Robust Dynamic and Stochastic Scheduling In Permutation Flow Shops

Mohanad Riyadh Saad AL-Behadili

A thesis submitted for the degree of
Doctor of Philosophy

The Logistics, Operational Research and Analytics Group
Department of Mathematics
University of Portsmouth

January 2018

Declaration

Whilst registered as a candidate for the above degree, I have not been registered for any other research degree award. The results and conclusions embodied in this thesis are the work of the named candidate and have not been submitteed for any other academic award.

Mohanad Riyadh Saad AL-Behadili

January 2018

Acknowledgements

I would like to extend thanks to the many people, in different countries, who so generously contributed to the work presented in this thesis. I would express my heartfelt gratefulness to my family. Without the support and encouragement of them, I would never have had the ability to finish this thesis, and for their belief in me I will be forever grateful and thankful. To my wonderful parents, they have been there for me all the time. Thank you for all you have done for me. To my wife and daughters, I am so sorry for the long after-work hours necessary to complete this thesis, but I am glad I was able to make you proud.

I would also like to thank my supervisors, Professor Djamila Ouelhadj and Professor Dylan Jones for their valuable support, motivation, encouragement, insightful comments, enthusiasm and continuous guidance during the development of this thesis. They were always open to discussing and clarifying different concepts and made me feel part of the department. Similar, profound gratitude to Professor Juan Angel at the Open University of Catalonia (Spain) and Professor Rubén Ruiz at Polytechnic University of Valencia (Spain) for sharing their knowledge, I cannot thank them enough for that.

I would also like to thank my sponsor, the Department of Mathematics, College of Science, University of Basra in Iraq, not only for providing the funding which allowed me to undertake this research, but also for the support provided by the staff members of the Department during the whole period of study.

Last, but not the least, I would like to thank my fabulous friends in the office for keeping me smiling, the fantastic staff in the of the Department of Mathematics and other people at the University of Portsmouth for their support and friendship, I am also indebted to them for their help, and everyone else who has been supportive of my work so far.

Dedication

This thesis is dedicated to the souls of brave men and women who defend my country against the terrorism.

Abstract

The Permutation Flow Shop Scheduling Problem (PFSP) is a fundamental problem underlying many operational challenges in the field of logistic and supply chain management. The PFSP is a well-known NP-hard problem whereby the processing sequence of the jobs is the same for all machines. The dynamic and stochastic PFSP arise in practice whenever a number of different types of disruptions or uncertainties interrupt the system. Such disruptions could lead to deviate the disrupted schedule from its initial plan. Thus, it is important to consider different solution methods including: an optimisation model that minimise different objectives that take into account stability and robustness, efficient rescheduling approach, and algorithms that can handle large size and complex dynamic and stochastic PFSP, under different uncertainties. These contributions can be described as follows:

1. Develop a multi-objective optimisation model to handle different uncertainties by minimising three objectives namely; utility, instability and robustness.

2. Propose the predictive-reactive approach to accommodate the unpredicted uncertainties.

3. Adapt the Particle Swarm Optimisation (PSO), the Iterated Greedy (IG) algorithm and the Biased Randomised IG algorithm (BRIG) to reschedule the PFSP at the reactive stage of the predictive-reactive approach.

4. Apply the Simulation-Optimisation (Sim-Opt) approach for the Stochastic PFSP (SPFSP) under different uncertainties. This approach consists of two methods, which are: the novel approach that hybridise the Monte Carlo Simulation (MCS) with the PSO (Sim-PSO) and the Monte Carlo Simulation (MCS) with the BRIG (Sim-BRIG).

The main aim of using the multi-objective optimisation model with different solution methods is to minimise the instability and keep the solution as robust as possible. This is to handle uncertainty as well as to optimise against any worst instances that might arise due to data uncertainty. Several approaches have been proposed for the PFSP under dynamic and stochastic environments, where the PSO, IG and BRIG are developed for the PFSP under different uncertainties. Also, hybridised the PSO and the BRIG algorithms with the MCS to deal with

SPFSP under different uncertainties. In our version of the approach, the first one is a PSO algorithm step after which an MCS is incorporated in order to improve the final solutions of problem. The second approach proposed the hybridisation of the BRIG algorithm with MCS to be applied on the SPFSP under different uncertainties. The developed multi-objective model and proposed approaches are tested on benchmark instances proposed by (Katragjini et al., 2013) in order to evaluate the effectiveness of the proposed methodologies, this benchmark is based on the well-known instances of Taillard's (Taillard, 1993). The computational results showed that the proposed methodologies are capable of finding good solutions for the PFSP under different uncertainties and that they are robust for the dynamic and stochastic nature of the problem instances. We computed the best solutions and found that they could be highly promising in minimising the total completion time. The results obtained are quite competitive when compared to the other models found in the literature. Also, some proposed algorithms show better performance when compared to others.

Table of contents

List of figures

List of tables

Glossary of Symbols and Abbreviations

AIS Artificial Immune System

B&B Branch and Bound

BRIG Biased Randomised Iterated Greedy

BRNEH Biased Randomised NEH

BR Biased Randomised

COP Combinatorial Optimisation Problems

DDT Discretised Decreasing Triangular Distribution

DE Differential Evolution

FFSP Flexible Flow shop Scheduling Problem

FJSP Flexible Job Shop Scheduling Problem

FSP Flow Shop Scheduling Problem

GA Genetic Algorithm

GC Greedy Constructive

IG Iterated Greedy

ILS Iterated Local Search

IPG Iterated Pareto Greedy

JSP Job Shop Scheduling Problem

LS Local Search

MA Memetic Algorithm

MCS Monte Carlo Simulation

MILP Mixed Integer Linear Programming

MSA Memetic Search Algorithm

MSR Our proposed Multi-Objective Model

PFSP Permutation Flow shop Scheduling Problem

PMSP Parallel Machine Scheduling Problem

PSO Particle Swarm Optimisation

SA Simulated Annealing

Sim-BRIG Simulation-Biased Randimised Iterated Greedy

Sim-Opt Simulation-Optimisation

Sim-PSO Simulation-Particle Swarm Optimisation

SMSP Single Machine Scheduling Problem

SPFSP Stochastic Permutation Flow shop Scheduling Problem

TFT Total Flow Time

TS Tabu Search

TWT Total Weighted Tardiness

VN Variable Neighbourhood

VRP Vehicle Routing Problem

Chapter 1

Background and motivation

1.1 Introduction

Scheduling is a decision-making process that is vital in many manufacturing and services industries. It deals with the assignment of a set of jobs to a set of machines in a reasonable amount of time with the goal of optimising one or more objectives (Pinedo, 2016). Scheduling problems are classified into different types of problems (Pinedo, 2016). One of the most important scheduling problems is the PFSP, in this problem it does not allow for the job sequence to change between machines. Because of the priority of the PFSP, we will consider this problem to be the subject of study in this thesis. There are also different categories of scheduling problems environments, these are; static, dynamic and stochastic scheduling problems (Jarboui et al., 2013). The real-life scheduling problems in manufacturing systems are dynamic and stochastic in nature. Due to the importance of dynamic and stochastic scheduling in real practical life, researchers addressed the nature of the gap between the scheduling theory and scheduling practice. There was a considerable gap until the late 1980s before interest in the subject was rekindled. Considerable studies have been done in the last forty years for scheduling problems under dynamic and stochastic environments. In dynamic, stochastic manufacturing environments, managers, production planners, and supervisors must not only generate high-quality schedules, but also react quickly to unexpected events and subsequently revise schedules in a cost-effective manner. These events are generally difficult to take into consideration while generating a schedule, disturbing the system and generating considerable differences between the predetermined schedule and its actual realisation on the shop floor. Rescheduling is then practically mandatory in order to minimise the effect of such disturbances in the performance of the system. There are many types of disturbances that can upset the plan. Rescheduling is the process of updating an existing production schedule in response

to disruptions or other changes. The following is a partial list of possible disruptions among others:

- New (urgent) job Arrival.
- Cancellation of a job, change to a job's due date, or other change in job specification.
- Machine breakdown, repair, or other failure in status.
- Delay in the arrival of required material or other problem with material delivery.
- Absentee workers or changes to worker assignments.
- Incorrect predictions of setup time, processing time, or other actions.
- Poor quality parts that require rework or manufacture of new parts.

There exists a vast variety of solution methods that have been proposed for a large range of scheduling problems (Pinedo, 2016). In particular, different solution methods have been proposed for dynamic and stochastic scheduling problems. Mathematical optimisation models have been used widely as solution method along with the exact and approximate techniques to solve the PFSPs. However, PFSPs in real shop floor mainly operates in highly dynamic and stochastic environments, where there are different real-time events and uncertainties that could lead to the schedule deviating from its initial plan, and therefore a previously feasible schedule may turn infeasible when it is released to the shop floor. Such a schedule is defined as schedule nervous (Steele, 1975), or often referred to as schedule instability. The instability could be disconcerting to production schedulers who often find that changes come faster than they could effectively respond to. Now, after more than 40 years of Steele's publication on this issue, schedule instability is still an ongoing issue both in real practice as well as in academic research despite the significant advancement of scheduling systems. For this, it is very important to consider optimisation models that aim to reduce instability, robustness and also utility (depending on the problem objectives). Rescheduling is one of main procedures that are used to accommodate the dynamic disruptions. It uses different approaches typical to the problem environments and the disruption types. The most well-known efficient approach used in the dynamic scheduling is the predictive-reactive approach, which is considered in this thesis. This approach is triggered at the time of disruption, and it uses any suitable algorithm at the reactive stage in order to accommodate the disruptions. Regarding the solution algorithms, exact or complete methods are the first proposed methods for different scheduling problems under dynamic real-time events or stochastic uncertainties, these methods have mainly concentrated on finding a guaranteed optimal solution for every instances of finite size in a specific time.

The most proposed well-known exact method for different scheduling problem are Lagrangian relaxation, dynamic programming, Branch and Bound (B&B), and Branch and Cut. Since the PFSP belongs to the class of NP-hard (mathematically intractable) problems (Graham et al., 1979), the computational complexity of the scheduling problem has special attention in the literature of scheduling. It is defined as a maximum number of computational steps required to reach an optimal solution. According to the concept of complexity, it may not be possible to find an optimal solution using the classical algorithms such as exact methods for medium or large scale problems of NP-hard class, as is the case of scheduling problems (in PFSP, medium instances size ranging from 50×5 to 100×20 and large size problems ranging from 200×10 to 500×20). Hence, alternative methods are proposed, such as; heuristics, metaheuristics, and so on. Regarding the stochastic scheduling problems, there are different techniques that have been used in the literature. Recently, the Sim-Opt methods have been applied successfully in many Combinatorial Optimisation Problems (COP), more precisely, in the SPFS area. In this thesis, we consider the dynamic and stochastic PFSP under different types of uncertainties. To solve this problem, we develop a multi-objective optimisation model that consider utility, stability and robustness and proposed the predictive-reactive approach with different efficient heuristics, metahueristics and Sim-Opt methods.

1.2 Definition of scheduling in a manufacturing system

Scheduling is the main key for most service and manufacturing systems. It is convenient to adopt manufacturing terminology with the definition of scheduling, where jobs represent activities and machines represent resources, while the range of application areas for scheduling theory are not limited to manufacturing but are extensive. Some of the realistic situations in which scheduling problems exist are:

- Technological planning of how the jobs should be completed in a manufacturing unit.
- Scheduling of aircraft waiting for landing clearance.
- Ordering of jobs for processing in a manufacturing plant.
- Scheduling of jobs under rental conditions in a non-deterministic environment.
- Scheduling of patients waiting in a hospital for different types of tests.

Currently, manufacturing services are facing new challenges for example, shorter product life cycles, changes in market demand, global competition, and so on. It is crucial for the manufacturing industries to improve the performance of their production scheduling systems

under different internal and external uncertainties such as job cancellation, new job arrivals, machine breakdown, stochastic processing times, and so on. Scheduling solution methods are very important to reduce the production cost in a manufacturing procedure in order to keep the company in the forefront of the competitive environment. Different scheduling approaches are required to allocate jobs to machines, when the manufacturing process experience a lack of resources and limited execution time or are facing different disruptions. It is vital for industries to meet the deadline committed to a customer in order to prevent failure, which may lead to a loss in customer satisfaction. Therefore, the industries are required to schedule tasks in the shop floor in an efficient method. The combinatorial scheduling problems belong to the class of representatives of problems. Thus, they are seeking a local optimal solution in the finite set of potential solutions. Production scheduling in manufacturing systems is continually assessed so as to manufacture reliable and high-quality merchandises at the given time and without any delays. These objectives can be achieved by manufacturers relying on some tools such as shop floor scheduling process, which is considered as the most substantial factor in the planning of manufacturing systems (Suwa & Sandoh, 2013). The scheduling problem is employed for different applications of technology and human resources to fulfill customer's demands. This function must organise the simultaneous execution of several activities while accounting for constraints on available resources. According to the shop floor conditions, jobs and machines perhaps take various shapes. The scheduling problem could have different formulations depending on the type of the problem, the sets of jobs, machines, the range of resources and the performance criteria during the optimisation process. For performance criteria, there are different performance measures that are employed to optimise schedules. For example, the objective function may consider reducing the total completion time to complete a sequence of jobs, also the objective function may minimise the Total Weighted Tardiness (TWT), and so on.

1.3 Classification of scheduling problems

As scheduling is the main key for manufacturing systems, it also plays an important role in most information processing environments. According to Chryssolouris (2006), there are four dimensions to classifying scheduling problems as follows:

- Requirement generations.
- Processing complexity.
- Scheduling criteria.

- Scheduling environment.

The first dimension, is referred to as the distinction between what is called an open shop versus a closed shop requirements generation. The second dimension, processing complexity, is concerned primarily with the number of processing steps associated with each production task or item. Scheduling criteria are measures by which schedules are to be evaluated, and may be classified broadly into schedule costs and schedule performance measures. The last dimension is the scheduling environments. A wide range of classification of scheduling problem models are introduced according to their environment nature. The scheduling environment is an important component of the rescheduling framework, which is to identify the set of jobs that need to be scheduled. The classifications of scheduling according to the problem environments are as follows:

1. **Static scheduling**
   The scheduling problems in which the nature of job arrival is different and a set of jobs over time does not change are called static scheduling problems. The setup times of jobs are available beforehand. In other words, the scheduling problems when all elements of the problems such as the arrival state of jobs at a shop floor, due date of jobs, ordering, processing time, availability of machines etc. do not include stochastic factor and are determined in advance are included in this category. Scheduling is called deterministic if all the attributes needed for constructing a schedule take constant values and they are known in advance.

2. **Dynamic scheduling**
   The problem of scheduling in the presence of real-time events, termed dynamic scheduling (Ouelhadj & Petrovic, 2008). An example for dynamic scheduling problem where a set of jobs changes over time and arrival rate of jobs is different. In other words, random disruptions may interrupt the system, which could change the scheduling plans. The schedule, which is actually executed on the shop floor, is called the realised (actual) schedule. This schedule may substantially differ from the initial schedule, depending on the degree or intensity of disruptions.

3. **Stochastic scheduling**
   The problem is stochastic if some information is not known exactly, i.e. at least one of the problem elements includes a stochastic factor. For example, the processing time of jobs are modelled as random variables. The stochastic processing could follow different disruptions depending on the use of models and systems, the following distributions are mainly considered in the literature.

(a) Uniform distribution; A processing time p_{ij} can uniformly be included between two values a and b. Then, p_{ij} follows a uniform distribution over the interval $[a,b]$. This kind of distribution is used to provide a simplified model of real industrial cases. For instance, it has already been used in Gourgand et al. (2010) and Kouvelis et al. (2000).

(b) Exponential distribution. A processing time p_{ij} may follow an exponential distribution. Exponential distributions are commonly used to model random events that may occur with uncertainty. This is typically the case when a machine is subject to unpredictable breakdowns. For example, processing times have been modeled by an exponential distribution in Cunningham & Dutta (1973) and Ku & Niu (1986) among others.

(c) Normal distribution. A processing time p_{ij} may follow a normal distribution $N(\mu,\sigma)$ where μ stands for the mean and σ stands for the standard deviation. This kind of distribution is especially usual when human factors are observed. A process may also depend on unknown or uncontrollable factors and some parameters can be described in a vague or ambiguous way by the analyst. Therefore, processing times vary according to a normal distribution Gourgand et al. (2010) and Wang et al. (2005).

(d) Log-normal distribution. A random variable X follows a log-normal distribution with parameters μ and σ if $\log X$ follows a normal distribution $N(\mu,\sigma)$. The log-normal distribution is often used to model the influence of uncontrolled environmental variables. For instance, this modeling has already been used in Dauzére-Pérés et al. (2010).

The scheduling problems are also categorised into the following problems Pinedo (2016):

- **Single Machine Scheduling Problem (SMSP):** This problem is defined as the process of assigning a number of jobs to a single machine.
- **Parallel Machine Scheduling Problem (PMSP):** In this problem, similar type of machines are available in multiple numbers and jobs can be scheduled over these machines simultaneously.
- **Flow Shop Scheduling Problem:** In FSP, there are n jobs where each job has to be processed on a series of m machines such that all jobs have to follow the same route.
- **Job Shop Scheduling Problem (JSP):** In this problem, there are n jobs and m machines where each job has its own predetermined route through the machines to follow.

- **Open Shop Scheduling Problem:** In this case, there are n jobs where each job has to be processed on each one of the m machines. There are no restrictions with regard to the routing of each job through the machine environment, this mean different jobs may have different routes. Also, some of the jobs processing times may be zero.

1.4 Computational complexity of scheduling problems

Computational complexity of a problem is defined as a maximum number of computational steps required to reach an optimal solution. The concept of complexity point out to the computing attempt needed by a solution algorithm. Computing attempt is represented by order-of-magnitude notation. Assume a specific proposed algorithm is employed to find the solution for problem of size n (for PFSP n represents the number of jobs). Then the total number of computations needed by the algorithm is usually restricted by a function of the number of jobs n. When the number of required computations is a polynomial function of n, then the algorithm is polynomial. For example, the order of magnitude function of n^2, which is denoted as $O(n^2)$. On contrary, when the function order of magnitude is not polynomial then the algorithm is called an exponential or non-polynomial. For instance, the order of magnitude function of 2^n is an exponential and it denotes as $O(2^n)$. Depending on the problem complexity in the literature, all problems are classified into **P** (polynomial) class and **NP** (non-polynomial) class. The first type of classes **P** is defined as all problems with the property that the execution time of the solution algorithm increases polynomially with the size of problem. On the other hand, the **NP** class consists of the problems, which are the time required for solution execution is grows exponentially. The algorithms that execution time grows polynomially are more preferred in real practice, since such algorithms obtained the solution in a reasonable time. However, some practical COPs are non-deterministic polynomial-time hard (NP-hard). For example, the scheduling problems are NP-hard (Graham et al., 1979). According to the concept of complexity, it is may not possible to find an optimal solution using the classical algorithms such as exact methods for large scale problems of NP-hard class, as the case of scheduling problems. For this, an alternative methods are proposed such as heuristics, metaheuristics, and so on.

1.5 Performance measures

In scheduling, it is usually difficult to state objectives as there are many complex and often conflicting objectives. The objective functions are called regular performance measures when the functions are non-decreasing in $C_1, ..., C_n$ where C_i, $i = 1, ..., n$ are jobs completion times.

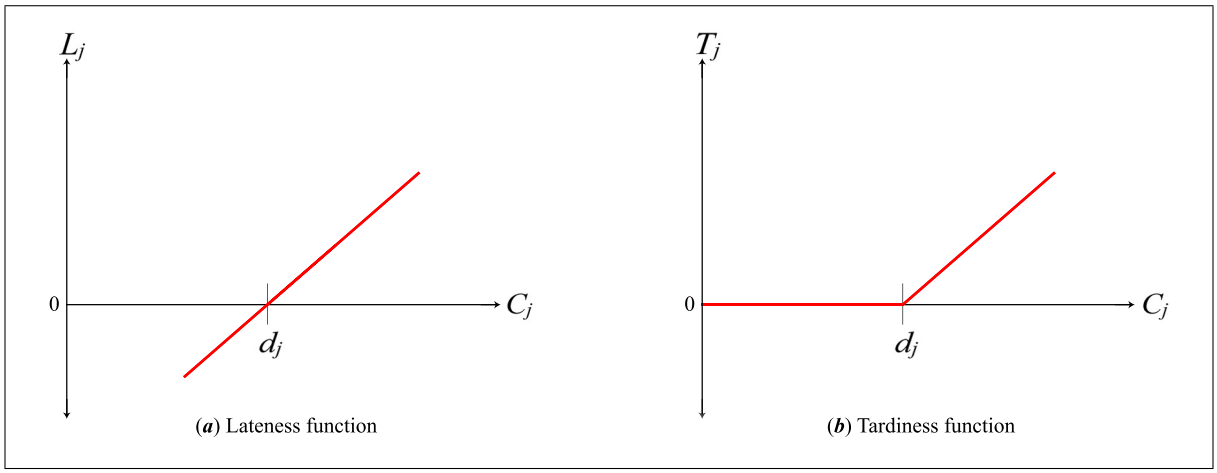
A noticeable number of scheduling problems with regular performance measures have been studied in the literature. Some commonly discussed regular performance measures among others are (Framinan et al., 2014):

1. **Makespan C_{max}:** The makespan is defined as the maximum completion time of the last job completed in the system. It can be seen as the time required to finish the scheduling plan completely since it measures from the time the first job starts processing, which is usually assumed to be zero (unless release times or other constraints exist), to the time the last job in the processing sequence is finished on the last machine it uses. The makespan is considered as one of the most common objective that have been studied in the literature of PFSPs.

2. **Total Completion Time ($\sum_j C_j$):** The sum of the completion times of all jobs is called the total completion time. The performance criteria of this measure is very important for scheduling problems so as to increase the maximum utilization and productivity of resources.

3. **Total Weighted Completion Time ($\sum_j [w_j C_j]$):** It is the sum of the weighted completion times of all jobs. The total weighted completion time is related to maximising system utilization and work in process work-in-process inventory.

4. **Total Weighted Tardiness ($\sum_j [w_j T_j]$):** This is a more general cost function than the total weighted completion time. It is related to job due dates where the tardiness is defined as follows: $T_j = max(0, L_j)$ Where L_j is lateness of job j, and is defined as the difference between the job completion time (C_j) and its due date (d_j), hence $L_j = C_j - d_j$. Tardiness ignores negative lateness values, the tardiness and lateness functions are shown in Figure 1.1.



(*a*) Lateness function (*b*) Tardiness function

Fig. 1.1 Tardiness and Lateness functions

5. **Weighted Number of Tardy jobs ($\sum_j [w_j U_j]$):** This objective has both academic and practical values. It is related to job due dates. For example, late delivery implies a penalty in the form of loss of goodwill and the magnitude of the penalty depends on the importance of the order or the client and the tardiness of the delivery. One of the objectives of the scheduling system is to minimize the sum of these penalties.

In the literature of scheduling, the majority of research addressed only the single objective for scheduling problems. However, the multi-objective performance measures for scheduling problems have considered a lot of attention since 1980 and since then research has reported the case of multi-objective shop scheduling problems. The main reason of considering multi-objective performance measures for scheduling problems is the companies environment nature, which could be conflicting, dynamic and/or stochastic, companies strive to attain multiple performance measures to ensure keeping in a good situation. The multi-objective models of scheduling problem have been considered by different researchers. However, the majority of researches were restricted to two or three objective performance measures. Developing approaches for manufacturing scheduling environments have been given a considerable coverage and effort in the literature. However, only few researches were successful in the practical solving of real life scheduling problems, while the majority of researchers rely on highly theoretical and unrealistic assumptions. Therefore, implementation of such approaches are generally impractical for scheduling problems in manufacturing environments of the real world, which are conflicting, dynamic, stochastic and complex in nature. Actually, the most real manufacturing scheduling problems are subjected to different perturbations because of a vast extent of dynamic and stochastic uncertainties. Some uncertainty examples are; machine breakdowns, new job arrivals, stochastic processing times, job ready times variation, and so on, these disruptions can delay a schedule's completion time. Some disruptions have a major effect on the system performance. For example, machine breakdown is consider as one of the most significant disruptions in shop scheduling problems. To minimse the effect of such uncertainties on the scheduling in manufacturing systems under dynamic and/or stochastic environments, two important measures have been studied in the literature (Cowling et al., 2004) namely; stability and robustness. The stability measure is defined as the schedule that does not deviate the completion time of the unaffected operations from the original schedule in a disrupted situation, while the robustness measure is the schedule performance, which does not deteriorate in a disrupting situation. These two measures have been proposed with the utility (makespan) measure implicitly. The utility, stability and robustness measures have the following assumptions: let n be the number of jobs where the jobs index is $j = 1,2\ldots.n$, and let m be the number of machines where $i = 1,2\ldots.m$ is the machines index. Now the performance measures mentioned can defined as follows:

- Makespan: The most common objective for the PFSP is the minimisation of the maximum total completion time $\sum_j C_j$, this is referred to as makespan. This measure is aiming to indicate the degree of optimisation of the scheduling problem where the completion time is the time at which processing time of last operation at the job j is completed.
- Stability: This measure is to indicate the deviation between the new schedule and the baseline.
- Robustness: This measure is to calculate the difference between the completion time of the baseline and new schedule.

These performance measures are studied in details in this thesis

1.6 Research aims and objectives

As we explained previously, real world manufacturing system usually operate in highly dynamic and uncertain environments, where random disruptions may cause non-optimal performances for scheduling problem. In addition, real world manufacturing scheduling is generally too complex and they are large scale problems. However, the robust dynamic and stochastic scheduling is rarely addressed in the literature, and hence, we aim to consider the gap between scheduling theory and practice and try to narrow it by discussing the dynamic PFSP and SPFSP under different uncertainties. The aim of this thesis can be summarised in the following points:

1. Consider the challenging dynamic PFSP and SPFSP under different uncertainties with the aim of proposing efficient frameworks and solution methods for these problems.
2. Design a multi-objective optimisation model that consider utility, stability and robustness for the PFSP under uncertainties to accommodate the disruptions that effect the schedule plan, in order to prevent the new schedule deviating too much from its initial plan.

Also, the novel research contributions for achieving the aims can be summarised as follows:

- Propose efficient predictive-reactive approach to handle the effect of different real-time events on the scheduling system.
- Propose efficient and simple IG, its randomised version and PSO and develop these methods with the predictive-reactive approach to solve the PFSP with different disruptions.
- Design a Sim-Opt framework by considering the case where the PFSP is stochastic and under different disruptions simultaneously. The propose framework consist of the

integration of MCS with BRIG and then with the PSO to handle dynamic and stochastic uncertainties for the PFSP with the consideration of minimising the utility, stability and robustness simultaneously.

The following two goals are necessary for such algorithms to be efficient methods: exploration and exploitation. The exploration ensures that the majority of areas of the solution space domain are explored well to obtain a good local optimum solution. On the other hand, the exploitation focuses the search direction procedure near the best solutions obtained in order to explore the neighbourhoods of the best found solution to potentially find better solutions. For this, in this thesis we consider more techniques to be used implicitly such as the Nawaz, Enscor, and Ham (NEH) heuristic and Local Search (LS) procedure.

1.7 Organisation of thesis

Eight chapters presented in this thesis are organised as follows:

Chapter 1: Background and motivation

This chapter is structured as follows; the concept of scheduling including basic applications and solution methodology are introduced first then the justification provided, the motivation aim of research and research objectives.

Chapter 2: Literature review and research gap

The purpose of this chapter is to provide a literature review of the base knowledge that is already available about PFSPs in dynamic and stochastic environments. This chapter will also highlight the solution proposed frameworks, methodologies and problems that related to different parts of the PFSPs in uncertain environments. Finally, the recent gap in the literature is presented and the conclusions of this chapter are summarised.

Chapter 3: Multi-objective Optimisation model

This chapter introduce the multi-objective optimisation model and discusses the benchmark instances of the PFSP with different uncertainties. The multi-objective optimisation model address three important measures, namely; utility to minimise the makespan, the stability to minimise the problem noise due to different uncertainties interruptions and robustness measure to keep the current schedule robust in face of different disruptions. Moreover, this chapter explain the generation of the benchmarks of dynamic PFSP and SPFSP including the different real-time events. Finally, the sensitivity analysis technique that is used to generate the objectives weights is discussed in this chapter.

Chapter 4: Particle Swarm Optimisation Algorithm

This chapter shows the adaption of the evolutionary PSO algorithm for the predictive-reactive approach with the proposed multi-objective optimisation model to generate robust and stable

schedule for the dynamic PFSP under machine breakdowns and new job arrivals. The experimental results of the proposed multi-objective optimisation model compared against other models to test our model efficiency. Finally, a statistical Analysis of Variance study (ANOVA) is used to find the effect of different models on the solution efficiency under the state of different real-time disruptions.

Chapter 5: Iterated Greedy Algorithm

This chapter introduce the IG algorithm for the predictive-reactive approach and our multi-objective optimisation model to solve the dynamic PFSP under different real-time events. This algorithm has been implemented to determine the local optimal solution for the problem and improve the solution by using other techniques implicitly including the NEH heuristic which is used to generate initial solution to the IG algorithm, and LS procedure to improve the solution in the algorithm. An experimental study and ANOVA is conducted to study the effect of different proposed models on the problem performance under uncertainty situation. Finally, comparative study between IG and PSO algorithms is discussed in this chapter.

