristic based method.

Anglani et al. (2005) proposed a fuzzy mathematical model of scheduling parallel machines with sequence-dependent cost while considering uncertainties in processing times. Recently, Bouyahia et al. (2009) presented a probabilistic generalization to design robust a priori scheduling that assumes the number of jobs to be processed on parallel machines as a random variable with respect to the total weighted flow time.

Guo and Nonaka (1999) studied how to reduce the effect of machine failure on a three-machine flow shop by proposing a method to evaluate initial schedules (preschedules) and a rescheduling method that is applied after machine failure. Matsveichuk et al. (2009) proposed a two-stage scheduling decision framework to execute schedules of a two-

machine flow shop with interval processing times. Qi et al. (2006) introduced a rescheduling approach for single and parallel two-machine environment subjected to random machine unavailability and processing time variations.

Artigues et al. (2003) proposed insertion techniques for static and dynamic resource-constrained project scheduling. Surico et al. (2006) suggested a hybrid meta-heuristic that integrates a mathematical programming, multi-objective evolutionary computation (genetic algorithm), and a problem-specific constructive heuristic that returns a number of solutions or the pareto sets (schedules), each with a cost and risk trade-offs, for the problem of Supply Network (SN) for ready-mixed concrete (RMC). Chtourou and Haouari (2008) presented a two-stage algorithm to produce robust resource-constrained project scheduling subjected to unpredictable increase in processing times. Lambrechts et al. (2008) proposed a tabu search algorithm that uses a free slack-based objective function to produce robust predictive-reactive project schedules in the presence of uncertain renewable resource availabilities.

Rangsaritratsamee et al. (2004) proposed a rescheduling method based on local search genetic algorithm for a job-shop with dynamically arriving jobs. Their proposed algorithm simultaneously considers the efficiency by preserving the makespan, tardiness and stability by minimizing the jobs starting time deviations. In their work, the rescheduling takes place at specific time intervals using all available jobs at the rescheduling moment. Fattahi and Fallahi (2010) combined the work of Rangsaritratsamee et al. (2004) and Fattahi et al. (2007) and developed a multi-objective genetic algorithm based method to scheduling a flexible job-shop with dynamically arriving jobs. Mahdavi et al. (2010) presented a real-time simulation-based decision

support system to control the production of a stochastic flexible job-shop subjected to stochastic processing times.

Vinod and Sridharan (2009) experimentally studied different scheduling decision rules for scheduling dynamic flexible job-shop (jobs arrive intermittently) using a discrete simulation-based model. They considered a partial flexible job-shop system. The system consists of eight machines wherein an operation can be executed on three different ones. Vinod and Sridharan (2011) continued the previous study by studying the interaction between due-date assignment methods and scheduling rules in a dynamic job-shop system using a discrete-event simulation model. Their simulation analysis showed that due-date assignment methods and the used scheduling rules are significantly affecting the performance measures of the shop.

For a recent overview discussing aspects of scheduling with uncertainties readers are referred to Davenport and Beck (2002), Aytug et al. (2005), Herroelen and Leus (2005) and Mula et al. (2006) who gave detailed review of literature related to scheduling under uncertainty.

In light of the above literature, approaches used to achieve schedule robustness are classified into two categories, preservation of solution quality approach and execution-oriented quality approach. In the first, robustness is considered as the ability to preserve some level of solution quality, such as preservation of makespan in Leon et al. (1994) and Jensen (2003) or preservation of tardiness and total flow-time as in Jensen (2001b), etc. In the later, also known as rolling time approach, robustness is achieved by producing partial schedules and the final decisions are delayed until the execution time is reached or

nearly reached as in Kutanoglu and Wu (1998); Kutanoglu and Wu (2004); and Policella et al. (2004); etc. Hence, two definitions for a robust schedule can be distinguished:

1. A schedule is considered to be robust if it has low cost relative to other schedule when facing disruption and when right-shifting is used as a rescheduling algorithm (Jensen, 2003).
2. A schedule is considered to be robust if it can absorb the external events (disruptions) without loss of consistency while keeping the pace of execution (Policella et al., 2004), i.e. without amplifying the effects of a change over all schedule components.

Despite of the apparent similarity between the two definitions, it can be concluded that the first definition is more suitable for offline scheduling such as predictive scheduling, whereas the second definition is falling in some category belonging to dynamic scheduling and more precisely to proactive/reactive (or predictive/reactive) scheduling and knowledge-based scheduling. Furthermore, the second definition has some resemblance with the definitions of stable and/or flexible schedule found in literature. In Cavalieri and Terzi (2006); Policella et al. (2004); and Policella et al. (2005) flexibility was defined as "the ability to respond effectively to changing circumstances". A more thorough definition of flexibility was given by Jensen (2001a) as "a schedule expected to perform well relative to other schedules, when facing disruption and when some rescheduling method using search (other than right-shift) is used". Similarly, Gören (2002); Liu et al. (2007b); and Wu et al. (1993) defined stable schedule as a schedule that has a very small deviation either in time or sequence between the predicted schedule and the realized schedule. At this point one may conclude that the two types of schedules named stable schedule and flexible schedule used in literature are actually describing the

same schedule. This schedule (stable or flexible) can be related to the system (or schedule) nervousness measure, i.e. if the performance measure of the schedule nervousness is high then the stability of the schedule is low (representing an unstable manufacturing system) and vice versa. Furthermore, a schedule is called robust or stable depending on how it was designed to adapt to changes and unforeseen future events.

Furthermore, it can be observed that unlike single machine environment, two machines environment, and job-shop environment, vast majority of the literature published in the area of stochastic scheduling gives less attention to the FJSP. The literature focuses either on deterministic FJSP, stochastic classical job-shop scheduling problem (JSP) or dynamic FJSP. To the best of the author's knowledge, the literature on predictive scheduling for the FJSP under random machine breakdowns is almost void. Therefore, the goal of this work is to improve robustness and stability of predictive schedule for the FJSP subjected to random machine breakdowns. This Chapter introduces a new methodology that integrates the non-idle time insertion approach, a modification of the idle time insertion method of Liu et al. (2007b), and the hGA proposed in Chapter 2. The proposed methodology is based on a bi-objective hybrid genetic algorithm. Moreover, the current work relates the robustness of a schedule to its degree of makespan degradation under disruptions and considers it to be stable when its sum of the absolute deviations of operation completion times from the realized schedule is small. The used measures are covered in Section 3.4.

3.3 Problem Definition

FJSP is a generalization of the classical job-shop scheduling problem (JSP). It forms when alternative production routing of operations is allowed in the classical job-shop. Therefore, FJSP is strongly NP-hard due to: (a) assignment decisions of operations to a subset of machines and (b) sequencing decisions of operations on each machine (Tay and Wibowo, 2004). Hence, allowing stochastic data (like random machine breakdowns in this works) further complicates the problem. A hypothetical or deterministic FJSP is usually formulated as follows:

- There are n independent jobs that are indexed by i.
- All jobs are ready to start at time zero.
- Each job i has O_i operations and the operations' sequence is given by O_{ij} for $j = 1, \ldots, O_i$.
- There are m machines indexed by k.
- Machines never breakdown and are always available.
- For each operation O_{ij}, there is a set of machines capable of performing it represented by $M_{kij}, M_{kij} \subseteq \{1,...,m\}$.
- The processing time of an operation O_{ij} on machine k is predefined and given by p_{ijk}.
- The setup time of any operation is sequence independent and included in its processing time.
- A started operation cannot be interrupted (non-preemption condition).

- Each machine can process at most one operation at any time (resource constraints).
- The precedence constraints of the operations in a job can be defined for any pair of operations.
- The objective is to find a schedule that has the lowest possible value of makespan.

In practical manufacturing environment, disruptions and unforeseen incidents occur. Therefore, a schedule that is built based on deterministic information of having perfect knowledge of all problem's parameters, is impractical and may lead to poor performance. In this study, it is assumed that the uncertainty of the machine breakdown disruption in terms of occurrence and duration can be quantified using some distributions determined from historical data. The prediction of machine breakdown time and duration is covered in more details in Section 3.4.

3.4 Scheduling with Machine Breakdown Disruptions

3.4.1 Robustness and stability for FJSPs

Dooley and Mahmoodi (1992) stated that the robustness of a schedule refers to its ability to perform well under different operational environments including dynamic and uncertain conditions. Thus, it is important to develop a scheduling heuristic (or method) that enhances the performance in the face of variations during the process implementations.

This study considers using a bi-objective optimization measure of performance where weighted sum is used to form a scalar objective function.

Three measures M_h of stability are suggested to be examined:

1) The average difference between the completion times of the predicted operations and the realized ones:

$$M_1 = min \frac{\sum_{i=1}^{n} \sum_{j=1}^{q_i} |CO_{ijP} - CO_{ijR}|}{\sum_{i=1}^{n} O_i} \quad (3.3)$$

Where n: number of jobs, q_i: number of operations of job i, CO_{ijP}: the predicted completion time of operation j of job i, CO_{ijR}: the realised completion time of operation j of job i, and O_i: the total number of operations of job i.

2) The difference between the completion times of the predicted operations and the realized ones:

$$M_2 = min \sum_{i=1}^{n} \sum_{j=1}^{q_i} \left| CO_{ijP} - CO_{ijR} \right| \quad (3.4)$$

3) The average difference between the completion times of the affected predicted operations and the affected realized ones:

$$M_3 = min \frac{\sum_{i=1}^{n} \sum_{j=1}^{q_i} |CO_{ijP} - CO_{ijR}|}{\sum_{i=1}^{n} \sum_{j=1}^{q_i} AO_i} \quad (3.5)$$

Where, AO_i: the total number of affected operations of jobs i.

These measures are linearly combined (using weighted sum) with the primary objective function (makespan) to form a bi-objective function to guide the genetic algorithm search procedure. Since this analysis may result in two schedules, the predictive schedule and the realized schedule, the performance of using either makespan is investigated by independently considering both of them in the bi-objective. This results in the following bi-objective expression:

$$\boldsymbol{Z_{type,h}} = \min \gamma MS_{type} + (1\text{-}\gamma)M_h \quad (3.6)$$

Where,

$$type = \begin{cases} P \text{ for predicted schedule makespan; or} \\ R \text{ for realized schedule makespan} \end{cases}$$

$$h = \begin{cases} 1 \text{ for stability measure } M_1; \\ 2 \text{ for stability measure } M_2; \text{ or} \\ 3 \text{ for stability measure } M_3 \end{cases}$$

and, γ is a parameter $\in$ [0,1].

This, in total, gives six possible combinations of the bi-objective $Z_{type,h}$. Each of these bi-objectives may lead to generate different solutions or schedules. Hence, number of benchmark problems are used to evaluate which $Z_{type,h}$ combination performs better in terms of solution quality. These experiments are addressed in Section 3.5.

3.4.2 Proposed approach

Schedules that are developed based on minimum makespan are not just very short, but also very dense and compact. Therefore, minimum makespan schedules are prone to a high level of disruption and instability. Moreover, strategies that are based on inserting idle times in the predictive schedule to serve as time buffering to absorb the effects of disruptions face two main difficulties. First, they have to decide how to find the appropriate locations of inserting that idle times. Second, they have to decide the amount of inserted idle times amount to be inserted in that location. Hence, inserting idle times to the schedule may turn the schedule to be inactive and reduce the resources utilization. For this reason, a non-idle time insertion method is proposed as explained below.

Historical records of a certain shop floor can provide an approximated distribution for the machine breakdowns. Such distribution can be used in generating the predictive schedule. The idea behind this is to simultaneously integrate the knowledge of the machine breakdown probability distribution along with the available flexible routing of machines to obtain a predictive schedule that assigns and sequence operations on machines in such a way that has less impact on the overall performance of the schedule in case of occurrence of machine breakdown, while still maintaining a high level of solution quality (minimum makespan in the current work). This is achieved in the decoding procedure of the genetic algorithm, where the effect of disruptions on the predicted activities is measured using the robustness and stability measure. As a result of this combination, the algorithm searches for a predictive schedule that can work around the expected breakdown. This increases the probability of finding a predictive schedule that is located on a broad beak.

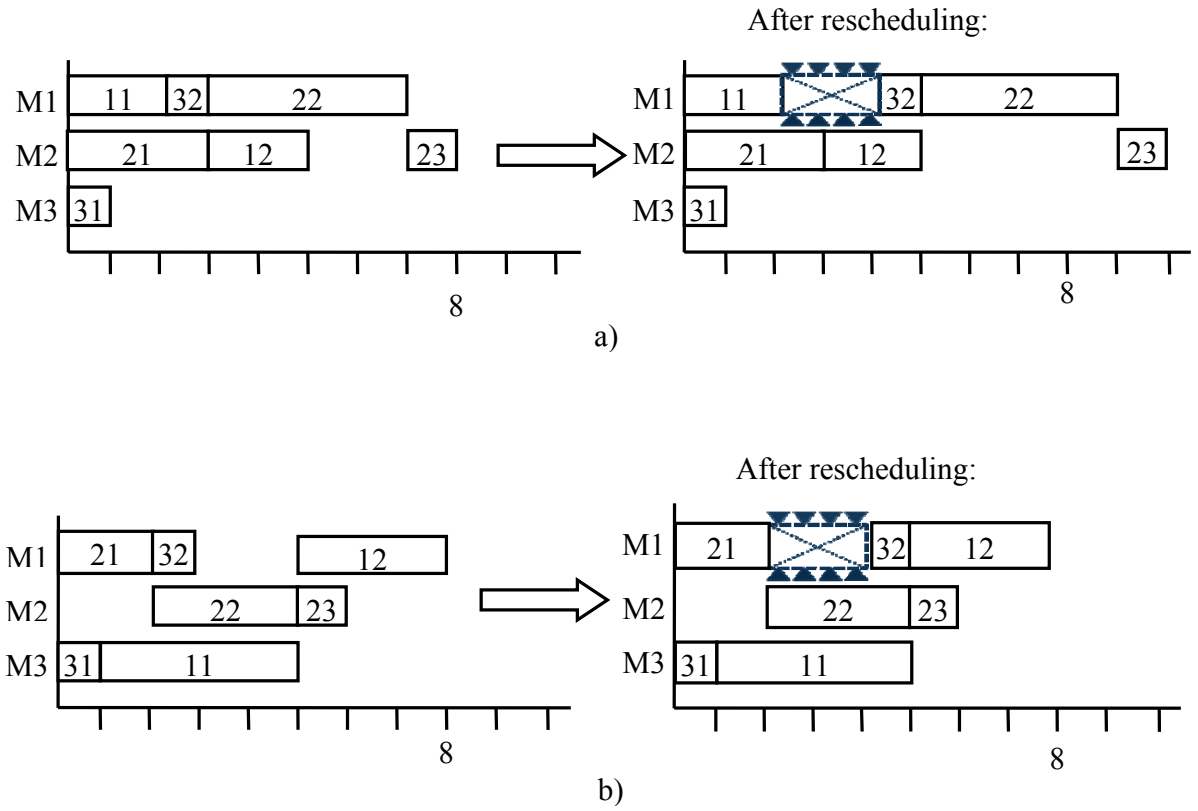


Figure 3.1: Example of schedule robustness and stability. For the breakdown specified by triangles, schedule (b) is more robust and stable than schedule (a).

To illustrate this, consider schedules shown in Figure 3.1. The left-hand-side of the figure shows two Gantt-charts of two possible schedules for a flexible job-shop with three jobs and three machines according to the processing times given in Table 2.1. When considering a deterministic FJSP with an objective to minimize makespan, there will be no preference in selecting either one to be released to the shop floor as both have the same makespan. However, when the probability of machine breakdown is integrated into the problem, then selecting a schedule that can absorb the impact of a possible future machine breakdown can be achieved. For example, if the historical data indicate that machine M1 has a high chance of failure, then the predicted schedule shown in Figure

3.1(b-left) is more appropriate to be selected. Consider the right-hand-side of Figure 3.1, both original predicted schedules are subjected to the same breakdown specified by the triangles and rescheduling is performed (the used rescheduling method is covered in Subsection 3.4.3). Since, after rescheduling, schedule (b) has smaller makespan and less number of disrupted operations than schedule (a), then schedule (b) is more robust and stable than schedule (a) when facing machine breakdown. The idea behind this is motivated by the work done in robust optimization of continuous functions which states that robust optima are located on broad beaks in the fitness landscape. For example, Branke (1998); and Tsutsui and Ghosh (1997) used genetic algorithm in which they perturb solutions according to a noise distribution before fitness evaluation and used this perturb fitness as the fitness of the evaluated phenotype. Similarly, the breakdown probability is used to perturb the predicted solutions (or schedules) in the decoding procedure of the proposed algorithm.