Chapter 6: Biased Randomised Iterated Greedy Algorithm

In this chapter, the BRIG algorithm with randomisation techniques are explained and adapted for the predictive-reactive approach to solve the dynamic PFSP under different real-time events. This algorithm has been implemented to determine the local optimal solution for the problem and consider the stability and robustness by using the proposed multi-objective optimisation model that introduced in chapter three. An experimental study and ANOVA is conducted to study the effect of different proposed measures on the problem performance under uncertainty situation. Also, comparisons between the BRIG, the IG and the PSO algorithms are implemented to test the performance and speed of algorithms.

Chapter 7: Simulation Particle Swarm optimisation method

This chapter presents the framework of the novel Sim-PSO for the SPFSP under different real-time events. The summary of the results, recommendations and scope for the algorithm are given in this chapter including the reliability analysis to compare different dynamic and stochastic solutions.

Chapter 8: Sim-Biased Randomised Iterated Greedy Algorithm

In this chapter, a Sim-BRIG approach is proposed and implemented to solve the SPFSP under different real-time events, with the goal of minimising three measures, which are; utility, instability and robustness, simultaneously. The experimental results and conclusions are given at the end of this chapter, including a comparative study between the Sim-BRIG and the Sim-PSO algorithms.

Chapter 9: Conclusions and future works

This chapter presents the conclusions, summary of the results, recommendations and domain

for future work in the direction of dynamic PFSPs and SPFSPs under different uncertainties. It also discusses the specific contributions made in this research work and the limitations there in. This chapter concludes the work covered in the thesis with implications of the findings and general discussions on the area of research.

Chapter 2

Literature review

2.1 Introduction

The previous chapter highlights the background and the concept of scheduling problems in manufacturing systems. The shop scheduling problems considered are complex and hard problems to be solved due to the fact that they belong to the NP-hard class (Graham et al., 1979) in addition when the problem under dynamic and/or stochastic environments and with multiple performance measures. Until today, most research has been done on the static PFSPs with less research being considered for the PFSP under dynamic and stochastic environments. This lack of research is due to the complicated scheduling approaches for such systems to guarantee the best employment for the scheduling system and to reduce the instability, also keep the schedule robust in the face of different uncertainties. This chapter provides sufficient reliance for the related approaches and the relevant gap in the previous literatures corresponding to the dynamic and stochastic PFSP in the presence of uncertainties. The existing literature about the PFSP is widely categorised depending on the problem environment (static, dynamic and stochastic), the optimisation models and on the solution approaches. According to this restriction, this chapter is conducted from the next points of view:

1. A review of the PFSP and the solution methods for this problem that existed in the literature.

2. A review of the advanced scheduling techniques that have been used for the PFSP effectively under different environments (static, dynamic and stochastic).

The overall aims of this chapter are given as follows:

- To summarise the PFSP (static, dynamic and stochastic) and the advanced solution techniques that handle this problem.

- To identify the research limitations in the existing approaches to static, dynamic and stochastic scheduling.
- To highlight the objectives of research for this thesis.
- To present the research outline to achieve the objectives of this research.

The classification of this chapter is shown in Figure 2.1. Furthermore, the literature gap that is pertinent to this work and solution methodologies are given in the following sections.

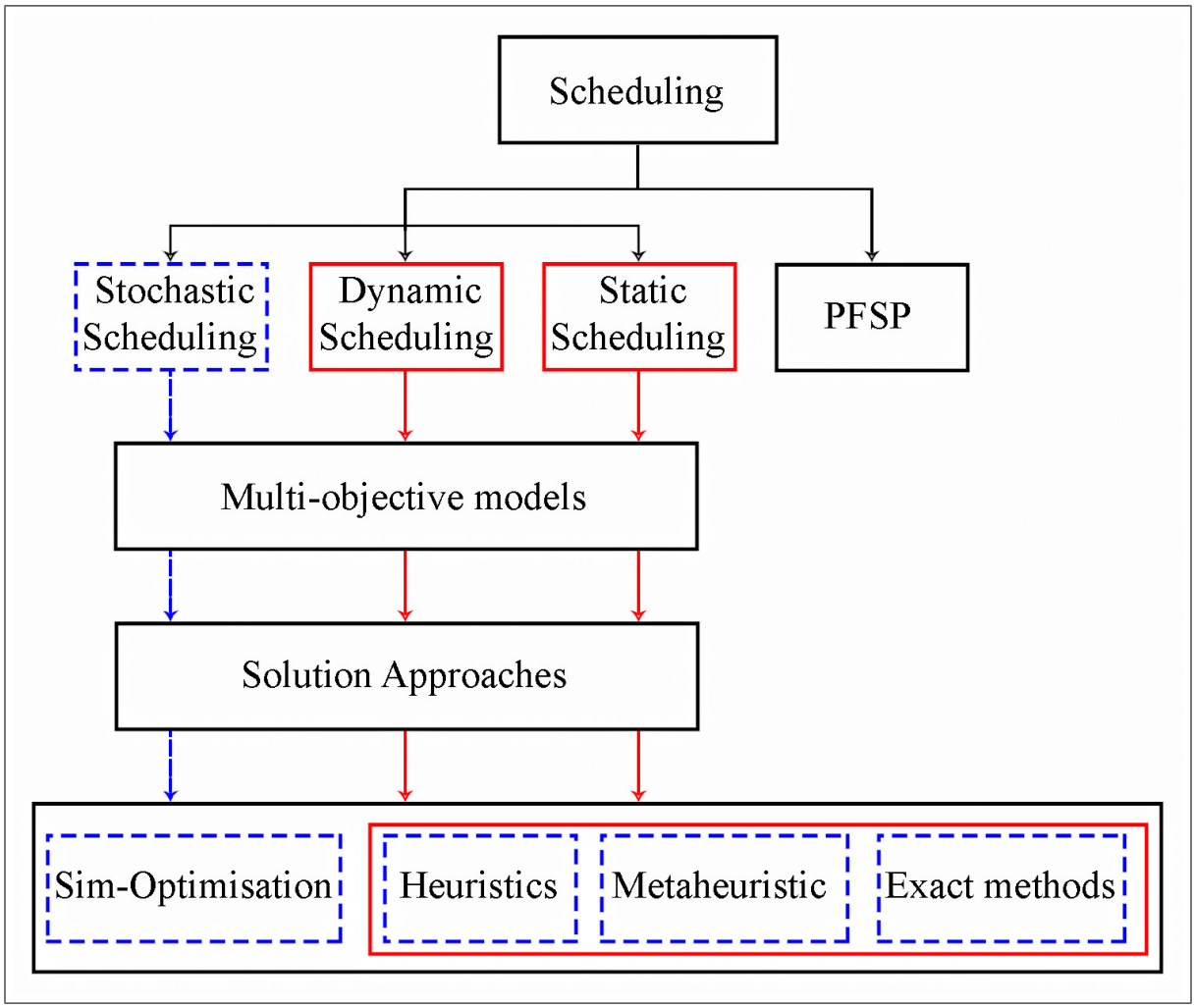


Fig. 2.1 Structure of the literature review

Figure 2.1 shows the path of introducing the literature review in this chapter.

2.2 Permutation Flow Shop Scheduling Problem

The PFSP is defined as a set of n-independent jobs that has to be executed on a set of m-independent machines. On each machine, each job has a fixed processing time value $p_{ij} \geq 0$. Also, each machine can process at most one job at a time, and the processing sequence of the jobs is the same for all machines, i.e., the job passing is not permitted. The definition of the PFSP dates from over seventy years. Since then, a large number of papers have been published about this problem and its variations. In this section, we present the literature related to the PFSP without focusing on the problem environments. However, the PFSP under dynamic and stochastic environments and uncertainties types will be discussed in details in the following sections. The early research on Flow Shop Scheduling Problem (FSP) is mostly based on Johnson's rule Johnson (1954). This work introduced the PFSP on an environment formed by two machines where the criterion is to minimise the makespan. The PFSP of n-jobs on m-sequential machines with the objective of minimising makespan is proven to be NP-hard (Graham et al., 1979), (Kan, 1976) and can be solved exactly for only small size problems. Because of this intractability, many authors proposed various techniques to solve this problem.

2.2.1 Exact methods

In the literature of PFSP, different solution methods have been developed and applied for this problem. Emmons & Vairaktarakis (2012) introduced the different methods including exact, heuristics and metaheuristics that were used for FSPs and hence PFSP. The first methods that were developed and proposed for the PFSPs are exact methods. However, such methods were successful for small size instances. In 1970, the PFSP has been reviewed by James & Michael (1970). Then, Campbell et al. (1970) studied the problem highlighting the strategy of solutions and diverse optimisation objectives. Ignall & Schrage (1965) were the first authors that introduced the B&B method for PFSP with minimising the makespan. Hariri & Potts (1989) proposed the B&B algorithm to minimise the number of late jobs in a PFSP. They proposed a technique of basing a lower bound on the simultaneous consideration of easily solved sub-problems of a SMSP. Carlier & Rebaï (1996) applied two B&B algorithms to minimise the makespan for the PFSP. Hejazi & Saghafian (2005) provided a comprehensive survey of FSPs subject to the minimisation of the makespan. This study also surveyed for small size instances some exact techniques and for larger size constructive heuristics approaches, metaheuristic and evolutionary methods. Moreover, the author introduced contributions from early research of Johnson (1954) until modern methods of metaheuristics in 2004.

2.2.2 Heuristic methods

Another methods that has been applied successfully for even large size PFSP instances are heuristics. Dudek & Teuton (1964) developed an m-stage rule for the PFSP subject to the minimisation of the idle time accumulated on the last machine when executing each job by employing the basic ideas of Johnson's rule. Palmer (1965) introduced the Slope Index Heuristic algorithm, which can be applied to large size problems even for hand calculations. This heuristic algorithm first calculates a slope order for each job, and then sequences the jobs according to the slope orders. This gives priority to the jobs with the strongest tendency to progress from short times to long times in the sequence of operations. Gupta (1971) presented an adjustment of Palmer's Slope Index which utilised some resemblance between sorting and scheduling problems. In a similar way, for the PFSP, Bonney & Gundry (1976) studied the idea of employing the geometrical properties of the jobs cumulative process times and a Slope Matching approach. Dannenbring (1977) attempted to integrate the advantages of the heuristic procedures introduced by Campbell et al. (1970). This approach is termed the Rapid Access technique where it aims to provide a quick and successful schedule by constructing an artificial 2-machine problem such that the processing times were specified from a weighting technique and then solved by using Johnson's rule. Nawaz et al. (1983) proposed an NEH heuristic for the PFSP to minimise the makespan. It basically uses the idea that jobs with high processing times on all the machines must schedule as early as possible before the jobs with less processing times. Hence, the heuristic NEH algorithm is based neither on Johnson's algorithm nor on Slope Indexes. However, the only obstacle is that a total of $\frac{n(n+1)}{2} - 1$ schedules must be computed, being n of those schedules complete sequences. Framinan et al. (2004) introduced a classification and review of heuristics for the PFSP under the objective of minimising the makespan. Also, Framinan et al. (2002) and Ruiz & Maroto (2005) introduced a comprehensive review and evaluation of PFSP heuristics, while a statistic review was introduced by Reisman et al. (1997). Ruiz & Stützle (2007) introduced one of the most efficient heuristic for the PFSP with minimising the makespan, which is the IG algorithm. This algorithm showed an extraordinary performance in reaching high quality solutions in reasonable computational time. Dong et al. (2015) applied a Self Adaptive Strategy for the Iterated Local Search (ILS) on the PFSP where the criterion is to minimise the Total Flow Time (TFT). Sharma et al. (2016) made an attempt to minimise the makespan for the m-machine FSP and compare the complexity time for this problem by reducing the sequences and to finding an optimal or near optimal makespan, where the CDS heuristic algorithm was proposed. Shao & Pi (2016) presented a self guided Differential Evolution (DE) with Neighbourhood Search for the PFSP where the criterion is to minimise the makespan. Rossi et al. (2017) addressed the PFSP with the criterion

of minimising the TFT. They developed heuristic methods that provide high-quality solutions with computational efficiency for this problem.

2.2.3 Metaheuristic and other methods

To solve the PFSP, a broad different metaheuristic approaches, which require fewer computations were used in the literature to generate a local optimal solutions. Some of these methods are; PSO algorithm, Genetic Algorithm (GA), Tabu Search (TS), Simulated Annealing (SA) and more. The first article discussing the application of PSO for solving the PFSP was presented by Tasgetiren et al. (2004). Also, Rajendran & Ziegler (2004) studied the use of Ant Colony Optimisation algorithm for the PFSP with the criteria of minimising both the makespan and the sum of the TFT of jobs. Moreover, Solimanpur et al. (2004) proposed a TS algorithm with neural networks for PFSP. Liu & Liu (2013) presented a hybrid discrete Artificial Bee Colony method for the PFSP with the minimisation of makespan. Bargaoui & Driss (2014) applied a Multi-Agent model based on a TS method to solve the PFSP. Mirabi (2014) developed one novel Hybrid GAs for the FSP with minimising the makespan. Robert & Kumar (2016) proposed the hybridisation of GA and SA algorithms for the PFSP to minimise the makespan. They compared this method against PSO and a Bacterial Foraging Optimisation algorithms. In this work, the obtained results demonstrated the viability of the proposed method. A novel PSO algorithm for the PFSP subject to the minimisation of the makespan was proposed by Jia et al. (2016). To adjust the PSO algorithm for discrete problems, some improvements and corresponding procedures were used. Li et al. (2015) employed the PSO algorithm by using the advantage of the swarm feature to determine the best particle in the solution space for the PFSP with the criteria of minimising the makespan. In the first step an initial solution was generated by the NEH heuristic. Then, they used some optimised strategy to set the parameters acceleration constant and nonlinear inertia weight strategy which is based on random self-adaptive by means of a Chaos method for setting parameters. Deng & Wang (2017) proposed a multi-objective Memetic Search Algorithm (MSA) for the distributed PFSP with the bi-objective function of both the makespan and TFT criterion. They first used the NEH algorithm to initialise the population to improve initial solution quality. Then they applied a Global Search Embedded with a perturbation operation to enhance the solution of the whole population. Moreover, the author employed a Single Insert Based LS technique to enhance each individual and then used a further LS strategy to determine a better solution for the non-improved individual in the Single Insert Based LS. Bessedik et al. (2016) studied the hybrid GA Based Artificial Immune System (AIS) for the PFSP with the makespan objective. The presented hybridisation technique was used in two trends: the first way is the hybrid of GA and AIS Vaccination (Jiao & Wang, 2000) into the field of GAs based on the theory of Immunity in biology. The second considered its

inspiration on the Immune network theory (Perelson, 1989), and applied it to the field of GAs. Greedy Randomised Adaptive Search Procedure metaheuristic for the scheduling problem in a PFSP environment in order to minimise the TWT was proposed by Molina-Sánchez & González-Neira (2016).

A number of selecting studies are also introduced in the following sections, concentrating on some of the key domain research. The aim of most of this search is to diminish the existence gap between scheduling theory and practice. The early literature extend back to the 1960s where Dutton (1962), Dutton (1964) tried to capture scheduling practice in a box manufacturer from a simulation model of scheduler behavior. Noticeable research efforts in the last four decades have been done to develop different approaches to support scheduling under real circumstances, Maccarthy & Liu (1993), introduced the failure of classical scheduling theory to respond to the needs of practical environments, and recent trends in scheduling research attempt to make it more relevant and applicable. Jackson et al. (2004) introduced a new model to understand and describe scheduling in real manufacturing industry. Mathematical optimisation models play an important role in scheduling solution approaches. Thus, in the following section, we introduce briefly the literature of optimisation models related to the problem under study.

2.3 Mathematical Optimisation models

The initial formulations of mathematical optimisation models for scheduling problems may be traced back to the late of 1950s. At that time, a few solution approaches were recognised to solve different types of mathematical optimisation models. However, there were no applicable computing technologies for the existing solution methods. The gap between computing technologies and mathematical models turn many practitioners to use mathematical models widely as their basic way that could obtain optimal solutions. The impact of this problem discourage both academics and practitioners, as computational technologies and alternative solution approaches could not handle large size problems. It is clear that mathematical programming would be considered as the best way to generate optimal solutions if computational technology could keep up. However, this problem did not stop researcher in academia from continuing to develop mathematical models as a portion of their solution methods in the hope of reaching the day where solving the mathematical models to optimality is possible. This hope was considered as unattainable in the imagination of many researchers. Thus, many researchers believe that the problem of reaching optimality could not be solved unless supercomputers were discovered. On the other hand, few researchers thought it is possible to solve mathematical models to optimality for small and medium-sized instances using the available technology at that time. For this, they worked to adapt modeling methods such that the decision variables and constraints of the

mathematical models were significantly reduced. Although the single-objective PFSP has been broadly studied, investigations about multi-objective PFSP have not been covered as much as the single objective.

2.3.1 Multi-objective Optimisation models

Initial work on the weighted sum method for multi-objective problems can be found in Zadeh (1963). The context of instability or nervousness first started being used by Steele (1975), the author refers to the significant changes occurring in Material Requirement Planning Systems. Different scheduling formulations that could lead to different robust and stable schedules where formulations consider different efficiency measures. There are some studied in the literature about efficiency measures. However, there are not much studies considering the weaknesses of continuously introducing changes in the schedule (Rangsaritratsamee et al., 2004). Chang et al. (2002) presented the Gradual Priority Weighting method to search the Pareto optimal solution for the multi-objective FSP, which has the following objectives; makespan, TFT, total tardiness and maximum tardiness. The presented solution methods search the feasible solution space starting from the first measure and towards the remaining measures step by step. Coello et al. (2004) proposed a method where the Pareto dominance integrated with the PSO such that the approach will have the ability to deal with multi-objective functions. Qian et al. (2006) introduced a DE based hybrid algorithm for multi-objective PFSP. Geiger (2008) tested the LS metaheuristic for multi-objective PFSP. Their work was based on two important principles of heuristic search, which are; intensification through Variable Neighbourhoods (VN), and diversification through perturbations and successive iterations in favorable regions of the search space. Geiger (2006) proposed an investigation of the search space topology in the context of global multi-objective PFSP. He showed that for the single objective problems a single global optimum has to be identified, while the multi objective problem need the identification of whole set of equalities. The significance of this work was shown in the context of metaheuristic LS methods for which meaningful implications derive. Mokotoff (2009) developed the multi-objective SA models for the multi-objective PFSP to provide the decision maker with high quality solutions. Rahimi-Vahed & Mirghorbani (2007) used the concept of the Ideal Point and a new multi-objective PSO method to solve the bi-objective PFSP with minimising both the weighted mean completion time and weighted mean tardiness. Wang et al. (2008) provided a comprehensive survey of multi-objective scheduling. They considered some basic concepts and prevalent approaches for multi-objective optimisation. As well they discussed several multi-objective scheduling models and a recent study on them. Rahimi-Vahed & Mirzaei (2008) applied a multi-objective Shuffled Frog Leaping approach to a bi-objective PFSP with the minimisation of the weighted mean completion time and the weighted mean

tardiness. Lei (2008) provided an extensive review of the literature on the scheduling problems with multiple objectives, among others. Also, a complete review of the literature for multi objective FSPs including Objective Weighting approach introduced by Minella et al. (2008). Qian et al. (2009) presented a hybrid DE algorithm to solve multi-objective PFSP with limited buffers between consecutive machines. They used a Largest-Order-Value rule to modify the continuous values of individuals in DE to job permutations, and hence, adjust the DE to solve scheduling problems. The authors also applied a LS based on the landscape of the multi-objective PFSP with limited buffers. Moreover, the Pareto dominance concept was applied to deal with the multi-objective nature. Sun et al. (2011) introduced a complete review of previous and recent methods on the multi-objective FSPs. They firstly gave a wide description and the complexity of these problems. They also provided a classification of multi-objective optimisations and presented an analysis of the publications on the proposed problem. Sioud et al. (2015) presented an algorithm that hybridising the principles of a GA and AIS introduced to solve the multi-objective PFSP with sequence-dependent setup times where the makespan and the total tardiness were the two objectives studied. For a literature review of the contributions to multi-objective PFSP we refer to Yenisey & Yagmahan (2014). Rahmani et al. (2014) proposed a multi-objective Mixed Integer Linear Programming model for the FSP with stochastic parameters. The multi-objective functions considers minimising each of makespan, TFT and total tardiness, simultaneously. The authors apply the Chance Constrained Programming method and Fuzzy Goal Programming to handle multi-objective function and the stochastic parameters. Amirian & Sahraeian (2016) presented a modification of multi-objective DE based on SA to solve a general tri-objective non-PFSP. The flow shop system considers the release dates, machine breakdowns, past-sequence-dependent setup times and learning effect for all the jobs. The algorithm proposed to tackle such a model combines the robustness of DE with the rapid convergence and conditional diversification of SA. Entezari & Gholami (2015) applied the Weighted Sum method to the multi-objective optimisation model of Flexible Flow shop Scheduling Problem (FFSP) with unexpected arrivals of new jobs. To solve the no-idle PFSP with the objective of minimising the total tardiness, Shen et al. (2015) proposed a Bi-Population Estimation of Distribution algorithm. For a multi-objective FSP, Tiwari et al. (2015) proposed the Pareto optimal block-based Estimation of Distribution Algorithm using bivariate model. Li & Li (2015) used a multi-objective LS based decomposition for the multi-objective FSP problem with and without sequence dependent setup times where the bi-objective function is to minimise the makespan and TFT. The proposed method decomposes a multi-objective problem into sub-problems of single objectives employing an Aggregation approach and optimise them simultaneously. Leisten & Rajendran (2015) proposed a new criterion in a PFSP that aims to reduce the gap between the completion times of each two consecutively scheduled jobs. The

authors compared some solution methods and discussed the influence of scheduling decisions on other systems related to this scheduling system by employing a new criterion. Deng & Wang (2017) presented the Memetic Algorithm (MA) to solve the multi-objective distributed PFSP with the bi-objective function of makespan and total tardiness. Lu et al. (2016) introduced a solution approach for a real-world scheduling problem of a welding process. This problem was formulated as a new multi-objective Mixed Integer Linear Programming model. Then a multi-objective discrete grey wolf optimiser was proposed to handle this problem. based on the NEH heuristic, Liu et al. (2016a) proposed a heuristic approach to solve the PFSP with the bi-objective function that minimise the makespan and machine idle time. de Siqueira et al. (2016) showed the implementations of two metaheuristics based on GAs for solving the multi-Objective hybrid FSP Problem. These two implemented metaheuristics were known as second generation methods of evolutionary multi-objective algorithms. Li & Ma (2016) presented a novel multi-objective MSA for the multi-objective PFSP. Hassanzadeh et al. (2016) developed a new metaheuristic technique to solve an integrated multi-objective production distribution FSP. This problem had two objectives where the first one concentrated on minimising makespan and TWT, while the goal of the second objective function was to minimise the summation of total weighted earliness, inventory costs, total weighted number of tardy jobs and total delivery costs. Zangari et al. (2017) introduced a novel general multi-objective Decomposition-based Estimation of Distribution algorithms using Kernels of Mallows models for solving multi-objective PFSP, this minimised the TFT and the makespan. Finally, the multi-objective PFSP with sequence-dependent setup times of minimising the makespan and TWT was introduced by Xu et al. (2017). They designed a multi-objective ILS to solve this problem.

2.4 Static Scheduling Approaches

Static problems are fully defined in the literature of scheduling, even theoretically, they can be solved to optimality. Generally, exact methods applied together with the formulation of the scheduling problems. This formulation could be dynamic programming, constraint programming or mathematical programming, which is then solved by a complete and systematic search of the solution space. Numerous complex scheduling problems have been formulated as Mixed Integer Linear Programming model (MILP) (Pan & Chen, 2005), which were usually solved by the B&B method. The methods used to solve static scheduling problems are explained in figure 2.2.

Practical scheduling problems (not small size problems) are usually very complex to be solved to optimality by exact methods in a reasonable amount of time. For this, the need of heuristics and other techniques have been raised. Such techniques concentrate on finding

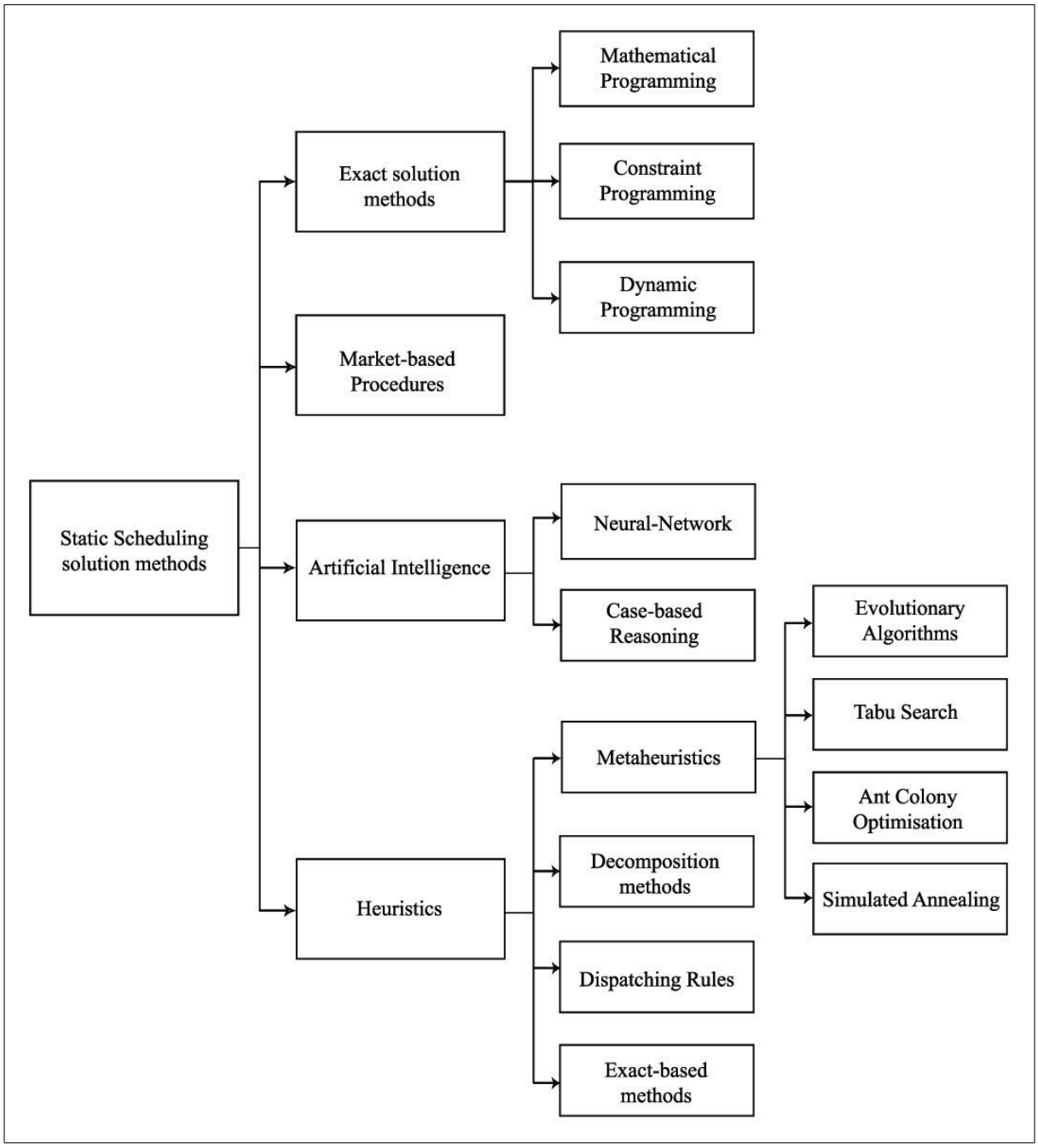


Fig. 2.2 Static scheduling solution methods

a good local optimal solution in a short time. Heuristics can used exact-based methods by restricting the exploration of the solution space to specific parts, or by limiting the time of running the algorithm, after which the best solution found so far is returned. A simple kind of heuristic method for scheduling problems are Dispatching Rules, these methods construct solutions gradually by scheduling one operation at a time. At any time, when there are jobs waiting to be processed on an available machine, a priority index for each job is calculating by Dispatching Rules as a function of some job and machine features, for example, job due date or weight, also, machine current setup, and schedule only the imminent operation of the job with the highest priority. Due to lack of a global perspective on the problem, Dispatching Rules produce less quality solutions when compared to other complex heuristic methods. However, their local horizon allows them to be processed very readily. Regardless of the complexity of the overall problem. An alternative search-based heuristics are known as metaheuristics. Such methods start to generate initial solutions (randomly or using heuristics methods) where they explore the solution space to improve upon by means of LS techniques in incorporation with additional techniques, which prevent them from remaining in local optima and seeking more new locations from the space. A well-known branch of metaheuristic methods are evolutionary algorithms, these methods have been widely used for scheduling problems (Dahal et al., 2007). There are different types of search-based heuristics, which are based on decomposing the scheduling problem into sub-scheduling problems of smaller sizes. Such methods are called Decomposition methods, which can solve the complex scheduling problem more easily. The partial sub-scheduling solutions are then recombined to obtain a final overall solution to the given scheduling problem. For static scheduling problems, the Decomposition approaches are usually machine-based such as the well-known Shifting Bottleneck heuristic or job-based Decomposition (Mason et al., 2002), (Pinedo, 2016). Market-based approach is another procedure which is based on local decision making (Toptal & Sabuncuoglu, 2010). In this approach, the scheduling decisions were modeled as a negotiation process between agents related to the jobs and agents representing the machines. Each job agent has a specific budget that can employed to pay for a processing time on the needed machines. The job agent calls for bids to which the agents of the machine(s) can execute the operation of a job they reply to keep a time slot for this job operation, where each bid has a time slot and price that is determine by the machine agents with the objective of maximising their own utility. Then the job agent can choose the bid with the best value for money, that way scheduling the operation of a job. The application of Market-based approaches have the challenge of determining the good budgets, also, acceptance rules and effective pricing that are probably will be problem-specific. Finally, Machine Learning techniques were also addressed in the static scheduling problems (Pinedo, 2016). To solve this problem, these techniques were used inference from good solutions to

similar instances. Therefore, they can only be used in combination with another method or a human expert, which or who generates solutions to example problems that can be used to train the Machine Learning algorithm.

2.5 Dynamic Scheduling Approaches

Unlike static problems, dynamic ones cannot be solved optimally since the optimal schedule depends on future unpredictable real-time events which only happen after a schedule has been executed. The main feature of dynamic scheduling solution approaches is the way of considering different types of unpredictable real-time events, which have the proactive or reactive shape. Figure 2.3 shows the different dynamic scheduling approaches based on the surveys by Aytug et al. (2005), Ouelhadj & Petrovic (2008) and the general overview of the FSP under uncertainties (González-Neira et al., 2017).

From figure 2.3, dynamic scheduling has been defined under three categories (Vieira et al., 2003); (Aytug et al., 2005):

- Completely-Reactive Scheduling or Dynamic Scheduling is also referred to as Online Scheduling. In this case, no firm (robust) schedule is generated at the beginning of the scheduling process, and the job schedule is obtained in a real-time manner. Priority Dispatching Rules are frequently used. The advantage of completely-reactive scheduling is that alterations due to unpredicted real-time events are considered as they stand out, which allows for immediate response. However, to provide the ability of scheduling in real shop floor, the reactive approaches have to depend on executions that provide low computational and information needs, for example, Dispatching Rules that take scheduling decisions on the basis of a locally restricted information horizon with little consideration of the overall problem structure. It is clear that when the effect and frequency of random real-time events is high, a globally optimised schedule becomes neglected shortly, up to a point where the assumptions underlying the schedule become invalid only moments after the very first part of the schedule has been implemented. Then, the global solution method effectively solves the wrong problem and thus may well lead to poorer scheduling decisions than a locally restricted approach which does not prepare a future plan at all.

- Robust Pro-active Scheduling, this approach concentrates on constructing predictive schedules that remains robust (flexible) despite real-time events that may affect the system during a scheduling horizon, and up to a certain degree, can adjust future alterations in order to ensure the objective function value does not deteriorate significantly. When

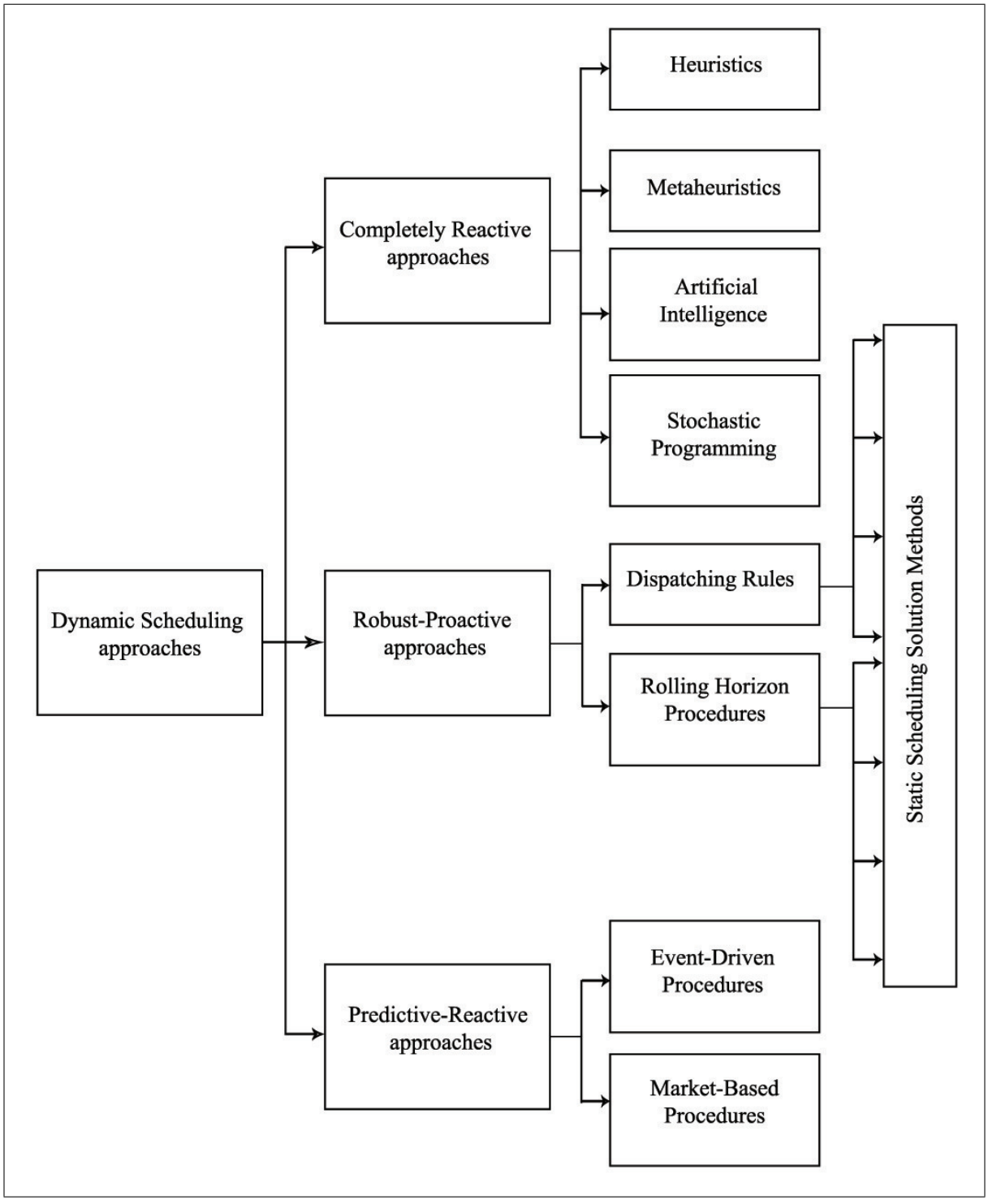


Fig. 2.3 Dynamic scheduling solution methods

using this approach, it is important to have enough information about the real-time events that present in the problem, their occurrence probability and their potential effect on the solution quality. These information can then be integrated within the solution method, for example, within the objective function or constraints of a stochastic mathematical programme (Kouvelis et al., 2000). For longer time horizons, the predictions related to the aftertime developments become increasingly inaccurate. Also, there is an exponential increase of possible combinations of future real-time events over time. Thus, for the problems with relatively short time horizons, the best approach is the robust pro-active scheduling.

- Predictive-Reactive Scheduling is defined as a rescheduling procedure such that schedules are modified at the time of disruptions. The approach is a two-stage process; in the first step, predictive scheduling (baseline) is generated. The second phase is about releasing the schedule to the shop floor and revising it in response to real-time events. In general, there is a broad agreement in the literature that the Predictive-reactive approach is the most common dynamic technique that can be applied in manufacturing systems. Figure 2.4 shows the idea of this approach.

Depending on the mechanism for starting the modification process of the schedule, predictive-reactive approaches can be categorised as Time-Driven or Event-Driven. In the Time-Driven techniques which are also termed as Rolling Horizon techniques, the schedule is reoptimised at uniform intervals of time. Event-Driven techniques triggers a revision procedure to the schedule in response to random real-time events. In the two stages of the predictive-reactive approach, any of the solution techniques for static scheduling problems (see Figure 2.2) can be used for the generation of a predictive solution and the revised schedule, where choosing the suitable technique for a given problem generally counts on the nature of the disruptions that present into the system in terms of their disruptive power and the available time to react. For example, when the only unexpected real-events are the arrivals of new jobs, the method of choice maybe the Time-Driven approach that periodically applies a B&B algorithm, which is a computationally expensive (Ovacikt & Uzsoy, 1994). Also, for scheduling problem under random disruptions of machine breakdowns which could cause unexpected and significant changes in the scheduling plan, it may be necessary to apply the Event-Driven method. This method quickly restores feasibility of the schedule by means of a simple heuristic in the case of machine breakdowns (Yamamoto & Nof, 1985). Furthermore, a hybridisation of these methods is applied in practice, which generally follow a periodic reoptimisation but are able to react flexibly, if the disruption caused by a specific real-time event is severe.

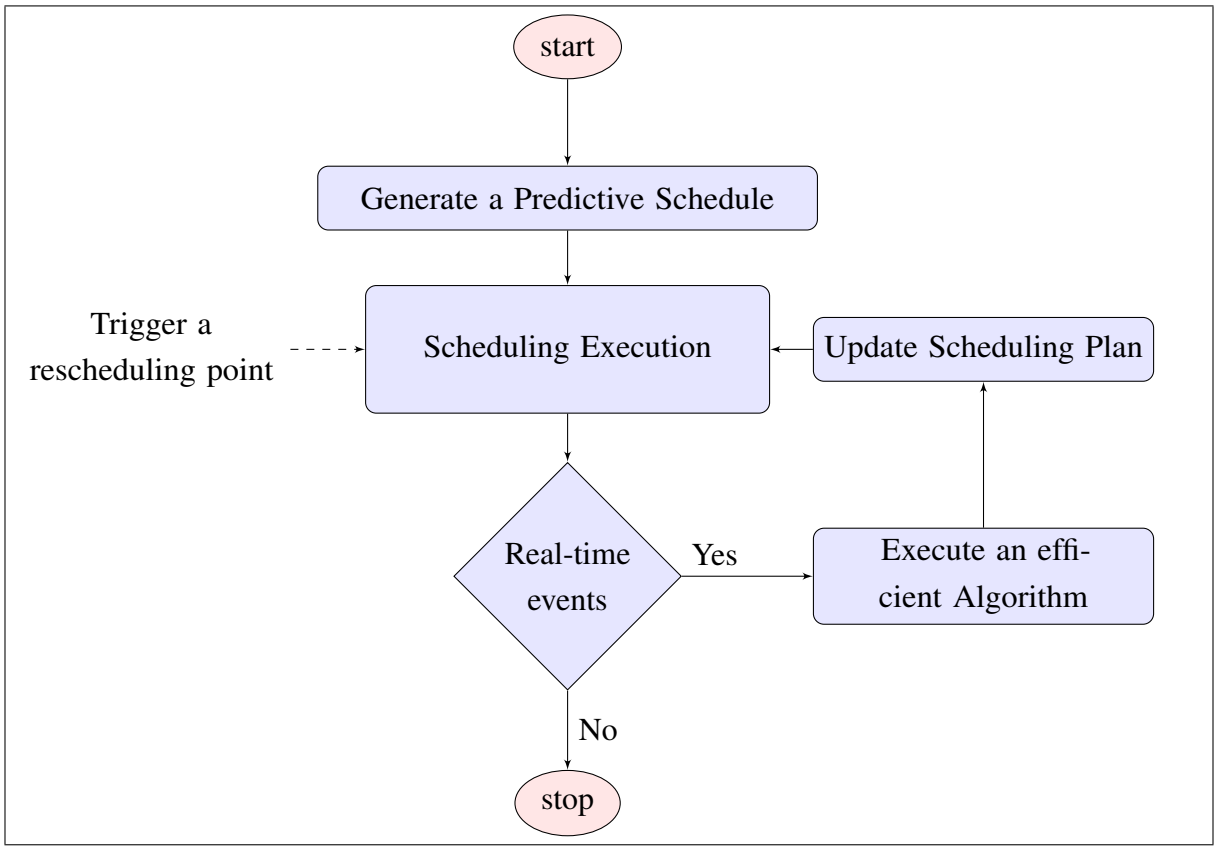


Fig. 2.4 Predictive-Reactive approach

There are different approaches existing in the literature for solving dynamic scheduling problems. It should be noted that, it is possible to hybridise any of the aforementioned approaches to deal with dynamic scheduling problems depending on the nature of the real-time events and the scheduling problem.

2.5.1 Disruptions classification

In scheduling for a real shop floor, the problem effected by single or different disruptions of real-time events. As we discussed previously, there are different methods existing in the literature that can be used to solve the PFSPs. Using a suitable method mainly depending on the problem environments and also on the type and frequency of disruption. For this, it is important to highlight these factors to be able to propose the best solution approaches. The literature of manufacturing systems under dynamic environment have considered a significant number of real-time events including their effects. Thus, real-time events can be categorised into the following groups (Vieira et al., 2003):

- **Resource-related:** such as; machine breakdown, unavailability or tool failures, operator illness, loading limits, defective material (material with wrong specification), delay in the arrival or shortage of materials, and so on.
- **Job-related:** for example; arrival of jobs, rush jobs, due date changes, job cancellation, change in job priority, changes in job processing time, and so on.

In this thesis, we consider some important disruptions of real-time events, which are; machine breakdowns and new jobs arrival. these disruptions are frequently occurred in real manufacturing systems.

2.5.1.1 Machine breakdown

The scheduling problems have been widely studied under static environment by assuming machines and jobs are available at time zero (Vieira et al., 2003); (Gholami et al., 2009). However, due to the uncertain environments in real shop floor, these assumptions become invalid. In this section, we highlight the work done for the scheduling problems under machine breakdown. Ali & John (1998) studied the bi-objective FSP that minimising both the makespan and maximum lateness, this problem considered the case of 2-machines under random machine breakdowns. When stochastic breakdowns effect the first or the second machine, respectively, the authors showed that the shortest and longest processing times orders are optimal with respect to both objectives in a sequence of FSP with 2-machines. To absorb the impacts of breakdowns, Mehta & Uzsoy (1998) have used the available information on uncertainties to generate a predictive schedule. To measure the effect of disruptions on planned activities, they used the difference between the planned completion times of jobs in the predictive schedule and their realised ones. The deviations of completion times were decreased by inserting extra idle time into the predictive schedule. The amount of inserted extra idle time based on the structure nature and frequency of the disruptions and the predictive schedule. Hence, in the predictive schedule, the completion times of jobs rely on the schedule and the amount of inserted extra idle time. For the Flexible Job Shop Scheduling Problems (FJSP), Jensen (2003) proposed the robust and flexible schedules (solutions), where the aim is to minimise the makespan. The author has used GA to define and investigate the robustness measure to obtain robust and flexible schedules. These solutions were used to improve rescheduling significantly after machine breakdown disrupted ordinary schedules.
To minimise makespan criteria for a stochastic FFSP that is effected by machine breakdowns, Allaoui & Artiba (2004) proposed a framework of robustness to deal with such a problem. By minimising the starting time deviations of jobs simultaneously, the algorithm handle the

efficiency by maintaining the objectives of makespan, tardiness and stability. In this case, the rescheduling triggered at specific intervals of time employing all obtainable jobs at the moment of disruption. Kasap et al. (2006) studied the policies of optimal sequencing of jobs for a single machine PFSP under random breakdowns and the objective of the expected makespan. The FFSP with sequence dependent setups and under a stochastic machine breakdown have been solved by a heuristic algorithm (Gholami et al., 2009). This heuristic used the random key GA to determine the best local solution. Also, the Right-Shift technique and the Event-Driven policy were incorporated in a simulator into the GA to evaluate the expected value of makespan. Al-Hinai & Elmekkawy (2011) addressed finding stable and robust solutions for the FJSP under random machine failures. They defined some bi-criteria performances considering both the stability and robustness of the predicted schedule and compared using the same rescheduling scheme. The Discrete Group Search Optimiser algorithm for the hybrid FSP under random machine breakdown was introduced in Cui & Gu (2014). The proposed method adopted the vector representation and several discrete operators, e.g., swap, insert, destruction, and construction in the process, DE, rangers phases and scroungers. Pugazhenthi & Saravanan (2015) proposed a new heuristic to analyse and solve the PFSP with breakdown nature. This heuristic proposed an Exponential Index method with known breakdown time, the break down occurs due to power shutdown. This heuristic has been applied for the problem with the objective of minimising the makespan time of n jobs and m machines. Wang et al. (2016) introduced the distributed PFSP with minimising the makespan and under machine failures. They used the distribution algorithm of Fuzzy Logic-based Hybrid Estimation for this problem. To find better solutions in the search space, the authors hybridised the probabilistic model of Estimation of the Distribution method with crossover and mutation operators of GA to generate new offspring. Proactive-scheduling approach is triggered for an uncertain machine failure under deteriorating production environments was considered by Wang et al. (2015a), where the usage and age of machine lead to a longer real processing time for jobs. They proposed a multi-objective Evolutionary algorithm based on Elitist Non-Dominated Sorting, in which a support vector regression surrogate model is built to replace the time-consuming simulations in evaluating the rescheduling cost. Fazayeli et al. (2016) proposed a Pro-active scheduling and a hybrid metaheuristic method based on GA and SA for the FSP with machine breakdown. The hybrid flow shop rescheduling problem with flexible processing time in steelmaking casting systems was studied by Li et al. (2016). They used a Hybrid Fruit Fly Optimisation algorithm to solve this problem under machine breakdown and processing variation disruptions, simultaneously. A serial-batching scheduling problem under machine breakdowns and new job arrivals events, where the objective is to minimise the makespan was addressed in Pei et al. (2016). A developed heuristic approach was applied to find near optimal solutions for

this problem. Adressi et al. (2016) considered a group scheduling problem in no-wait FSP by considering two stages with group sequence-dependent setup times under machine breakdowns. GA and SA based heuristics have been presented for this problem where the primary objective is to minimise the makespan for two classes of small and large scale problems. A SMSP under machine breakdown was considered by Imed & Hans (2016). The authors studied two different criterions for this problem, namely; minimising the makespan and maximum lateness. Also, two different algorithms were proposed to solve two types of the problem; without release dates or with different release dates. To solve a PMSP in dynamic environment under random machine failures, the Learning Agent was presented (Yuan et al., 2016). They tested this method for the problem with two different criterions; minimising the maximum lateness and minimising the percentage of tardy jobs.

2.5.1.2 New job arrivals

The scheduling in static environment have the most efforts in the literature. Such problems frequently assume the number of jobs are fixed, the processing times is deterministic and there is no unpredicted disruptions which would influence the processing of job when the schedule is under way. However, in reality, new orders arrive at production systems randomly, which leads to sheer complexity in scheduling due to the dynamic changes given various constraints of resources. Previous studies simply attach new orders directly after the existing schedule. Liu et al. (2005) proposed an approach based on the Support Vector Machine for the flexible manufacturing scheduling with minimising earliness and tardiness penalties of all jobs, to achieve the goal of dynamical scheduling. For the just-in-time scheduling of a manufacturing environment under new jobs that coming randomly into the system, Weng & Fujimura (2009) introduced two distributed feedback mechanisms, where the simulation proves that distributed feedback mechanisms showed a high performance. Some important improvements to formerly employed intelligent production system handling the dynamic scheduling problem of FSP with the factory environment of a multi-stage multi-machine were presented by Weng & Fujimura (2010), where the criterion is to minimise the total earliness and tardiness penalties of all jobs during any given period of time. Guo et al. (2011) applied an Adaptive Job-Insertion based heuristic for the FSP minimising the mean flow time in a dynamic environment, where the job arrival or release dates are not known in advance and new order jobs arrive randomly to the system. Rahman et al. (2013) proposed a method based on GA for the PFSP with multiple jobs arriving at different time points. The problem of combining new rush orders with the recent schedule of a manufacturing shop floor level was addressed in Madureira et al. (2013), where a Self-Organised Integration Mechanism Module based on Case-based Reasoning was presented so as to determine independently that combination mechanism have been employed

to incorporate new orders in the recent plan. Joo et al. (2013) used two Dispatching Rule-based scheduling methods to the three-stage FSP with maximising the quality rate and the mean tardiness of the finished jobs in a dynamic environment, where jobs of multiple types arrive to the system over time dynamically. Pickardt (2013) proposed three methods based on Evolutionary Algorithms to automate and support the design of Dispatching Rules for dynamic and complex scheduling problems. Kaplanoğlu (2014) used a collaborative Multi-Agent based Optimisation approach for the SMSP with sequence-dependent setup times and under constraints of regular and irregular maintenance activities when the order arrivals are dynamic. The effects of inserted idle times on the performance of a selection of Dispatching Rules for FSP with a new job arrival was examined in El-bouri (2013) where the objectives are to minimise the mean flow time, mean tardiness, or number of tardy jobs. Rao & Ranga Janardhana (2014) gave a literature review which analysed the rescheduling activity for the case of uncertainties from the manufacturer, supplier and customer. They also considered rescheduling factors, rescheduling environments, and rescheduling algorithms. Xingbao et al. (2015) discussed a Predictive Scheduling approach for the PFSP under new jobs arrival. They developed this approach to reduce the impact of the new job arrival by inserting moderate slack times into the baseline. A two-step approach was used for the two-stage FSP with unexpected arrival of new jobs (Entezari & Gholami, 2015). In the first step, an initial schedule is obtained considering makespan as an objective function. After the initial schedule was applied, now assume that a new job arrives during the execution of the initial schedule. In the second step, they proposed three criteria as a measure based on a classical objective and performance measures. This Measure consists of makespan, stability and variation of completion times. Sahin et al. (2015) proposed a Multi-Agent based System to the flexible scheduling problem in a dynamic manufacturing environment. Kaplan & Rabadi (2015) studied the PMSP under job-related disruptions, namely; departure of an existing job, the new jobs arrival and changes to job priority. They considered the minimisation of the bi-criterion of both of the TWT and instability of schedule, simultaneously. The authors first built a MILP using a scheduling problem proposed by Kaplan & Rabadi (2012). Then they applied five different heuristic approaches for the PMSP with the bi-objective function, using the developed methods by Kaplan & Rabadi (2012) and Kaplan & Rabadi (2013). Rahman et al. (2015) presented a heuristic based decision process for the dynamic PFSP. As each new order arrives, they used the developed GA based method over and over to re-optimise the problem. Gao et al. (2016) introduced a two-stage Artificial Bee Colony algorithm and an effective model for solving the scheduling problem of re-manufacturing. This problem was modeled as FJSP with minimising the makespan, when the new job arrives the problem was splitted into two steps; scheduling and rescheduling. Zhang et al. (2016) presented an innovative method to investigate the flexible

scheduling problem in dynamic environment in order to minimise or maximise the consumption of energy into account. To solve this problem, they used a rescheduling method based on the GA, also they proposed a new goal programming model that considers the consumption of energy and the efficiency of schedule simultaneously. Xia et al. (2016) formulated a new dynamic Integrated Process Planning and Scheduling (IPPS) model, the combination of hybrid algorithm and rolling window technology was applied to solve the dynamic IPPS problem, and two kinds of disturbances were considered, which are the machine breakdown and new job arrival. The characteristics of dynamic integrated process planning and scheduling problem with job arrivals were studied in Liangliang et al. (2017). A novel MILP model was established to accommodate new job arrivals, and three criteria; makespan, stability, and tardiness were considered. New periodic and Event-Driven rescheduling strategies were also presented. Sahin et al. (2015) proposed a Multi-Agent based System for the dynamic flexible machine scheduling groups and material handling system working. A novel scheduling strategy by integrating Match-Up strategy and Real-Time strategy in order to make use of the remaining time before the old order due date for PFSP with new job arrivals was introduced by Liu et al. (2017).

2.5.1.3 Scheduling in the presence of different disruptions

In real manufactures, it is very likely that different disruptions interrupt the system simultaneously. However, there are only few examples in academia regarding the scenario of scheduling system under different types of disruptions. Turkcan et al. (2009) considered the problem of PMSP with controllable processing times where the presented models were revised to incorporate a stability performance for rescheduling unpredicted events such as machine failure, new job arrival, delay in the arrival or shortage of materials in rescheduling. Liu et al. (2016b) addressed the PFSP with sequence dependent setup time which was effected by six different types of real-time events, which are; arrival of new jobs machine failure, variation of setup times, variation of processing time, job cancellation and job priority upgrading, simultaneously. Katragjini et al. (2013) introduced a novel benchmark for the PFSP under different types of disruptions including; machine breakdown, new job arrival and uncertain ready job. They introduced a bi-objective model to minimise makespan and instability performances. Also, the authors applied different heuristics methods and compared them against an IG algorithm, the aforementioned algorithm showed extraordinary better results when compared against other heuristics. A 2-machine FSP under stochastic processing times and unpredicted new jobs arrival were considered in Rahmani & Heydari (2014). They introduced a novel multi-objective optimisation model that consider three measures; makespan, stability and robustness to reduce the noise that disturb this system because of unpredictable uncertainties. Li et al. (2015) proposed a Discrete Teaching-Learning-based Optimisation to solve the flow shop rescheduling

problem under five types of uncertainties including machine breakdown, arrival of new jobs, cancellation of jobs, job processing variation and job release variation. The authors used the bi-objective model that was proposed by Katragjini et al. (2013) to minimise the makespan and instability measures. Park et al. (2017) proposed the Genetic Programming rules for the dynamic JSP under machine failures and arrival of new jobs.

2.6 Solution methods related to dynamic and static scheduling

As shown in figure 2.3, the solution methods of static scheduling are also used in some levels of the framework solutions of the dynamic scheduling problems. In this section, we present the literature of the techniques that proposed implicitly or explicitly for the dynamic PFSP under different real-time events. We start with the literature of the PSO algorithm for this problem, then the NEH heuristic is discussed. This algorithm has been used to generate an initial solution for both of the IG and its Biased Randomised (BR) version, which are also presented in this section.

2.6.1 Particle Swarm Optimisation

The PSO algorithm was described as a stochastic global optimisation method, it was introduced by Kennedy & Eberhart (1995). Thus, the PSO algorithm is one of the most efficient algorithms that have been applied successfully for the dynamic and stochastic COPs. The PSO algorithm has been implemented to solve the PFSP and other scheduling problems. For example, Lian et al. (2006) suggested a PSO algorithm for solving the PFSP with respect to minimisation of makespan and computational experiments showed that it was more efficient than GA. However, some problems cannot be solved to guarantee optimality. Tasgetiren et al. (2007) presented a PSO for solving the PFSP so as to minimise the makespan and the TFT. Liu et al. (2007) proposed a PSO-based MA for the PFSP in order to minimise the makespan. On the other hand, Wang & Yang (2007) developed a PSO to solve FSP where the processing times were uncertain. Lian et al. (2008) presented a novel PSO algorithm, which was successfully applied for the PFSP with minimising the total completion time. Pan et al. (2008) proposed a discrete PSO algorithm for solving the no-wait FSP in order to minimise both the makespan and the TFT. Liu et al. (2008a) introduced a hybrid PSO algorithm for the PFSP with the limited buffers between consecutive machines with the objective of minimising the makespan. Zhang et al. (2008) used an improved PSO method that combined the PSO algorithm with genetic operators for solving the FSP in order to minimise the makespan. (Sha & Hung Lin, 2009) provided a

PSO based multi-objective method for FSP. This method searches the Pareto optimal solution for criterions by taking in account the modified PSO algorithm and the objectives of makespan, mean flow time, and machine idle time. Kuo et al. (2009) presented a new hybrid PSO model which combined random-key encoding techniques, individual enhancement techniques and PSO algorithm for solving the FSP so as to minimising the makespan. Sha & Hung Lin (2009) implemented PSO algorithm for multi-objective FSP. They proposed an evolutionary algorithm that searches the Pareto optimal solution for objectives by considering the makespan, mean flow time, and machine idle time. Zhang et al. (2010) proposed a hybrid alternate two phases PSO algorithm to solve the FSP with the objective of minimising makespan which combined the PSO with genetic operators and annealing strategy. Liu et al. (2011) proposed a hybrid PSO with Estimation of Distribution algorithm to solve PFSP. Subashini & Bhuvaneswari (2011) presented a multi-objective PSO approach where a Non-Dominated Sorting PSO which combined the operations of Non-Dominated Sorting GA II (NSGA–II) was applied to schedule tasks in a heterogeneous environment. Liao et al. (2012) proposed the PSO method to solve the hybrid FSP subject to minimising the makespan. Kamble & Kadam (2012) studied the simultaneous scheduling problem of machine and automated guided vehicle in a flexible manufacturing system with minimising the objective of makespan, they applied the PSO algorithm for this problem. Chen et al. (2014) proposed a revised discrete PSO to solve the PFSP with the objective of minimising makespan. Damodaran et al. (2013) presented a PSO method to schedule batch processing machines arranged in a PFSP with the objective of minimising the makespan. Marinakis & Marinaki (2013) proposed an algorithmic Nature-Inspired method which employed a PSO algorithm with different neighbourhood topologies to solve one of the PFSPs. Vijay chakaravarthy et al. (2013) studied the flow shop with equal size sub lots with the criterion of minimising the makespan and TFT. To solve this problem, they used a DE and PSO algorithms. Moreover, A PSO algorithm was applied for the PFSP with the objective of minimising the makespan (Ramanan et al., 2014). Akhshabi et al. (2014) presented a PSO algorithm based on the MA which combined with the LS approach to solve a no-wait FSP with the criterion of minimising the TFT. Behnamian (2014) extended the hybrid PSO-based metaheuristic for solving the fuzzy PMSP with bell-shaped fuzzy processing times where the criterion is to minimise the fuzzy value of makespan. Li et al. (2014) proposed a hybrid approach of an ILS and PSO to solve the hybrid FSP with preventive maintenance activities. Zhang & Wu (2014) investigated the PFSP with the objectives of minimising the makespan and the TFT and proposed a hybrid metaheuristic based on the PSO algorithm. Ramanan et al. (2014) used a PSO approach for the objective of optimising the makespan of an FSP. Dongdong et al. (2015) applied a Discrete PSO for the FFSP which minimises the maximum time used in the FSP. Zhang et al. (2015) introduced a comprehensive survey that

investigated the advances with PSO algorithm, including its modifications, population topology, hybridisation with other optimisation approaches, extensions, theoretical analysis and parallel implementation. They also provided a survey on PSO algorithm applications to different fields. A hybrid algorithm based on PSO and SA was proposed by Kamble et al. (2015) and considered the FJSP with minimising five criterion, namely; the makespan, the maximal machine workload, the total workload, the machine idle time and the total tardiness, simultaneously.