3.4.3 Generating machine breakdown

A shop floor may experience number of machine breakdowns during a scheduling horizon. Having extensive simulations to evaluate the effects of all possibilities of machine breakdowns at the initial schedule is a cumbersome task and may take tremendous time. Hence, this work proposes aggregating all breakdowns as one breakdown and accordingly evaluates the schedule stability. When a machine breakdown occurs, the operation being processed is resumed after the repair is finished. In this work, modified Affected Operations Rescheduling (mAOR) proposed by Subramaniam and Raheja (2003) is used to reschedule operations after the occurrence of the machine

breakdown. In mAOR operations' sequence remains the same as in the predictive schedule to avoid the setup costs incurred by sequence deviations and only the operations directly and indirectly affected are pushed in time to account for the disruption. Thus, the realized operations' completion time after the occurrence of machine breakdown can be calculated as follows:

$$CO_{ijR} = \begin{cases} SO_{ijR} + p_{ijk}, & \text{for unaffected operations, or} \\ SO_{ijR} + p_{ijk} + \tau_{k,duration} & \text{for affected operations} \end{cases} \quad (3.7)$$

Where, p_{ijk}: the processing time of operation j of job i on machine k, $\tau_{k,duration}$: the aggregated breakdown duration of machine k; and SO_{ijR}: the realized starting time of operation j of job i and is given by:

$$SO_{ijR} = max\{CO_{i(j-1)R}, t_k\} \quad (3.8)$$

Where, t_k: is the progressive time of machine k.

In practice, the repair duration of a machine may not precisely be known at the breakdown time. However, its estimate will probably be more precise as the time progresses. This aspect is ignored in the present work due to the difficulties of its formulation. Hypothetically, to simulate a machine breakdown, three parameters are required a) choosing which machine to breakdown, b) the breakdown time, and c) the breakdown duration. Where most of the previous works randomly select a machine to breakdown, the current work takes a more practical direction by relating the three previous parameters to the busy time of the machine. This work assumes that a machine put in service more than others has a higher probability to fail and its breakdown time

and duration are based on the its elapsed busy time or hours of operations. Thus, all these parameters are approximated based on the machine's busy time (*MBT*).

The probability of machine *k* to fail is approximated by the following empirical relation:

$$\rho_k = \frac{MBT_k}{MBT_{tot}} \tag{3.9}$$

Where, MBT_k: the busy time of machine *k*, and MBT_{tot}: the total busy time of all machines.

Furthermore, the current work assumes two levels of machine breakdown disruptions, low level and high level; and two intervals of machine breakdown time, early and late. This leads to four breakdown parameters combinations. The machine breakdown time and the aggregated breakdown duration are generated using the following uniform distributions, respectively:

$$\tau_k = [\alpha_1 MBT_k, \alpha_2 MBT_k] \tag{3.10}$$

$$\tau_{k,duration} = [\beta_1 MBT_k, \beta_2 MBT_k] \tag{3.11}$$

Where, τ_k: is the breakdown time of machine *k*, MBT_k: is the busy time of machine *k*, and the parameters α and β: are given by Table 3.1. Limiting the values of β between 0.1 and 0.15 keeps the disruption level to a relatively low level, whereas increasing these values to be between 0.35 and 0.4 raises the disruption level to be from 35% to 40% of the machine's busy time. Similarly, limiting the values of α between 0 and 0.5 ensure that the breakdown occurs during the first half of the scheduling horizon, while the values 0.5 and 1 ensure that the breakdown occurs at a later stage during the second half of the scheduling horizon.

Table 3.1: Breakdown combinations

Breakdown type	Disruption level & occurrence time	β_1	β_2	α_1	α_2
BD1	Low, early	0.1	0.15	0	0.5
BD2	Low, late	0.1	0.15	0.5	1
BD3	High, early	0.35	0.4	0	0.5
BD4	High, late	0.35	0.4	0.5	1

3.4.4 Framework of the two-stage hGA

As highlighted in Section 3.2, literature related to finding predictive schedules for the FJSP with stochastic data is almost void. To the best of the author's knowledge, there is not previously study that addresses obtaining predictive schedules for the FJSP subject to random machine breakdowns. Hence, comparing the currently proposed method against other previous methodologies is not attainable. However, to facilitate the efficiency assessment of the proposed method, we will adopt the slack-based robustness measure proposed by Leon et al. (1994) and the neighborhood robustness measure proposed by Jensen (2003) that are proposed for the classical JSP. These two measures are selected for three main reasons. First, similar to the proposed method, both are non-idle time insertion methods. Second, both were originally proposed using genetic algorithm. Third, classical JSP can be considered as a special case of FJSP without alternative routing of operations.

In the previous Chapter, Chapter 2, a hybrid Genetic Algorithm (hGA) is proposed to solve deterministic FJSP and the computational results show that the approach is very efficient in solving these problems. Therefore, eleven different hGAs are created to minimize eleven objective functions: MS_{min} (minimum makespan for deterministic schedules), $Z_{type,h}$ (minimum bi-objectives of Eq. (3.6) for robust and stable schedules), $Z_{r=1}(s)$ (Eq. (3.1) for minimizing the slack-based robustness, Leon et al. (1994)), $Z_{MS_{min}}(s)$ (Eq. (3.2) for minimizing the neighborhood robustness measure, Jensen (2003)), and a robustness measure $Z_{comb}(s)$ (Eq. (3.12)) combining the latter two.

$$Z_{comb}(s) = \frac{1}{|N_1(s)|}\sum_{s` \in N_1(s)} Z_{MS_{min}}(s`) \qquad (3.12)$$

All algorithms used to solve the previous objective functions are based on the hGA proposed in Chapter 2. This is done by replacing the fitness function of the hGA by the so-called robust evaluation functions. Replacing the fitness function of the hGA by different objective functions will lead to obtaining different solutions. Hence, number of experiments is conducted to evaluate and compare their performance in Section 3.5. At this stage it may worth highlighting that the advantage of using GA in contrast of other techniques is the fact that GA utilizes a population of solutions in its search. Taking this fact along with the fact that finding robust solutions requires the exploration of a diverse set of solutions, GA may have a higher chance of locating solutions that are located on broad beaks (see Subsection 3.4.2).

The originally proposed hGA in Chapter 2 to minimize the makespan turned to be easy to modify so that it minimizes the other robustness and stability measures. The hGA is modified to work in two stages except for the hGA that minimizes MS_{min} which consists

of only one stage. Both stages of the proposed two-stage hGA use the same chromosome representation, chromosome decoding, and genetic evolution. However, in the first stage, the population is evolved based on the objective of minimizing the makespan assuming deterministic problem parameters where no disruptions are to occur. This stage is a common stage shared by all eleven algorithms. After a given number of generations are reached, the algorithm switches to the second stage. The second stage starts by taking the final population generated in the first stage as its initial population. Then the algorithm continues to evolve based on the bi-objective robustness and stability measure with the random machine breakdown, or by the other robustness measures given in Eq. (3.1), Eq. (3.2) or Eq. (3.12). Figure 3.2 shows the general flow chart of the proposed two-stage hGA.

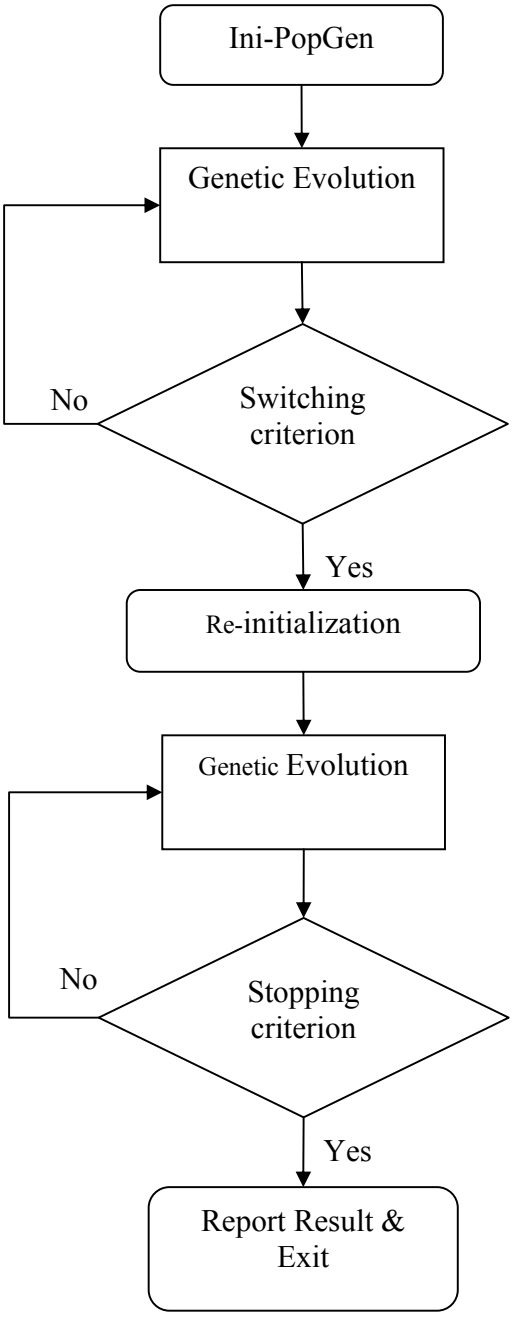


Figure 3.2: Flow chart of the two-stage hGA

3.4.4.1 Shared two-stage elements

The used chromosome representation is a permutation-based representation; where each operation is represented by triples (*k*,*i*,*j*) such that *k* is machine assigned to the operation, *i* is current job number, and *j* is the progressive number of that operation within job *i*. A schedule for FJSP with three jobs and three machines can be represented by (221-131-111-212-322-223-332). This representation has the advantage of solving the scheduling problem using a concurrent approach as well as modeling alternative routing of the problem by changing the index *k*.

The initial population for the first stage is produced using two techniques. The first technique is to generate half of the population by randomly selecting a progressive operation number of an unscheduled operation and then randomly assigning it to an appropriate machine. The second half of the population is generated using a schedule construction heuristic called *Ini-PopGen* that randomly assigns priority to jobs. Based on this priority an operation is scheduled on the machine that can finish processing sooner from the set of appropriate machines. This procedure considers the processing time and the work load on the machine while assigning operations.

Chromosomes decoding follows an active decoding procedure, wherein no operation can be started earlier without delaying at least one other operation or violating the technological constraints. The active decoding starts by scheduling left operation before right ones. After the active decoding, the schedule is improved by a local search procedure that results in a local optimal schedule (Lamarckian learning). However, this local search procedure is only applied every d^{th} generation and number of moves is

limited to a maximum *loc_iter* moves without improvements. At the end of this procedure, the schedule is transferred back to a chromosome.

The selection of individuals for mating is divided into two phases. The first phase uses roulette wheel technique to form donors' mating-pool based on a selection probability given by Eq. (2.1).

The second phase starts if the individual in the donors' mating-pool passes a crossover probability P_c. Here, an *n*-Size tournament method is used to select *n* chromosomes from the population to form the receivers' mating-sub-pool. Then, the best individual (one with lowest makespan or lowest robustness measure) in the sub-pool is chosen for reproduction.

The genetic operators are specially designed for the proposed chromosome representation to avoid creation of invisible chromosomes. The designed crossover operator is based on the *Precedence Preserving Order-based Crossover* (POX) (Kacem et al., 2002a) and was modified to not treat the parents symmetrically. Two mutation operators are used here. The first operator is *Machine Based Mutation* (MBM), where a random number of operations (denoted as *nrand*) are selected and reassigned to another machine. After subjecting the chromosome to POX, modified *Position Based Mutation* (PBM) (Mattfeld, 1996) is applied. PBM was originally designed for JSP using single triple permutation-based chromosomes representation. Thus, the PBM is modified so that no infeasible chromosomes are produced. This mutation starts by randomly selecting an operation within the chromosome and then reinserting it at another position in a way that does not violate the technological constraints. Readers are referred to Chapter 2 for a detailed description of this hGA.

3.5 Benchmark Problems

Since there are no standard benchmarks for stochastic flexible job-shop scheduling problem, a number of deterministic FJSP benchmarks found in literature are used for the experiments. Selected benchmarks are chosen to represent realistic flexible job-shop systems. Most literature used systems consisting of four to ten machines (Vinod and Sridharan, 2009). Hence, the following benchmarks are selected to be similar to the contemporary research of FJSP.

1- Three T-FJSP benchmarks; Ex1 consisting of 10 x 7, Ex2 consisting of 15 x 10, and Ex3 consisting of 10 x 10, where the first two are taken from Kacem et al. (2002b), and the third is taken from Mesghouni et al. (1997).

2- Twelve P-FJSP benchmarks; Ex4 consisting of 5 x 3 taken from Lee and DiCesare (1994), Ex5 consisting of 8 x 8 taken from Kacem et al. (2002a), and examples MK01 – MK10 with different sizes varying between 10 x 6 and 20 x 15 proposed by Brandimarte (1993).

3.5.1 Computational results

Although great care is paid in selecting representative benchmarks that covers wide range of reported FJSP systems in literature, it is unlikely to generalize the conclusions. This is because the results depend on the assigned values to the parameters of the simulated system. However, since the used benchmarks cover wide range of systems and

conditions, it may be sensible to anticipate that results and conclusions are relevant in a broader sense.

The experimental work in this section is divided into three parts. In Subsection 3.5.1.1 experiments are conducted to analyze the performance of the predictive schedule generated by the method proposed in this Chapter while using different bi-objective robustness and stability measures given in Eq. (3.6). In Subsection 3.5.1.2 the performance of the predictive schedule generated using the proposed method while using the best found bi-objective robustness and stability measure with the random machine breakdown from Subsection 3.5.1.1 is compared with the predictive schedules obtained using other robustness measures given in Eq. (3.1) Eq. (3.2) and Eq. (3.12) against schedule generated by optimizing the MS_{min}. The last Subsection 3.5.1.3 discusses why predictive schedules obtained by the proposed algorithm outperform the predictive schedules obtained by other methods.

The two-stage hGA is coded and executed using C++ on an Intel® Core™ 2 Quad CPU @ 2.4 GHz with 3.24 GB RAM. For comparability and ease of implementation reasons, all hGA are closely related and the parameters are experimentally tuned according to the performance of the deterministic hGA (minimizing MS_{min}). The parameter values that are chosen for the two-stage hGA algorithm are as follows: population size 500, crossover probability 0.7, mutation probability 0.3, number of generations 500, number of parents in the receivers' mating sub-pool 4, number of generations to perform local search d = 10, maximum number of moves without improvement in the local search *loc_iter* = *min* [*tot_noper*, 150], the worst chromosome is replace every k = 3 generations, the algorithm switches to stage-two after 100 generations, and for Eq. (3.4) γ = 0.6 (Liu et al., 2007b).

3.5.1.1 Analysis of robustness and stability measures

To analyze the performance of the different robustness and stability measures with the random machine breakdowns proposed in this Chapter in terms of solution quality, four different test cases are used. These are Ex3, Ex1, MK02, and MK05; and will be referred to by test case 1, test case 2, test case 3, and test case 4, respectively. The analysis of variance (ANOVA) is performed using the commercial statistical software Minitab 14. Each test case is subjected to the four different levels of the breakdown type, and each time is solved using one of the six robustness and stability measures (Eq. (12)), resulting in 4 x 6 = 24 settings per problem. In order to draw more accurate responses, five replications for each of the 24 different settings of each test case is used. Then each generated predictive schedule is subjected to 400 random machine breakdowns which results in 24 x 5 x 400 = 48000 test problems per test case and 48000 x 4 = 192000 test problems in total.

Since this comparative study is done to evaluate the performance of the predictive schedule obtained using robust method against the obtained schedule using the deterministic method, both obtained schedules are subjected to the same machine breakdown disruption and their performance is compared in terms of the predictive makespan, realized makespan, and stability measure. Usually a scheduler requires a schedule that has a minimum makespan when released to the shop floor and that this same schedule, if a disruption occurred, has the minimum realized makespan after rescheduling is implemented with the highest stability (i.e. lowest system nervousness). Hence, the pervious performance criteria of each obtained schedule using the two

methods are combined in a relative solution quality measure. This relative measure enables us to examine the results' quality with respect to the average values of replications at each combination, and analyze the effect of the considered factors on the performance measures and their interaction effects using analysis of variance (ANOVA). The relative solution quality of the obtained robust schedule compared to the deterministic schedule for each test problem is given by:

$$\min RQULT = \left(\frac{RMS_P - DMS_P}{DMS_P}\right) + \left(\frac{RMS_R - DMS_R}{DMS_R}\right) + \left(\frac{RSTB - DSTB}{DSTB}\right) \quad (3.13)$$

Where, *RMS* and *RSTB*: are the makespan and the stability of the obtained schedule using robust method, respectively; *DMS* and *DSTB*: are the makespan and the stability of the obtained schedule using deterministic method, respectively; and the subscripts *P* and *R*: refer to predictive (or the original released schedule to the shop floor) and realized (or the actual schedule after breakdown), respectively.