2.6.2 NEH Algorithm

The NEH algorithm is a heuristic algorithm that was designed for the PFSP by Nawaz et al. (1983). It has been broadly studied and various developed versions of the NEH heuristic have been presented in the literature. Examples of studies that show the NEH heuristic outperforming old version of methods are given in Turner & Booth (1987) and Taillard (1990), it has also been proved to give better results than other highly cited heuristics such as CDS method (Campbell et al., 1970). Several more recent studies established that the NEH heuristic showed better performance, even when compared with modern and more complex heuristics. An important study was given by Ruiz & Maroto (2005), where NEH heuristic was tested and compared against 25 other heuristics algorithms, including the more recent and complex algorithms of Hundal & Rajgopal (1988), Ho & Chang (1991), Koulamas (1998), Suliman (2000) and Pour (2001). In this work, careful statistical analyses of results of comparing NEH heuristic against different heuristics showed that NEH was superior to all tested heuristic methods and it was much faster at the same time. Moreover, Nagano & Moccellin (2002) used a new developed constructive heuristic called N&M in order to minimise the makespan for the FSP. This algorithm have been compared with the constructive NEH heuristic. The N&M algorithm outperforms, on average, the NEH algorithm. However, the study showed that there is no significant difference regarding computation effort for both N&M and NEH algorithms.
To minimise the maximum completion time of the PFSP, Framinan et al. (2003) tested 176 rules employed to obtain the initial list of jobs and illustrated that the ordering proposed initially in the NEH algorithm was the one where it showed the best results. However through these rules, the Nagano & Moccellin (2002), the Pour (2001) and the Profile Fitting procedures were not included. Since then, the NEH heuristic has been used to generate an initial solution for many modern heuristics and meta-heuristics. Some examples where the NEH was used as a seed sequence for GAs, SA, ILS, TS and many of other metaheuristic methods were proposed in Reeves (1995), Chen et al. (1995), Murata et al. (1996), Stützle (1998), Zheng & Wang (2003), Rajendran & Ziegler (2004), Bożejko & Wodecki (2004) and Ruiz et al. (2006). More recently, Ruiz & Stützle (2007) applied NEH heuristic to initialise the solution of the proposed IG algorithm. Also, the IG construction phase used the NEH procedure to construct the new

solution. Dong et al. (2008) studied several different Priority Rules for the NEH heuristic. They proposed a new strategy to solve job insertion ties which may exist in the original NEH heuristic. The authors also showed that the priority rule that combines the average processing time of jobs and their standard deviations, was not statistically significantly better than that used in NEH but it can get slightly better performance. Moreover, their new tie-breaking strategy has improved the performance of NEH significantly. Kalczynski & Kamburowski (2008) presented a combination of both the Priority Order and a simple Tie-Breaking approach. This new approach outperformed the NEH heuristic. Ribas & Mateo (2010) studied the FSP with and without buffer constraints and proposed an improved NEH-based heuristics algorithm for this problem. Li & Yin (2013) proposed a DE based MA for the PFSP where a heuristic NEH algorithm integrated with random initialisation to the population with certain quality and diversity. Marichelvam et al. (2014) proposed a developed Cuckoo Search algorithm for the multistage hybrid FSP with the objective of minimising the total completion time. In this algorithm, the NEH algorithm was incorporated with the initial solutions to reach a local optimal solutions quickly. Ramanan et al. (2014) used a PSO approach with the objective of optimising the makespan of an FSSP. The problems were tested on Taillard's benchmark problems. The results of NEH heuristic were initialised to the PSO to direct the search into a quality space. Vasiljevic & Danilovic (2015) showed the importance of the inclusion of the information about the sort of ties in the initial phase of the NEH heuristic, which applied for the PFSP with the makespan objective. The conclusion obtained by this study, was that the range of the objective values for different sorts of ties was often greater than the improvements, published in literature. This allowed them to construct a very simple algorithm that outperformed published NEH improvements, maintaining NEH's exceptional efficiency. The proposed algorithm also used the information about the ties in the insertion phase to improve the objective value. Rossi et al. (2016) analysed the PFSP with makespan minimisation criteria, and proposed constructive methods that make use of the principles of the NEH heuristic and recent studies to significantly improve its performance. For this purpose, 67 new constructive heuristic approaches were presented. Ding et al. (2016) proposed a modified NEH heuristic algorithm based on the multi-objective concept and developed the multi-objective IG approach to solve the FSP. Danilovic & Ilic (2016) considered the PFSP and used a generalised constructive algorithm which based on the extension of the NEH algorithm. Shao et al. (2017) used a modified speed-up NEH method at the initialisation stage and the random initialisation was utilised to generate more promising solutions with a reasonable running time, then a MA with Hybrid Node and Edge Histogram was applied to solve no-idle PFSP with the criterion to minimise the maximum completion time.

2.6.3 Iterated Greedy method

The IG algorithm is one of the most powerful and efficient heuristic algorithms developed so far for solving the PFSP. It is a constructive technique that was introduced by Jacobs & Brusco (1995). This algorithm was applied successfully for the PFSP by Ruiz & Stützle (2007). It is state of the art in terms of simplicity, speed and accuracy. Ruiz & Stützle (2007) compared the IG algorithm against different methods; the IG algorithm showed the best performance throughout all other methods. The authors showed in this research the IG algorithm was far superior and simpler than the hybrid GA proposed by Ruiz et al. (2006). Also, Ribas et al. (2015) proposed several SA based techniques and compared them to the IG algorithm. The authors applied the proposed methods for all classical benchmarks and the results illustrated that the IG algorithm performed better and uses fewer parameters than their methods. Ying (2007) proposed an IG algorithm for the non-PFSP. Also, Ying (2009) proposed the IG algorithm for the multistage hybrid FSP with multiprocessor tasks where the objective is to minimise the total completion time. The computational experiment results illustrated that the IG algorithm showed better performance when compared with other three metaheuristics. In addition, Ying (2012) presented an IG algorithm for the Wafer Sorting Scheduling Problem in order to minimise the primary objective of the total setup time and minimise the secondary objective of the number of testers. The same author Ying et al. (2009) proposed simple IG for solving the Single Machine Tardiness problem with sequence dependent setup times. Ying & Cheng (2010) presented an IG approach to the dynamic PMSP with sequence-dependent setup times in this problem. As well, Lin et al. (2011) considered an improved IG with a sinking temperature to minimise the maximum lateness of an identical PMSP with sequence-dependent setup times and job release dates. Moreover, Lin et al. (2013) presented a modified IG algorithm for the distributed PFSP with minimising the makespan. The IG algorithm applied with five other heuristics for the PFSP under different types of disruptions (Katragjini et al., 2013). In their work, the IG algorithm outperformed all other heuristics in achieving a better near optimal solution in a reasonable amount of time. Tasgetiren et al. (2013) presented a variable IG algorithm with DE, designed to solve the no-idle PFSP. Kang et al. (2013) used an IG algorithm for the problem of allocating parallel application tasks to processors in heterogeneous distributed computing systems where the objective is to maximise the system reliability. Ciavotta et al. (2013) presented Iterated Pareto Greedy (IPG) for the multi-objective sequence dependent setup times PFSP. Also, an IPG method was presented for solving the hybrid FSP in order to minimise both of the total completion time and the total tardiness (Ying et al., 2014). Campos & Arroyo (2014) addressed an Elitist Non-dominated Sorting GA with IG algorithm to solve the three-stage assembly FSP in order to minimise both of the TFT and the total tardiness, simultaneously. Deng & Gu (2014) proposed an enhanced IG algorithm which explores inserting and swapping neighbourhoods to

solve the SMSP with sequence-dependent setup times and the objective is to minimise the TWT. Pan et al. (2017) proposed a MILP model for the mixed no-idle extension where only some machines have the no-idle constraint where the objective is to minimise the total completion time. The authors used an IG algorithm to solve this problem. Ding et al. (2015) proposed a Tabu-mechanism improved IG method for solving a no-wait FSP in order to minimise the total completion time. Abdollahpour & Rezaeian (2015) proposed three approaches which are; IG, an AIS and a hybrid AIS-IG algorithms to solve the PFSP with the limited buffers between consecutive machines where the objective is to minimise the total completion time. Very recently, Pan et al. (2017) used an IG algorithm for the hybrid FSP in order to minimise the bi-criteria function of the weighted earliness and tardiness objective from the due window. Also, a comparative study between the IG algorithm and nine other competing approaches were given in this work, the IG algorithm showed the best performance against all of the nine approaches. Elias C. Arroyo (2017) addressed the scheduling problem of n jobs with arbitrary job sizes and non-zero ready times on m unrelated parallel batch machines with different capacities in order to minimise the total completion time. They provided a lower bound for the problem and a MILP model. To solve this problem, a metaheuristic based on the IG algorithm was proposed. Ribas et al. (2017) considered the parallel blocking FSP with minimising the makespan among lines. They presented a mathematical model along with some constructive and improvement heuristics to solve the presented problem where the constructive technique employed two methods which were completely different from those presented in the literature. These approaches are applied as generating initial solution techniques of an ILS and an IG algorithm, where they both combined with a VN search. Li et al. (2017) extended a simple IG algorithm for the two-sided assembly line balancing problem and introduced a modified NEH-based heuristic to obtain a high quality initial solution. Dubois-Lacoste et al. (2017) discussed the possibility of re-optimising the subsequence obtained from the destruction phase of the IG algorithm. For the PFSP with the objective of minimising makespan, the authors showed that the performance of the IG algorithm can be significantly improved with this extension. Also, they proved in experiments that the LS on subsequence of jobs is the key component of the powerful performance of the algorithm. Tasgetiren et al. (2017) proved that the IG algorithm performance depends significantly on the speed-up approach used. The parameters of the presented IG method were tuned through a design of experiments on randomly generated benchmark instances. Regarding the application of IG algorithm for the multi-objective PFSP, Framinan & Leisten (2008) pioneered the use of IG methodologies for solving the multi-objective scheduling problems. The authors presented an IG search technique to solve the PFSP in order to minimise both of the makespan and the flowtime. On the other hand, Minella et al. (2011) proposed an algorithm based on the IG approach for solving the multi-objective PFSP.

2.6.4 Biased Randomisation

The randomised or probabilistic are recent solution approaches in the field of COPs. Such methods are usually proposed for the problems under issues of uncertainty or local optima. These kind of algorithms have been used widely for various classes of COPs, such as: Scheduling Problems, Vehicle Routing Problems (VRP) (Belloso et al., 2017), Packing and Partitioning Problems, and more. For more details about randomised approaches we refer the reader to (Collet & Rennard, 2007). An example of the successful hybridisation of the BR with some heuristic was the application of the BR technique with the NEH heuristic that was used to generate the initial solution for Sim-heuristic approach that was proposed by Juan et al. (2014a). The authors applied the Discretised Decreasing Triangular Probability Distribution (DDT) to generate random variates at the BR step.

2.7 Stochastic Scheduling Approaches

When the processing times of jobs are considered as random variables whereas the population of jobs is assumed to be known in advance, the scheduling problem is called stochastic scheduling (Pinedo, 2016). In these types of problems, the random processing times of all jobs follow a specific probability distribution where μ_j is the expected value and σ_j is the standard deviation. Stochastic scheduling models have been mainly introduced since the 1980's where researches have traditionally concentrated on non-anticipative policies which intent to minimise the criteria in expectation. Additionally, it is usually supposed that the processing times of jobs are independent stochastically. A policy of scheduling is non-anticipative if its decisions about the jobs that must be scheduled at a time t depend only on the jobs which are already finished at time t and on the conditional distributions of the remaining processing times of jobs that are still active at this time. Rothkopf (1966) showed that for the scheduling problem of m immediately available jobs with random variable service times. It is certain that such problems can be reduced to equivalent deterministic problems. Möhring et al. (1984) investigated the analytic properties in scheduling of various classes of policies, also for special cases, the optimal policies were determined. Weiss (1991) and Weiss (1992) derived additive performance bounds for a PMSP without release dates in stochastic environment. In addition to this, Modarres et al. (1999) developed COP approaches for different scheduling problems in stochastic environment. The authors examined the power of linear programming based priority policies, and compared them to the expected performance of an optimal stochastic scheduling policy. Bertsekas & Castanon (1999) showed how rollout approach can be implemented in an efficient way, also they showed that the performance of these policies is local optima, and is substantially better than the performance of their underlying heuristics. The authors, concentrated on a

different class of scheduling problems in stochastic environments. Koole (2000) applied event-based dynamic programming to stochastic scheduling problems. Skutella & Uetz (2005) derived approximation policies for stochastic machine scheduling with precedence constraints. Kamburowski (2000) studied a stochastic 3-machines scheduling problem in Johnson's flow shops with the objective of minimising the expected total completion time. Alcaide et al. (2002) addressed the FSP with minimising the expected total completion time under machine breakdowns in stochastic environment. The authors presented a method that converted a scheduling problem under breakdowns into a finite sequence of problems without-breakdowns. Yang et al. (2004) studied using TS to optimise the parameters of a FSP. Empirical results showed TS as a promising method to solve the FSP. Wang et al. (2005) applied a Hypothesis-Test method incorporated into a GA for solving the FSP problem in stochastic environment and to avoid premature convergence of the GA. Hentenryck & Bent (2006) provided the main algorithm for online stochastic COPs, they have given an interesting review of many classical COPs in a stochastic environment such as; stochastic scheduling, stochastic VRP and stochastic reservations. Kalczynski & Kamburowski (2006) proposed a job sequencing rule which includes Talwar's and Johnson's rules for the 2-machines FSP so as to minimise the total completion time. In this problem, the processing times are assumed independently and follow the Weibull distribution. Liu et al. (2008b) proposed a class of PSO algorithm with SA and hypothesis test to solve the FSP with no-wait constraint in stochastic environment, where the criterion is to minimise the total completion time. The developed PSO algorithm showed better feasibility, effectiveness and robustness when compared to other proposed algorithms. Parajuli (2010) compared stochastic scheduling performance with deterministic scheduling, given that the problem involves stochastic processing times. He also focused on due date performance as a scheduling objective, considering that both early and tardy completion is undesirable. Baker & Altheimer (2012) applied heuristics for the stochastic FSP and general distributions for processing times. Almeder & Hartl (2013) dealt with a scheduling problem of a real-world offline stochastic FFSP with limited buffers. The scheduling problem with impatience to the end of service or impatience to the beginning of service in stochastic environment have been considered by Salch et al. (2013). The impatience of a job was considered as an uncertain due date and both of the processing times and due dates were stochastic variables. The criteria is to minimise the expected weighted number of tardy jobs. Elyasi & Salmasi (2013) presented a stochastic approach based on Chance Constrained Programming for two different scheduling problems; SMSP and 2-machine scheduling in stochastic environment. Cai et al. (2014) provided a comprehensive and unified coverage of studies in this area. Wang & Choi (2014) presented a Decomposition-based Holonic approach to solve the FFSP under stochastic processing times, in order to minimise the total completion time. Ebrahimi et al.

(2014) proposed two metaheuristics for the hybrid FSP with sequence dependent family setup time. The objective is to minimise the total completion time and the total tardiness. Also, the due date was considered as a stochastic variable following the Normal distribution. Baker (2014) proposed a developed B&B method for the stochastic SMSP in order to minimise the total expected earliness and tardiness costs. Aydilek et al. (2015) considered the setup and processing times as stochastic variables for the problem of a 2-machines production FSP with the criterion of minimising the total completion time. Saravanan & Pugazhenthi (2015) pertained to heuristic technique to obtain an optimal scheduling in PFSP where the jobs were associated with probability, and the criterion of minimising the makespan was associated with probability nature. Framinan & Perez-Gonzalez (2015) proposed some heuristics from the literature for the PFSP under stochastic processing times in order to minimise the expected total completion time.

2.7.1 Simulation-Optimisation

The terms Optimisation for Simulation or Simulation for Optimasation are commonly mentioned in the field of stochastic COPs (Amaran et al., 2017). Both the comprehensive surveys of Fu (1994) titled Optimisation via Simulation, and (Andradóttir, 1998), which was titled Simulation Optimisation, reflect the two terms mentioned previously. The main aim of hybridising simulation and optimisation is to handle the COPs in the presence of stochastic components. Recently, in stochastic scheduling, the Sim-Opt is used widely where heuristics or metaheuristics are used for the optimisation part. The concept of SA algorithm has been developed into an algorithm that can be used to solve a variety of optimisation problems. This was shown in work by Manz et al. (1989), where SA was used to optimise parameters for an Automated Manufacturing System Simulation. Sabuncuoglu & Kizilisik (2003) investigated the problems of reactive scheduling in a stochastic and dynamic flexible manufacturing systems. Tekin & Sabuncuoglu (2004) proposed a total survey on Sim-Opt approaches with emphasis introduced on modern developments. They provided a taxonomy about the existing approaches depending on the problem characteristics and discussed the main advantages and possible drawbacks of the various approaches. Fu et al. (2005) introduced a review of the important techniques of Sim-Opt and described some modern theoretical and algorithmic developments in the field of Sim-Opt. Simulation-heuristic algorithm has been applied successfully for other stochastic COPs. Work by Konak & Kulturel-Konak (2005) discussed optimising simulation problems using TS, including discussion of the profound effect that parameter selection has on the performance of the search. Grasas et al. (2016) presented the Sim-ILS approach that extends the ILS algorithm by combining simulation to provide the algorithm with the ability of dealing with stochastic COPs in a natural way. Juan et al. (2014a) presented a Sim-heuristic

approach for solving the PFSP under uncertain processing times. This approach hybridised an ILS metaheuristic with the MCS so as to handle the stochastic nature of the problem. A set of methods related with planning and/or scheduling, many of which are a hybridisation of optimisation and simulation have been presented by Pereira (2016).

As an alternative to the heuristic methods, metaheuristic approaches allow generation of high-quality solutions to the real-life COPs in relatively short computing times. For example, GA and PSO algorithm are showing a wide range of applicability and robustness for solving a wide range of different COPs and have received a large amount of research. In work by Joines et al. (2002), they used GA to optimise simulations of a supply chain to set optimal order quantity and time between orders. Koyama et al. (2004) worked on optimising routing algorithms with GA using a simulated system. Dahal et al. (2005) used a standard GA optimiser to solve a simulation of an actual port facility to minimise total costs by reducing delays. Jeong et al. (2006) developed a hybrid solution where the GA was used to optimise schedules and the simulation was applied to minimise the makespan of the last job while reflecting stochastic characteristics with the fixed input from the GA. Persson & Stablum (2006) was able to use GA to solve a multi-objective Mail Sorting Simulation Created in Arena. Gu et al. (2008) addressed the FSP with random breakdown and random repair time. They applied a Quantum Genetic Based Scheduling Algorithm for this problem, this approach combined stochastic simulation theory, stochastic programming, quantum computing and GA together. Juan et al. (2011) integrated routing metaheuristics with MCS for solving the VRP with stochastic demands; Cáceres-Cruz et al. (2012), also combined a MCS and routing metaheuristic to solve the inventory routing problem with stochastic demands and stock-outs. Wang et al. (2015b) addressed the PFSP under unknown processing times and applied a two-stage Sim-based hybrid Estimation algorithm to solve this problem. Juan et al. (2015) presented a review of Sim-heuristics by extending metaheuristics to deal with stochastic COPs. González-Neira et al. (2016) addressed the Integral Analysis method for the stochastic FFSP with the bi-objectives criteria, where the cardinal analysis implemented both a MILP model and a Sim-Opt technique for the TWT solution. Noura et al. (2016) applied a Sim-based GA with the stochastic MILP model to construct Quay Crane scheduling that account for the dynamics and the uncertainty inherent to container handling process. Finally, Frazzon et al. (2016) introduced and examined a Sim-Opt technique to solve the production and logistic processes along a global supply chain involving a production JSP and intermodal transport.

2.8 Benchmark problem

A comprehensive comparisons against well-known and established benchmarks of instances are frequently required in the field of COPs and when a set of benchmarks is recognised, then various approaches can be applied and compared depending on this set. Recently, there are benchmarks available online for different scheduling problems and for different sizes. Frequently, the best known solutions are the best known upper bounds in minimisation problems, these solutions are used so as to compare the proposed method. In any study of combinatorial optimisation benchmarks, poor quality instances can lead to experiments that are far removed from real manufacturing problems. Thus, the design of the benchmark is of extreme importance. Moreover, the benchmark instances might be too easy, limited size or specific for a given combination of input parameters. In such situations, if the proposed technique outperforms another in the set of instance, there is still no guarantees that the performance can be generalised over the real world instances. As we define previously, in the FSP there are a list of n jobs that proceed sequentially on all the set of m machines in series. Each job j has a non-negative, deterministic and known amount of processing time p_{ij} where $i = 1,...,m$ and $j = 1,...,n$. In the FSP the jobs start processing on the first machine and continue processing until the last machine m. The solution space that has all the possible sequences is given as; $n!$, since at each machine there are $n!$ possible permutations of job sequences. On the other hand, in the PFSP, there are only $n!$ possible sequences to be considered and once the sequence of jobs for the first machine is determined, is kept unchanged for all other machines. Taillard (1993) introduced the well-known benchmark for different sizes of deterministic PFSP. His benchmarks have been used widely in the literature of shop scheduling. Taillard benchmark was designed for the case of deterministic PFSP. However, the real world PFSPs are dynamic and stochastic in nature. Actually, there is a lack in the literature to consider dynamic and/or stochastic benchmark for PFSP and there are no standard benchmarks in the literature of dynamic and stochastic PFSP except the arguably limited benchmark of the SPFSP introduced by Baker & Altheimer (2012). Recently, Katragjini et al. (2013) introduced a novel benchmark for PFSP under different types of disruptions, they considered three different types of real-time event during time horizon, which are; machine breakdown, arrivals of new jobs and ready time variations. From our knowledge, there is no other given benchmark in the literature for the PFSP with different uncertainties. This benchmark available online in http://soa.iti.es/. Vallada et al. (2015) proposed a new benchmark of hard instances for the PFSP where the objective is to minimise the makespan. This benchmark consisted of 480 instances including 240 small size and 240 large size instances with up to 800×60 ($jobs \times machines$).

2.9 Conclusion

The dynamic and stochastic PFSP under different uncertainties are considered as essential studies in scheduling area. Such problems are widely studied due to the practical relevance of the applications, where the research in this area is growing faster than in the last decades. From the findings, current research on dynamic and stochastic PFSP under different uncertainties focused more on improving the schedule by minimisation of specific objective(s) such as makespan. In the same direction, others have attempted to reduce different objectives by using multi-objective models, e.g., the bi-objective model with the objectives of minimising both makespan and instability measures (Cowling et al., 2003), simultaneously. Furthermore, other researcher defined such models and applied different rescheduling approaches to accommodate different scheduling disruptions. Also, they proposed different efficient algorithms depending on the problem environment and the disruptions types, for example; the Sim-heuristic algorithm that was proposed for the SPFSP (Juan et al., 2014a). A central feature in dynamic and stochastic PFSP under different uncertainties is the source of dynamicity (stochasticity) in terms of machine breakdowns and new job arrivals (stochastic processing times). The aim in this case is to make the new schedule feasible and optimal after any types of different uncertainties (failures). In terms of multi-objective optimisation models, a literature review showed that this could be a competitive approach in addressing uncertainty for the dynamic and stochastic PFSP, when compared to the deterministic solution, as it does not require distributing assumptions on the uncertainty. Also, multi-objective optimisation models can be adapted to address stability and robustness (in addition utility) in order to accommodate the new disruptions with other solution approaches. There exists broad range of different approximate and exact methods have been used to solve dynamic and/or stochastic PFSP under different uncertainties. Exact methods have the ability to only solve relatively small size instances. On the other hand, approximate methods success to solve even large size problems in many cases. Hence, several promising ways of research are worth more attention. A better heuristic should be more flexible to accommodate the various disruptions in rescheduling approaches encountered in most of real-life applications. In terms of Sim-Opt, nowadays, the combination of Simulation and Optimisation is becoming quite popular in the research community. We presented several studies on Sim-Optimisation methodologies, which are related to the combination of simulation with heuristics/metaheuristic, in order to improve and to find a better way to solve COPs, in particular, SPFSP. Additionally, the advantages of these algorithms are that they are flexible, quite efficient and can be implemented in most practical applications. Also, the uncertainty modelling feature of MCS hybridised with efficient and fast heuristic/metaheuristic can create interesting approaches for real-life problems. Sim-Opt offers a practical perspective which is able to deal with more realistic scenarios; by integrating MCS in the heuristic/metaheuristic,

it is possible to naturally consider any probabilistic distribution for modelling the stochastic jobs processing times. To conclude a literature review showed that the use of Sim-Opt is a well-established and increasingly relevant topic in COPs. There is the potential applications of distributed computing to solve large-size PFSPs.

Chapter 3

Multi-objective Optimisation model for Robust PFSP under different disruptions

3.1 Introduction

In the literature of manufacturing scheduling, one of the gaps is presented by considering only classical efficiency performance measures for economic performances of the scheduling systems. Examples of these measures are; the maximum flow time, makespan, tardiness, earliness, and so on. As we mentioned previously, in real manufacturing systems, scheduling frequently operated in highly dynamic and uncertain environments where random disruptions may lead to non-optimal performances. Therefore, rescheduling actions will be required to re-optimise the new schedule. The deviation of the current schedule could cause significant impact such as additional costs in the case of storage costs, material handling costs, setup costs, and more. Thus, it is important to minimise additional objectives in scheduling systems to reduce any instability or deviations. The reminder of this chapter is orginised as follows; a multi-objective optimisation model for robust PFSP under different real-time events is proposed in section 3.2. Section 3.3 demonstrates the weighted objectives. In section 3.4, the uncertainties and real-time events are discussed. Finally, related conclusions are presented in section 3.5.

3.2 The proposed multi-objective optimisation model for robust PFSP

In this chapter, we propose a new scheduling multi-objective optimisation model based on the optimisation model introduced by Cowling et al. (2004) and Rahmani & Heydari (2014). The new proposed model has been extended for the case of n jobs and m machines for the PFSP. It

takes into account three important measures; utility (makespan), stability and robustness. The new multi-objective optimisation model (MSR) for robust PFSP is then given as follows;

$$Min\ MSR = \alpha U_n(S^*) + \beta I_n(S^*) + \gamma R_n(S^*) \tag{3.1}$$

Where:
S^* refers to the new schedule after the disruption. Similarly, S denotes the schedule before the disruption time t_D.
$U_n(S^*) = \sum_{j'} CR_{mj'}$
$I_n(S^*) = \sum_i \sum_{j'} |CR_{ij'} - CP_{ij'}|$
$R_n(S^*) = |\sum_{j'} CR_{mj'} - \sum_{j'} CP_{mj'}|$
In this model we used the following notations:
n is the number of jobs.
m is the number of machines.
j index for jobs $\{1,2,...,n\}$.
i index for machines $\{1,2,\ldots,m\}$.
j' index of jobs that have not been processed on any machine yet and the newly arrived job.
$CP_{ij'}$ is the predictive completion time of job j' on the machine i.
$CR_{mj'}$ is the real completion time of job j' on the machine i.
$U_n(S^*)$ is the real makespan in real scheduling.
$I_n(S^*)$ is the stability measure.
$R_n(S^*)$ is robustness measure.
$\sum_{j'} CP_{mj'}$ is the predictive makespan according to the initial schedule.
α, β and γ are weights used to indicate the importance of the objectives, where $\alpha + \beta + \gamma = 1$.
In this model, the utility measure is used to indicate the degree of optimisation for the schedule. While, the stability measure represents the difference between the completion times of the jobs in the baseline and the new schedule. Also, robustness is used to determine the deviation in performance of the baseline and new schedule.
In multi-objective optimisation problems things get complicated if the original functions are not in the same scale. Thus, the model (3.1) will be normalised to enable a reasonable comparison. The following model (NMSR) is the normalised objective function of model (3.1).

$$NMSR = \alpha NU_n(S^*) + \beta NI_n(S^*) + \gamma NR_n(S^*) \tag{3.2}$$

Where $NU_n(S^*)$, $NI_n(S^*)$ and $NR_n(S^*)$ represent the normalised makespan, instability and robustness, respectively and α, β, γ are the weight coefficients. The functions $NU_n(S^*)$,

$NI_n(S^*)$ and $NR_n(S^*)$ are calculated as following:
The normalised utility is giving by the following equation:

$$NU_n(S^*) = \frac{U_n(S^*) - Min(U_n)}{Max(U_n) - Min(U_n)} \tag{3.3}$$

Where $Max(U_n)$ and $Min(U_n)$ are the upper and lower bounds respectively for the makespan at the time of disruption t_D (Katragjini et al., 2013). To calculate $Min(U_n)$ we first define π_{AD} and π_{BD}, where π_{AD} is the partial sequence of jobs on the first machine at the time of disruption t_D, such that these jobs have already been executed or are in progress, and π_{BD} denotes the subsequence of jobs that have not proceed yet on the first machine and can be permuted. Now $Min(U_n)$ is calculated using the following steps:

1. Determine π_{AD} and π_{BD}.
2. Calculate $U_n(\pi_{AD})$ the makespan of π_{AD}.
3. Calculate $\sum_{j=1}^{n(\pi_{BD})}(P_{mj})$ The total processing time of all jobs of π_{BD} on the last machine.

Then the lower bound of makespan is as follows:

$$Min(U_n) = U_n(\pi_{AD}) + \sum_{j=1}^{n(\pi_{BD})}(P_{mj}) \tag{3.4}$$

To calculate $Max(U_n)$, first we determine π_{AD} and π_{BD}. For every job $j's$ in π_{BD}, the job j start to be executed only after the previous job is terminated in the sequence. This calculation process is explained in example in figure (3.1). From this figure, job number 3 is the first job in π_{BD}, and is starts proceeding after the termination of job number 5. Similarly, job number 1 starts after the termination of job number 3.
Also, figure 3.2 explain how to determine j' and n' where the only permutation order of jobs is 3 and 1 can be changed. The partial fixed sequence including the jobs that have already been executed or are in progress on the first machine at the moment of the disruption is defined by π_{AD}. As well π_{BD} denotes the permutable subsequence containing the jobs whose succession order can be modified.

The normalised stability is given as follows:

$$NI_n(S^*) = \frac{I_n(S^*) - Min(I_n)}{Max(I_n) - Min(I_n)} \tag{3.5}$$

Where $Min(I_n)$ and $Max(I_n)$ represent the lower and upper bounds for instability at the moment of disruption t_D. $I_n(S^*)$ represents the instability calculated as the sum of operations whose starting times have been anticipated or delayed in the new schedule S^*.

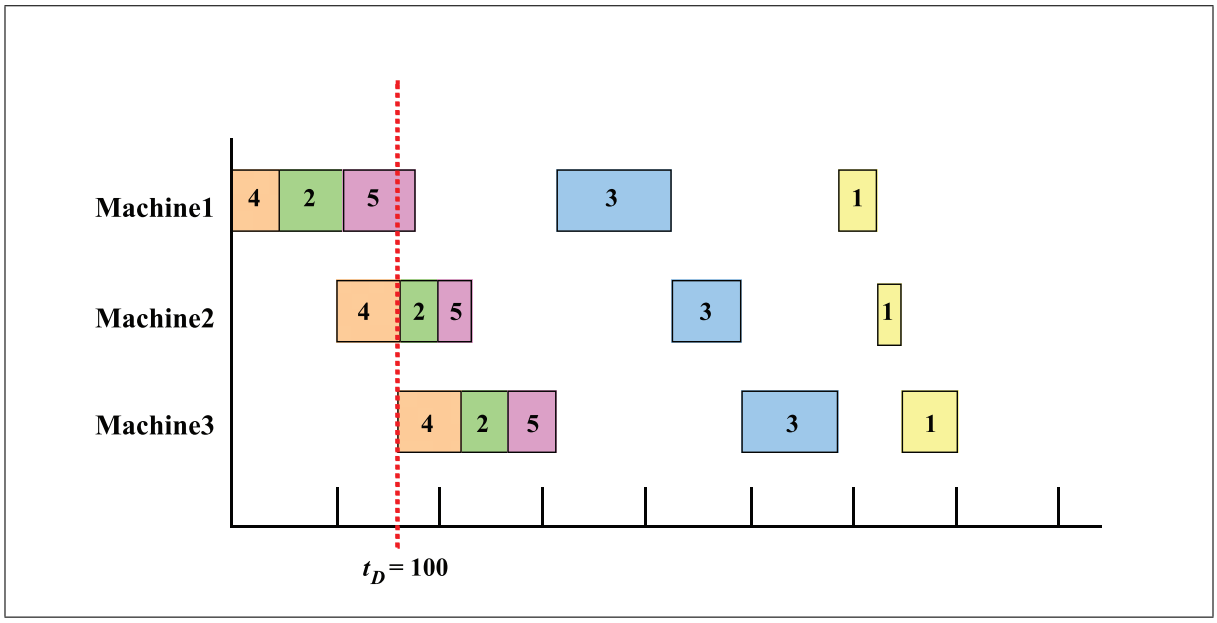


Fig. 3.1 Example of calculating $Max(U_n)$ for PFSP where $n = 5$ and $m = 3$

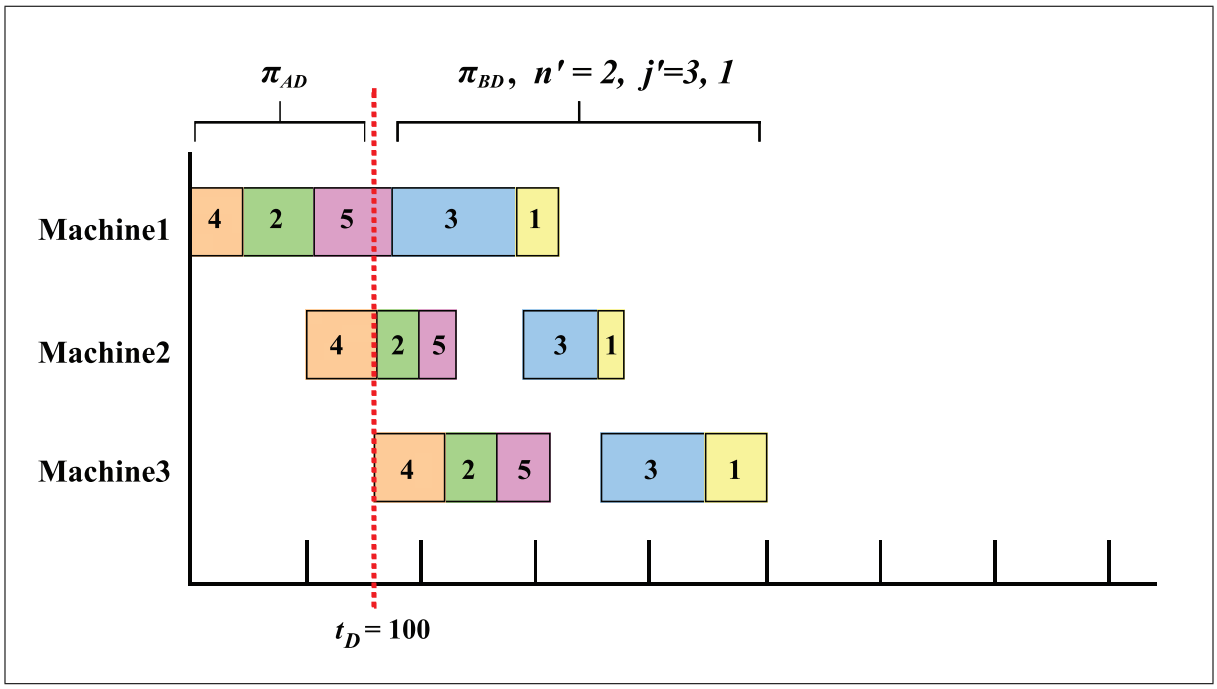


Fig. 3.2 Example showing how to determine j' and n' for PFSP where $n = 5$ and $m = 3$

$I_n(S^*) = \sum_{i=1}^{m} \sum_{j=1}^{n} K_{ij}$ and

$$K_{ij} = \begin{cases} 1 & ,|q_{ij}^* s - q_{ij} s| > h \\ 0 & ,otherwise \end{cases}$$

Where $q_{ij}^* s$ denotes job $j's$ starting time on machine i after the rescheduling, and q_{ij} s refers to the starting times of the same task before the disturbance. h is a parameter to indicate an alteration of an operation's starting times up to h time units such that schedule stability is not affected. By setting its value to 0 we consider the more general situation in which every single change contributes to the instability of the final value. The normalised robustness function is defined as follows:

$$NR_n(S^*) = \frac{R_n(S^*) - Min(R_n)}{Max(R_n) - Min(R_n)} \tag{3.6}$$

Where $R_n(S^*)$ is the robustness after the disruption time t_D. $Max(R_n)$ and $Min(R_n)$ are the upper and lower robustness bounds respectively. To obtain these bounds, we first consider every criterion to solve the problem alone, then according to three criteria, each problem should be solved three times. The results are shown as a 3×3 matrix in equation (3.7)

$$\begin{array}{c} \\ U_n^* \\ I_n^* \\ R_n^* \end{array} \begin{array}{c} \begin{array}{ccc} U_n & I_n & R_n \end{array} \\ \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} \end{array} \tag{3.7}$$

In matrix (3.7), the first row corresponds to the case where considering model (3.1) with weight $\alpha = 1, \beta = 0, \gamma = 0$ is solved. The second and third rows correspond to the state that by considering the problem with weights $\alpha = 0, \beta = 1, \gamma = 0$ and $\alpha = 0, \beta = 0, \gamma = 1$ are solved. Values of $Min(R_n)$ and $Max(R_n)$ are calculated as follows:

$$Min(R_n) = min\{a_{1,3}, a_{2,3}, a_{3,3}\}$$

$$Max(R_n) = max\{a_{1,3}, a_{2,3}, a_{3,3}\}$$

In the next chapters, the MSR model is compared against the bi-objective model (Katragjini et al., 2013) and the classical makespan model. The next section explains the way of getting the values of weights α, β and γ.

3.3 Weighted objectives

The aim of designing different optimisation models is to determine the best model that minimises the objective function by changing design variables while satisfying design constraints. It is often that during design optimisation it is required to consider objective functions or several design criteria simultaneously. The multi-objective optimisation problem can then be defined as the problem of finding "a vector of decision variables which satisfies constraints and optimises a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term 'optimise' means finding such a solution which would give the values of all the objective functions acceptable to the decision maker" (Osyczka, 1985), it is also called multi-performance, multi-criteria optimisation or vector optimisation problem. The most common and simple approach used for multi-objective optimisation is based on summarising the multi-objectives in a new single objective, where the multiple objectives are transformed into an aggregated objective function by using a linear combination of the proposed weights. This approach is called the Weighted Sum. As state previously, the weight factors in the proposed MSR model are; α, β and γ where $0 \leq \alpha, \beta, \gamma \leq 1$. These weights are factors of the utility, stability and robustness objective functions, respectively. The weighted sum $\alpha + \beta + \gamma = 1$ is said to be a convex combination of objectives. The decision parameters α, β and γ are used to indicate the importance of each of the three objectives. The optimisation approach of weighted objectives is also referred to as the "a Priori" optimisation, since the weights are calculated before the optimisation process. Although weighted approach was applied for many multi-objective COPs, it is still unobvious how the weights should be established. Thus, a Revised Weight Sensitivity algorithm introduced by Jones (2011), which is performed to examine a part of weight space that is important to the decision maker in a multi-objective scheduling problems. This work is based on the work of Jones & Tamiz (2010) that allows to explore the whole weight space, but even so, many decisions have a priority for some data beside their initial weighting estimate. Such important information avoid the method of exploring the whole weight space by reducing the area which is required to be searched.

3.3.1 Initial estimate of weights

1. The starting point of the method which is an initial estimate by the decision maker.

2. A valid initial point for the method, this is obtained by using the Equal Weighting technique, with the favorite weights of all the unfavorable deviations being given the value of one.

3. Supply a starting point, the solution obtained by using the pair wise comparison methods (Saaty, 1981) after the decision maker(s) supplies pair wise information according to the importance of the objectives.

There are an additional favorable data employed to restrict the exploration weight space area where the decision maker consider only one or hybrid of any of the following potential forms of favorable data expressions:

- Absolute information about the relative importance of a single weight.
- Absolute information about a set of weights.
- Pair wise ordinal information regarding weights.
- Pair wise cardinal information regarding weights.

The revised weight sensitivity algorithm is then given in figure 3.3.

1. Select initial starting point
2. Let $S = \phi$
3. Let $w =$ Initial set of weights
4. Add $[SolveWGP(w)]$ to S
5. For $n = 1$ to $TMax$
6. For each subset T^* of deviations of cardinality n
7. Form new vector w^*by:
8. $Calc_{max}(T^*, max\_weight\_vector)$
9. Set $w = max\_weight\_vector$
10. If $[SolveWGP(w)]$ is not in S then add $[SolveWGP(w)]$ to S
11. Let $w = w\_low, w^* = w\_up$
12. Examine_Weight_Line $(w_{low}, w_{up}, 1)$
13. End_For
14. Next n
15. End
16. Subroutine Examine_Weight_Line $(w_{low}, w_{up}, level)$
17. If $[SolveWGP(w_{low})] = [SolveWGP(w_{up})]$ then EXIT
18. If $level > MaxLevel$ then EXIT
19. Form a new weight vector w_{mid} by setting the weight of each
20. deviation to $\frac{(w_{low}+w_{up})}{2}$
21. If x^* of $[SolveWGP(w_{mid})]$ is not in S then add $[SolveWGP(w_{mid})]$ to S
22. Examine_Weight_Line $(w_{low}, w_{mid}, level + 1)$
23. Examine_Weight_Line $(w_{mid}, w_{up}, level + 1)$
24. End

Fig. 3.3 The revised weight sensitivity algorithm (Jones, 2011)

Where the parameter $TMax$ is used to control the number of varied weights simultaneously, $MaxLevel$ is a parameter to control the maximum number of bi-sections of the line of direction between the maximum level and initial estimate. In addition, $[SolveWGP(w)]$ is a sub-code which solves the dependent weighted goal programme using a weight set w to the unfavorable deviations in the accomplishment objective. Also, $Calc_{max}(T^*, max\_weight\_vector)$ is a sub-program used to compute the maximal level of weight that can be parted among the deviations in T^* in ratio with the proportions of importance given in the initial solution whilst staying within the bounds specified by the additional preference information supplied by the user. The remaining weight is then shared amongst the remaining weights whilst coming as close as possible to maintaining the ratios of importance given in the initial solution. In some cases this

can be calculated very simply from the preference information given while in other situations a combination of two priority level lexicographic goal programme with the weight values as the decision variables and the preference information as constraints must be constructed from the preference information and solved to find max_weight_vector.

3.3.2 A revised weight sensitivity algorithm for MSR model

In this thesis, thirteen different weights (α, β, γ) are derived from the Practical Weight Sensitivity algorithm to evaluate the performance of the proposed multi-optimisation model, these weights represent the relative importance of each objective in model (3.1). The selection of the thirteen weights is based on a Practical Weight Sensitivity algorithm, which explained above. By setting $TMax = 1$, $MaxLevel = 1$ and an sequential weight starting solution is set to be one, ten of these weights are given in Table (3.1):

Table 3.1 The weights values

Solution	α	β	γ
W_1	0.333	0.333	0.333
W_2	0.666	0.166	0.166
W_3	0.498	0.498	0.002
W_4	0.416	0.416	0.166
W_5	0.166	0.666	0.166
W_6	0.002	0.498	0.498
W_7	0.166	0.416	0.416
W_8	0.166	0.166	0.666
W_9	0.498	0.002	0.498
W_{10}	0.416	0.166	0.416

In this Table, the weights are represented as $W_i, i = 1, 2, \ldots, 10$. The remaining weight are the unity weights $(1,0,0)$,$(0,1,0)$ and $(0,0,1)$, these weights are only used to obtain the lower and upper bounds that are used to demonstrate the normalised model (3.2).

3.4 Uncertainties and real-time events

The proposed optimisation model is used for the dynamic PFSP under different types of disruptions. In this thesis, we proposed these benchmark instances for the PFSP under different uncertainties. In the first part, we consider the dynamic PFSP under different real-time events including; machine breakdown and arrival of new jobs. We also consider these two disruptions

in addition to the stochastic processing times for the SPFSP in the second part. These real-time events interrupt the initial predictive schedule individually or both simultaneously. Katragjini et al. (2013) simulated the baseline of shop floor execution by generating different real-time disruptions randomly at time t where $0 \leq t \leq C_{max}(BL)$ and $C_{max}(BL)$ represents the makespan of the predictive baseline BL. They generated different disruptions till the end of the baseline time horizon and not of the revised schedule for two reasons; the first reason is that new jobs arrive continuously into the schedule sequence, which could delay the completion time of the revised schedule and hence the process of disruption generation would be unending if halted at the completion of each revised schedule. The second reason is that the authors aim to generate a confined benchmark of disruptions and ensure reproducibility of the results when comparing different rescheduling techniques. Since the revised schedule clearly depends on the algorithm providing the best solution, the disruptions generated after the completion time of the baseline are strongly related to the shop floor status determined by this algorithm and hence they cannot ensure the reproducibility of results similar to a simulation process. They try to avoid lengthy and difficult-to-reproduce simulation processes. Moreover, unless the new job arrival rate is set to a very high level, as time goes by the number of jobs to be scheduled decreases and the problems resolved at every rescheduling point tend to become trivial.

3.4.1 Machine breakdown

It is assumed that the breakdown time and interval are not known a priori. Then the schedule disruption is simulated to generate random machine breakdowns at time t where $0 \leq t \leq C_{max}(BL)$. The failure time duration is detected directly after the disruption occurs, where the failure times are generated by applying a uniform distribution in the range $U[1,...,99]$. In this case, a job that is preceded due to a machine breakdown resumes its processing from the point at which the event occurred.

3.4.2 New jobs arrivals

The scenario of a dynamic problem is considered in this research by generating arrivals of new jobs randomly to the scheduling system. In other words, there is a probability of generating one new job arrival at every point t where $0 \leq t \leq C_{max}(BL)$. All jobs are characterised by the arrival time, which is the time they enter the system, the ready times that identify the time at which they can be released to the shop floor, and the processing times of operations on all shop floor machines. The distribution of the processing times for the new jobs is fixed to $U[1,...,99]$ following Taillard's processing times generation. All the disruptions are saved as a rescheduling event benchmark, which can be found on http://soa.iti.es/. It should be noticed

that there does not exist any similar benchmark of disruptions in the literature, even for a single type of disruption.

3.4.3 Stochastic processing time

For each PFSP instance in the Taillard's problems, we consider the processing time of each job as a random variable following a well-known probability distribution with a given mean and a given variance. In other words, we replaced the deterministic processing time p_{ij} to stochastic processing times P_{ij} with $E[P_{ij}] = p_{ij}$ of job i on machine j. Any probability distribution with a known mean could be used for modeling processing times. It should be noticed that, in a real-world systems, historical data would be employed to generate each processing time by a different probability distribution. The Log-Normal distribution is used when modeling non-negative processing times (Juan et al., 2014a), it is a more natural choice than the Normal distribution. Thus, a Log-Normal distribution is selected to be used in this thesis. This probability distribution has two parameters; the location parameter, μ_{ij}, and the scale parameter, σ_{ij}. According to the properties of the Log-Normal distribution, these parameters will be given by the following expressions:

$$\mu_{ij} = ln(E[P_{ij}]) - 1/2(1 + Var[P_{ij}]/[E[P_{ij}]^2) \tag{3.8}$$

$$\sigma_{ij} = \left| \sqrt{\ln\left(1 + \frac{Var[P_{ij}]}{E[P_{ij}]^2}\right)} \right| \tag{3.9}$$

3.4.4 Interaction between real-time events

It may have a machine breakdown, a new job arrival and/or stochastic processing time simultaneously during the time horizon. At the beginning of every disruption, reactive actions are used to cope with the disruptions and to preserve a balance between schedule performance, stability and robustness. Also, when the processing time is a random variable then simulation is applied to handle the stochastic behavior.

3.5 Conclusion

The rescheduling literature, provide evidence of the lack of a standard methodology when dealing with dynamic and stochastic manufacturing settings and the existence of a gap between theory and practice in production scheduling. In this work we introduced a new multi-objective

optimisation model for robust PFSP under different types of disruptions. The proposed model considers three different measures namely; makespan, stability and robustness to handle the effect of different disruptions on the schedule. We have addressed the FSP rescheduling under two different types of disruptions that dynamically affect the shop floor layout, these disruptions are; machine breakdowns, new job arrivals. However, real-life manufacturing operations are affected by other types of events that need to be accommodated. Therefore, we also consider the stochastic case when the stochastic processing time is adding to the shop floor. These three disruptions are very common in every day manufacturing operations and negatively affect the overall system performance. Hence, this work introduced the generation of a new disruptions and benchmark by (Katragjini et al., 2013) as explained in details previously. The shop floor layout considered in this work is a PFSP, yielding very stiff permutations that can be reoptimised only partially at every rescheduling point.

Part I

Dynamic PFSP under different real-time events

Chapter 4

Particle Swarm Optimisation Algorithm for Robust PFSP

4.1 Introduction

In the literature of COPs, evolutionary methods have been widely used to solve different types of these problems. The PSO algorithm belongs to the category of evolutionary computation optimisation family that was introduced by Kennedy & Eberhart (1995). Evolutionary algorithms are usually inspired by nature such as the well-known GAs, Bee colony Optimisation and Ant Colony Optimisation. The PSO algorithm simulates a social behaviour such as swarm bird migration. It is a stochastic optimisation technique, and hence, it is a good option for many dynamic and stochastic COPs. The PSO algorithm optimises the problems using improvement solutions in a multi-dimensional space. It conducts a search using what is called a swarm (population) of individuals and updated Iteratively, also, the individuals are called particles. Each particle represents a solution (a candidate position) to the problem. A particle is treated as a point in an n-dimension space, the position and velocity of each particle characterises its status. The PSO algorithm stores populations of particle swarm, which moves around in the solution space; the best position associated with the best fitness value of the particle obtained so far is called the personal best, also the best experience or position ever found by all particles is called global best. The nature of the PSO algorithm is fairly robust to changes caused by random disruptions in a dynamic scheduling environment. This algorithm belong to the metaheuristic techniques and it has been proposed by many researchers because of its advantages over more traditional methodologies, the following are some advantages of this algorithm:

1. The dynamic and stochastic nature of the algorithm and its simplicity of implementation (Blackwell, 2007); (Li et al., 2006); (Zhang et al., 2015).

2. Another features are the use of self-information, individual best information and global best information to generate effective and optimal results, as well as the convergence speed of the swarm is very high (Blum & Merkle, 2008).

3. The main advantage of PSO is that, it requires fewer parameters to be adjusted when compared to different optimisation methods (Li et al., 2012).

The contribution of this chapter is to apply the PSO algorithm for the predictive-reactive approach with the MSR model for the robust dynamic PFSP under different types of real-time events. The chapter is structured as follows; in section 4.2, the framework of the PSO algorithm for dynamic PFSP is introduced. Section 4.3 gives an example to illustrate the procedure of the PSO algorithm for the PFSP. Section 4.4 provides the experimental results. The related conclusions are introduced in section 4.5.

4.2 Predictive-reactive based PSO framework for robust PFSP

In order to solve the PFSP under different disruptions, the PSO algorithm has to be defined. First of all, the PSO algorithm is proposed to solve the dynamic PFSP under different real-time events. Also, the MSR model is proposed to maintain the stability and robustness for this problem. The predictive-reactive approach is proposed to control the effect of different real-time events on the scheduling process. Thus, the methodology is to apply the predictive solution first, then rescheduling is triggered when a real-time event is indicated in the scheduling system. There are two real-time events that disturb the scheduling system, these are; machine breakdowns and new job arrivals. These real-time events interrupt the initial planned schedules simultaneously. At the beginning of every machine failure, rescheduling PSO algorithm is applied to accommodate the new disruption, while the MSR model is used to preserve the stability and robustness of the system. As well as, when a new job arrives to the system, the PSO algorithm is applied at the reactive stage to reschedule partial sequence of jobs such that the new job is located in an optimal sequence. Figure 4.1 shows the framework of the predictive-reactive approach and how it applies to the PSO algorithm at the reactive stage.

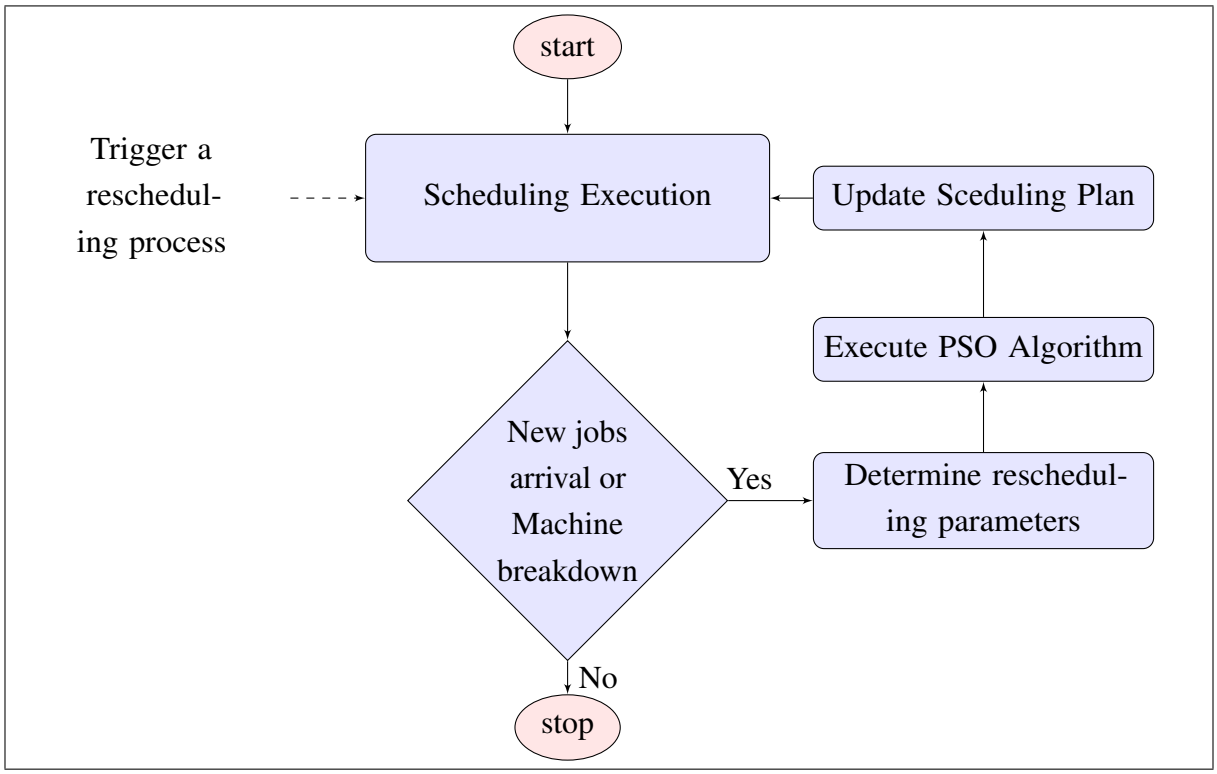


Fig. 4.1 Predictive-Reactive based PSO approach

In the PSO algorithm, the swarm consists of n number of particles. Each particle corresponding to a position vector and a velocity vector where the position vector represents the position particle in the solution space, while the position vector is used to create a schedule from the particle position. This vector is decoded as explained in section 4.2.3 in order to represent a solution to the problem. The PSO algorithm is an iterative procedure where each particle move its position in the search space in order to improve the solution quality. Every single particle continue looking for the best solution, this is named as the personal best. Also the algorithm continues looking for the best solution from the personal best found in the swarm, this is called the global best. The particles velocity control the movement of a particle to a different position. The velocity of a particle is updated at each iteration and the particle change it position in the search space, which generates a new position vector. The velocity has all information about the way of moving the position of the particle from the position of the global best solution in the whole swarm and the personal best solution that found by the particle itself. The dimension of position, velocity, personal best and global best vectors are σ. This dimension represents the required number of columns or variables in order for the position vector to accurately represent a schedule to the problem. To explain PSO algorithm let us define the following:

Problem dimension: n is the problem dimension (the number of jobs) and j is the index of the problem dimension $j = 1, 2, \ldots, n$.

Population size: σ is the population size (the number of particles) and i is the index of population size $i = 1, 2, \ldots, \sigma$.

Iteration number: t is the number of iterations.

Particle: X_i^t is the i^{th} particle (position vector) in the swarm at iteration t, it consists of n particles that is $X_i^t = x_{i1}^t, x_{i2}^t, \ldots, x_{in}^t$ where x_{ij}^t is the position value of particle i of job j at iteration t.

Particle velocity: V_i^t is the i^{th} particle velocity (velocity vector) at iteration t, it consists of n particle velocities $V_i^t = v_{i1}^t, v_{i2}^t, \ldots, v_{in}^t$ where V_{ij}^t is the velocity of particle i of job j at iteration t.

Population: The set consists of σ particles in the swarm at iteration t is called a population po^t where $po^t = X_1^t, X_2^t, \ldots, X_\sigma^t$.

Permutation: The permutation of job sequence implied by the particle X_i^t is defined as $\pi_i^t = \pi_{i1}^t, \pi_{i2}^t, \ldots, \pi_{in}^t$ where π_i^t is the assignment of job j of the particle i in the permutation at iteration t.

Inertia Weight: The inertia weight w^t is used to control the impact of the velocities from the previous step on the current velocity.