Table 3.2: ANOVA results concerning *RQULT*

Factor	*F*-ratio	*P*-value
A: Measure	80.93	0.000
B: Test case	23.16	0.000
C: BD Type	34.70	0.000
AB	10.95	0.000
AC	3.33	0.000
BC	2.29	0.017

Table 3.2 shows *F*-ratio and *P*-value of the ANOVA results from the experiments. This test results show the effect of the used robustness and stability measure, the considered test case, breakdown type and the interaction between these factors on the relative quality of the predictive schedule. In this study, effects are considered significant if the *P*-value is less than 0.05. The *F*-ratio indicates that the used robustness and stability measure has the most statistical significant effect on the relative quality of the obtained results followed by the breakdown type (BD type). However, the interaction between them does not have the same significance. Figure 3.3 shows the *RQULT* is generally constant when applying different robustness and stability measures to the first two test cases except for measures $Z_{R,1}$ and $Z_{R,3}$. However, $Z_{R,2}$ and $Z_{P,2}$ show their superiority over other measures in terms of providing lower *RQULT* per test case and both provide significantly better results for test cases 3 and 4. This result can be related to the nature and the problem size

of the test cases. Test cases 1 and 2 are both T-FJSP and have relatively smaller total number of operations (30 and 29, respectively), whereas test cases 3 and 4 are P-FJSP and have higher total number of operations (58 and 106, respectively). The domino effect in small shop floors has less impact than on the large shop floors. Furthermore, it can be observed from Figures 3.3 and 3.4 that the different measures have a similar behavior to the different BD types and that all measures perform better with early expected breakdowns rather than with late expected breakdowns. Moreover, it can be concluded that the third robustness and stability measure, $Z_{R,2}$, offers the best *RQULT* when comparing it against different test cases and BD types and hence it is used for the computational results in the next subsection.

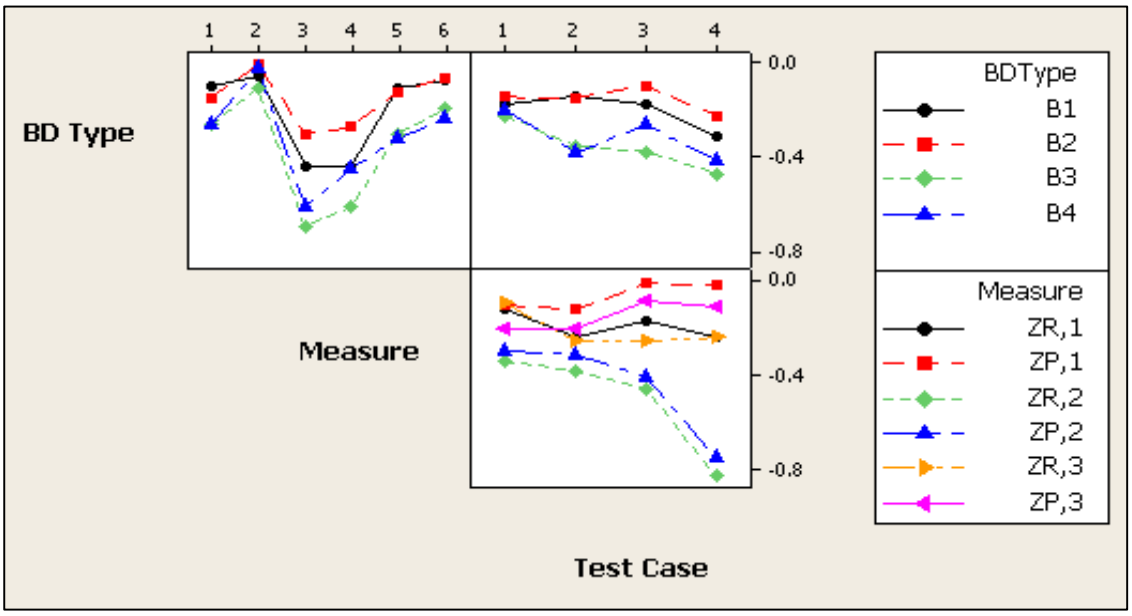


Figure 3.3: Significant interaction effects of factors on *RQULT*.

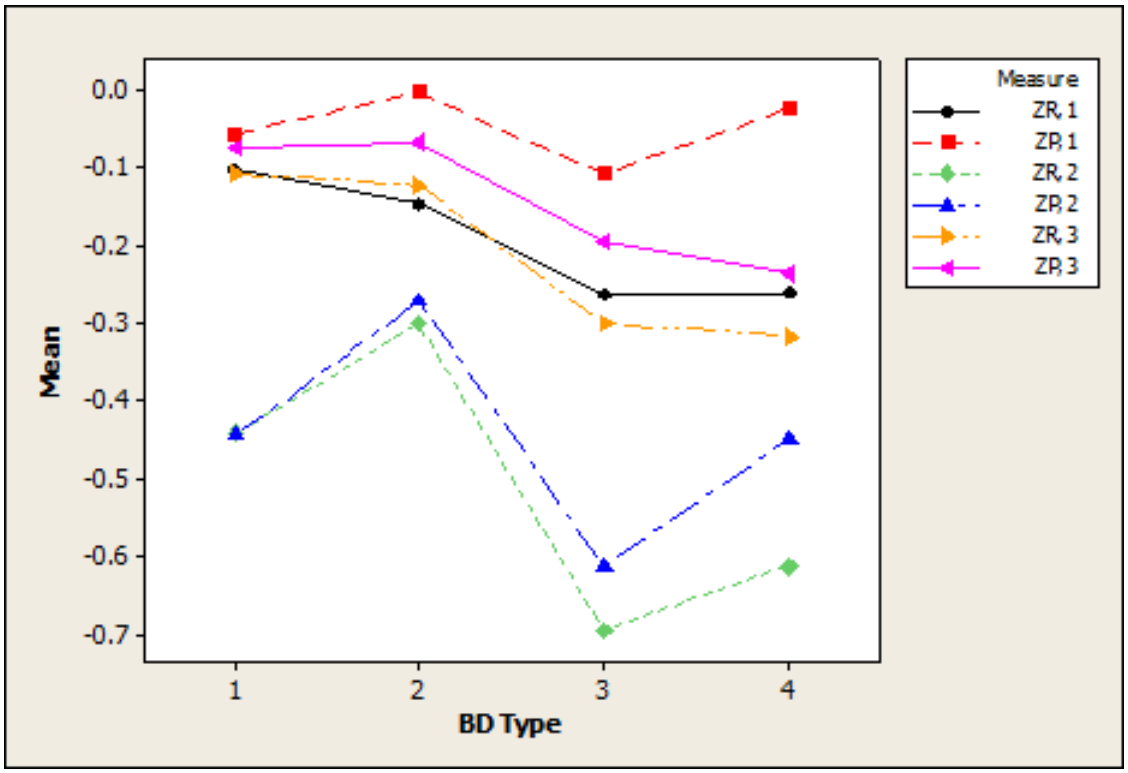


Figure 3.4: Significant interaction effects between different measures and BD type on *RQULT*.

3.5.1.2 Predictive schedules performance

The previous subsection concluded that optimizing the bi-objective $Z_{R,2}$ offers the best *RQULT*. Hence, this subsection compares the performance of the predictive schedules obtained using the bi-objective $Z_{R,2}$ and the predictive schedules obtained using the robustness measures given in Eq. (3.1) Eq. (3.2) and Eq. (3.12) against the schedules obtained using deterministic scheduling approach that minimizes MS_{min}. Similar to Subsection 3.5.1.1, each benchmark is subjected to the four different BD types with 5

replications per instance per BD type. After that, each generated predictive schedule is subjected to a 400 random machine breakdowns. Table 3.3 and Table 3.4 show the compared results in terms of the average improvement percentage for BD1 and BD2; and BD 3 and BD 4, respectively.

The average realized makespan improvement percentage (AMS_RI) is obtained by:

$$AMS_RI = \frac{\sum_{q=1}^{5}\sum_{p=1}^{400} RMS_R(q)_p - \sum_{q=1}^{5}\sum_{p=1}^{400} DMS_R(q)_p}{\sum_{q=1}^{5}\sum_{p=1}^{400} DMS_R(q)_p} \times 100 \quad (3.14)$$

The average stability improvement percentage (*ASTBI*) is obtained by:

$$ASTBI = \frac{\sum_{q=1}^{5}\sum_{p=1}^{400} RSTB(q)_p - \sum_{q=1}^{5}\sum_{p=1}^{400} DSTB(q)_p}{\sum_{q=1}^{5}\sum_{p=1}^{400} DSTB(q)_p} \times 100 \quad (3.15)$$

Where, $\boldsymbol{q}$ and $\boldsymbol{p}$ are the replication predictive schedule, and the breakdown number, respectively.

Tables 3.3 and Table 3.4 consist of 11 columns. The first and second columns represent the instance name and size. The third column refers to the average improvement percentage of each instance. The remaining columns are labeled according to the performance robustness measure compared against deterministic scheduling approach when subjected to a specific breakdown type. For each column, the best performance; lowest average improvement percentage; is printed in bold. Furthermore, negative values in the tables indicate that there are improvements whereas positive values indicate that there are degradations when comparing different methods.

When comparing the proposed approach to the deterministic approach computational results in Table 3.3 and Table 3.4 show that it has improved the stability in all instances, whereas the efficiency has improved in 76.67%, slightly degraded in 18.33%, and remained the same in 5% of the instances. A closer look at Table 3.3 and Table 3.4 reveals that the degradation is very minimal as in average the degradation was less than 1.5%. Furthermore, the proposed approach is performing better in stability in 95% and slightly worse in 5% of the instances; and the efficiency is better in 75%, similar in 6.67% and worse in 18.33% of the instances when compared to the other robustness approaches. This finding confirms that using the proposed bi-objective methodology is very useful in improving efficiency and stability of predictive schedules. Also, it shows that stability is improved far more the efficiency even for instance where the efficiency is degraded. This is a very tempting outcome in real FJSP. In practice schedule stability is often very important as changing starting of activities may complicate prior commitments with suppliers or subcontractors and increases the system nervousness.

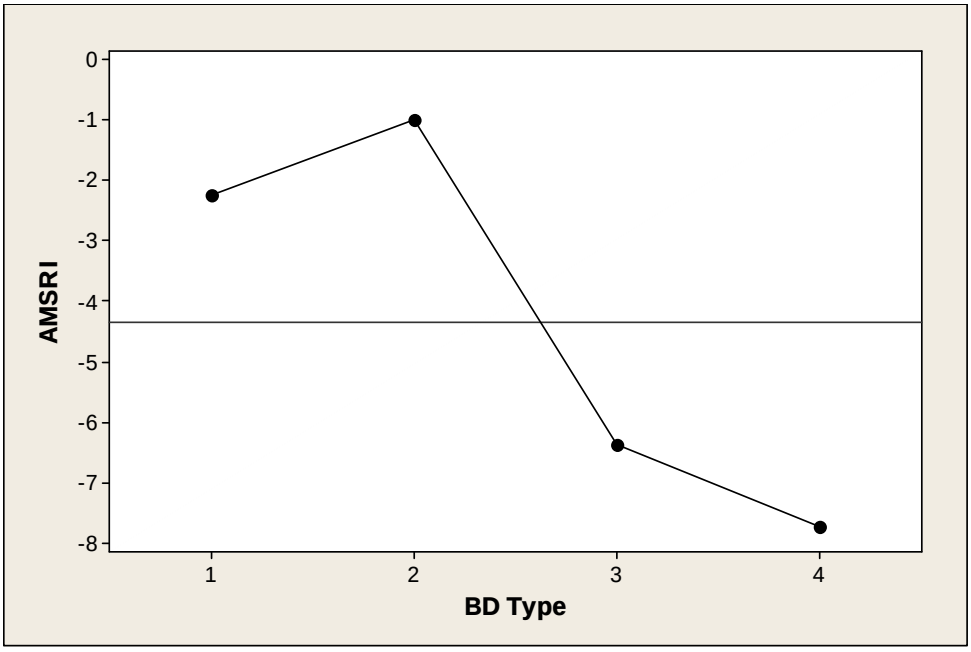
To further investigate the impact of the breakdown type on the performance of the proposed approach, a one-way ANOVA test is performed. Table 3.5 shows the *F*-ratio and the *P*-value of the one-way ANOVA results. Since the AMS_RI and *ASTBI* distribution functions are not known, the results of this statistical test has to be taken with care as ANOVA tests requires the observations to be normally distributed. Nevertheless, ANOVA test is robust with respect to violations of this condition (Rutherford, 2001). Results from Table 3.4 indicate that the BD type has a significant effect on both measures. This is supported by Figure 3.5 which shows the main effects of BD type on the performance of efficiency and stability of the proposed approach.

Table 3.3: Computational results of instances subjected to breakdown type BD1 and BD2

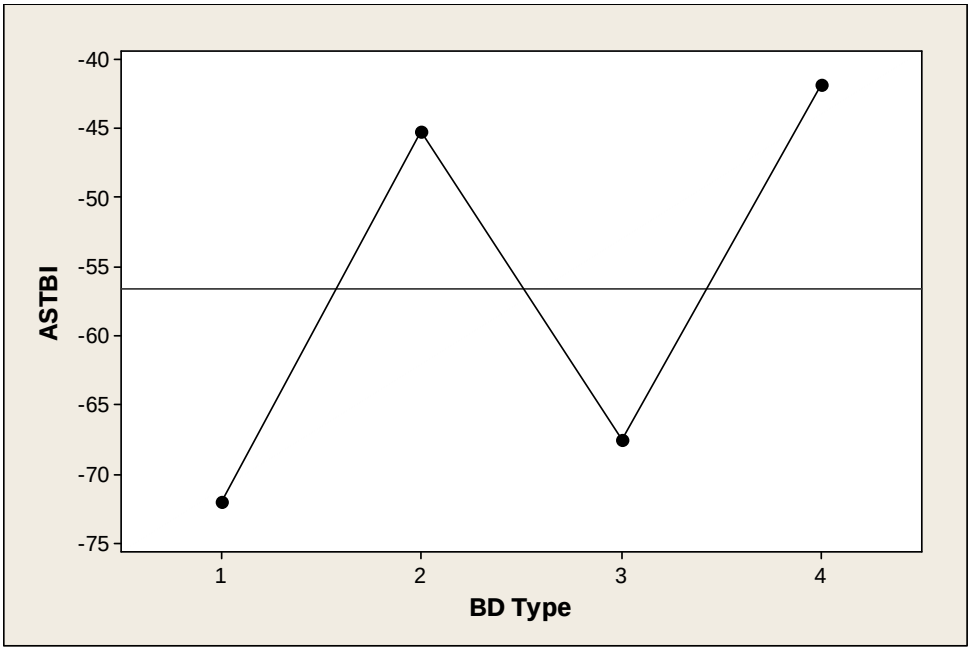
Instance	Size		BD1				BD2			
			$Z_{R,2}$	$Z_{r=1}$	$Z_{MS_{min}}$	Z_{comb}	$Z_{R,2}$	$Z_{r=1}$	$Z_{MS_{min}}$	Z_{comb}
Ex1	**10 x 7**	***AMS$_R$I***	**-1.64**	**-1.64**	4.92	1.64	**-1.75**	3.51	0	0
		ASTBI	**-53.66**	-26.83	26.83	24.39	**-61.11**	-22.22	-38.89	-38.89
Ex2	**15 x 10**	***AMS$_R$I***	**-1.64**	4.82	6.56	1.64	**3.28**	4.92	6.56	**3.28**
		ASTBI	**-96.97**	-3.56	9.09	0	**-58.26**	11.3	32.17	73.91
Ex3	**10 x 10**	***AMS$_R$I***	**-5.41**	0	-2.7	-2.7	**-5.41**	0	-2.7	0
		ASTBI	**-100**	50	-40	50	**-90.91**	-63.64	-72.73	11.14
Ex4	**5 x 3**	***AMS$_R$I***	**-5.03**	1.51	-0.5	-2.51	2.65	4.76	**2.12**	2.65
		ASTBI	**-86.91**	-9.06	-15.77	-19.46	**-24.14**	8.62	14.66	12.07
Ex5	**8 x 8**	***AMS$_R$I***	**-1.35**	0	1.35	1.35	**-4.37**	-2.56	-2.56	1.28
		ASTBI	**-73.33**	-20	-20	-36.67	**-37.12**	38.46	-30.77	61.54
MK01	**10 x 6**	***AMS$_R$I***	-4.7	**-4.98**	-3.17	-3.42	**0.91**	2.87	4.78	2.87
		ASTBI	**-62.64**	-40.64	-44.65	-34.06	**-81.74**	1.68	11.77	1.68
MK02	**10 x 6**	***AMS$_R$I***	**0**	2	4.67	4	0	**-1.33**	1.32	-0.66
		ASTBI	**-55.61**	18.56	15.81	-1.92	-5.89	51.9	13.92	**-8.86**
MK03	**15 x 8**	***AMS$_R$I***	-4.45	**-5.45**	-4.45	**-5.45**	-7.27	**-8.52**	-7.18	**-8.52**
		ASTBI	**-85.85**	-20.12	-21.89	-26.92	**-66.79**	-35.95	-50.55	-62.63
MK04	**15 x 8**	***AMS$_R$I***	**0.29**	4.39	3.51	2.92	**0.3**	3.25	4.14	5.92
		ASTBI	**-49.09**	31.63	26.18	-20.36	**-32.63**	-3.76	-11.68	22.96
MK05	**15 x 4**	***AMS$_R$I***	**-1.33**	0.41	1.64	0.72	**-1.05**	-0.51	1.11	0.2
		ASTBI	**-61.02**	5.81	-9.27	-15.21	**-9.01**	1.71	-1.81	-8.21
MK06	**10 x 15**	***AMS$_R$I***	**-1.31**	0.28	1.01	1.96	0.25	**-0.85**	2.54	3.11
		ASTBI	**-58.78**	-23.6	-29.68	-30.36	-42.61	**-51.66**	21.54	6.93
MK07	**20 x 5**	***AMS$_R$I***	**-1.87**	0.79	2.53	3.54	**-2.5**	-0.13	3.87	1.72
		ASTBI	**-83.42**	-8.32	-7.18	-26.06	**-40.26**	56.12	41.82	-17.96
MK08	**20 x10**	***AMS$_R$I***	**-4.05**	1.35	1.28	-0.04	1.72	2.32	3.8	**0.86**
		ASTBI	**-86.57**	7.54	15.81	-1.7	**-31.54**	79.02	161.22	70.13
MK09	**20 x 10**	***AMS$_R$I***	**-4.23**	-1.61	-2.39	-2.39	**-2.51**	-0.87	-0.66	-0.66
		ASTBI	**-66.60**	-10.96	-10.75	-10.75	**-46.96**	-17.24	-16.15	-16.15
MK10	**20 x 15**	***AMS$_R$I***	2.87	**-0.48**	3.36	2.98	0.74	**0.49**	3.3	4.04
		ASTBI	**-58.23**	-16	-3.96	-2.13	**-48.32**	6.10	-7.82	10.21
Average		***AMS$_R$I***	**-2.26**	0.09	1.17	0.28	**-1**	0.49	1.36	1.07
		ASTBI	**-71.91**	-4.37	-7.3	-10.08	**-45.15**	4.03	4.45	7.86