Personal Best: The personal best P_i^t represents the best position of particle i with the best fitness at iteration t where $P_i^t = p_{i1}^t, p_{i2}^t, \ldots, p_{in}^t$ and p_{ij}^t is the position value of the i^{th} personal best with respect to j. In a minimisation problem with the objective function $f(\pi_i^t)$, the personal best P_i^t of the i^{th} particle in the swarm can be obtained such that $f(\pi_i^t) \leq f(\pi_i^{t-1})$ where π_i^t is the corresponding permutation of P_i^t and π_i^{t-1} is the corresponding permutation of P_i^{t-1}. The fitness function of P_i^t is simplified as $f_i^{pb} \leq f(\pi_i^t)$.

Global Best: The global best among all the swarm of particles achieved so far is called global best G^t. It is defined as $G^t = g_1^t, g_2^t, \ldots, g_n^t$ where g_i^t is the position value of G^t. To obtain the global best we use the criterion $f(\pi^t) \leq f(\pi_i^t)$ where π^t and π_i^t are the corresponding permutation of global best G^t and personal best P_i^t respectively. For simplicity we use f^{gb} for the fitness function of the global best instead of $f(\pi^t)$.

Termination Criterion: The search process of PSO algorithm will be terminated after a maximum number of iterations or a maximum CPU time.

Now the algorithm can be described after defining these notations. The general structure of the PSO algorithm is given in figure 4.2 below.

1. $t = 0$
2. Initialise particles X_i^t;.
3. **While** termination condition are unsatisfied **do**
4. $t = t + 1$;
5. Update w^t;
6. Select f_i^{pb} for each particle;
9. Select f^{gb} from X_i^{t-1};
11. Calculate particle velocity v_{ij}^t
12. Update particle positions x_{ij};
14. end

Fig. 4.2 General structure of the PSO algorithm

An initial solution is required to start the PSO algorithm. Every single particle is placed randomly in the search space and its position is evaluated. After the evaluation of all the solutions, the global best solution can be identified and hence decide whether to keep the old solution or replace it with the current one. The fitness function is given as $f(\pi_i^t) = NMSR$ and it is rewritten as f_i^t in short. The PSO algorithm then moves to the next iteration. These steps are explained in details in the following subsections.

4.2.1 The initialisation of the PSO algorithm for the PFSP

The PSO algorithm is start at iteration $t = 0$, also the population size is set to $\sigma = 2 \times n$ where n is the number of dimensions (Tasgetiren et al., 2004). The initial population of particles generated randomly and the initial position values for the particle are generated randomly as follows:

$$x_{ij}^0 = x_{min} + r_1(x_{max} - x_{min}) \tag{4.1}$$

Where $i = 1, 2, \ldots, \sigma$, $[x_{min}, x_{max}] = [0, 4]$ and $r_1 \in (0, 1)$ is a uniform random number determined later as discussed in Section 4.3. Once the position vectors have been initialised, the initial velocities are established similarly as follows:

$$v_{ij}^0 = v_{min} + r_2(v_{max} - v_{min}) \tag{4.2}$$

where $[v_{min}, v_{max}] = [-4, 4]$ and $r_2 \in (0,1)$ is a uniform random number.
The next step is to apply the decoding rule (described in Section 4.3) to the vector $X_i^0 = x_{i1}^0, x_{i2}^0, \ldots, x_{in}^0$ for every particle to obtain π_i^0. We use this schedule to compute the value of f_i^0 and thus for every particle we have:

$$P_i^0 = X_i^0 \tag{4.3}$$

and

$$f_i^{pb} = f_i^0 \tag{4.4}$$

for $i = 1, 2, ..., \sigma$. The final step in fully initialising the PSO is to set the global best vector P_g^t and the best objective value (makespan) of the swarm as follows:

$$g^0 = P_z^0 \tag{4.5}$$

and

$$f_i^{gb} = \min_i (f_i^{pb}) \tag{4.6}$$

for $i = 1, 2, ..., \sigma$. where the z^{th} particle position is the one that yields the lowest objective value out of all the particles in the swarm. At this point, all the particles have been initialised and their position in the search space has been evaluated. The algorithm begins its iterative procedure as described in following section.

4.2.2 PSO Algorithm

After all the particles in the swarm have been initialised, the inertia weight (w^0) is given its initial value. According to Tasgetiren et al. (2004), a reasonable value for the inertia weight, which starts from 0.9 and never decreased below 0.4. The inertia value decreases the importance of the previous velocity vector. Therefore, as the value of w increases, the previous velocity has a bigger impact on the new velocity. If the value of w decreases, the previous velocity becomes less relevant.

Step 1: Go to next iteration;

$$t = t + 1 \tag{4.7}$$

Step 2: Update inertia weight;

$$w^t = w^{(t-1)} \times \beta_{PSO} \tag{4.8}$$

Where β_{PSO} is the decrement factor. The decrement factor is similar to the cooling rate of the temperature in SA. In this case, as time goes by β_{PSO} decreases the inertia weight, which reduces the importance of the prior iteration. The selected value for β_{PSO} for this problem is discussed in Section 4.3. In general, it ranges from $(0,1)$.

Step 3: Update velocity;

$$v_{ij}^t = w^{(t-1)} v_{ij}^{(t-1)} + c_1 r_1 (p_{ij}^{(t-1)} - x_{ij}^{(t-1)}) + c_2 r_2 (g_{ij}^{(t-1)} - x_{ij}^{(t-1)}) \tag{4.9}$$

Equation (4.9) above is repeated along every dimension of a particle in order to populate the velocity vector for particle i. This has to be repeated for each particle of the swarm. In this equation, c_1 and c_2 are constants. It was found that a reasonable value for both constants is 2 (Tasgetiren et al., 2007). However, further analysis (presented in Section 4.3) yields different values. Finally, r_1 and r_2 are uniformly distributed random numbers from 0 to 1. If after updating the velocity any value along any dimension (i.e. v_{ij}^t) exceeds v_{max} or v_{min}, then the corresponding value is replaced with v_{max} or v_{min}, respectively.

Step 4: Update position;

$$x_{ij}^t = x_{ij}^{(t-1)} + v_{ij}^t \tag{4.10}$$

The above equation, just like with velocity, is repeated for every dimension of a particle and for all particles in the swarm. It should be noted that if the value of x_{ij}^t exceeds either x_{max} or x_{min} then it is replaced by x_{max} or x_{min}, respectively.

Step 5: Schedule decoding; Once the new position vectors are obtained, utilise the procedure described in the next section to decode a solution. This solution will yield a schedule π_i^t, for the position vector of every particle in the swarm.

Step 6: Update personal best; Each particle of the swarm is evaluated according to the corresponding π_i^t obtained in step 5. If $f_i^t < f_i^{pb}$, then $f_i^{pb} = f_i^t$; this is repeated for every particle. If the new f_i^t is better than f_i^{pb}, we also update the personal vector as $P_i^t = X_i^t$ and $\pi^t = \pi_i^t$. On the other hand, if $f_i^t \geq f_i^{pb}$, then we leave the current value of f_i^{pb} unchanged and we set $P_i^t = P_i^{(t-1)}$.

Step 7: Update global best; We find the minimum value of personal best in the whole swarm, that is $f_z^{pb} = \min_i[f_i^{pb}]$. If $f_z^{pb} < f^{gb}$, then $f^{gb} = f_z^{pb}$, and $g^t = X_z^t$ and $\pi^t = \pi_i^t$. Otherwise, f^{gb} remains unchanged and $g^t = g^{(t-1)}$.

Step 8: Stopping; If the stopping criterion is reached, then the procedure is stopped. Otherwise, return to Step 1. A common stopping criterion is to set a predetermined maximum number of iterations. If the maximum number of iterations has been reached, then π^t is the optimal schedule and the makespan of the assignment is given by f^{gb}. Additional stopping criteria can be implemented. For example, the algorithm may be stopped once a certain number of iterations are carried out, during which there have been no improvements to the global best. In this case, π^t would also be the optimal schedule and the makespan of the assignment would be given by f^{gb}.

4.2.3 Decoding of Solution

The decoding mechanism is considered as the key elements in the PSO algorithm, it is also the most challenging step of the procedure. The decoding step is defined as how a particle position vector is mapped into a solution to the scheduling problem. There are several techniques that have been used for this purpose. The most common method is the Smallest Position Value (SPV) rule that was introduced by Tasgetiren et al. (2004). Another method was proposed by Sha & Hsu (2008), who have presented a coding mechanism that uses a priority list vector representation to map a solution to the problem of open shop where the lowest value has a higher priority. Lian et al. (2006) used a different coding methodology for solving a JSP with the objective of minimising the makespan. They translated an $n \times m$ matrix into a sequence where jobs are sorted by location in numerical order. In this thesis, the SPV rule is used, which is explained within the example given in section 4.3. Thus, the pseudocode of the PSO algorithm is given in figure (4.3).

1. Set initial iteration $t = 0$ and $\sigma = 2 \times n$.
2. Generate σ initial particles $X_i^0 = \{x_{i1}^0, x_{i2}^0, ..., x_{in}^0\}$ where x_{ij}^0 is selected randomly from the range $[0,4]$.
3. Generate σ initial velocities $V_i^0 = \{v_{i1}^0, v_{i2}^0, ..., v_{in}^0\}$ where v_{ij}^0 is chosen randomly from the range $[-4,4]$.
4. Use the SPV rule to detect $\pi_i^0 = \{\pi_{i1}^0, \pi_{i2}^0, ..., \pi_{in}^0\}$ of particle X_i^0.
5. Apply the fitness function f_i^0 to evaluate each particle i in the swarm.
6. For each particle set $P_i^0 = X_i^0$ where $P_i^0 = \{p_{i1}^0, p_{i2}^0, ..., p_{in}^0\}$, $p_{i1}^0 = x_{i1}^0, p_{i2}^0 = x_{i2}^0, ..., p_{in}^0 = x_{in}^0$ together with its best fitness function $f_i^{pb} = f_i^0$.
7. Detect the best fitness value in the whole swarm such that $f_L^0 = min\{f_i^0\}$ with its corresponding particle X_L^0. Set global best to $G^0 = X_L^0$ such that $g_1^0 = x_{L1}^0, g_2^0 = x_{L2}^0, ..., g_n^0 = x_{Ln}^0$ with its fitness value $f^{gb} = f_L^0$.
8. Update the iteration $t = t + 1$.
9. Update the inertia weight $w^t = w^{t-1} \times \beta_{PSO}$ where β_{PSO} is the decrement factor.
10. Update the velocity as follows $v_{ij}^t = w^{t-1} v_{ij}^{t-1} + c_1 r_1 (p_{ij}^{t-1} - x_{ij}^{t-1}) + c_2 r_2 (g_j^{t-1} - x_{ij}^{t-1})$ where c_1 and c_2 are social and cognitive parameters and $r_1, r_2 \in (0,1)$ are uniform random numbers.
11. Update position values $x_{ij}^t = x_{ij}^{t-1} + v_{ij}^t$.
12. Use the SPV rule to detect the permutation $\pi_i^t = \{\pi_{i1}^t, \pi_{i2}^t, ..., \pi_{in}^t\}$.
13. Use the permutation to evaluate particles by checking if there is any improvement at iteration t for the personal best. That is, if $f_i^t < f_i^{pb}$, then P_i^t is updated as $P_i^t = X_i^t$ and $f_i^{pb} = f_i^t$.
14. Find the minimum value of P_i^t as $f_L^t = min\{f_i^{pb}\}$, $L \in \{i | i = 1,2,...,\rho\}$. If $f_L^t < f^{gb}$, then update the global best as $G^t = X_L^t$ and $f^{gb} = f_L^t$.
15. If the number of iteration exceeds the maximum number of iterations, then stop; otherwise go back to step.

Fig. 4.3 PSO algorithm for the PFSP

4.3 An Example of the PSO Algorithm for the PFSP

To demonstrate the PSO methodology to the PFSP, we explain the procedure of the algorithm for a single particle and for small number of iterations. In this case, we will assume a PFSP with three jobs and two machines, and hence the problem dimension is equal to the number of jobs, which means $n = 3$. As we mentioned above, the population size σ is set to be twice of the problem dimension, so $\sigma = 6$. This means the dimension of both of position and velocity vectors is six. Also, the decrement factor β_{PSO} is set to 0.975 and w^0 is set to 0.9

(Tasgetiren et al., 2007). As well as, $c_1 = c_2 = 2$, $[x_{min}, x_{max}] = [0,4]$ and $[v_{min}, v_{max}] = [-4,4]$. The processing times for this example are given in Table 4.1.

Table 4.1 Jobs processing times

Machines	J_1	J_2	J_2
Machine 1	7	3	3
Machine 2	1	5	5

The first step is starting with set $t = 0$ and use equations (4.1) and (4.2) to initialise X_i^0 and V_i^0 respectively. Now, we apply the SPV rule as shown in Table 4.2. This Table shows these initial values where π_i^0 represent the sequence of jobs by applying SPV rule. Note that the SPV rule sort the jobs by sorting the position values X_i^0 in decent manner.

Table 4.2 Positions, velocity and sequence of jobs

Jobs	1	2	3
X_1^0	0.329	2.453	3.559
V_1^0	-0.397	-1.136	3.838
π_1^0	3	2	1
X_2^0	1.224	3.904	1.433
V_2^0	-1.570	0.098	-1.057
π_2^0	3	1	2
X_3^0	3.406	1.267	1.881
V_3^0	-3.763	2.997	-3.521
π_3^0	1	3	2
X_4^0	2.137	0.694	0.339
V_4^0	0.015	2.432	-1.847
π_4^0	1	2	3
X_5^0	0.455	2.213	0.174
V_5^0	-2.013	1.609	-2.517
π_5^0	2	1	3
X_6^0	0.965	3.242	0.260
V_6^0	-1.767	-1.285	-2.594
π_6^0	2	1	3

Once all the jobs have been assigned, we calculate the fitness function (makespan) corresponding to each position value X_i^0. Since we are in the first initial iteration $t = 0$, we set $P_i^0 = X_i^0$. Also, since the dimension is 3, so we have six sequences of jobs, which are; π_1^0, π_2^0, π_3^0, π_4^0, π_5^0 and π_6^0. The makespan for these sequences are shown in Table 4.3 below:

Table 4.3 Fitness functions

Schedule	π_1^0	π_2^0	π_3^0	π_4^0	π_5^0	π_6^0
f_i^0	14	16	20	20	18	18

From Table 4.3 the personal best will be $P_1^0 = X_1^0$, $P_2^0 = X_2^0$, ..., $P_6^0 = X_6^0$ because this the initial step of the algorithm. It is obvious that the global best will be given as follows; $g^0 = f(\pi_1^0)$.
Now we set $t = 1$. The velocity is then updated according to the updated iteration described in the algorithm (see figure 4.3) as follows;

$$v_{ij}^1 = w^0 v_{ij}^0 + c_1 r_1 (p_{ij}^0 - x_{ij}^0) + c_2 r_2 (g_{ij}^0 - x_{ij}^0) \tag{4.11}$$

Once the velocity vector has been updated, we calculate the new values for the position vector as follows;

$$x_{ij}^1 = x_{ij}^0 + v_{ij}^1 \tag{4.12}$$

The new values of position vector and velocity vector for particle s are shown in Table 4.4 below.

Table 4.4 Positions, velocity and sequence of jobs

Jobs	1	2	3
X_1^0	-0.058	1.350	7.608
V_1^0	-0.387	-1.104	4.048
π_1^0	3	2	1
X_2^0	0.189	4.975	0.413
V_2^0	-1.035	1.070	-1.020
π_2^0	3	1	2
X_3^0	-0.044	4.401	-1.275
V_3^0	-3.450	3.134	-3.157
π_3^0	2	1	3
X_4^0	2.491	3.107	-1.455
V_4^0	0.353	2.413	-1.794
π_4^0	2	1	3
X_5^0	-1.470	4.123	-2.296
V_5^0	-1.925	1.910	-2.469
π_5^0	2	1	3
X_6^0	-0.488	2.045	-2.290
V_6^0	-1.453	-1.197	-2.550
π_6^0	2	1	3

Where $c_1 = c_2 = 2$ and r_1, r_2 are random variables from the interval $(0,1)$. Finally, the inertia weight will updated as follows:

$$w^1 = w^0 \times \beta_{PSO} = 0.9 \times 0.975 = 0.877$$

Now the new position vector is arranged using the SPV rule and the current schedule and its corresponding makespan values are shown in Table 4.4. Also, the new fitness functions are given in Table 4.5.

Table 4.5 Fitness functions

Schedule	π_1^1	π_2^1	π_3^1	π_4^1	π_5^1	π_6^1
f_i^0	14	18	18	18	18	18

At this point, from Table 4.5, the value of f_1^1 is equal to f_1^0, for this, we have the choice to keep or replace the previous personal best by setting $P_1^1 = P_1^0$ and we apply the same procedure for the remaining values of $f_i^1, i = 2,3,4,5,6$. Thus, we have: $P_2^1 = P_2^0, P_3^1 = P_3^0, P_4^1 = P_4^0, P_5^1 = P_5^0$ and $P_6^1 = P_6^0$. In this iteration, the global best f^g requires to be recomputed based on the new personal best values of the swarm. However, in this example the value of particle x_1^1 does not change, which yields an unchanging global best and concluding the first iteration. The movement of the algorithm to a better solution is continue until stopping criteria yield. As can be seen from the few iterations of the algorithm (Table 4.5), the value of the objective function does not improve but it still the best. It is important to acknowledge that the algorithm can not reach a better solution after more iterations are carried out. Furthermore, the performance of the algorithm may improve if there are different values for job processing time and different values for the parameters are set.

4.4 Experiment Results

This section gives the experimental results and comparison study for the PSO algorithm. This algorithm has been applied for the dynamic PFSP under different real-time events, where the PSO algorithm is triggered for the predictive-reactive approach with the MSR model and used to maintain the problem stability and robustness. All experiments are coded in Java, eclipse platform and run on an Intel Cori5 2.6 GHz PC with 6GB of memory RAM. In the PFSP under different real-time events, three models are used and compared namely; the MSR model, the bi-objective model of (Katragjini et al., 2013) (it has only makespan and stability objectives) and the classical makespan model. A sensitivity analysis using a Practical Weighted Analysis approach (Jones, 2011) is also coded to test the versatility of the MSR model in producing differing Pareto efficient solutions with respect to the three objectives. The Jones (2011) algorithm produced thirteen distinct weight sets (α, β, γ), each representing different levels of relative importance of the objectives in the MSR model. The parameters used in this algorithm are as follows:
$TMax = 1$, $MaxLevel = 2$ and a sequential weight starting solution is set to be one. The unity weights are applied to obtain the normalised model of the MSR model, these three weights are; $(0.999, 0.001, 0.001)$,$(0.001, 0.999, 0.001)$,$(0.001, 0.001, 0.999)$ While the sets of remaining ten different weights are using to test this experiment. These weights are given in Table 3.1. Furthermore, the bi-objective model given in (Katragjini et al., 2013) is also evaluated using only the first (α) element of weights sets W_1 to W_{10}. These elements are listed as follows:

$$\alpha = 0.333, 0.666, 0.498, 0.416, 0.166, 0.002, 0.166, 0.166, 0.498, 0.416$$

Taillard (1993) has provided extensive sets of generated test problems for minimising makespan in PFSP. The total number of problems he generated are 120, including 12 different size of problems ranging from 20 to 500 jobs and 5 to 20 machines. He has provided 10 different instances for each PFSP from the same size. The performance of the PSO algorithms with different weights are evaluated by using the benchmark set of Katragjini et al. (2013). They have reported the PFSP under different real-time events and its predictive solution for all Taillard's benchmarks, which can be found in http://soa.iti.es/. The real-time events are simulated such that they interrupt the system in specific disruption points t_D, for example, with the PFSP face machine breakdown disruption, the schedule disruption is simulated by generating random machine breakdowns at time t, $0 \leq t \leq C_{max}(B)$. For each instance, the baseline B is generated by applying the IG algorithm. Also, machine breakdowns happen only on busy machines, in other words, machines do not undergo failures during idle times. In addition, the downtime duration is detected directly after the event occurs. At this point, the down times are obtained from applying the uniform probability distribution in the range $U[1,\ldots,99]$. Another assumption is that no other real-time event is recorded on the same machine before the breakdown event is recovered. At most, only one machine can have a breakdown event at time t. For more information about the real-time events including the new job arrival see chapter 3.

In the PSO algorithm, the permutation representation is used, and the population size is taken as twice the number of jobs. Also, the stopping criteria of the algorithm is depending on the number of iterations and it is set to be $3 \times n$. Regarding the PSO parameters, social and cognitive parameters are taken as $c_1 = c_2 = 2$ consistent with the literature (Tasgetiren et al., 2004). Initial inertia weight is set to $w^0 = 0.9$ and is never decreased below 0.4. Finally, the decrement factor β_{PSO} is taken as 0.975. The solution quality is measured with the relative percentage deviation (RPD), to be more specific, RPD is computed as follows:

$$RPD = \frac{M - Best_{Sol}}{Best_{Sol}} \times 100$$

where M is the solution obtained by the proposed model and PSO algorithm. $Best_{Sol}$ is the average of 10's Taillard instances of lower bound solution from the same size. The predictive-reactive approach with the PSO algorithm are applied for the MSR model, the bi-objective model introduced by (Katragjini et al., 2013) and the classical makespan model, we run each instance five independent times to obtain more reliable results for the proposed model and algorithm.

Table 4.6 shows the results with respect to the RPD. In terms of the objective weights we solve the problem for 10 weights as shown in chapter three, then we select the solution corresponding to the weight that associated with lower values of the RPDs. In comparison to different weights,

the weight $W_8 = (0.166, 0.166, 0.666)$ had the lowest RPD values which were relatively better results.

Table 4.6 RPD for MSR and bi-obj models using the PSO algorithm

	Ta	20×5	20×10	20×20	50×5	50×10	50×20	100×5	100×10	100×20	200×10	200×20	500×20	**Average**
W_1	**MSR**	29.446	22.907	20.532	27.014	22.872	23.768	15.085	23.259	20.938	22.923	22.559	17.737	22.420
	bi-obj	34.345	6.838	21.109	25.106	22.311	24.256	28.078	30.075	22.222	25.297	22.860	20.764	23.605
W_2	**MSR**	29.381	23.974	20.558	27.083	22.548	23.157	17.130	26.133	21.384	24.939	22.734	17.765	23.066
	bi-obj	31.384	24.540	20.702	24.847	22.241	23.259	22.343	30.012	21.158	25.517	21.896	23.592	24.291
W_3	**MSR**	29.062	23.921	20.621	27.905	23.539	23.203	18.788	26.151	21.636	24.693	22.518	17.960	23.333
	bi-obj	32.807	23.506	20.76	23.575	22.308	24.023	25.127	32.409	21.901	27.009	20.427	22.8	24.721
W_4	**MSR**	29.512	23.612	20.706	28.567	22.775	23.372	16.81	23.805	21.54	24.415	22.973	17.69	22.981
	bi-obj	32.611	23.065	20.961	25.234	23.332	23.988	24.437	33.679	20.91	25.176	22.83	22.763	24.916
W_5	**MSR**	30.796	23.75	21.892	28.64	23.369	23.09	15.386	24.082	21.604	22.86	22.6	17.44	22.959
	bi-obj	31.769	23.618	23.52	25.811	23.375	24.774	24.453	32.38	22.264	25.187	23.909	24.521	25.465
W_6	**MSR**	26.871	23.025	21.274	16.737	21.771	23.672	14.022	21.687	19.604	11.474	19.088	12.117	19.279
	bi-obj	31.278	24.514	24.129	23.684	25.36	24.999	23.103	34.158	23.255	24.553	23.554	20.359	25.246
W_7	**MSR**	28.163	23.447	20.138	19.146	22.358	23.624	14.085	20.92	19.896	11.331	19.082	13.794	19.665
	bi-obj	31.769	25.285	23.52	25.811	23.375	24.774	24.453	32.38	22.264	25.187	23.909	24.521	25.604
W_8	**MSR**	26.609	23.157	20.876	19.003	21.461	22.75	13.781	20.831	19.993	11.533	18.843	12.277	19.260
	bi-obj	31.769	24.514	23.520	25.811	23.375	24.774	24.453	32.38	22.264	25.187	23.909	24.521	25.54
W_9	**MSR**	29.422	23.664	20.464	29.659	22.184	22.267	14.98	23.648	22.117	23.118	22.889	17.177	22.632
	bi-obj	32.807	24.514	20.76	23.575	22.308	24.023	25.127	32.409	21.901	27.009	20.427	22.8	24.805
W_{10}	**MSR**	29.798	23.684	20.648	29.882	22.451	21.983	15.434	24.007	21.96	22.564	20.303	17.031	22.479
	bi-obj	32.611	23.065	20.961	25.234	23.332	23.988	24.437	33.679	20.91	25.176	22.83	22.763	24.916

In the first row of this Table, Ta is the problem of size of $n \times m$. As well as, the bi-obj denotes the solution values obtained by using the bi-objective model of Katragjini et al. (2013), where $\alpha = 0.166$. According to Table 4.6, the objective functions are sensitive to different sets of weights. The results show that the stability and robustness measures play an important role in obtaining better solution. As shown in Table 4.6, the weights $W_6 = (0.002, 0.498, 0.498)$, $W_7 = (0.166, 0.416, 0.416)$ and $W_8 = (0.166, 0.166, 0.666)$ produce better solutions in comparison with other weights. Furthermore, the solution corresponding to the weight $W_8 = (0.166, 0.166, 0.666)$ has in general the lowest RPD values, this shows that giving more priority to the robustness measure leads to better solutions. For this reason, the weight $W_8 = (0.166, 0.166, 0.666)$ is selected in this experiment.

Now, to compare the quality of solution for different models, the MSR and the bi-obj are compared with the classical model of makespan where is denoted as Utility for the RPD values corresponding to the weight $W_8 = (0.166, 0.166, 0.666)$ as shown in figure 4.4. In this figure, it is obvious that the RPD values relating to the MSR model are generally less than the values of both of the bi-obj and Utility models.

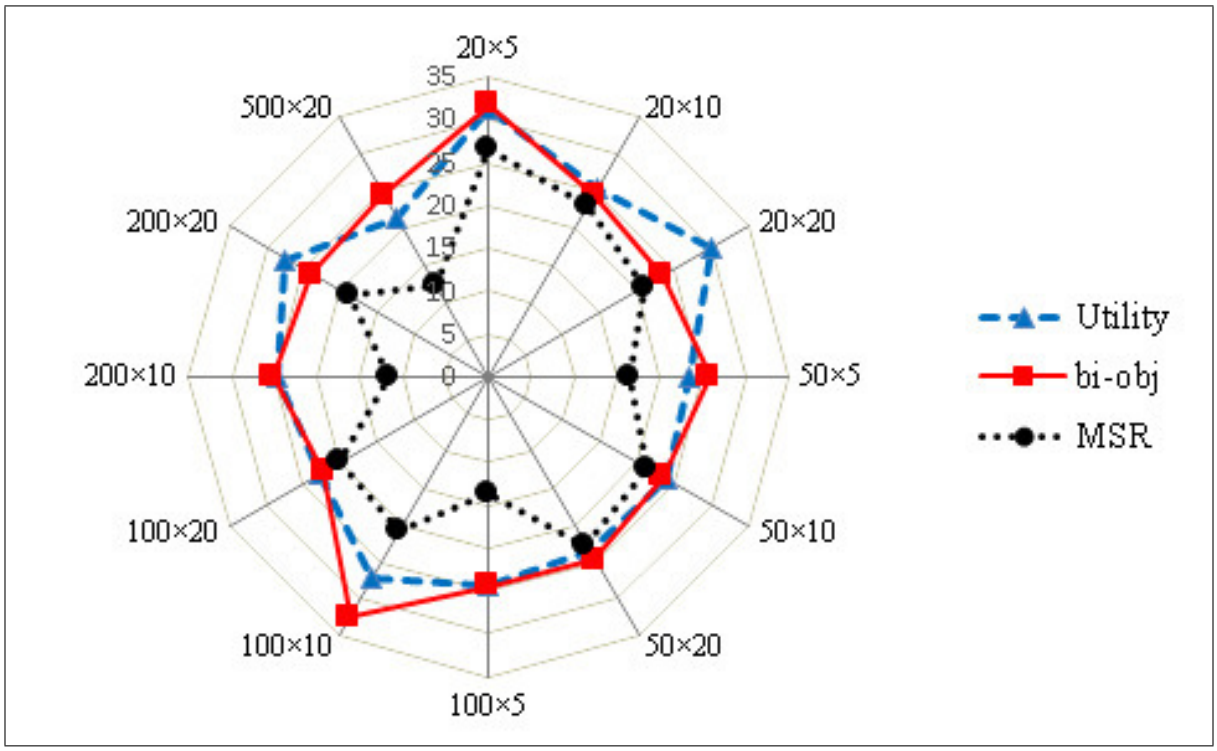


Fig. 4.4 RPD for all models with weight W_8 using the PSO algorithm

To further investigate the impact of the proposed predictive-reactive based PSO and models on the dependent variable RPD, the single factor mean of an Analysis of Variance (ANOVA) is performed. The results of MSR model corresponding to the weight $W_8 = (0.166, 0.166, 0.666)$, bi-obj and Utility model are statistically tested by ANOVA. For our analysis with the three models, the null and alternative hypotheses in ANOVA are:

H_0: all means are same.

H_A: at least one mean is different.

The F-ratios and the p-values are reported in Table 4.7. Since the p-value is smaller than 0.05, we reject the null hypothesis of no difference, and conclude these factors have a statistically significant effect on RPD at the 95% confidence level. The ANOVA F-test only permit to reject the null hypothesis, but it does not give indication about which groups have different mean values. Hence, the 95% confidence interval is used to specify which of the mean RPD differences are statistically significant as shown in figure 4.5.