Table 3.4: Computational results of instances subjected to breakdown type BD3 and BD4

			BD3				BD4			
Instance	**Size**		$Z_{R,2}$	$Z_{r=1}$	$Z_{MS_{min}}$	Z_{comb}	$Z_{R,2}$	$Z_{r=1}$	$Z_{MS_{min}}$	Z_{comb}
Ex1	**10 x 7**	***AMS$_R$I***	**-11.11**	-8.33	-8.33	-9.72	**-16.67**	7.25	-8.7	-11.59
		ASTBI	**-94**	-6	-15	-13.28	**-18.75**	70	130	45
Ex2	**15 x 10**	***AMS$_R$I***	**-2.86**	8.57	1.79	11.43	-1.76	**-2.7**	5.41	2.7
		ASTBI	**-82.73**	23.21	4.96	56.96	**-37.34**	-23.81	-19.05	-23.81
Ex3	**10 x 10**	***AMS$_R$I***	**-6.98**	-2.33	-2.33	2.07	**-9.52**	-7.14	0	-4.76
		ASTBI	**-46.15**	0	-5.13	40.13	**-60**	-6.67	26.67	-13.33
Ex4	**5 x 3**	***AMS$_R$I***	**-0.83**	3.75	0	1.67	**-11.67**	-7.5	-1.25	1.25
		ASTBI	**-69.03**	2.53	-4.41	16.44	**-34.69**	-13.7	16.62	-11.95
Ex5	**8 x 8**	***AMS$_R$I***	**-13.33**	-5.56	-2.22	-3.67	**2.47**	6.17	6.17	8.64
		ASTBI	**-83.96**	0.53	-5.88	15.51	**-86.96**	139.13	143.48	121.74
MK01	**10 x 6**	***AMS$_R$I***	**-9.39**	-6.56	-6.53	-5.31	**-8.8**	-6.4	-3.2	-7.2
		ASTBI	**-69.71**	-8.42	-35.62	-9.14	**-44.67**	-18.27	-18.78	-23.35
MK02	**10 x 6**	***AMS$_R$I***	**0**	3.28	**0**	2.73	**-7.73**	-3.31	4.97	-4.42
		ASTBI	**-57.64**	20.56	-20.94	-1.92	**-11.3**	75.67	80.27	5.98
MK03	**15 x 8**	***AMS$_R$I***	**-5.24**	-5.14	-4.24	**-5.24**	-13.18	-11.09	-6.11	**-17.19**
		ASTBI	**-70.51**	-9.24	-15.13	-17.66	**-49.82**	0.93	14.58	-30.64
MK04	**15 x 8**	***AMS$_R$I***	**-8.74**	4.92	3.55	1.37	**-13.83**	-1.61	-0.25	-0.25
		ASTBI	**-96.38**	28.05	22.28	-25.49	**-92.11**	-4.1	-13.82	-13.82
MK05	**15 x 4**	***AMS$_R$I***	**-7.32**	0.61	1.5	0.62	**-13.87**	-0.34	1.52	0.59
		ASTBI	**-70.19**	-28.63	-28.2	-29.98	**-60.6**	20.14	26.36	3.96
MK06	**10 x 15**	***AMS$_R$I***	**-7.28**	6.31	4.09	1.86	**-4.23**	-0.32	1.38	1.2
		ASTBI	**-66.23**	23.16	1.23	-12.04	**-36.99**	-21.77	-23.63	-28.99
MK07	**20 x 5**	***AMS$_R$I***	**0.31**	2.15	1.02	3.15	**-6.28**	-2.94	2.53	0.14
		ASTBI	**-31.18**	-14.22	-30.34	-21.35	**-32.28**	4.8	-19.41	3.52
MK08	**20 x10**	***AMS$_R$I***	**-8.4**	0.47	-2.51	-1.44	**-1.93**	4.4	5.94	7.07
		ASTBI	**-61.78**	6.65	-13.47	-0.15	**-33.38**	23.87	78	95.04
MK09	**20 x 10**	***AMS$_R$I***	**-5.4**	-2.05	-2.77	-2.8	**-5.58**	-1.29	-0.46	-0.46
		ASTBI	**-36.99**	-4.3	-11.4	1.16	**-26.34**	-18.75	-25.78	-25.78
MK10	**20 x 15**	***AMS$_R$I***	**-9**	-0.38	1.73	3.72	**-3.11**	-0.46	2.35	2.74
		ASTBI	**-75.88**	-9.11	-7.56	-4.13	-2.04	**-6.61**	53.56	56.33
Average		***AMS$_R$I***	**-6.37**	-0.02	-1.02	0.03	**-7.71**	-1.82	0.69	-1.44
		ASTBI	**-67.49**	1.65	-10.97	-0.33	**-41.82**	14.72	29.94	10.66



a)



b)

Figure 3.5: Main effects of BD type on: a) AMS_RI, b) *ASTBI*.

Furthermore, most of the degradation in efficiency happened in the predicted schedules obtained for the instances when BD2 was expected, 63.64% of degraded cases to be exact. This may be explained by recalling that the inspiration of the proposed algorithm is to find a predictive schedule that is located on a broad beak where the breakdown is expected (see Subsection 3.4.2). Since BD2 assumes that the breakdown disruption is low and its occurrence is expected at a late stage of the schedule implementation, then the chance of obtaining a schedule with a relatively narrow beak is high. Hence, more emphasis was put to make the predictive schedule stable at the cost of sacrificing some of the efficiency (due to the bi-objective function that is used to guide the hGA convergence). Another possible cause of this can be attributed to the intuition since the expected late occurrence of breakdown disruption means that fewer operations are affected by the breakdown. Thus, the chance of critical operations to be among the affected operations by the breakdown is also less. This reduces the possibility of delaying the schedule (i.e. increasing its makespan) and hence minimizing the stability term in the bi-objective function is emphasized more. These conclusions are supported when we consider the results of the predictive schedules obtained when BD1 and BD4 are expected. Similar to BD2, BD1 assumes a low disruption level, but at an early stage. This means that more operations are affected by rescheduling and as result solutions with narrow beaks will be rejected in order to minimize the degradation in efficiency measured by the makespan. Moreover, BD4 assumes a late disruption occurrence, but with a high level. The expected high level disruption influences the hGA to converge towards solutions that are less sensitive to such disruptions. This seems to forcing the

hGA to search for predictive schedules that are located at a broader beak around the expected disruptions to reduce its possible effects in efficiency and stability.

Table 3.5: One-way ANOVA results concerning AMS_RI and *ASTBI* of the proposed method

	AMS_RI		*ASTBI*	
Factor	*F*-ratio	*P*-value	*F*-ratio	*P*-value
BD Type	10.22	0.000	7.63	0.000

3.5.1.3 Overview of the proposed bi-objective approach

Results in the previous subsection suggest that the proposed bi-objective approach is able to find predictive schedules that can perform better (in terms of both, efficiency and stability) than predictive schedules obtained by using slack-based robustness measure ($Z_{r=1}$) and neighborhood robustness measure ($Z_{MS_{min}}$). The superiority of the proposed approach in increasing the predictive schedule's stability can be explained by recalling the fact that both later algorithms do not consider stability in their objective function whereas ours does. However, this argument does not explain the achieved efficiency. Nevertheless, the improved performance in terms of efficiency and stability can probably be justified by summon up the basic principle behind the different algorithms.

The driving motor of the slack-based robustness measure (Eq. (3.1)) is to generate a schedule in which operations have slack. Since slack of an operation is defined as the time by which an operation can be delayed without worsening the schedule performance

(makespan), this slack works as a buffering time that can absorb effects of disruptions. Furthermore, according to the *Slack Hypothesis* proposed by Jensen (2003), neighborhood robustness measure (Eq. (3.2)) generates a predictive schedule that has slack too. However, the slack created by neighborhood robustness measure is less than that created by the slack-based robustness measure. This is supported by results shown in Table 3.3 and Table 3.4. In general, these results suggest that the predictive schedules' performance of these two measures are first dependent on the details of the experiments, and second in most cases the performance of the slack-based robustness measure is better than the neighborhood robustness measure when especially the breakdown duration is high. This finding is in fact complying with the finding of Jensen (2003). The enhanced performance of the slack-based robustness measure in favor of neighborhood robustness measure is actually related to the slack amount imposed by each of the two measures. Unlike the slack-based robustness measure, the amount of slack formed by the neighborhood robustness measure is probably less. This is due to legitimacy that neighborhood-based robustness measure is intended to find a predictive schedule close to a set of schedules in the neighborhood $\mathcal{N}_1$ where all have a good performance, whereas the slack generated by this measure in the obtained predictive schedule is only a secondary of that search.

In Subsection 3.4.2, it was stated that the aim of this Chapter is to develop a non-idle time insertion approach so that a predictive schedule that can work around an expected machine breakdown is obtained. This was achieved by simultaneously integrating the knowledge of the machine breakdown probability distribution along with the available flexible routing of machines and accordingly assigning and sequencing operations of the

predictive schedule. The efficient performance of predictive schedules subjected to machine breakdown disruptions obtained using this approach suggests that these schedules have more slack than ordinary schedules obtained by minimizing MS_{min} as well as schedules obtained by minimizing $Z_{r=1}$and $Z_{MS_{min}}$. This slack is a result of small delays in schedule caused by considering operations swaps and alternative machines assignments done to accommodate the expected machine breakdown disruption in the decoding procedure of the genetic algorithm. In order to have smaller objective function value for the hGA (low bi-objective value $Z_{R,2}$), the predictive schedule is necessitated to still have good performance even with these delays. This two conflicting requirements forces the hGA to search for schedules that concentrate more slack for operations close to where the breakdown disruption is expected to occur and reduce it elsewhere. To verify this hypothesis, predictive schedules obtained for the instance MK08 by the proposed approach when BD2 machine disruptions are expected were randomly selected. After that, these schedules were subjected to BD3 machine breakdown disruptions instead of BD2 machine disruptions. The results showed that on average the expected AMS_RI and *ASTBI* of these schedules with respect to predictive schedules obtained by minimizing MS_{min}, i.e. deterministic approach, have increased from -8.4 and -61.78 to 0.60 and 6.98, respectively. This increase indicates that these predictive schedules are more compact and dense around the region where the actual breakdown disruptions occurred and the schedule slack was not enough to absorb the effect of the disruptions.

3.6 Conclusion

This Chapter proposed a preservation of solution quality approach that considers both robustness and stability in obtaining predictive schedules of flexible job-shops subject to machine breakdowns. The stability of the predictive schedule is evaluated by an aggregated machine breakdown. Furthermore, this Chapter defined six different bi-objective performance robustness and stability measures and investigated their effectiveness in producing robust and stable predictive schedules. For this, an ANOVA comparative study is conducted to compare the performance of these bi-objective measures. ANOVA results revealed that the robustness and stability measure called $Z_{R,2}$ can significantly improve the relative quality of the predictive schedule (*RQULT*) of FJSP where longer and/or early breakdown are expected.

The performance of the bi-objective robustness and stability measure $Z_{R,2}$ was compared against a neighborhood-based and a slack-based robustness measures taken from the literature. The predictive schedules for the FJSP were obtained using a two-stage genetic algorithm that was proposed to solve the previous measures. Computational results showed that predictive schedules generated using the proposed approach is superior in most cases in terms of robustness and stability than the referred methods. Moreover, results showed that the breakdown details of the experiments have a significant effect on the relative performance of the proposed approach.

CHAPTER 4

ROBUST SCHEDULING OF FLEXIBLE JOB-SHOP WITH PROCESSING TIME UNCERTAINTY: A COMPARISON STUDY

This Chapter considers the flexible job-shop scheduling problem when processing times of some operations are represented by or subjected to low-to-medium uncertainty. This uncertainty is represented by a uniform distribution with given lower and upper bounds. The objective is to find a predictive schedule that can deal with this uncertainty while maintaining its robustness. In this Chapter, two genetic approaches to obtain predictive schedule are compared. An approach based on expected processing times and an approach based on sampling technique. To determine the performance of the predictive schedules obtained by both approaches with respect to two types of robustness, an experimental study and Analysis of Variance (ANOVA) are conducted on a number of benchmark problems.

4.1 Introduction

Flexible job-shop scheduling problem (FJSP) is computationally difficult problem to solve. Hence, classical FJSP where all related data about the problem is assumed to be fixed or *deterministic* are considered as combinatorial optimization problems and classified as NP-hard problems (Garey et al., 1976). Recently, the use of *meta-heuristics* methods such as simulated annealing (SA), tabu search (TS) and genetic algorithm (GA)

in solving FJSP proved that an optimal or near optimal solution can be found with relatively small computational effort.

However, in real manufacturing systems unforeseen incidents happen. For this, classical models that assume deterministic data about processing times of operations, machines availability; etc; may; in theory; produce an optimal or near optimal schedule, but its performance may deteriorate when implemented in practice; i.e.; released to the shop floor; due to unexpected disruptions. Nevertheless, when incorporating the data uncertainty in the formulation of the already NP-hard FJSP, the problem becomes even more difficult and complicated to solve.

A number of methods are suggested in literature to deal with stochastic parameters of a certain scheduling problem. However, based on the desire of the decision maker these methods can be classified and accordingly choose a method that fulfills his need. For example, some decision makers favor a solution that can hedge against the worst possible scenario; others prefer a solution that has a high quality on average; whereas some look for a solution that minimizes the risk of ending with a bad solution. Sevaux and Sörensen (2004) presented a GA that uses sampling technique to estimate the robustness of a single machine schedule subjected to small variation in release dates. They stated that, in a similar way, other types of stochastic problem data can be easily incorporated. This Chapter modifies the hybridized genetic algorithm (hGA) proposed in Chapter 2 to deal with FJSP when some operations are represented by or subjected to variations characterized by a uniform processing time. Furthermore, the study compares two methods, a method based on sampling technique similar to Sevaux and Sörensen (2004)

and a method that optimizes the objective function based on the expected processing time of the operations (i.e. simple method similar to deterministic approach).

Schedules obtained from both methods are compared based on two robustness measures, *quality robustness* and *solution robustness*. This Chapter adopts the same definitions for both robustness measures adopted by Sevaux and Sörensen (2004). Quality robustness is measured by the objective function value (makespan of the predictive schedule). Here, a schedule is said to be of quality robust when its objective value does not deviate much from the best obtained optimal or near optimal solution. On the other hand, solution robustness measures how much the solution has deviated after the disturbance from the original solution (makespan of the realized schedule). This means that the first robustness measure, quality robustness, is concerned with the objective function space whereas solution robustness is concerned with the solution space. The advantage of considering these two measures when comparing obtained schedules is emphasized more in Section 4.5.

The remainder of this Chapter is structured as follows: after the literature review in Section 4.2, Section 4.3 describes the FJSP. Section 4.4 discusses the modified hGA architecture. Analysis of the computational results is presented and discussed in Section 4.5. Finally, the research summary is covered in Section 4.6.

4.2 Literature Review

For decades the emphasis of literature that discusses scheduling problems is put towards deterministic scheduling problems where the data parameters are assumed to be fixed and

known beforehand. Nevertheless, recently more attention is given to schedule systems where some data parameters are unknown or are represented by some probabilistic distributions. Since most scheduling problems are classified as NP-hard, heuristic and meta-heuristic approaches received much attention to deal with the presence of uncertainty in the problem's data parameters. This section gives a brief survey of stochastic scheduling approaches found in literature.

Daniels and Kouvelis (1995); Kouvelis et al. (2000); Möhring et al. (1985); Montemanni (2007); Pinedo (1982); Wu et al. (2009); and Xia et al. (2008) addressed stochastic single machine with uncertain jobs processing times. Single machine environment subjected to machine breakdowns was considered by others like Al-Turki et al. (1996); Cai and Tu (1996); and Liu et al. (2007b). Similarly, Sevaux and Sörensen (2004) used a modified GA to find robust solution in single machine environment subjected to stochastic release dates of jobs.

Also, Leon et al. (1994) analyzed effects of machine breakdowns and processing time variability on the quality of job-shop schedules using slack-time based robustness measure. The performance of simple dispatching heuristics versus algorithmic solution techniques in job-shops subjected to uncertain processing times were studied by Lawrence and Sewell (1997) and Sabuncuoglu and Karabuk (1999) showed that dispatching rules are more robust to interruptions than the optimum seeking off-line scheduling algorithms. Mehta and Uzsoy (1998) proposed a two-step algorithm based on disjunctive graph representation to minimize maximum lateness and absorb the impact of random machine breakdowns on the predictive schedule of a job-shop by inserting idle time. Furthermore, Jensen (2001b, and 2003) used GA (proposed in Mattfeld, 1996) to

improve the robustness and flexibility of the job-shop schedules when minimizing maximum tardiness, summed tardiness and total flow-time measures using two robustness measures, a neighborhood-based robustness measure and a lateness-based robustness measure.

Guo and Nonaka (1999) studied how to reduce the effect of machine failure on a three-machine flow shop by proposing a method to evaluate initial schedules (predictive schedules) and a rescheduling method that is applied after machine failure. Shafaei and Brunn (1999) and Shafaei and Brunn (2000) used the rolling time approach to investigate the robustness of schedules. Byeon et al. (1998); Kutanoglu and Wu (1998); Kutanoglu and Wu (2004); and Wu et al. (1999) applied decomposition heuristic to divide the classical job-shop scheduling problem with uncertain processing times into a series of subproblems and then the problem parameters are iteratively updated to analyze the effect of the processing time variation using *a priori* stochastic information.

Anglani et al. (2005) presented a fuzzy mathematical model of scheduling parallel machines with sequence-dependent cost while considering uncertainties in processing times. Matsveichuk et al. (2009) proposed a two-stage scheduling decision framework to execute schedules of a two-machine flow shop with interval processing times. Also, Bouyahia et al. (2009) proposed a probabilistic generalization to design robust *a priori* scheduling that assumes the number of jobs to be processed on parallel machines as a random variable with respect to the total weighted flow time. Mahdavi et al. (2010) presented a real-time simulation-based decision support system to control the production of a stochastic flexible job-shop subjected to stochastic processing times. Readers are referred to Davenport and Beck (2002), Herroelen and Leus (2005), Aytug et al. (2005)

and Mula et al. (2006) who gave detailed review of literature related to scheduling under uncertainty.

In light of the literature, scheduling under uncertainty can be classified into number of categories depending on the adopted strategy by the decision maker on how to react to uncertainties. Hence, methods compared in this chapter falls under the category *proactive (robust) scheduling* which is defined as a schedule with relatively insensitive quality to a changing environment (Aytug et al., 2005; Herroelen and Leus, 2005). At this point, it is worth mentioning that Aytug et al. (2005) highlighted a very interesting conclusion stating that "*if the level of uncertainty is low enough, an optimization-based predictive scheduling algorithm can outperform an on-line dispatching algorithm*". Accordingly, the choice of the optimization method depends on the level of uncertainty. For this, while an algorithm based on finding a predictive schedule that can hedge against worst possible scenario may end up with a very conservative schedule, decision makers and/or schedulers may prefer a predictive schedule that minimizes risk of ending up in bad scenario. This is because an extremely conservative predictive schedule may come at the cost of sacrificing the initial predictive performance measure of that schedule. In such case, decision maker and/or scheduler may be in a trade-off situation between sacrificing the performance of the predictive schedule allowing it to absorb almost all of the disruptions, or having a higher performance of the predictive schedule with a lesser ability to absorbing effects of disruptions. Moreover, algorithms that are purely based on inserting additional idle times in predictive schedules may turn it to be inactive schedule. Therefore; in our opinion; if the expected level of uncertainty is low enough decision makers and/or schedulers might consider using two possible optimization-based

algorithms. The first is to adopt an algorithm that optimizes a schedule based on the average values of parameters with uncertainties (such as processing time). The second is to implement an algorithm based on sampling technique from the random distributions of these parameters. The later approach subjects different schedules' sequences to different sets of uncertainties and then selects the one that performs well on average. To the best of the author's knowledge, there is not a previous study that addresses a comparison of the two former algorithms for obtaining predictive schedules of the FJSP when some operations are represented by or subjected to low-to-medium processing time variations. Hence, the goal of this work is to evaluate and compare the quality and the solution robustness of predictive schedules obtained using these two choices in flexible job-shop environment where the processing times of some operations are represented by or subjected to low-to-medium uncertainty. Specifically, processing times of these operations are represented by an interval of equally possible real value between given lower and upper bounds. For clarity and ease of referencing, the algorithm that optimizes expected average data will be referred to as MS_{exp} and the algorithm that is based on sampling will be referred to as $MS_{\%\text{Rob}}$.

4.3 Problem Description

This work considers a non-preemptive flexible job-shop scheduling problem (FJSP) with the objective of minimizing the makespan. There is a set of $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ jobs and each job i has a set of $O = \{O_{i1}, O_{i2}, \ldots, O_{iq_i}\}$ operations where q_i denotes the total number of operations of job J_i. Each operation O_{ij} is to be processed in a subset of

machines $\mathcal{M}_{ij} \in \mathcal{M} = \{M_1, M_2, \ldots, M_m\}$. An operation O_{ij} cannot start processing until its precedence operation $O_{i(j-1)}$ has finished its processing. All n jobs are available at time $t = 0$ and the processing time $p_{ijk}$of some operations O_{ij} of job J_i in machine M_k may equally take any real value between given lower $\underline{p}_{ijk}$ and upper $\bar{p}_{ijk}$ bounds. This processing time variation of operation is due to, e.g., incomplete or unreliable information or unavoidable stochastic variability related to machine's tools and/or workers skills, etc. The processing time uncertainty can be described by a set of all possible scenarios (infinite) ζ. Each unique set of processing times ξ is obtained by equally selecting a value from the associated interval of each operation:

$$p_{ijk}^{\xi} \in \left[\underline{p}_{ijk}, \bar{p}_{ijk}\right] \forall\ i \in \{1, 2, \ldots, n\}, j \in \{1, 2, \ldots, q_i\}, k \in \{1, 2, \ldots, m\} \qquad (4.1)$$

In practice the actual operations' processing time of some operations may not be known or difficult to verify until the operation has finished processing. In such case, assuming an expected value helps the scheduler or decision maker in obtaining a schedule that satisfies a certain performance measure. In this work, the expected processing times $E\left[p_{ijk}\right]$ of operations that are represented by or subjected to variations according to a uniform time intervals are given by:

$$E\left[p_{ijk}\right] = \left(\left(\underline{p}_{ijk} + \bar{p}_{ijk}\right)/2\right) \qquad (4.2)$$

4.4 Hybridized Genetic Algorithm for the FJSP

Chapter 2 proposed hybridized genetic algorithm (hGA) architecture for the deterministic FJSP and results illustrated that the approach is very effective in minimizing the

makespan of this problem. The work conducted in Chapter 3 shows that this hGA can be modified to deal with FJSP subjected to random machine breakdowns by replacing its fitness function. In the following sections, describes the original deterministic hGA and then show how it can be modified to minimize the makespan of schedules according to average expected processing times data or according to the sampling technique method.

4.4.1 Deterministic hGA for the FJSP

The proposed hGA in Chapter 2 used permutation-based representation chromosome representation; where each operation is represented by triples (k,i,j) such that k is machine assigned to the operation, i is current job number, and j is the progressive number of that operation within job i. A schedule for FJSP with three jobs and three machines can be represented by (221-131-111-212-322-223-332). In this architecture, the initial population is created by two ways. The first way is to generate half of the population randomly. The second half of the population is generated using a schedule construction heuristic called *Ini-PopGen*. Ini-PopGen starts by randomly assigning priority to jobs. Then, based on this priority an operation is scheduled on the machine (from the set of appropriate machines) that can finish it sooner. This procedure considers the processing time and the work load on the machine while assigning operations.

Chromosomes decoding follow an active decoding procedure, wherein no operation can be started earlier without delaying at least one other operation or violating the technological constraints. After the active decoding, the schedule is improved by a local search procedure that results in a local optimal schedule (Lamarckian learning).

However, this local search procedure is only applied every d^{th} generation and number of moves is limited to a maximum *loc_iter* moves without improvements.

Two chromosomes are selected from the population. At first, roulette wheel technique is used to form donors' mating-pool based on a selection probability given by:

$$P_{sel} = \frac{F_{ind}}{F_{tot}}, ind = 1, \dots, N \quad (4.3)$$

Where, P_{sel}: is the probability of choosing the ind^{th} individual; N: is the population size; F_{ind}: is the ind^{th} individual fitness; and F_{tot}: is the total fitness of all individuals in the current generation.

Then, if the individual in the donors' mating-pool passes a crossover probability P_c, an *n*-Size tournament method is used to select *n* chromosomes from the population to form the receivers' mating-sub-pool. Then, the best individual (one with lowest fitness value, makespan) in the sub-pool is chosen for reproduction.

The crossover operator is based on the *Precedence Preserving Order-based Crossover* (POX) (Kacem et al., 2002a) and was modified not to treat the parents symmetrically. Mutation of individuals is implemented through using two operators. The first operator is a *Machine Based Mutation* (MBM), where a random number of operations (denoted as *nrand*) are selected and reassigned to another machine. After that, modified *Position Based Mutation* (PBM) (Mattfeld, 1996) is applied. PBM was originally designed for JSP using single triple permutation-based chromosomes representation. Thus, the PBM is modified so that no infeasible chromosomes are produced and it starts by randomly selecting an operation within the chromosome and then reinserting it at another position.

4.4.2 Modified hGA for the FJSP

In this section, a description of how the original deterministic hGA can be modified to solve the FJSP to find schedules using the two methods, MS_{exp} and $MS_{\%Rob}$ is provided.

Historical records of a certain shop floor can provide approximated distribution uncertainties that can affect it; such as machine breakdowns, processing times variations, cancelations or arrivals of new jobs, etc. In FJSP, Chapter 3 shows that such distributions can be used as a guide when generating the predictive schedule. This can be achieved by integrating the probability distribution of that specific uncertainty with the machine routing and sequencing of operations so that overall performance, measured by makespan, of the schedule is not affected to a high degree in case such disruption occurs.

Previous Chapter as well as previous studies like Jensen (2001b, and 2003), Leon et al. (1994), Sevaux and Sörensen (2004), etc, showed that such objective can be achieved by replacing the fitness function of the GA by a fitness function that satisfies the new objective, usually referred to as *robust fitness function*. The main purpose of such robust fitness evaluation functions is to guide the evolution of solutions towards solutions that are not or slightly affected by perturbed data parameters. This is motivated by the robust optimization of continuous mathematical functions which suggests that robust optima are located on broad beaks in the fitness space. For example, Tsutsui and Ghosh (1997), Tsutsui (1999) and Branke (1998, and 2001) used genetic algorithm in which they perturb solutions according to a noise distribution before fitness evaluation and used this perturb fitness as the fitness of the evaluated phenotype.

For the suggested comparison between the two methods in this work, the ordinary objective function of the deterministic FJSP with minimum makespan is given by:

$$MS_{min} = min\,\{max(C_i)\}\forall J_i = \{J_1, J_2, \dots, J_n\} \tag{4.4}$$

where, MS_{min} is the minimum makespan, and C_i is the completion time of job J_i; can be applied and/or modified as follows. First, the same ordinary objective function Eq. (4.4) is used for optimizing the MS_{exp} method. The only difference is when using the processing times of operations represented by or subjected to uncertainty. In this case, the expected processing operations' times replaces the uniform interval processing times and then these expected processing time values are used to generate the sequence of the predictive schedule. The advantage of using such method is reducing the complexity of dealing with sampling of data values to a single data value making it similar to the classical deterministic approach. However, for the second method $MS_{\%\text{Rob}}$ the procedure is not straight forward. Here, according to Sörensen (2001), the solution of such objective fitness function has to be implemented on a randomly modified sample set of characteristics (or data parameters) and then combining a number of evaluations of the same schedule *s* sequence solution in the objective fitness function. A possible sampling objective fitness for uncertain processing times can be represented by a weighted average of *m* derived evaluations such that:

$$MS_{\%\text{Rob}}(s) = \frac{1}{m}\sum_{l=1}^{m} w_l \varphi\left(MS_{min}(s), \zeta_l\left(p_{ijk}^{\xi}\right)\right)$$

$$\forall\, i \in \{1, 2, \dots, n\}, j \in \{1, 2, \dots, q_i\}, k \in \{1, 2, \dots, m\} \tag{4.5}$$

where, w_l is the weight related to the derived schedule s sequence evaluation, $\zeta_l\left(p_{ijk}^{\xi}\right)$ is sampling function that takes a random sample of a certain processing time scenario p_{ijk}^{ξ}, and m is the number of samples used to evaluate the schedule s.

Therefore, the previously described hGA in Subsection 4.4.1 is modified to first use Eq. (4.4) with the expected processing times for MS_{exp} method, and then modified by replacing its fitness function by Eq. (4.5) for $MS_{\%\text{Rob}}$ method. The hGA used for each method will be referred to as MS_{exp}-hGA and $MS_{\%\text{Rob}}$-hGA, respectively. Using different objective fitness function in the hGA will lead to obtaining different schedule sequences. Thus, each schedule's sequence may respond differently to disruptions as some may be able to absorb their effects more than others. Furthermore, since GA utilizes a population of solutions in its search, this gives higher chances to explore a diverse set of solutions. Hence, GA will have higher chances of finding schedule sequences that are less sensitive to data uncertainties.

4.5 Analysis and Results

At this stage, it may worth emphasizing that details of reported results here depend on the assigned values to the parameters of the simulated system. As a result, it may not be possible to generalize conclusions reached in this section to all situations. Nevertheless, since great care is paid in selecting representative benchmarks covering wide range of FJSP systems and conditions, it may be sensible to anticipate that results and conclusions are relevant in a broader sense and acknowledge that they serve as a reason to be carefully optimistic.

Numbers of FJSP benchmarks with a wide range of sizes, from 5 x 3 to 20 x 15 found in literature are used for the experiments. These benchmarks are Ex1 taken from Lee and DiCesare (1994), and examples MK01 – MK10 proposed by Brandimarte (1993). For this work, the expected processing time $E[p_{ijk}]$ of an operation is to be equal to processing time of that operation in the original problem. Hence, the upper and lower processing time bounds of an operation affected by uniform variation in its processing time is calculated by:

$$\left[\underline{p}_{ijk}, \bar{p}_{ijk}\right] = E\left[p_{ijk}\right] \times [(1-\beta), (1+\beta)] \tag{4.6}$$

where, β is the percentage difference from the original expected processing time.

Parameter β represents the level of variability of the operation's processing time. In order to control the number of operations that are affected by the processing time variability, the parameter α has been used. Table 4.1 shows different combinations of the two parameters, α and β, that are used to generate the different test cases of the experiments.

Table 4.1: Different processing time variation's combinations

Disturbance Type	% of affected operations & disturbance level	α	β
DT1	Low, low	0.20	0.15
DT2	Low, medium	0.20	0.40
DT3	Medium, low	0.40	0.15
DT4	Medium, medium	0.40	0.40

There is no clear definition of low, medium or high variation of processing time in the literature. However, based on experience and personal judgment each researcher sets his own definition of low, medium or high variation. For example, Shafaei and Brunn (1999) considered approximately 5% variation of expected processing time to be low and 25% variation to be high. On the other hand, both Daniel and Kouvelis (1995) and Kouvelis et al. (2000) considered 10% variation to be low, 30% variation to be medium and 50% variation to be high. Nonetheless, from the industrial experience is noted that such classification is a problem dependent. It depends on number of factors related mainly to the used unit time and how time variation is interpreted in the overall cost. For simplicity, it is defined that a 15% variations in the expected processing times to be low level and up to 40% variations to be medium.

4.5.1 hGA parameters

The number of function evaluations m, i.e. the number of samples used to evaluate the schedule s, in Eq. (4.5) requires being sufficient. This is due to the fact that using a smaller m value may lead to selecting a non-robust solution, whilst using a larger value leads to unacceptable increase in the computational time. Hence, three different m values of evaluations are tested in all experiments. Here, all values of m are related to the total number of operations of each instance. The *m value* is set to 50%, 75% and 100% of the total number of operations, and hence, the corresponding $MS_{\%Rob}$-hGA is referred to as MS_{50Rob}-hGA, MS_{75Rob}-hGA and MS_{100Rob}-hGA, respectively. All sequence evaluations are given the same importance and hence the weight w_l in Eq. (4.5) is set to 1.

All test codes are implemented and executed using C++ on an Intel® Core™ 2 Quad CPU @ 2.4 GHz with 3.24 GB RAM. For comparability and ease of implementation, all hGA are closely related and the parameters are experimentally tuned according to the performance of MS_{exp}-hGA (minimizing MS_{exp}). The parameter values that are chosen for the two-stage hGA algorithm are as follows: population size 200, crossover probability 0.7, mutation probability 0.3, number of generations 200, number of parents in the receivers' mating sub-pool 4, number of generations to perform local search d = 10, maximum number of moves without improvement in the local search *loc_iter* = *min* [*tot_noper*, 150], and the worst chromosome is replace every k = 3 generations.