Table 4.7 ANOVA between models using the PSO Algorithm

Groups	**Count**	**Sum**	**Average**	**Variance**
MSR	12	242.196	20.183	22.17
bi-obj	12	308.617	25.718	17.768
Utility	12	307.253	25.604	9.702

ANOVA						
Source of Variation	*SS*	*df*	*MS*	*F*	*P*-value	*F* crit
Between Groups	240.165	2	120.082	7.257	0.002	3.284
Within Groups	546.053	33	16.547			
Total	786.218	35				

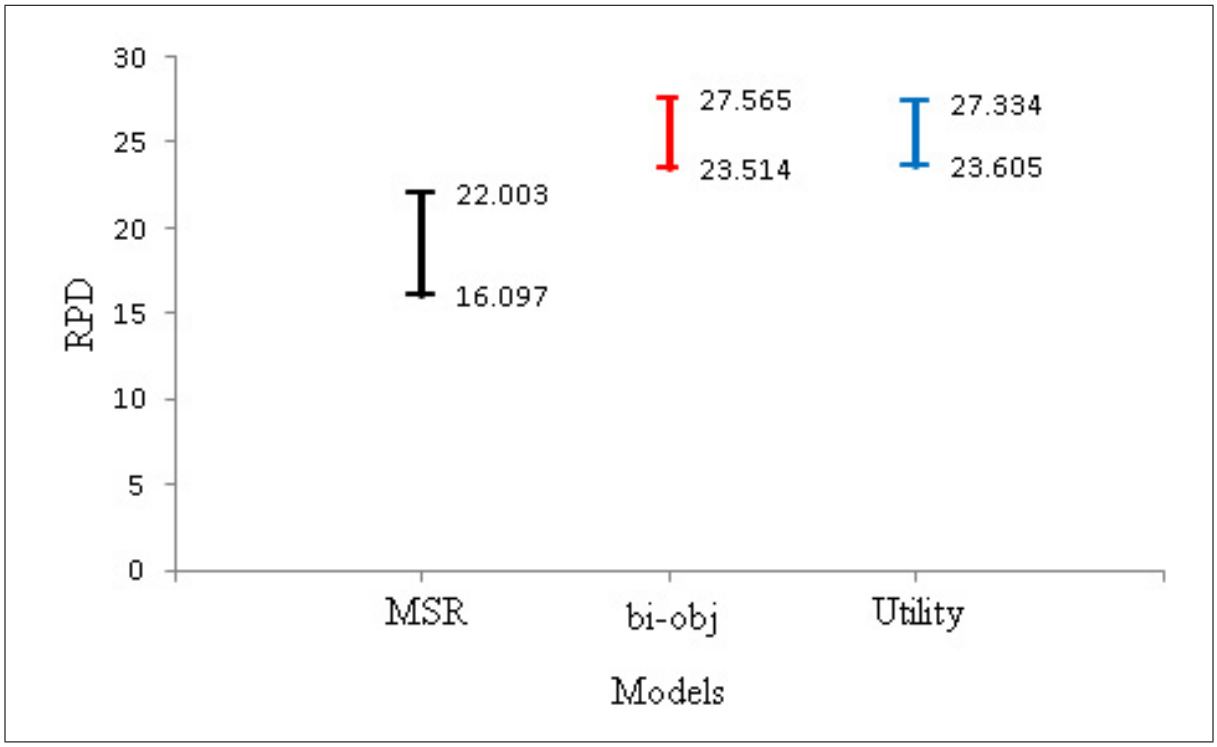


Fig. 4.5 95% Tukey confidence interval for all models using the PSO algorithm

This figure shows that the MSR model has significance difference in comparison with the other two models. This support the good performance of the proposed MSR model when comparing with the other two models, and hence, the importance of the robustness measure in getting better performance.

The times required to reach the solution is given in Table 4.8.

Table 4.8 Computational time of PSO algorithm in seconds

Ta	20×5	20×10	20×20	50×5	50×10	50×20	100×5	100×10	100×20	200×10	200×20	500×20
C-T (S)	0.044	0.041	0.105	0.27	0.364	0.488	1.35	1.923	2.126	10.728	16.051	292.2

In this Table, the C-T(S) represent the computational time in seconds, which is obtained from the average of five independent runs. The PSO algorithm reach good quality results in the time mentioned in Table 4.8.

4.5 Conclusion

In this chapter, we applied the multi-objective model proposed in chapter 3, which considers utility, stability and robustness to obtain robust and stable schedules for the PFSP under different real-time events. Also, we proposed a predictive-reactive approach based on generating robust schedule and reacting at every disturbance point, the PSO is applied for rescheduling at the reactive stage. This algorithm shows the ability to deal with the dynamic nature of this problem successfully. On the other hand, a bi-objective model Katragjini et al. (2013) and the classical model of makespan are compared with our proposed model. For this, an ANOVA comparative study is applied to compare the efficiency of these models. The ANOVA results revealed that the MSR model has a significant effect on the RPD than other referred models. Also the computational results of sensitivity analysis indicated that giving higher priority for robustness measure leads to better solutions. Even though it is not included in this work, a LS may also be implemented in the future to achieve better solutions and avoid being trapped in a local optimal solution. Similarly, generating an initial solution to start the PSO algorithm instead of starting randomly using efficient and simple heuristic such as NEH algorithm may significantly improve the algorithm solution.

The PSO algorithm is a population-based method and it requires a relatively huge time to reach a high quality solution as shown in Table 4.8. For this reason, in the following chapter, we propose an efficient and simple heuristic method, which is an IG algorithm to reduce the computational time and examine the ability of this simple algorithm to reach a better quality solution than the PSO algorithm.

Chapter 5

Iterated Greedy Algorithm for Robust PFSP

5.1 Introduction

Often, heuristic methods are applied to solve COPs, which are very complicated to be solved to optimality. In the previous chapter, we explained the PSO algorithm for the dynamic PFSP under different types of real-time events. However, the PSO algorithm is a metaheuristic approach that requires different parameters and consume a considerable computational time. Thus, in this chapter, we proposed a simple and efficient heuristic approach, which applied successfully for the PFSP with the objective of minimising the makespan (Ruiz & Stützle, 2007). This algorithm is known as an IG algorithm, which needs less parameters at the applications. Also, the NEH heuristic algorithm and LS technique are used to improve the algorithm quality, and hence, we explained these approaches in details this chapter. A possible definition of the IG algorithm could be, a heuristic method that aims to optimise single or multi-objective function, it is going from one solution to another solution by generating a sequence of solutions. These sequences are obtained by iterating over Greedy Constructive (GC) heuristics. The algorithm has two main phases, namely; destruction and construction phases. In the destruction phase some jobs are selected to be removed from a previously constructed complete candidate set of jobs. Then during the construction phase a GC heuristic is applied to reconstruct a complete candidate solution. After completing a candidate solution set, the algorithm attempts to improve the current solution by applying an optional LS method. Then an acceptance criterion is called into action to decide whether the newly constructed solution will replace the incumbent solution. Finally, the algorithm continue iterating these steps while some stopping criterion is satisfied. An initial solution is required to start the algorithm, The NEH constructive heuristic (Nawaz

et al., 1983) is used to generate such initial solution. It should be noticed that the constructive of the NEH procedure is also used in the construction phase of the IG algorithm. The IG heuristic algorithm is closely similar to ILS (Lourenço et al., 2003), the main difference between these two algorithms is that the ILS is iterating over a LS while IG algorithm iterates in an identical way over construction heuristics. IG algorithm is a conceptually simple yet powerful heuristic that has proven to be very efficient in solving complex COPs (Juan et al., 2014c). In the next sections, we explain the NEH heuristic algorithm that is used to initialise the IG algorithm. Also, the optional LS technique is explained.

5.2 Predictive-reactive based IG framework for robust PFSP

In many industrial processes machine failures continuously affect the planned activities. Preventive maintenance may reduce the breakdown rate, but it is almost impossible to eradicate this type of disruption from the system. Similarly, other parameters such as material availability and market demand are highly likely to undergo modifications and, hence, it is crucial to react rapidly and produce new schedules that take into account the new system variables. The stage of applying the IG algorithm mentioned below is designed as a deterministic heuristic for the dynamic PFSP under different real-time events. Hence, it is necessary to develop a predictive-reactive approach which can absorb the impact of different real-time events while maintaining high shop performance. For this, the predictive-reactive approach is applied for the dynamic PFSP. In this approach, the procedure starts with a predictive solution, then rescheduling is triggered in response to unexpected different types of real-time events that alter the current system status. Figure 5.1 illustrates the predictive-reactive approach and how it uses the IG algorithm at the reactive stage.

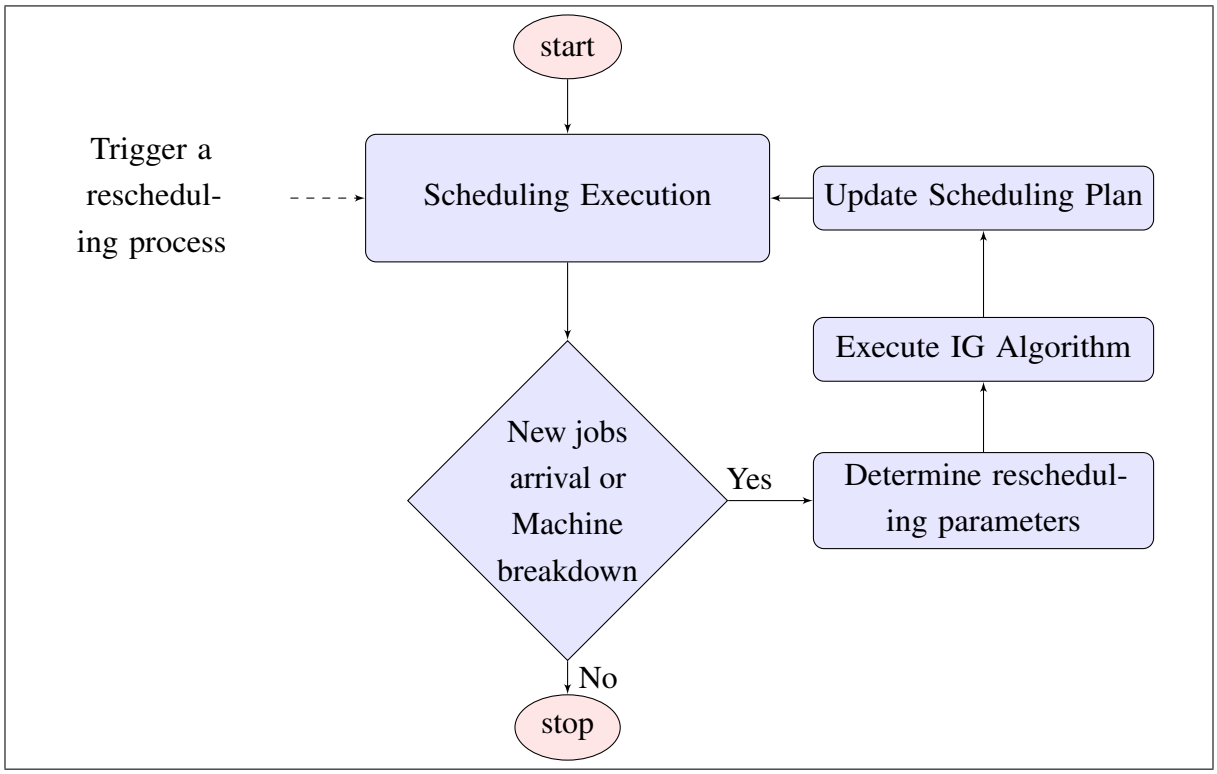


Fig. 5.1 Predictive-Reactive based IG approach

5.2.1 The NEH constructive heuristic

This section discusses the heuristic NEH algorithm in the context of scheduling problems. This algorithm is used to initialise an initial solution for the IG algorithm. The NEH heuristic, which stands for the first letters of the authors, Nawaz, Enscore and Ham (Nawaz et al., 1983), has been distinguished as the highest performing old version method, it was designed for the PFSP under the objective of minimising the total completion time (makespan). This algorithm is commonly used to generate initial solutions for most modern algorithms designed to solve the PFSP. For this, we use the NEH heuristic to generate an initial solution to initialise IG algorithm. The idea of the NEH heuristic is very simple. First of all, the job list is sorted in the decreasing order according to jobs total processing time. Thereafter, the jobs of indexes 1 and 2 in this sequence are scheduled in order to obtain a sequence that minimises the partial makespan. And so on for all the remaining jobs in the sequence, for each job, all possible positions in the partial list are explored then the position that minimises the partial makespan is selected, that way keeping the relative order of jobs. The procedure is continued until the last job in the sequence is included into the partial list, the obtaining list is the list of jobs that minimises the total makespan. The complexity of NEH heuristic algorithm is given by $O(n^2)$

where n is the number of jobs and m is the number of machines. The heuristic NEH algorithm can be stated as in figure 5.2.

1. **procedure**
2. NEH_Jobs_List = sort_Jobs_Using_NEH_Criterion
3. NEH_Sol = NEH_Algorithm(NEH_Jobs_List) % NEH solution
4. base_Sol = NEH_Sol
5. $nIter = 0$
6. **while** cost(base_Sol)$\geq$ cost(NEH_Sol) and $nIter < nJobs$ **do**
7. $nIter = nIter + 1$
8. new_Jobs_List = NEH_Job_List
9. new_Sol = NEH_Algorithm(new_Jobs_List)
10. **if** new_Sol = getCost(new_Sol) < getCost(base_Sol) **then**
11. base_Sol = new_Sol
12. **end if**
13. **end while**
14. Return baseSol
15. **end procedure**

Fig. 5.2 The NEH heuristic Algorithm

5.2.2 Local Search approach

LS approach is an optional step in IG algorithm (as we explain in the next section), it is a simple and efficient method to improve the solution in many COPs. In this section, we explain what the LS method is. First of all, let us consider the general mathematical definition of LS, for the following general minimisation problem:

$$\min_{x} f(x)$$

$$\textit{Subject to } x \in U$$

where $U \subseteq R^n$ is a subset of non-negative values and $f : R^n \rightarrow R$ is a real valued function. The key characteristic of a LS method used to solve this problem is that it is an iterative procedure which starts with some $x_0 \in R^n$(not necessarily feasible) and in each iteration i attempts to move to an x_i with some relation to $x_{(i-1)}$; x_i has to be a neighbour of $x_{(i-1)}$ (a proper definition of a neighbour will be given later on). The move is accepted if x_i is a 'better' solution than $x_{(i-1)}$, if the moves rejected we take $x_i = x_{(i-1)}$. The point x_i is called a state.The idea of LS procedure is sufficiently simple, first of all the solution space is the space containing all solutions to the

problem, it includes feasible and infeasible solutions. Now, in LS method the search process considers all list of solutions and select the feasible solution that gives minimum makespan. The method moves in the solution space, and it perturbs the given solution in each iteration. LS considers the observation by replacing a current feasible or infeasible solution with an obtained different solution. The method then records the best obtained feasible solution so far after checking each solution feasibility if required, in this case the best obtained feasible solution is called the incumbent solution. The stopping criteria of the method is finally applied to terminate the procedure after the stopping criteria condition is met, such as the number of iterations without any improvement or running time to the incumbent solution. To give more clear idea about LS method, we represent the current set of solutions as S, where S is produced by executing an operations (Oper) on solution set S, which is named the neighbourhood $N(S)$ of the current solution. By applying a neighbourhood operator $No(S)$ to S, several parameters are usually taken by the operator $No(S)$ to indicate which jobs of S will insert in the perturbation list. The neighbourhood size $N(S)$ of S based on the different parameters considered for $No(S)$. The neighbourhood of S is constructed and evaluated at each iteration of the method. Then the new solution is chosen from one of the neighbouring solutions. It should be noted that, in LS, there often more than one neighbourhood operator is applied. Figure 5.3 shows a visualisation of the different concepts. To select the neighbouring solution to move, there are two improvement strategies that are typically used; first and best improvement strategies. The search process in the first improvement strategy evaluates the neighbouring solutions during the neighbourhood construction step. The search direction moves to a neighbouring solution that improves the current solution quality. The best improvement strategy constructs and evaluates the complete neighbourhood. The strategy selects a new current solution that most improved the solution quality.

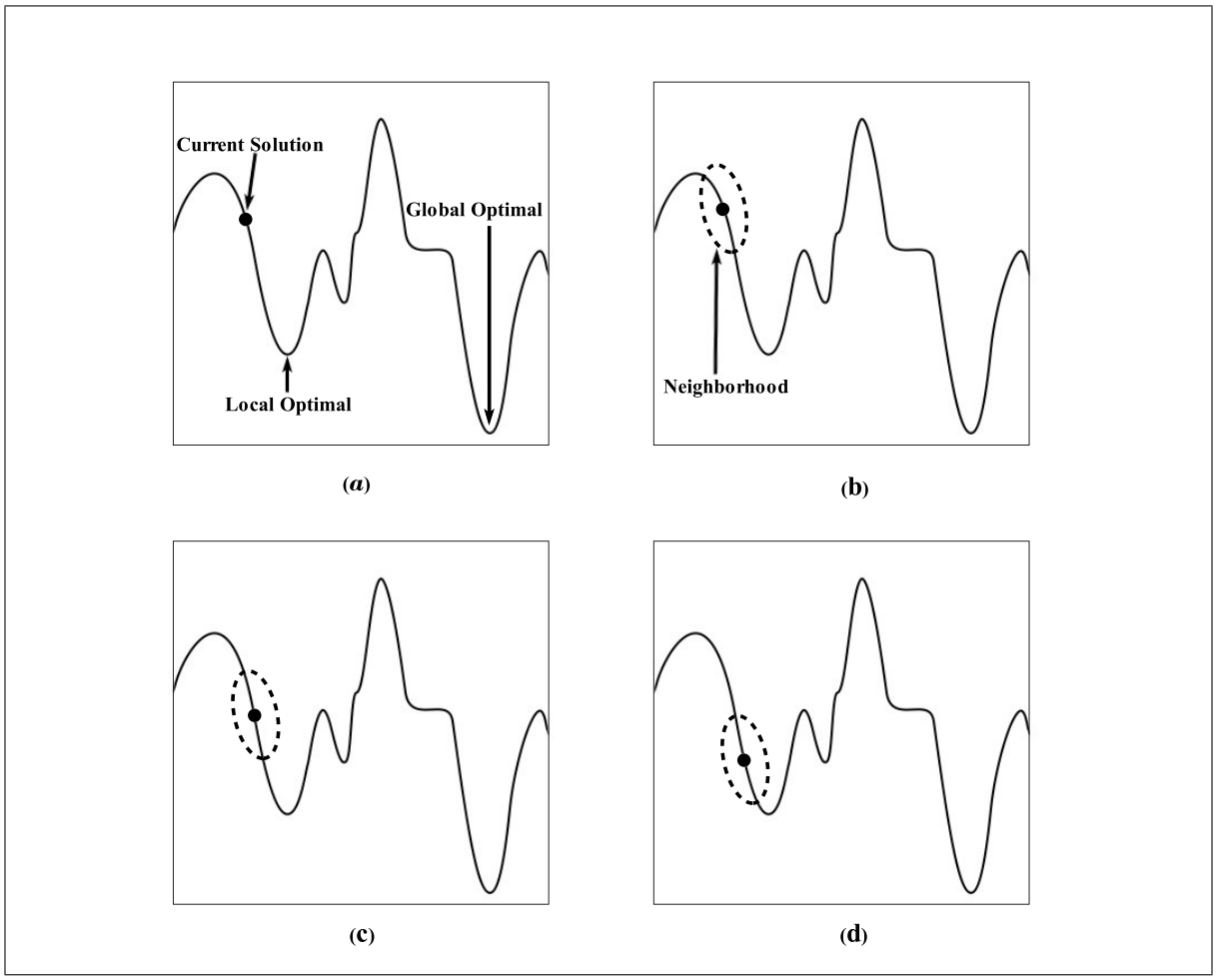


Fig. 5.3 LS moving through the solution space towards a local optimum

LS method has two important concepts, which are; intensification and diversification. These two concepts are usually implemented in the selection and evaluation of neighbouring solutions. Intensification defines as a measure that concentrates the search in the most promising area of the solution space. Diversification moves the search to explore new areas in the solution space so as to insure the search process does not remain stuck in the same local optimum solution. For a more extensive review about LS, we refer to (Gaspero et al., 2003). In the literature there are many different versions of LS algorithm, which can be taken into account. In this thesis, the LS algorithm based on the insertion neighbourhood, by Ruiz & Stützle (2007), is considered. This type of LS is very practical and a good procedure to be applied for the PFSP (Taillard, 1990). To explain this type of LS, we define permutation of jobs as π, then the insertion neighbourhood approach of π could be defined by taking into account all possible positions pairs (j,k) where $1 \leq j,k \leq n$ of π and j not equal to k. In this case, the job in the j position is removed from permutation sequence π and it

reinserted into position k. Thus the list of permutation obtained from such a movement is define as $\pi' = (\pi(1),...,\pi(j-1),\pi(j+1),...,\pi(k),\pi(j),\pi(k+1),...,\pi(n))$ if $j < k$, or $\pi' = (\pi(1),...,\pi(k-1),\pi(j),\pi(k),...,\pi(j-1),\pi(j+1),...,\pi(n))$ if $j > k$. Moreover, $I = (j,k) : j \neq k,\ 1 \leq j,\ k \leq n \bigwedge j \neq k\ ,1,\ 1 \leq j \leq n,\ 2 \leq k \leq n$, where I is the set of all insertion moves, and the insertion neighbourhood of the permutation π is defined as $V(I,\pi) = \{\pi_v : v \in I\}$. The iterative improvement algorithm is implemented by applying a first improvement type pivoting rule. The proposed LS algorithm is given in Figure 5.4.

1. Procedure: Iterative_Improvement_Insertion
2. improve := true;
3. while (improve = true) do
4. improve := false;
5. for $i := 1$ to n do
6. remove a job k at random from π (without repetition)
7. obtain best permutation by inserting job k in any position of π;
8. If $C_{max}(\pi') < C_{max}(\pi)$ then $\pi = \pi'$;
9. improve := true;
10. endif
11. endfor
12. endwhile
13. return
14. end

Fig. 5.4 Iterative improvement of neighbourhood LS (Ruiz & Stützle, 2007)

5.2.3 IG algorithm

Simplicity is the main feature of IG algorithm, the reason behind its simplicity is that it has very few parameters. Also, the IG algorithm has shown best performance for different FSPs with different objectives. To construct an initial solution for the PFSP, IG algorithm starts with an initial solution generated by the NEH heuristic of Nawaz et al. (1983). It is based on the idea that jobs with higher total processing times on all machines should be scheduled as early as possible. IG algorithm consists of two phases; destruction and construction. In the destruction phase d jobs are selected randomly and extracted from the current permutation π and inserted into a list of removed jobs π_R. Then, in their construction phase, the NEH insertion procedure is applied to reinsert all jobs in π_R individually into π again. There is one more optional step in IG algorithm used to improve each solution that is generated in the construction phase by a LS algorithm. This algorithm based on the insertion neighbourhood, is

commonly regarded as being a very good choice for the PFSP (Ruiz & Stützle, 2007). The next step is to decide whether to keep the incumbent solution or replace it with the new one, using an acceptance criterion based on the constant temperature SA-like criterion (Osman & Potts, 1989). It basically calculates a constant temperature as;

$$Temperture = T\frac{\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}}{m \times n \times 10}$$

where T is another value to calibrate. Hence, the final proposed IG method is given in figure 5.5.

1. Generate initial solution π_0;
2. Apply LS to π_0 , and put modified solution into π_s;
3. repeat
4. $\pi_d = Destruction\ (\pi_s)$;
5. $\pi_c = Construction\ (\pi_d)$;
6. $\pi_l = Local\ Search(\pi_c)$;
7. $\pi_f = AcceptanceCriterion(\pi_s, \pi_l)$;
8. Until termination condition met;
14. end

Fig. 5.5 The IG Algorithm

5.3 Experiment Results

In this section, the experimental study for the PFSP under different real-time events including; machine breakdown and new job arrival is discussed. The MSR model and the predictive-reactive based IG algorithm are used to solve this problem to verify the effectiveness of the model and the algorithm. The IG algorithm is implemented in Java using the eclipse platform. The numerical experiments were carried out on a computer with CPU of Intel Cor i5 3.2 GHz, RAM: 6 Gb. For each instance, the RPD over the best solution for each compared model is given as follows:

$$RPD = \frac{M - Best_{Sol}}{Best_{Sol}} \times 100$$

Where $Best_{Sol}$ is the average lower bound solution of ten Taillard instances that have the same size $n \times m$ and M is the solution obtained from the presented models and methods. Table 5.1 illustrates the values of average RPD for instances of Katragjini et al. (2013) corresponding to

the objectives weights. In this Table, the bi-objective model uses the following components:

$$\alpha = 0.333, 0.666, 0.498, 0.416, 0.166, 0.002, 0.166, 0.166, 0.498, 0.416$$

On the other hand, the MSR model use the weights given in Table 3.1 (see chapter 3).

Table 5.1 RPD for MSR and bi-obj models using the IG algorithm

	Ta	20×5	20×10	20×20	50×5	50×10	50×20	100×5	100×10	100×20	200×10	200×20	500×20	**Average**
W_1	**MSR**	15.00	15.36	19.07	8.86	15.69	15.09	14.79	13.16	11.80	11.08	10.71	8.38	13.249
	bi-obj	14.22	13.99	19.32	9.20	14.44	15.18	16.02	11.45	12.22	10.71	10.74	8.81	13.025
W_2	**MSR**	11.77	16.81	20.37	8.16	16.03	14.45	10.89	11.65	11.70	12.00	11.62	8.83	12.857
	bi-obj	15.95	16.71	17.71	9.10	16.59	14.30	18.20	12.00	12.35	11.62	10.88	9.62	13.753
W_3	**MSR**	12.38	15.29	15.91	10.33	15.00	13.48	14.70	12.30	11.95	13.16	10.77	9.47	12.895
	bi-obj	16.00	15.65	19.19	11.54	14.85	14.20	13.72	10.86	12.28	10.77	10.48	8.36	13.158
W_4	**MSR**	13.61	8.83	15.87	9.39	15.00	15.02	16.71	11.35	11.70	10.97	11.48	9.47	12.450
	bi-obj	12.72	10.64	16.26	9.82	14.85	14.70	15.35	11.70	11.95	11.48	10.75	8.36	12.382
W_5	**MSR**	12.95	14.74	15.60	9.47	15.32	14.12	16.78	10.92	11.99	15.06	10.46	8.91	13.027
	bi-obj	14.13	15.71	16.00	10.57	15.71	13.77	15.48	12.15	12.59	10.46	11.10	8.55	13.018
W_6	**MSR**	12.37	10.48	15.58	8.23	14.75	14.81	13.36	12.19	11.86	11.78	10.51	8.90	12.068
	bi-obj	13.01	11.98	15.97	8.85	17.47	14.22	19.87	11.34	11.93	10.51	10.66	8.99	12.900
W_7	**MSR**	12.46	10.19	16.23	9.37	15.21	14.59	12.13	12.08	11.74	13.07	10.36	8.12	12.129
	bi-obj	13.37	12.03	16.08	8.74	15.21	14.26	15.00	12.34	12.62	10.36	10.92	8.45	12.448
W_8	**MSR**	13.71	13.02	16.68	10.82	15.48	13.72	14.92	11.75	12.35	10.93	10.73	8.83	12.745
	bi-obj	15.37	13.30	16.06	8.86	16.58	14.63	14.58	11.98	11.92	10.73	10.91	8.46	12.782
W_9	**MSR**	13.05	13.23	16.92	8.76	15.36	13.82	11.76	12.15	12.48	11.07	10.82	8.91	12.361
	bi-obj	14.63	13.84	17.06	9.92	15.94	15.83	13.33	10.97	12.09	10.82	10.85	8.40	12.807
W_{10}	**MSR**	12.72	17.37	17.57	7.36	15.34	15.83	16.98	11.66	12.16	11.83	10.71	9.49	13.252
	bi-obj	12.43	13.86	16.39	10.88	15.43	14.56	17.41	11.02	11.94	10.71	10.71	8.99	12.861

According to this Table, different sets of weights produce different objective functions values, which means the solution is sensitive to different weights. It is clear from Table 5.1 that the stability and robustness performances are key to improve the solution quality. This table shows that the weights $W_4 = (0.416, 0.416, 0.166)$, $W_6 = (0.002, 0.498, 0.498)$ and $W_7 = (0.166, 0.416, 0.416)$ have minimum RPDs comparing to the other weights. Moreover, the weight $W_6 = (0.002, 0.498, 0.498)$ produces the lower values of RPDs, which proves that giving more priority to the stability and robustness objectives produces better quality solutions. For this reason, we select the weight corresponding to the lowest RPD value, which is $W_6 = (0.002, 0.498, 0.498)$.

The RPD for models MSR, bi-obj and Utility corresponding to the weight $W_6 = (0.002, 0.498, 0.498)$ are given in Figure 5.6 where Utility represents the solution of a classical model of minimising total completion time (makespan) and bi-obj the bi-objective model proposed by Katragjini et al. (2013) corresponding to $\alpha = 0.449$. From this figure, it is clear that the MSR model has minimum RPD values in general comparing to the other models.

The single factor mean of an ANOVA is applied for further investigation for the impact of the proposed predictive-reactive based IG and models on the dependent variable RPD.