4.5.2 Analysis of robustness measures

To compare the performance of methods, MS_{exp}-hGA and $MS_{\%Rob}$-hGA, a simulation procedure is applied. Consequently, a standard MS_{exp}-hGA using the ordinary evaluation function (described in Subsection 4.4.2) minimizing MS_{exp} is first run to obtain schedules with sequence referred to as *expected sequence*. Then, $MS_{\%Rob}$-hGA, with the systematic application of sampling function evaluation (Eq. (4.5)), is used to obtain schedule with sequence referred to as *sampling sequence*. In order to draw more accurate responses, five schedules for each of the hGA different settings of each test case is used. After the sequences are obtained, 400 replications of each problem instance with randomly modified processing times according to the disturbances are evaluated. This results in 5 (number of obtained schedules' sequences) x 4 (disturbances levels) x 400 (replications) = 8000 test runs per test instance. The comparative study of the two methods, $MS_{\%Rob}$-

hGA and MS_{exp}-hGA, is performed using analysis of variance (ANOVA) through the use of the commercial statistical software Minitab 15.

Since this comparative study is done to compare the performance of the predictive schedules' sequence obtained using MS_{exp} method and predictive schedules' sequence obtained using the $MS_{\%Rob}$ method, all obtained sequences are subjected to the same processing time variation disturbances and their performance is compared in terms of:

1- The relative error (*RE*) predictive makespan deviation with respect to the best-known lower bound value defined as:

$$RE = \left[\left(MS_{comp} - LB\right)/LB\right] \times 100 \quad (4.7)$$

where, MS_{comp} is the initial predicted makespan obtained using either method, and *LB* is the best-known lower bound. It is worth pointing out that since for every replication the processing times are randomly modified, estimating its *LB* value is not possible. Therefore, the used *LB* is the same *LB* reported in literature for the same test case group. The relative error measures the robustness in the objective function space, i.e. quality robustness.

2- The average relative deviation (*Ave. RDev*) of the 400 modified instances with respect to the initial expected makespan defined as:

$$Ave.RDev = \frac{1}{N}\sum_{p=1}^{400}\left[\left(MS(q)_{Rp} - MS(q)_{P}\right)/MS(q)_{P}\right] \times 100 \quad (4.8)$$

The relative deviation evaluates the robustness in the solution space, i.e. solution robustness, which quantifies how much the solution has deviated after the disturbance from the original solution.

3- The average absolute relative makespan deviation between the initial predicted schedule makespan and the actual realized makespan after the 400 disturbances'

replications according to the following equation:

$$Ave.Abs\ RMS_{\Delta} = \frac{1}{N}\sum_{p=1}^{400}\left\{\sqrt{\left[MS(q)_{Rp} - MS(q)_{P}\right]^{2}/[MS(q)_{P}]^{2}}\right\} \times 100 \quad (4.9)$$

where, $\boldsymbol{q}$ is the replication predictive schedule, and the subscripts P, R and p : refer to predictive (or the original released schedule to the shop floor), realized (or the actual schedule after disturbance simulation), and the processing time disturbance number, respectively.

The performance measures addressed above examine the average values related to the obtained replications' schedules at each combination before or after the disruptions. One of the essential concerns associated with any proposed method, a heuristic or a meta-heuristic method, to solve a problem is arbitrating the quality of its obtained solutions (predictive schedules in this study). Therefore, the first measure *RE* seeks to answer that concern by measuring how far are the obtained schedules from the optimal or near optimal schedules? This ensures that only methods that are capable of obtaining predictive schedules of high quality, minimum makespan, as well as proving their repeatability, or robustness in obtaining such schedules, are given the credit. For this reason, *RE* measure is designed to work on the objective function space by comparing the quality of obtained solution of any method to a standard benchmark solution.

While *RE* measure quantifies the quality robustness of obtained predictive schedules, *Ave. RDev* and $Ave.Abs\ RMS_{\Delta}$ measures, on the other hand, assist evaluating the solution robustness of these schedules and their sustainability in the face of uncertainties. To achieve this evaluation it is required to find a way of comparing the original solution, i.e. predictive schedule released to the floor shop, to the final solution, i.e. the realized

schedule after the disruptions. Levenshtein (1966) was first to introduce the concept of *edit distance* in the context of correcting binary codes. This concept was used as a measure determining the similarity between two strings. He defined it as the sum of the minimum costs of the elementary edit operations; insertion, deletion and substitution; that transform string s into string t.

The edit distance concept can be generalized to measure the distance between two schedules by interpreting the edit distance as the *changes* that have to be done in order to transfer a schedule and turn it to be like a given baseline schedule. These changes can be in the form of an operation insertion, an operation deletion or the substitution of an operation by another one in the schedule. Nevertheless, the exact interpretation of the edit distance concept is highly problem nature dependent. As previously presented in Section 4.4, in this Chapter the role of the studied methods, $MS_{\%Rob}$-hGA and MS_{exp}-hGA, is to obtain a schedule sequence that is released to the shop floor without future alternation in its sequences. Thus, a more appropriate distance measure is to consider the difference in the main performance measure between the predictive schedule and the realized schedule, i.e. makespan deviation. For this reason, *Ave. RDev measure* is used and is interpreted as the relative mean of the deviations between the realized schedules after disruptions and the originally released predictive schedules. Moreover, $Ave.Abs\ RMS_{\Delta}$ is also used to draw more accurate conclusions. It is considered as a quantity that measures how close, in average, the realized schedules are to the predictive schedules.

At this point, it is important to stress that while it is enough to consider the *RE* measure to test the solution quality of an algorithm or method developed to solve deterministic scheduling problems, *Ave. RDev* and/or $Ave.Abs\ RMS_{\Delta}$ cannot be used as the only

measure for the stochastic problems with uncertain data. Practically, they have to be considered along with a quality robustness measure. To illustrate this, consider the Gantt-charts shown in Figure 4.1. The Gantt-charts show two possible schedules for an arbitrary flexible job-shop with three jobs and three machines. In Figure 4.1, schedule (b) differs from schedule (a) by inserting a 2 unit idle time in-between an operation and its preceding operation. Implementing schedule (b) as is makes that idle time a buffering time zone that absorb a possible variation in the processing time of operations. Thus, it is highly expected that *Ave. RDev* and/or $Ave.Abs\,RMS_{\Delta}$ will have a very low value indicating that schedule (b) is of high solution robustness in face of disruptions. However, considering *RE* measure demonstrate that this solution is of low quality as it is far from the optimal baseline solution, schedule (a). Furthermore, such idle time insertion has turned schedule (b) to be an inactive schedule. On the other hand, if the decision maker or scheduler implemented schedule (a) as is, because of its very low *RE* measure, then any possible processing time variation in operations that form the critical-path of schedule (a) will have a direct impact on increasing its makespan leading to a low solution robustness.

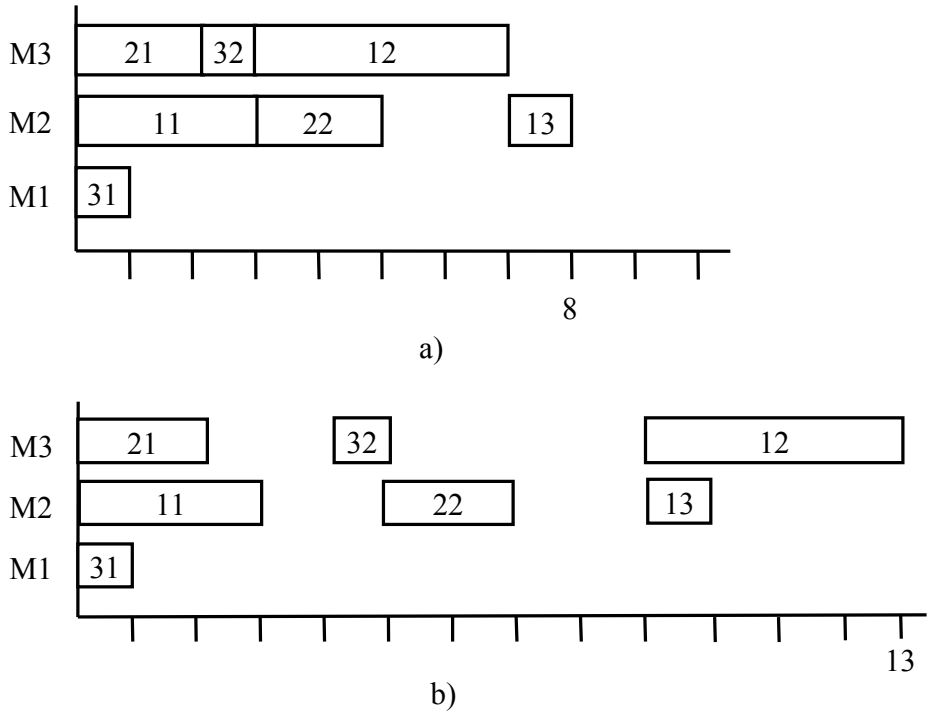


Figure 4.1: Two schedules illustrating the difference between quality robustness and solution robustness.

4.5.3 Computational results

Table 4.2 shows the detailed results obtained using MS_{exp}-hGA and $MS_{\%Rob}$-hGA. Table 4.2 consists of 15 columns. The first and second columns represent the instance name and size. The third column refers to used method to obtain the corresponding schedule. The remaining columns are labeled according to the performance measures given above and give the results of the 400 replications of specific disturbance type for the modified instances, i.e. test case groups. For each column, the best performance; lowest average

deviation percentage; is printed in bold-face. When considering *RE* results in Table 4.2, it can be noted that including variations of the processing times in the objective function ($MS_{\%Rob}$-hGA) to obtain a schedule has a negligible effect on increasing the makespan of the predictive schedule. Therefore, the maximum increase in *RE* when using the robust sampling objective function, $MS_{\%Rob}$-hGA, compared to the expected ordinary fitness function evaluation, MS_{exp}-hGA, is 6.66% (for the MK06 test case group) and on average 0.86% for all modified instances. Furthermore, in some cases the sampled solutions obtained by $MS_{\%Rob}$-hGA were sometimes slightly better than the expected solutions obtained by MS_{exp}-hGA like for the test cases MK01, MK04, MK07, MK09 and MK10. This may be explained by the change in how the population is handled when using the sampling function evaluation, Eq. (4.5), which may allow escaping from local optima. In addition, in terms of $Ave.Abs\ RMS_{\Delta}$ and *Ave. RDev*, the robust solutions results acquired by $MS_{\%Rob}$-hGA are outperforming those obtained by MS_{exp}-hGA. Thus, using a schedule sequence obtained by $MS_{\%Rob}$-hGA performs mostly better after disturbance occurrence compared to a schedule obtained by MS_{exp}-hGA. These findings highlight the capability of $MS_{\%Rob}$-hGA to find solutions that are both quality robust and solution robust.

Table 4.2: Computational results – deviation of schedules when subjected to random uniform processing time variations

Inst.	Size	Method	DT1			DT2			DT3			DT4		
			RE	*Ave. Abs* MS_Δ	*Ave. RDev*	*RE*	*Ave. Abs* MS_Δ	*Ave. RDev*	*RE*	*Ave. Abs* MS_Δ	*Ave. RDev*	*RE*	*Ave. Abs* MS_Δ	*Ave. RDev*
Ex1	5 x 3	MS_{exp}-hGA	**52.17**	1.47	1.3	**52.17**	4.09	3.97	**52.17**	1.97	1.76	**52.17**	5.48	5.12
		MS_{50R}-hGA	56.52	**1.37**	**0.44**	58.26	**3.56**	2.28	53.91	1.91	1.37	54.78	4.91	3.82
		MS_{75R}-hGA	54.78	1.50	0.77	55.65	3.58	**2.18**	55.65	**1.77**	**0.91**	55.65	4.81	3.51
		MS_{100R}-hGA	56.52	1.46	0.72	55.65	3.67	2.53	53.91	1.81	1.21	56.52	**4.47**	**3.11**
MK01	10 x 6	MS_{exp}-hGA	13.89	1.14	0.52	13.89	2.77	1.87	13.89	1.49	0.85	**13.89**	3.81	2.91
		MS_{50R}-hGA	13.33	1.16	**-0.03**	**11.11**	**2.43**	0.41	**12.22**	**1.41**	0.35	16.67	**3.68**	0.21
		MS_{75R}-hGA	**11.11**	1.07	0.06	13.33	2.70	**0.28**	16.67	1.48	**-0.01**	16.67	3.74	**-0.07**
		MS_{100R}-hGA	**11.11**	**1.03**	0.10	13.33	2.74	0.46	14.44	1.44	0.37	16.67	4.11	0.16
MK02	10 x 6	MS_{exp}-hGA	**15.83**	1.11	1.07	**15.83**	2.99	2.9	**15.83**	1.84	1.79	**15.83**	4.97	4.90
		MS_{50R}-hGA	16.67	**1.08**	**0.19**	**15.83**	**2.89**	**2.55**	16.67	**1.24**	**0.65**	**15.83**	5.88	5.84
		MS_{75R}-hGA	17.5	1.13	0.81	**15.83**	3.38	3.10	16.67	1.85	1.39	16.67	**4.18**	**3.66**
		MS_{100R}-hGA	16.67	1.14	0.62	16.67	3.36	3.32	16.67	1.75	1.48	16.67	5.48	5.38
MK03	15 x 8	MS_{exp}-hGA	**0.00**	0.88	0.00	**0.00**	**2.29**	0.00	**0.00**	1.17	0.02	**0.00**	3.24	0.04
		MS_{50R}-hGA	**0.00**	0.90	0.00	**0.00**	2.44	**-0.06**	**0.00**	1.19	**-0.05**	**0.00**	3.12	**0.03**
		MS_{75R}-hGA	**0.00**	0.92	**-0.04**	**0.00**	2.37	0.04	**0.00**	**1.11**	0.00	**0.00**	3.03	0.11
		MS_{100R}-hGA	**0.00**	**0.10**	0.01	**0.00**	2.35	-0.01	**0.00**	1.18	-0.10	**0.00**	**3.01**	0.11
MK04	15 x 8	MS_{exp}-hGA	37.92	0.97	0.55	37.92	2.66	2.21	37.92	1.32	0.92	37.92	3.83	3.55
		MS_{50R}-hGA	37.92	0.88	**0.08**	37.08	**2.06**	0.58	36.67	1.21	0.48	37.5	**2.98**	1.64
		MS_{75R}-hGA	**37.08**	**0.67**	**0.08**	**36.25**	2.33	1.43	37.08	1.19	**0.11**	39.17	3.12	**1.47**
		MS_{100R}-hGA	38.33	0.87	0.10	40	2.64	**0.33**	**36.25**	**1.18**	0.32	**37.08**	3.58	3.20
MK05	15 x 4	MS_{exp}-hGA	**4.167**	0.59	0.43	**4.17**	1.71	1.54	**4.17**	0.90	0.74	**4.17**	2.70	2.59
		MS_{50R}-hGA	**4.167**	0.56	0.41	4.76	1.72	1.58	4.40	0.90	0.74	5.00	2.50	2.31
		MS_{75R}-hGA	4.52	**0.49**	**0.17**	5.24	**1.55**	**1.28**	5.24	**0.75**	**0.30**	4.64	2.84	2.73
		MS_{100R}-hGA	5.00	0.55	0.24	4.64	1.63	1.40	5.00	0.82	0.50	4.76	**2.47**	**2.27**
MK06	10 x 15	MS_{exp}-hGA	**98.79**	1.03	1.03	**98.79**	2.84	2.83	**98.79**	1.70	1.70	**98.79**	4.72	4.71
		MS_{50R}-hGA	101.21	1.47	1.42	101.81	**2.00**	**1.83**	103.64	**1.09**	**0.92**	103.64	3.90	3.83
		MS_{75R}-hGA	103.64	0.86	0.79	103.64	2.44	2.37	102.42	1.41	1.38	104.85	**3.47**	**3.40**
		MS_{100R}-hGA	102.42	**0.71**	**0.56**	101.81	2.50	2.43	101.82	1.11	0.98	105.45	3.88	3.84
MK07	20 x 5	MS_{exp}-hGA	10.53	0.83	0.77	10.53	2.49	2.40	10.53	1.37	1.29	10.53	4.11	3.99
		MS_{50R}-hGA	10.98	**0.80**	0.55	**9.32**	2.38	2.21	9.92	**1.17**	**0.89**	11.28	3.92	**3.81**
		MS_{75R}-hGA	**10.23**	0.88	0.75	10.98	2.43	2.30	**9.62**	1.27	1.03	**9.77**	4.17	4.10
		MS_{100R}-hGA	**10.23**	**0.80**	**0.51**	11.28	**2.08**	**1.84**	11.58	1.28	1.13	11.73	**3.91**	**3.81**
MK08	20 x10	MS_{exp}-hGA	**0.00**	0.40	0.26	**0.00**	1.19	0.94	**0.00**	0.57	0.38	**0.00**	1.61	1.32
		MS_{50R}-hGA	**0.00**	0.38	**-0.01**	**0.00**	**0.95**	0.05	**0.00**	**0.52**	0.02	**0.00**	1.38	0.19
		MS_{75R}-hGA	**0.00**	**0.36**	0.01	**0.00**	0.98	**0.02**	**0.00**	**0.52**	**-0.01**	**0.00**	1.42	**0.07**
		MS_{100R}-hGA	**0.00**	0.37	0.00	**0.00**	0.97	0.04	**0.00**	0.53	0.02	**0.00**	**1.32**	0.08
MK09	20 x 10	MS_{exp}-hGA	3.41	0.71	0.65	**3.41**	2.15	2.13	**3.41**	0.99	0.92	**3.41**	3.26	3.24
		MS_{50R}-hGA	**3.34**	0.73	**0.05**	4.01	1.95	0.98	**3.41**	0.90	0.23	4.08	2.24	1.32
		MS_{75R}-hGA	3.48	0.76	0.20	3.68	1.93	0.59	**3.41**	0.88	0.27	3.68	2.93	2.54
		MS_{100R}-hGA	4.15	**0.70**	0.12	4.21	**1.68**	**0.51**	3.55	**0.87**	**0.07**	4.15	**2.15**	**1.21**
MK10	20 x 15	MS_{exp}-hGA	39.64	0.80	0.71	39.64	2.34	2.30	39.64	1.17	1.12	**39.64**	3.80	3.79
		MS_{50R}-hGA	39.39	**0.62**	0.30	40.36	2.23	2.19	**38.67**	0.90	0.62	40.48	3.41	3.39
		MS_{75R}-hGA	**39.15**	0.64	0.48	41.09	**1.73**	**1.56**	39.76	**0.82**	**0.43**	41.82	**3.30**	**3.26**
		MS_{100R}-hGA	40.61	0.68	**0.27**	**39.52**	2.05	1.73	39.27	0.99	0.87	40.48	3.46	3.44

Values written in bold are the best values

To get a deeper insight about the effects of using the four different methods MS_{exp}-hGA, MS_{50Rob}-hGA, MS_{75Rob}-hGA and MS_{100Rob}-hGA along with the different levels of disturbances on the performance of schedules before and after disturbance we used ANOVA. Table 4.3 shows the *F*-ratio and *P*-value of the ANOVA results. The *F*-ratio of the *RE* measure confirms the previous conclusions that solutions obtained using $MS_{\%Rob}$-hGA are quality robust since the used method, $MS_{\%Rob}$-hGA, does not have significant effect on them. Furthermore, the *F*-ratio indicates that $Ave.Abs\ RMS_{\Delta}$ and *Ave. RDev* of the schedules are mainly affected by disturbance type (DT) whereas the method used to obtain the schedule does not have the same significance. This is supported by the similar values noted in Table 4.2.

Table 4.3: ANOVA results concerning *RE*, $Ave.Abs\ RMS_{\Delta}$, and *Ave. RDev*

	RE		$Ave.Abs\ RMS_{\Delta}$		*Ave. RDev*	
Factor	*F*-ratio	*P*-value	*F*-ratio	*P*-value	*F*-ratio	*P*-value
A:Method	0.05	0.985	3.58	0.014	16.78	0.000
B:DT	0.02	0.996	557.92	0.000	165.16	0.000
AB	0.00	1.00	0.57	0.825	0.88	0.538

The previous findings raise an essential concern about the benefit of using a simulation based objective function method, like $MS_{\%Rob}$, in favor of an ordinary evaluated objective function method that works by using an expected processing time, like MS_{exp}. Specifically, the concerns are related to obtaining predictive schedules' sequences for the

FJSP when processing times of some operations are represented by low-to-medium uniform distribution. These concerns are raised from two points. This first one is deduced from the finding that though simulation based objective function methods $MS_{\%Rob}$ produce solutions that are both quality and solution robust, yet these improvements in the solution robustness, when compared with the expected evaluated objective function method MS_{exp}, are marginal. The second drawback of simulation based methods is their required high computational time compared to the expected processing time method. For example, when solving Ex1, which is a small instance problem used in this study, the computational time increased by 38%, 70%, and 108% when MS_{50Rob}-hGA, MS_{75Rob}-hGA and MS_{100Rob}-hGA were used compared to using MS_{exp}-hGA, respectively.

4.6 Conclusion

This Chapter presented how a hybridized genetic algorithm for a flexible job-shop problem can be modified to find robust solutions when it is subjected to low-to-medium random variations in the operations' processing times. For this, two methods were compared. Computational results showed that obtained solutions are both solution robust and quality robust. Furthermore, computational results revealed an interesting finding showing if an FJSP is subjected to a low-to-medium level of processing time uncertainty, then an optimization-based method working with expected processing times value may obtain schedules that are as good as schedules obtained using a sampling technique method and hence saving the computational efforts.

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

5.1 Conclusions and Contributions

In this thesis, efficient concurrent meta-heuristic scheduling architectures for the flexible job-shop scheduling problem are proposed. The main focus of this work is on the offline scheduling aspect for the FJSP. It considers the two cases of offline scheduling, when the problem's parameters are deterministic, and the case when some of the problem's parameters are stochastic. The developed architectures in this research work are an integration and/or modification tools detailed as follows:

- A heuristic method (called Ini-PopGen) capable of obtaining optimal or near optimal schedules for small and medium flexible job-shop systems. The method optimizes a completion time-based objective function, makespan, while maintaining an acceptable balanced machine work load (Subsection 2.43). Furthermore, this method can be modified to consider other time-based objective criteria such as minimizing the mean flow time.
- A genetic algorithm approach that can work in larger systems where optimal schedules cannot be achieved due to computational limitations. The genetic algorithm is designed such that its modification to accommodate different objectives is achievable while sustaining a high performance in obtaining optimal or near optimal schedules.

- A local search (LS) approach that exploit the neighborhood of a given schedule and accordingly improving it.

In the previous chapters, the proposed methods were introduced in details and evaluated to validate their performances. At first, these main three components are first integrated forming an efficient hybridized genetic algorithm (hGA) structure that is very effective and has a good potential of obtaining optimal or near optimal results for the deterministic FJSP. The experimental results advocate the good performance of the proposed Ini-PopGen heuristic by outperforming some of the existing approaches in the literature. The performance of Ini-PopGen is further improved when combining it with the proposed LS for small to medium sized T-FJSP and P-FJSP indicating that Ini-PopGen with LS can be used as a stand-alone tool.

Furthermore, in Chapter 3, a non-idle time inserting methodology to deal with FJSP subjected to machine breakdown disruptions is proposed. The methodology is based on modifying the previously proposed deterministic hGA and converting it to a two-stage hGA. For that, the stability of the schedule is approximately evaluated by an aggregated machine breakdown. Also, the chapter provided an extensive ANOVA comparative study conducted to compare the performance of six different bi-objective performance robustness and stability measures. The chapter is concluded by an experimental study comparing the performance of the proposed method against the performance of other methods found in literature and gave an insight explaining its superiority to them.

Following that, another type of disruption that usually faces a manufacturing system, processing time uncertainty, is considered in Chapter 4. It discussed two possible methods that can be used to find robust schedules when a FJS is subjected to low-to-

medium random variations in the operations' processing times. Furthermore, computational results revealed an interesting finding. The chapter shows that if a FJSP is subjected to a low-to-medium level of processing time uncertainty, then a method based on optimizing expected processing times value may obtain schedules that are as good as schedules obtained using a sampling technique method and hence saving the computational efforts.

5.2 Future Work

Many interesting extensions of the current work can be investigated. For example, the proposed Ini-PopGen heuristic that have been shown to be efficient in scheduling small to medium FJSP follows a greedy nature that schedules operations after the other. This limits its efficiency when considering large scale FJSP. Hence, a more global approach can be exploited to obtain schedules. This global approach may consider a more realistic case that detects deadlock situations[3] and embed techniques that resolve such situations.

Furthermore, the proposed methodology in Chapter 3 may be explored to handle other uncertainties such as arrival of new jobs, job cancellations, etc, and find out how such uncertainties affects the operational decisions. Moreover, developing a detailed framework that relates the different hierarchy levels, managerial and operational levels, of a manufacturing system and then use it to show how different performance measures, either time-based like makespan and flow time or cost-based measures, are sensitive to different uncertainties. Such framework model can be then used to associate a risk factor

[3] A deadlock situation is a situation that occurs when a set of jobs enter a circular wait. Here, each job in this set is blocking a resource it is using waiting for another resource to become available, which is in the same time is being block by another job from that set.

aiding the decision-making process. Another interesting direction for future research is to develop a multi-objective approach that returns Pareto frontier solutions for the decision maker to select a preferable robust and/or stable schedule.

The presented work in Chapter 4 can be extended to study the impact of other kinds of processing time distributions on the conclusions found in this study. Another interesting direction for future research is to develop a multi-objective approach that returns Pareto frontier solutions for the decision maker to select a preferable robust and/or stable schedule.

In addition, the current work did not consider material handling. Such consideration is of quite importance for any manufacturing systems as deciding the most appropriate type of the material handling will help in determining the overall production cost. Furthermore, it helps in the design aspect of the shop floor. Hence, the material handling can be added to the hGA by changing the chromosome representation so that each gene is represented by four digits instead of three. The fourth digit represents the material handling assignment to the operation represented by that gene.

REFERENCES

Al-Turki, U.M., Mittenthal, J. and Raghavachari, M. (1996) 'The single-machine absolute-deviation early-tardy problem with random completion times', *Naval Research Logistics*, Vol. 43, pp. 573–587.

Al-Hinai, N. and ElMekkawy, T. (10-12 Feb 2009) 'A robust genetic algorithm approach for the flexible job-shop scheduling problem', *PEDAC'2009*, Alexandria, Egypt.

Al-Hinai, N. and ElMekkawy, T. (2011a) 'An efficient hybridized genetic algorithm architecture for the flexible job-shop scheduling problem', *Flexible Services and Manufacturing Journal*, Vol. 23, pp. 64-85, doi: 10.1007/s10696-010-9067-y.

Al-Hinai, N. And ElMekkawy, T.Y (2011b) 'Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm', *International Journal of Production Economics*, Vol. 132, pp. 279-291, doi: 10.1016/j.ijpe.2011.04.020.

Al-Hinai, N. And ElMekkawy, T.Y (2011c) 'Robust scheduling of flexible job shop with processing time uncertainty: A comparison study', *Submitted to the Journal of Manufacturing Systems*, June 2011, Manuscript No. SMEJMS-D-11-00123.

Anglani, A., Grieco, A., Guerriero, E. and Musmanno, R. (2005) 'Robust scheduling of parallel machines with sequence-dependent set-up costs', *European Journal of Operational Research*, Vol. 161; pp. 704–720.

Artigues, C., Michelon, P. and Reusser, S. (2003) 'Insertion techniques for static and dynamic resource-constrained project scheduling', *European Journal of Operational Research*, Vol. 149, pp. 249-267.

Aytug, H., Lawley, M.A., McKay, K., Moha, S., and Uzsoy, R. (2005) 'Executing Production Schedules in the Face of Uncertainties: A Review and Some Future Directions', *European Journal of Operational Research*, Vol. 161, pp. 86-110.

Barnes, J.W., and Chambers, J.B. (1996) 'Flexible job shop scheduling by tabu search', Graduate Program in Operations Research and Industrial Engineering, The University of Texas, Austin, Technical Report Series, ORP 96-09.

Babiceanu, R.F., and Chen, F.F. (2006) 'Development and applications of holonic manufacturing systems: A survey', *Journal of Intelligent Manufacturing*, Vol. 17, pp. 111-131.

Bierwirth, C. (1995) 'A generalized permutation approach to job shop scheduling with genetic algorithm', *OR Spektrum*, Vol. 17, pp. 87-92.

Bona, B., Brandimarte P., Greco, C., and Menga, G. (1990) 'Hybrid hierarchical scheduling and control systems in manufacturing', *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 6, pp. 673-686.

Bouyahia, Z., Bellalouna, M., Jaillet, P., and Ghedira, K. (2009) 'A priori parallel machines scheduling', *Computers & Industrial Engineering*, doi: 10.1016/j.cie.2009.11.009

Branke, J. (1998) 'Creating robust solutions by means of evolutionary algorithms', *Parallel Problem Solving from Nature V – PPSN V*, LNCS, Vol. 1498/1998, pp. 119-128.

Branke, J. (2001) 'Reducing the sampling variance when searching for robust solutions', In: Spector L et al. (ed), *GECCO 2001 – Proceedings of the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann Publishers, pp. 235-242.

Brandimarte, P. (1993) ‘Routing and scheduling in a flexible job shop by tabu search’, *Annals of Operations Research*, Vol. 41, pp. 157–183.

Brucker, P. and Neyer, J (1998) ‘Tabu-search for the multi-mode job-shop problem’, *OR Spektrum*, Vol. 20, pp. 21-28.

Byeon, E., Wu, S.D. and Storer, R.H. (1998) ‘Decomposition heuristics for robust job-shop scheduling’, *IEEE Transactions on Robotics and Automation*, Vol. 14, No.2, pp. 303-313.

Cai, X. and Tu, F.S. (1996) ‘Scheduling jobs with random processing times on a single machine subject to stochastic breakdowns to minimize early-tardy penalties’, *Naval Research Logistics*, Vol. 43, pp. 1127–1146.

Cavalieri, S. and Terzi, S. (2006) ‘Proposal of a performance measurement system for the evaluation of scheduling solutions’, *International Journal of Manufacturing Technology and Management*, Vol. 8, No. (1/2/3), pp. 248-263.

Chan, F.TS., Chung, S.H., and Chan, P.LY. (2006) ‘Application of genetic algorithms with dominant genes in a distributed scheduling problem in flexible manufacturing systems’, *International Journal of Production Research*, Vol. 44, No. 3, pp. 523-543.

Chen, Q. and Luh, J.Y.S., (1993) ‘Generation of Optimum Schedules in Multi-Robot Workcells With High Processing Flexibility’, *Proceedings of The 1993 IEEE/RSJ International Conference on Intelligence Robots and Systems*, Yokohama, Japan, pp. 662-669.

Chen, S.C. and Jeng, M.D., (1995) ‘A heuristic approach based on the state equations of petri nets for FMS scheduling’, *Proceedings of The 1995 IEEE Transactions on Industry Applications*, Piscataway, NJ., USA, pp. 275-281.

Chen, H., Ihlow, J. and Lehmann, C. (1999) 'A genetic algorithm for flexible job-shop scheduling', *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, Detroit, Michigan, Vol. 2, pp. 1120-1125.

Cheng, R., Gen, M., and Tsujimura, Y. (1996) 'A tutorial survey of job-shop scheduling problems using genetic algorithms – I. Representation', *Computers & Industrial Engineering*, Vol. 30, No. 4, pp. 983-997.

Chiu, Y-F, and Fu, L-C, (1997), 'A GA embedded dynamic search algorithm over a petri net model for an FMS scheduling', *Proceedings of the 1997 IEEE International Conference on Robotics & Automation*, Albuquerque, New Mexico, pp. 513-518.

Chtourou, H. and Haouari, M. (2008) 'A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling', *Computers & Industrial Engineering*, Vol. 55, pp. 183-194.

Cowling, P. I., Ouelhadj, D. and Petrovic, S. (2004) 'Dynamic Scheduling of Steel Casting and Milling Using Multi-Agents', *Production Planning & Control*, Vol. 15(2), pp. 178 – 188.

Dauzére-Pérés, S., and Paulli, J., (1997) 'An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search', *Annuals of Operations Research*, Vol. 70, pp. 281–306.

Daniels, R.L., and Kouvelis, P,. (1995) 'Robust scheduling to hedge against processing time uncertainty in single-stage production', *Management Science*, Vol. 41, No. 2, pp. 363-376.

Davenport, A.J. and Beck, J.C. (2002) 'A survey of techniques for scheduling with uncertainty', unpublished manuscript. Available online from

<http://www.eil.utoronto.ca/profiles/chris/gz/uncertainty-survey.ps> or <http://eil.utoronto.ca/profiles/chris/chris.papers.html>

Dooley, K.J. and Mahmoodi, F. (1992) 'Identification of robust scheduling heuristics: Application of Taguchi methods in simulation studies', *Computers & Industrial Engineering*, Vol. 22, No. 4, 359- 368.

Fattahi, P., Saidi Mehrabad, M., and Jolai, F. (2007), 'Mathematical modeling and heuristic approached to flexible job shop scheduling problems', *Journal of Intelligent Manufacturing*, Vol. 18, pp. 331-342.

Fattahi, P., and Fallahi, A., (2010) 'Dynamic scheduling in flexible job shop systems by considering simultaneously efficiency and stability', *CIRP Journal of Manufacturing Science and Technology*, Vol. 2, No. 2, Pages 114-123.

Falkenauer, E., and Bouffouix, S. (1991) 'A genetic algorithm for job shop', *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento-California, pp.824-829.

French, S. (1982) Sequencing and scheduling: An introduction to the mathematics of the job-shop, Ellis Horwood Limited, West Sussex: John Wiley.

Garey MR, Johnson, DS, and Sethi R (1976) 'The complexity of flow shop and job-shop scheduling', *Mathematics of Operations Research*, Vol. 1-2, pp. 117-129.

Gen, M., and Cheng, R. (2000) Genetic algorithms & engineering optimization, Wiley Series in Engineering Design and Automation, John Wiley & Sons.

Giovanni, L.D., and Pezzella, F. (2010) 'An improved genetic algorithm for the distributed and flexible job-shop scheduling problem', *European Journal of Operational Research*, Vol. 200, pp. 395-408.

Girish, B.S., and Jawahar, N. (2008) 'Scheduling job shop associated with multiple routings with genetic and ant colony heuristic', *International Journal of Production Research*, Vol. 99999, Issue 2, pp. 1-27, (doi: 10.1080/00207540701824845).

Girish, B.S., and Jawahar, N. (2009) 'A particle swarm optimization algorithm for flexible job shop scheduling problem', *5th Annual IEEE Conference on Automation Science and Engineering*, Bangalore, India, August 22-25, pp. 298-303.

Gilkinson, J.C., Rabelo, L.C., and Bush, B.O. (1995) 'a real-world scheduling problem using genetic algorithms', *Computers in Industrial Engineering*, Vol. 29, No. (1-4), pp. 177-181.

Gören, S. (2002) *Robustness and stability measures for scheduling policies in a single machine environment.* MSc Thesis. Department of Industrial Engineering, The Institute of Eng. and Sci., Bilkent University, Ankara , Turkey.

Guo, B. and Nonaka Y (1999), 'Rescheduling and optimization of schedules considering machine failure', *International Journal of Production Economics*, Vol. 60-61, pp. 503-513.

Hart, E., Ross, P., and Corne, D., (2005) 'Evolutionary scheduling: A review', *Genetic Programming and Evolvable Machines*, Vol. 6, pp. 191-220.

Herroelen, W., and Leus, R. (2005) 'Project Scheduling under uncertainty: Survey and research potentials' *European Journal of Operational Research*, Vol. 165, pp. 289-306.

Hmida, AB., Haouari, M., Huguet, M-J., and Lopez, P. (2010) 'Discrepancy search for the flexible job shop scheduling problem', *Computers & Operations Research*, Vol. 37, pp. 2192-2201.

Ho, NB., and Tay, JC. (2004) 'GENACE: An efficient cultural algorithm for solving the flexible job-shop problem', *Proceedings of the Congress on Evolutionary Computation CEC2004*, pp. 1759–1766.

Ho, NB., Tay, JC., and Lai, E M-K. (2007) 'An effective architecture for learning and evolving flexible job-shop schedules', *European Journal of Operational Research*, Vol. 179, pp. 316-333.

Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.

Hurink, E., Jurisch, B., and Thole, M., (1994) 'Tabu search for the job shop scheduling problem with multi-purpose machines', *Operations Research-Spektrum*, Vol. 15, pp. 205-215.

Hussain, MF., and Joshi, SB. (1998) 'A genetic algorithm for job shop scheduling problem with alternate routing', *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 3, pp. 2225-2230.

Jensen, M.T. (2001a) *Robust and flexible scheduling with evolutionary computation*. PhD Thesis, Dept. Comput. Sci., University of Aarhus, Aarhus, Denmark.

Jensen, M.T. (2001b) 'Improving robustness and flexibility of tardiness and total flow-time job shops using robustness measure', *Applied Soft Computing*, Vol. 1, pp. 35-52.

Jensen, M.T. (2003) 'Generating robust and flexible job shop schedules using genetic algorithms', *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 3, pp. 275-288.

Jia, HZ., Nee, AYC., Fuh, JYH., and Zhang, YF. (2003) 'A modified genetic algorithm for distributed scheduling problems', *Journal of International Manufacturing*, Vol. 14, 351–362.

Kacem, I., Hammadi, S., and Borne, P., (2002a) 'Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems', *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 32-1, pp. 1-13.

Kacem, I., Hammadi, S., and Borne, P., (2002b) 'Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic', *Mathematics and Computers in Simulation*, Vol. 60, No. 3, pp. 245-276.

Kacem, I. (2003) 'Genetic algorithm for the flexible job-shop scheduling problem', *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 4, pp. 3464 – 3469.

Kouvelis, P., Daniels, R. L. and Vairaktarakis, G. (2000) 'Robust scheduling of a two-machine flow shop with uncertain processing times', *IIE Transactions*, Vol. 32, 421-432.

Kumar, R., Tiwari, MK., and Shankar, R. (2003) 'Scheduling of flexible manufacturing systems: An ant colony optimization approach', *IMechE*, Vol. 217(B), pp. 1443-1453, (doi: 10.1243/095440503322617216).

Kutanoglu, E. and Wu, S.D. (1998) 'Improving schedule robustness via stochastic analysis and dynamic adaptation', *IMSE Technical Report 98T-001.*

Kutanoglu, E. and Wu, S.D. (2004) 'Improving scheduling robustness via preprocessing and dynamic adaptation', *IIE Transactions*, Vol. 36, No. 11, pp. 1107 – 1124.

Lee, DY., and DiCesare, F., (1994) 'Scheduling flexible manufacturing systems using petri nets and heuristic search', *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 2, pp. 123 – 132

Lee, EJ., and Mirchandani, PB. (1988) 'Concurrent routing, sequencing, and setups for a two-machine flexible manufacturing cell', *IEEE Journal of Robotics and Automation*, Vol.4, No. 3, pp.256-264.

Liu, H., Abraham, A., and Wang, Z., (2009) 'A multi-swarm approach to multi-objective flexible job-shop scheduling problems', *Fundamenta Informaticae,* Vol. 95, pp. 465–489.

Ling, QI., Jian-dong, Y., Bao, LI., and Han-cheng, YU., (2010) 'Flexible job-shop scheduling problem based on adaptive ant colony algorithm' *Journal of Mechanical and Electrical Engineering*, (doi: CNKI:SUN:JDGC.0.2010-02-016)

Lambrechts, O., Demeulemeester, E. and Herroelen, W. (2008) 'A tabu search procedure for developing robust predictive project schedules', *International Journal of Production Economics*, Vol. 111, pp. 493-508.

Laslo, Z., Golenko-Ginzburg, D. and Keren, B. (2008) 'Optimal booking of machines in a virtual job-shop with stochastic processing times to minimize total machine rental and job tardiness costs', *International Journal of Production Economics*, Vol. 111, pp. 812-821.

Lawrence, S.R. and Sewell, E.C. (1997) 'Heuristic, optimal, static, and dynamic schedules when processing times are uncertain', *Journal of Operations Management*, Vol. 15, pp. 71-82.

Lee, D.Y. and DiCesare, F. (1994) 'Scheduling flexible manufacturing systems using petri nets and heuristic search', *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 2, pp. 123 – 132.

Leon, V. J., Wu, S.D. and Storer, R. H. (1994) 'Robustness measures and robust scheduling for job shops', *IIE Transactions*, Vol. 26, No. 5, pp. 32-43.

Levenshtein VI (1966) 'Binary codes capable of correcting deletions, insertions, and reversals', *Soviet Physics – Doklady*, Vol. 10, pp. 707-710.

Liu, N., Abdelrahman, M.A. and Ramaswamy, S. (2007a) 'A complete multiagent framework for robust and adaptable dynamic job shop scheduling', *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, Vol. 37, No. 5, pp. 904-916.

Liu, L., Gu, H. and Xi, Y. (2007b) 'Robust and stable scheduling of a single machine with random machine breakdowns', *International Journal of Advanced Manufacturing Technology*, Vol. 31, pp. 645-654.

Mastrolilli, M., and Gambardella, LM., (2000) 'Effective neighbourhood functions for the flexible job shop problem', *Journal of Scheduling*, Vol. 3, pp. 3-20.

Mattfeld, DC (1996) Evolutionary search and the job shop: investigations on genetic algorithms for production scheduling, Germany, Heidelberg: Physica-Verlag.

Mahdavi, I., Shirazi, B. and Solimanpur, M. (2010) 'Development of a simulation-based decision support system for controlling stochastic flexible job shop manufacturing systems', *Simulation Modelling Practice and Theory*, Vol. 18, No. 6, pp. 768-786.

Matsveichuk, N.M., Sotskov, Yu.N., Egorova, N.G. and Lai, T.-C. (2009) 'Schedule execution for two-machine flow-shop with interval processing times', *Mathematical and Computers Modelling*, Vol. 49, pp. 991-1011.

Mellor, P., (1966) 'A review of job shop scheduling', *Operation Research Quarterly*, Vol. 17, No. 2, pp. 161-171.

Mesghouni, K., Hammadi, S., and Borne, P., (1997) 'Evolution programs for job-shop scheduling', *IEEE International Conference on Systems, Man, and Cybernetics*, Orlando, Florida, Vol. 1, pp. 720-725.

Mehta, S.V. and Uzsoy, R.M. (1998) 'Predictable scheduling of a job shop subjected to breakdowns', *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 3, pp. 365-378.

Mirchandani P., Lee, EJ., and Vasquez A. (1988) 'Concurrent scheduling in flexible automation', *Proceedings of the 1988 IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 2(8-12), pp.868-872.

Montemanni, R. (2007) 'A mixed integer programming formulation for the total flow time single machine robust scheduling problem with interval data', *Journal of Mathematical Modelling Algorithms*, Vol. 6, pp. 287-296.

Möhring, R. H., Radermacher, F. J. and Weiss, G. (1985) 'Stochastic scheduling problems II: Set strategies', *ZOR – Zeitschrift für Operations Research*, Vol. 29, pp. 65–104.

Mula, J., Poler, R., García-Sabater, J.P., and Lario, F.C. (2006) 'Models for production planning under uncertainty: A review', *International Journal of Production Economics*, Vol. 103, pp. 271-285.

Murovec, B., and Šuhel, P., (2004) 'A repairing technique for the local search of the job-shop problem', *European Journal of Operational Research*, Vol. 153, pp. 220-238.

Najid, NM., Dauzére-Pérés, S., and Zaidat, A. (2002) 'A modified simulated annealing method for flexible job shop scheduling problem', *Oct 2002 IEEE International Conference on Systems, Man and Cybernetics*, Vol.5, No. 6, pp. 1-6.

Paulli, J. (1995) 'A hierarchical approach for the FMS scheduling problem', *European Journal of Operational Research*, Vol. 86, No. 1, pp. 32–42.

Pezzella, F., Morganti, G., and Ciaschetti, G. (2008) 'A genetic algorithm for the flexible job-shop scheduling problem', *Computers & Operations Research*, Vol. 35, pp. 3202-3212.

Pinedo, M. (1982) 'On the computational complexity of stochastic scheduling problems', *In M. Dempster, J. Lenstra, & A. R. Kan (Eds.), Deterministic and stochastic scheduling. Dordrecht: Reidel*.

Pinedo, M., (2002) Scheduling: Theory, Algorithms and Systems, Prentice-Hall, Englewood cliffs, NJ.

Policella, N., Oddi, A., Smith, S.F. and Cesta, A. (2004) 'Generating robust partial order schedules', *M. Wallace (Ed.): CP 2004, Lecture Notes in Computer Science*, Vol. 3258, pp. 496-511.

Policella, N., Cesta, A., Oddi, A. and Smith, S.F. (2005) 'Schedule robustness through broader solve and robustify search for partial order schedules', *S. Bandini and S. Manzoni (Eds.): AI*IA 2005, Lecture Notes in Artificial Intelligence*, Vol. 3673, pp. 160-172.

Qi, X., Bard, J.F., and Yu, G. (2006) 'Disruption management for machine scheduling: The case of SPT schedules', *International Journal of Production Economics*, Vol. 103, pp. 166-184.

Rangsaritratsamee, R., Ferrell, W.G., and Kurtz, M.B. (2004) 'Dynamic rescheduling that simultaneously considers efficiency and stability', *Computers & Industrial Engineering*, Vol. 46, No. 1, pp. 1-15.

Reddy, J.P., Kumanan, S. and Chetty, O.V.K., (2001) 'Application of petri nets and a genetic algorithm to multi-mode multi-resource constrained project scheduling', *International Journal of Advanced Manufacturing Technology*, Vol. 17, pp. 305-314

Rutherford, A. (2001) Introducing ANOVA and ANCOVA: A GLM Approach, SAGE Publications Ltd, Athenaeum Press, Gateshead.

Sabuncuoglu, I. and Karabuk, S. (1999) 'Rescheduling frequency in an FMS with uncertain processing times and unreliable machines', *Journal of Manufacturing*, Vol. 18, No. 4, pp. 268-283.

Sevaux, M. and Sörensen, K. (2004) 'A genetic algorithm for robust schedules in a one-machine environment with ready times and due dates', *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 4OR 2; pp. 129–147.

Shafaei, R. and Brunn, P. (1999) 'Workshop scheduling using practical (inaccurate) data Part 2: An investigation of the robustness of scheduling rules in a dynamic and stochastic environment', *International Journal of Production Research*, Vol. 37, No. 18, pp. 4105 – 4117.

Shafaei, R. and Brunn, P. (2000) 'Workshop scheduling using practical (inaccurate) data Part 3: A framework to integrate job releasing, routing and scheduling functions to

create a robust predictive schedule', *International Journal of Production Research*, Vol. 38, No. 1, pp. 85 – 99.

Sörensen, K. (2001) 'Tabu searching for robust solutions', *Proceedings of the 4th metaheuristics international conference*, Porto, Portugal, pp. 707-712

Surico, M., Kaymak, U., Naso, D. and Dekker, R. (2006) 'Hybrid Meta-Heuristic for Robust Scheduling', *ERIM Report Series Reference No. ERS-2006-018-LIS*, Available at SSRN: http://ssrn.com/abstract=902747

Subramaniam, V. and Raheja, A.S. (2003) 'mAOR: A heuristic-based reactive repair mechanism for job shop schedules', *International Journal of Advanced Manufacturing Technology*; Vol. 22, No. (9-10), pp. 669-680.

Tay, JC., and Wibowo, D., (2004) 'An effective chromosome representation for evaluating flexible job shop schedules', *Genetic and Evolutionary Computation (GECCO 2004)*, Vol. 3103, pp. 210-221.

Tsutsui, S. and Ghosh, A. (1997) 'Genetic algorithms with robust solution search scheme', *IEEE Transactions in Evolutionary Computation*, Vol. 1, No. 3, pp. 201-208.

Tsutsui, S. (1999) 'A comparative study on the effects of adding perturbations to phenotypic parameters in genetic algorithms with a robust solution searching scheme', *Proceedings of the 1999 IEEE systems, man, and cybernetics conference (SMC'99 Tokyo)*, pp. III-585-591.

Vinod, V. and Sridharan, R. (2009) 'Development and analysis of scheduling decision rules for a dynamic flexible job shop production system', *International Journal of Business Performance Management*, Vol. 11, No. 1/2, pp. 43-71.

Vinod, V. and Sridharan, R. (2011) 'Simulation modeling and analysis of due-date assignment methods and scheduling decision rules in a dynamic job shop production system', *International Journal of Production Economics*, Vol. 129, pp. 127-146.

Wang, J., (1998) Timed Petri Nets: Theory and Application, Kluwer Academic Publishers, Boston, ISBN: 0-7923-8270-6.

Wei, Q., and Qiaoyun, L., (2009) 'Solving the flexible job shop scheduling problem based on the adaptive genetic algorithm', *2009 International Forum on Computer Science-Technology and Applications*, Vol. 1, pp. 97-100.

White, T., and Oppacher, F., (1994) 'Adaptive crossover using automata', *Lecture Notes in Computer Science*, Springer Berlin – Heidelberg, Vol. 866, pp. 229-238.

Wu, D.S., and Wysk, R.A., (1989) 'An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing', *International Journal of Production Research*, Vol. 27, pp. 1603-1624.

Wu, D.S., Storer, R.H., and Chang, P.C., (1993) 'One-Machine rescheduling heuristics with efficiency and stability as criteria', *Computers in Operations Research*, Vol. 20, pp. 1-14.

Wu, S.D., Byeon, E., and Storer, R.H., (1999) 'A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness', *Operations Research*, Vol. 47, No. 1, pp. 113-124.

Wu, C.W., Brown, K.N. and Beck, J.C. (2009) 'Scheduling with uncertain durations: Modeling β-robust scheduling with constraints', *Computers & Operations Research*, Vol. 36, pp. 2348-2356.

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

Xia, Y., Chen B. and Yue J. (2008) 'Job sequencing and due date assignment in a single machine shop with uncertain processing times', *European Journal of Operational Research*, Vol. 184, pp. 63-75.

Xing, L-N., Chen,Y-W., and Yang, K-W., (2009) 'An efficient search method for multi-objective flexible job shop scheduling problems', *Journal of Intelligent Manufacturing*, Vol. 20, pp. 283-293.

Xing, L-N., Chen, Y-W., Wang, P., Zhao, Q-S., and Xiong, J., (2010) 'A knowledge-based ant colony optimization for flexible job shop scheduling problems', *Applied Soft Computing*, Vol. 10, pp. 888–896.

Xu, D-S., Ai, X-Y., and Xing, L-N., (2009) 'An improved ant colony optimization for flexible job shop scheduling problems', *Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization*, Vol. 1, pp. 517-519.

Yang, J-B., (2001) 'GA-based discrete dynamic programming approach for scheduling in FMS environments', *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, Vol. 31, No. 5, pp. 824-835.

Zhang, GH., Shao, XY., Li, PG., and Gao, L., (2009) 'An effective hybrid particle swarm optimization algorithm for multiobjective flexible job-shop scheduling problems', *Computers and Industrial Engineering*, Vol. 56, Issue 4, pp. 1309-1318.

Zribi, N., Kacem, I., and Kamel, AE., (2007) ‘Assignment and scheduling in flexible job-shops by hierarchical optimization’, *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, Vol. 37, No. 4, pp. 652-661.