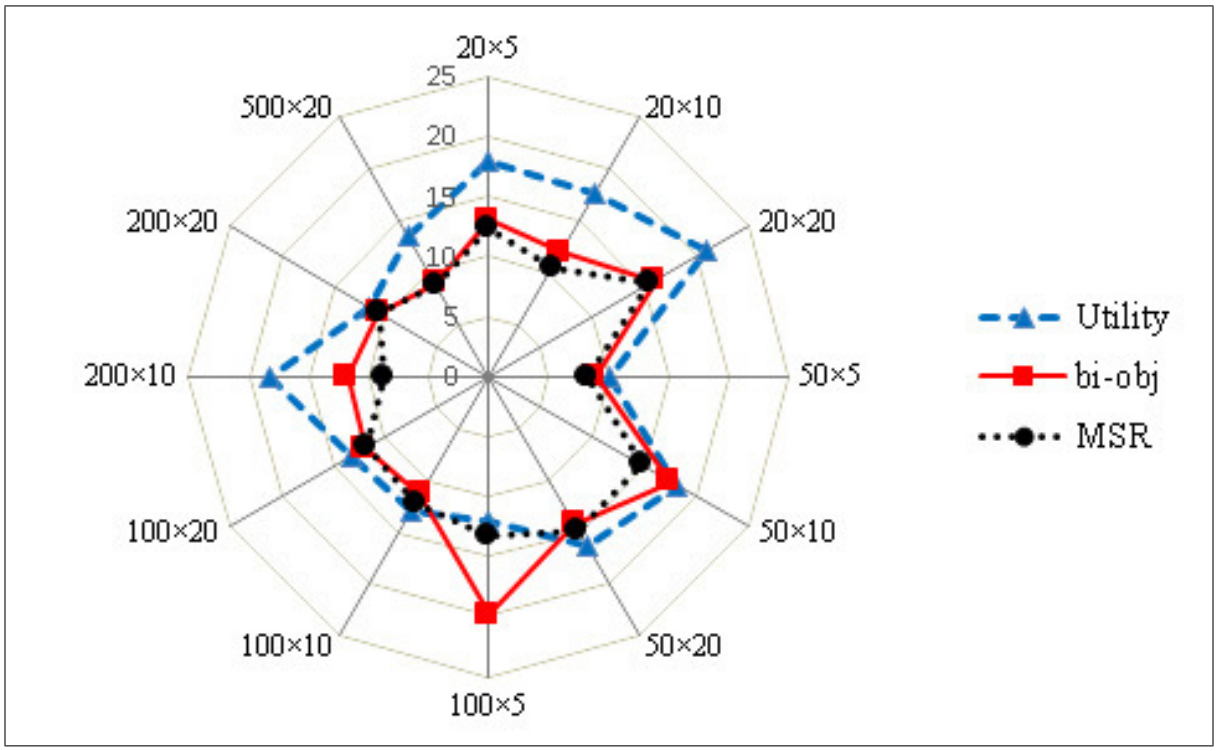


Fig. 5.6 RPD for all models with weight W_8 using the IG algorithm

The ANOVA statistically tested the results of all models where MSR corresponding to $W_6 = (0.002, 0.498, 0.498)$ and bi-obj corresponding to $\alpha = 0.449$. In ANOVA, both hypotheses (null and alternative) are given as follows:

H_0: all means are same.

H_A: at least one mean is different.

Table 5.2 shows the p-value and F-ratio; it is clear that the p-value are smaller than 0.05, and hence the null hypothesis of no difference is rejected. This means statistically, that these factors have a significant impact on RPD.

We perform a 95% confidence interval test to determine the models that are significantly different. Also, Figure 5.7 shows that both of the MSR and bi-obj models are significantly different with the Utility model. This shows how important the objectives of stability and robustness are reaching better quality solutions.

Table 5.2 ANOVA between models using the IG Algorithm

Groups	**Count**	**Sum**	**Average**	**Variance**
MSR	12	141.768	11.814	6.254
bi-obj	12	156.061	13.005	11.146
Utility	12	182.843	15.237	11.594

ANOVA						
Source of Variation	*SS*	*df*	*MS*	*F*	*P*-value	*F* crit
Between Groups	72.462	2	36.231	3.749	0.034	3.285
Within Groups	318.939	33	9.665			
Total	391.402	35				

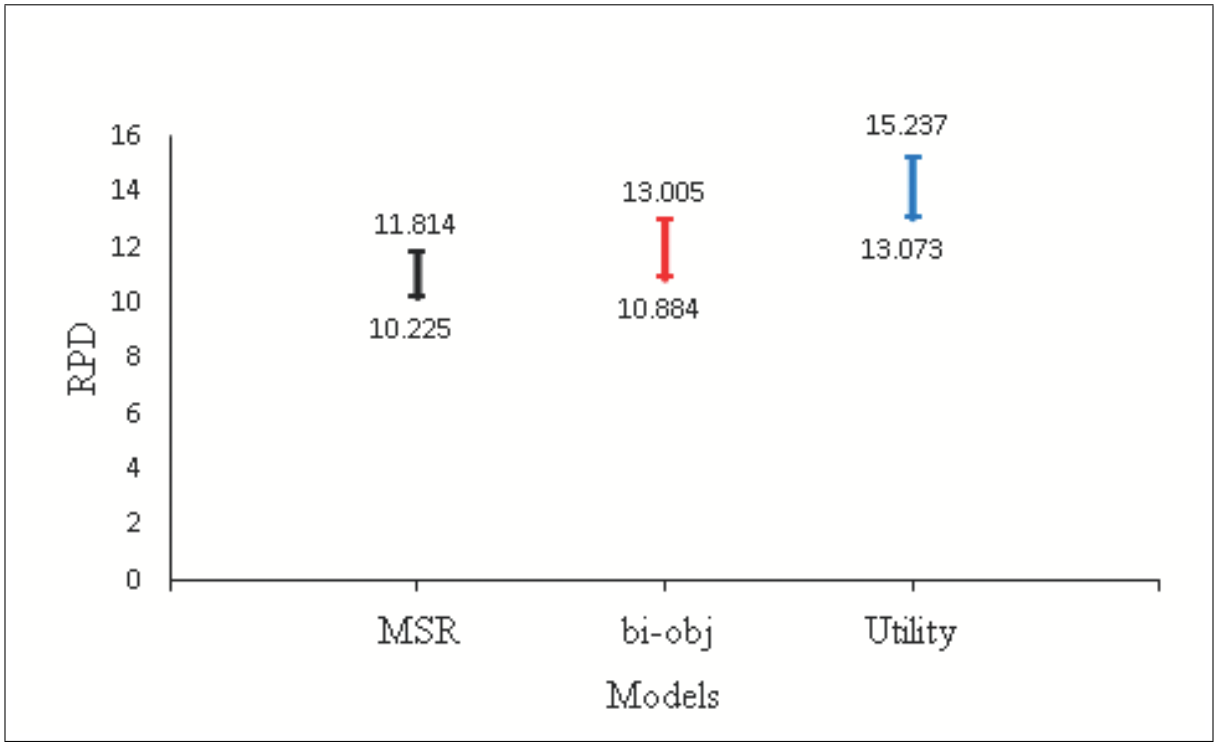


Fig. 5.7 95% Tukey confidence interval for all models using the IG algorithm

5.3.1 Comparison Study between PSO and IG algorithms

In this chapter and the previous one we considered the PFSP under different real-time events. The predictive-reactive approach based on the PSO and IG algorithms along with the MSR model were employed to solve this problem. In this section, the proposed IG algorithm is compared against the PSO algorithm for the same benchmark of the PFSP in the presence of different real-time events, where the proposed MSR model is used for this comparative

study with the weights $W_6 = (0.002, 0.498, 0.498)$ and $W_8 = (0.166, 0.166, 0.666)$ that showed relatively lower RPD values. The average of each instance solutions values for both algorithms are calculated from five independent runs, as well, a predictive-reactive approach is applied. To make fair comparison, we run both algorithms for the approximately same maximum number of iterations $Iter = 3 \times n$, where n is the number of jobs for the Taillard problem of size $n \times m$. Figure 5.8 shows the RPD values associated with both weights, it is obvious that, in general, the RPD values related to IG are much lower than the values corresponding to the PSO algorithm.

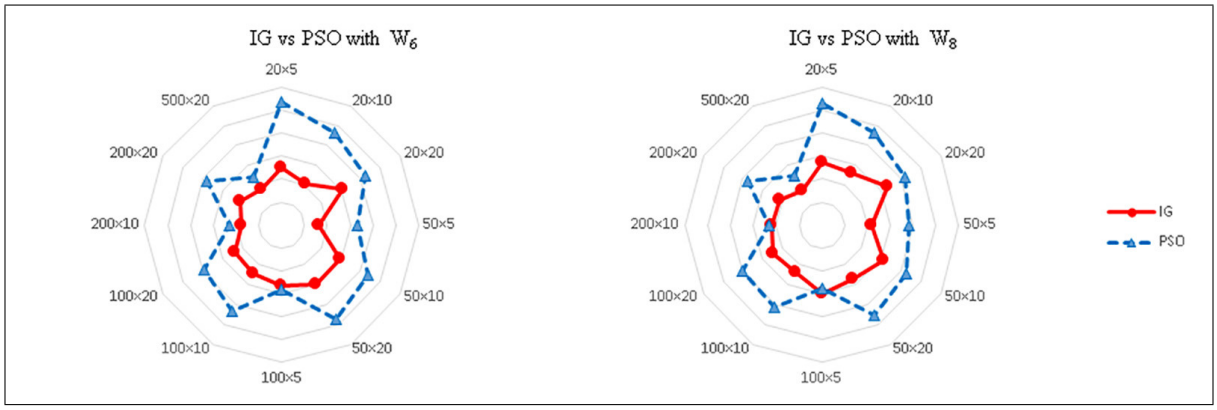


Fig. 5.8 The average RPD values obtained by using the PSO and IG algorithms with weights W_6 and W_8

Regarding the computational time, Table 5.3 includes the computational time for both of PSO and IG algorithms.

Table 5.3 Computational time of PSO and IG algorithms in seconds

Problem	PSO (S.)	IG (S.)
20×5	0.044	0.002
20×10	0.041	0.004
20×20	0.105	0.007
50×5	0.27	0.018
50×10	0.364	0.03
50×20	0.488	0.048
100×5	1.35	0.095
100×10	1.923	0.301
100×20	2.126	0.482
200×10	10.728	3.692
200×20	16.051	8.13
500×20	292.2	175.972
Average	27.141	15.732

In this Table, the computational time is given in seconds and it is represented as the average of five independent runs. From Table 5.3, the computational time of the PSO algorithm is much higher than the time required by IG algorithms. This is due to the simplicity of the IG algorithm that requires less parameters than the PSO algorithm. Also, the reason of reaching the IG algorithm better quality solution, is that the algorithm has the ability of exploring bigger parts from the solution space when comparing to the PSO algorithm. In summary, the IG algorithm outperform the PSO algorithm in getting a better quality solution in less computational time.

5.4 Conclusion

In this chapter, we proposed the IG algorithm for the predictive-reactive approach along with a MSR model that aims to minimise the makespan, stability and robustness simultaneously. This algorithm and the model have been proposed for the dynamic PFSP in the presence of machine breakdown and new job arrival. The obtained results show the high performance of the IG algorithm in handling the dynamic PFSP even for the large size instances and generating robust solutions successfully. The results also illustrate that the proposed MSR model outperforms the bi-objective models given in (Katragjini et al., 2013) and the single objective model of makespan. This emphasises the importance of stability and robustness measures where these measures provide better robust solutions.

The proposed IG algorithm uses few parameters and apply the LS technique implicitly, this gives the ability for the algorithm to explore larger portion from the solution space, and hence, it generates and rates a huge number of local optima during a short amount of computational time when compared to the PSO metaheuristic algorithm.

Recently, the biased randomisation techniques have been applied to improve the performance of many heuristics in COPs area (Juan et al., 2014b). The IG algorithm is a destructive-constructive procedure, and thus, in the next chapter, we hybridised the biased randomisation with the IG algorithm to improve the solution quality.

Chapter 6

Biased Randomised Iterated Greedy Algorithm for Robust PFSP

6.1 Introduction

One of the solution methods that applied to solve some of COP is probabilistic or randomised algorithms, often, when there is some uncertainty or local optima involved. Randomised techniques apply random variates or pseudo-random numbers in the constructive phase of the method. One of the advantage point of these techniques is that they are more likely to generate various outputs for various runs for the same input data. Hence, such approaches have the ability to explore the solution space extensively, which leads to find numerous solutions of local optima. In this chapter, we introduced a new approach based on the hybridisation of IG algorithm with a BR technique, which is the termed as BRIG algorithm. The biased randomised NEH heuristic (BRNEH) is also used to generate the initial solution for this approach. Moreover, the DDT probability distribution is used to generate random variates at the BR step. The contribution of this chapter is to implement the predictive-reactive based BRIG approach along with the MSR model for the dynamic PFSP in the presence of different real-time events. In addition, the BRIG algorithm is compared against both of the IG and PSO algorithms. The remainder of the chapter is organised as follows; in section 6.2 a BR heuristic is presented including the main advantages of this technique. In section 6.3, the predictive-reactive based BRIG framework for robust PFSP under machine breakdown and new job arrival is proposed and explained. Section 6.4 shows experimental results that illustrate the proposed methodology and comparative study. Finally, section 6.5 highlights the conclusions of this chapter.

6.2 Biased Randomised Heuristic

This section introduces how to transform deterministic heuristics such as NEH and IG algorithms into more efficient probabilistic algorithms. More precisely, it discusses the way of randomising these heuristics. The procedure of transforming a deterministic algorithm into probabilistic one is by applying a probability distribution. In fact using a non-symmetric probability distribution is more interesting for this transformation. BR means the method that use a non-uniform (skewed) probability distributions where they are defined as distribution probabilities in non-symmetric shape. As well, the BR technique can be induced into the algorithm by using the non-symmetric distributions. In PFSPs, the hybridisation of classical NEH or IG heuristics with BR process is applied in the step of inserting a job in a new position. That is, instead of choosing the inserted job randomly, the BR procedure selects the job with highest probability to be inserted first. The NEH heuristic for the PFSP employs an iterative process in order to construct a feasible and hopefully a better solution. Usually, during the iterative process, the next constructive movement is selected from the priority list of potential movements where the list is ordered in accordance with some criteria. The criteria used to order the sequence is based on the particular heuristic approach being used. For this, the constructive heuristic approach is actually considered as an IG approach that builds a good feasible solution to the problem by choosing the best option from a sequence at each iteration, then arranged according to some logical criterion. This is consider as a deterministic process, because as soon as the criterion is defined, the process provides a unique sort for the jobs potential movement sequence. It is obvious that if the order is randomised such that the jobs of the sequence are chosen, then a different list is probably obtained every time the entire approach is carried out. However, a non-symmetric (uniform) randomisation of that sequence will ruin the basic methodology of the heuristic greedy manner, hence, it is improbable that the randomised algorithm will provide a good output solution. This means, the randomised algorithm could be running thousands of times and the generated solutions are probably much worse than the solution that is generated by the original heuristic procedure. To keep the logic beyond the heuristic approach, the Greedy Randomised Adaptive Search Procedure (GRASP) (Prabhaharan et al., 2006) is suggested to take into account a list of restricted candidates. This list is defined as a partial list that have only a part of the most promising movements, i.e., the jobs at the top of the list, and then it applies a symmetric randomisation to arrange the list so as to select the restricted list of jobs as shown in Figure 6.1. It should be noticed that, the complexity of a randomised heuristic algorithm is the same as the complexity of the original deterministic heuristic (which is used as a basis) (Juan et al., 2014b). This is because the only extra operation added is the generation of a random decision instead of a greedy one.

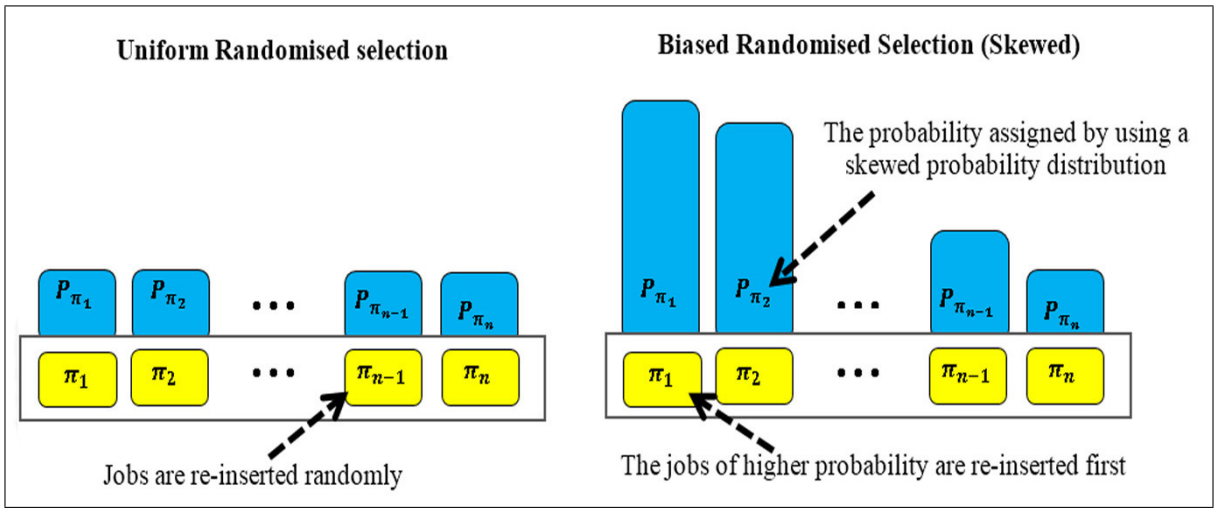


Fig. 6.1 BR selection versus uniform selection

The application of the BR technique with heuristic algorithms provides better results for many COPs problems (Juan et al., 2014b). Some of the main advantages of using BR heuristic approaches are as follows:

- The most important purpose of designing BR heuristic approaches is to support different strengths of metaheuristic approaches such as; simplicity, speed, flexibility and accuracy (Cordeau et al., 2002). Where Simplicity is referred to the facility of implementation and the number of parameters that need to be used in the algorithm. This is very important feature because it can be used for various instances other than the ones tested without the necessity of a long run test and without losing performance or quality. Also, speed feature represents the computational time of the algorithm. Moreover, flexibility is defined as the possibility of accommodating new side constraints and also with the adaptation to other similar problems. Finally, accuracy represents the degree of deviation of the current obtained solution from the initial planned solution.

- Because usually the used probability distributions do not need any parameters such as the DDT distribution or they require only one parameter such as the Geometric distribution, the BR techniques permit a simplification of the fine-tuning procedure. In general, this is not popular in recent metaheuristic techniques that typically use many parameters and, hence require more complex and more time consuming fine-tuning procedures to adapt their related values.

- The classical well-tested heuristics are relatively simple and easy to implement approaches that can be adjusted to account for new flexibility. Thus, for COPs there are high ranked efficient heuristics that can be chosen and transformed into BR. This happens,

among many other cases, with the Clarke and Wright savings heuristic for the VRP, and with the NEH heuristic for the PFSP.

- A BR with uniform distribution does not keep the logic behind the heuristic technique, because it specifies the same probabilities for all jobs to be selected during all movements. On the other hand, the application of skewing (non-uniform) distributions instead of uniform ones, produces a more efficient and natural way to choose the next movement from the priority list. BR with skewed distributions allows consideration of the common concept of the heuristic by assigning more probabilities of being selected to those movements which better fulfill the heuristic criteria. A good example is that in PFSP jobs requiring larger processing times in the NEH heuristic.

- The last advantage of BR heuristic is it can also be hybridised with many different approaches, for example BR is combined with MCS for stochastic variants of COPs (Juan et al., 2011); (Cáceres-Cruz et al., 2012).

6.3 Predictive-reactive based BRIG framework for robust PFSP

In the dynamic PFSP, two types of real-time events are employed namely; machine breakdowns and new job arrivals. These real-time events interrupt the initial planned schedules simultaneously. The reason for having initially considered only these two types of real-time events relies fundamentally on the fact that the concern of the research is not to address all types of disruption that may affect manufacturing settings. In this chapter, at the beginning of every machine failure, rescheduling BRIG algorithm are triggered to cope with this disruption where the MSR model is used to keep the system performance on a stable and robust level. Also when a new job arrives into the system, reactive procedure using a BRIG algorithm is applied to accommodate the new job in the current schedule with using the MSR model.

Figure 6.2 shows the framework of the predictive-reactive approach and how it applies the BRIG algorithm at the point of disruptions. The proposing predictive-reactive rescheduling approach and the MSR model seeking a good trade-off between schedule quality, stability and robustness. Consider a predictive initial schedule S_0, which is generated at the beginning of the planning schedule horizon. Let C_i (S_0) be the completion time of job i in this schedule. S_0 that is executed on the shop floor and revised using the proposed rescheduling method. At each periodic rescheduling point, all those unprocessed jobs on the first machine are performed. A new schedule S_n is generated by the proposed rescheduling BRIG algorithm. A

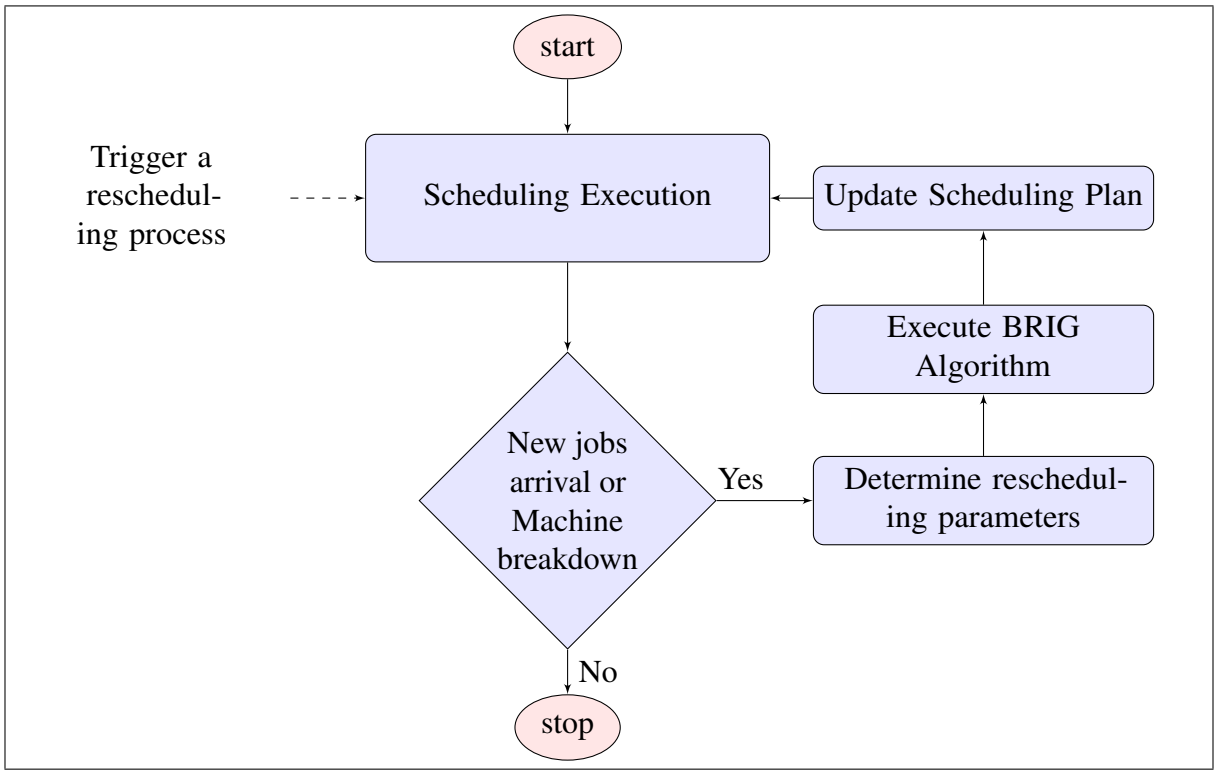


Fig. 6.2 Predictive-Reactive based BRIG approach

new rescheduling is triggered while a machine breaks down or new job arrival interrupt the schedule plan. Rescheduling employed a BRIG algorithm at every time t_D at which an event occurs to accommodate the partial sequence of jobs that have not already started processing by the first machine. This reason is due to the fact that in PFSP, the order of the jobs proceeds on the first machine and must be continued throughout all of the remaining machines.

6.3.1 BRIG algorithm

This work proposed the BRIG algorithm that hybridises the IG algorithm with BR. First of all, to construct an initial solution in the BRIG, we use the BR version of the NEH heuristic, which is explained in the section 6.2. In the BRNEH algorithm, the jobs are ordered in construction phase depending on the probability of each job. Thus, the job with higher probability is more likely to be constructed first. The probability distribution used for this task is the DDT distribution. This distribution is preferred for its practicality where it provides good results and also it does not have relatively straight forward parameters to set (Juan et al., 2014a). The next step is applying the hybrid BRIG algorithm, which combines BR with the IG algorithm. As we discussed before, the IG algorithm has two main phases; in the first phase, d jobs are selected randomly from the set of all jobs, in this case we suppose π_d is the set of d jobs that are selected

randomly from the list of all jobs, and π_R the set of the remaining jobs. In the second phase, each job from π_d is reinserted into π_R and the sequence corresponding to the minimum partial makespan is chosen, this process continues until $\pi_d = \phi$. To transform the IG algorithm from a deterministic version into a probabilistic method, the BR is induced to perturbation behaviour (Juan et al., 2014a). The idea of BR as in the BRNEH heuristic is to pick out jobs from the list π_d depending on their probability. Thus, to select a candidate job to be inserted from the list π_d, the BR assigns a different probability for each job in the list, then the job corresponding to higher probability is more likely to be inserted into π_R first. This probability is obtained by using a skewed distribution, specifically the DDT distribution as it applied for the BRNEH algorithm, where jobs with higher probability are more likely to be inserted to the permutation list π_R than the jobs with lower probability. At the construction phase, there is an optional step of applying a LS technique to improve the generated solutions. This LS step is based on the insertion neighbourhood technique, which is an efficient and common LS procedure for the PFSP (Ruiz & Stützle, 2007). This technique has been explained in details in chapter 5. Figure 6.3 shows the BRIG algorithm.

1. Generate initial solution π_0;
2. Apply *LS* to π_0 , and put modified solution into π_s;
3. repeat
4. $\pi_d = Destruction(\pi_s)$;
5. $\pi_c = Construction(\pi_d)$; % apply BR
6. $\pi_l = LocalSearch(\pi_c)$;
7. $\pi_f = AcceptanceCriterion(\pi_s, \pi_l)$;
8. Until termination condition met;
14. end

Fig. 6.3 The BRIG algorithm

6.4 Experiment Results

In this section we present the results of numerical experiments designed for the PFSP in the presence of machine breakdown and new job arrival. Once the best solution is found, the RPD is calculated over 10 of Taillard problems of the same size $(n \times m)$ and is given as follows:

$$RPD = \frac{M - Best_{Sol}}{Best_{Sol}} \times 100$$

Where the value M represents the acquired solution using the proposed model and solution methods. $Best_{Sol}$ is the average of lower bound solution of 10s Taillard's instances that have the

same number of jobs and machines. Table 6.5 illustrates the RPD for each instance of model MSR corresponding to the weight sets (α, β, γ) given in Table 3.1 (see chapter 3). Also, the bi-objective model uses the following components:

$$\alpha = 0.333, 0.666, 0.498, 0.416, 0.166, 0.002, 0.166, 0.166, 0.498, 0.416$$

Where the solution obtained from the bi-objective model of Katragjini et al. (2013) is termed as bi-obj. From Table 6.1, it can be seen that the objective functions components are sensitive to different weight sets. For example, for the solution corresponding to the weight $W_8 = (0.166, 0.166, 0.666)$, the RPD increases in the bi-objective while it decreases for the MSR model.

Table 6.1 RPD for MSR and bi-obj models using the BRIG algorithm

	Ta	20×5	20×10	20×20	50×5	50×10	50×20	100×5	100×10	100×20	200×10	200×20	500×20	**Average**
W_1	**MSR**	14.179	12.959	16.881	9.746	16.093	14.556	12.118	12.160	11.898	9.797	10.305	9.902	12.550
	bi-obj	14.392	15.778	16.545	8.559	15.916	14.355	16.343	12.119	12.996	11.318	10.346	11.170	13.320
W_2	**MSR**	12.479	15.205	15.530	10.620	14.712	13.589	12.062	12.391	11.706	12.658	10.151	8.362	12.455
	bi-obj	12.356	11.516	16.536	9.516	15.796	14.567	15.876	11.151	11.802	9.844	11.274	10.667	12.575
W_3	**MSR**	11.424	11.878	15.261	9.414	15.666	13.688	13.288	12.524	11.536	12.633	10.446	9.517	12.273
	bi-obj	14.981	14.270	16.254	8.109	15.542	13.881	15.786	10.950	12.924	9.913	10.637	9.236	12.707
W_4	**MSR**	14.457	13.874	14.823	8.347	15.262	14.227	12.495	11.739	12.528	12.257	10.462	9.673	12.512
	bi-obj	14.703	15.666	15.377	10.024	15.362	14.436	14.496	11.703	11.924	10.023	10.507	9.388	12.801
W_5	**MSR**	14.049	14.487	15.319	9.845	15.042	13.699	12.203	11.730	12.107	10.727	10.691	8.091	12.333
	bi-obj	13.452	13.828	15.530	7.521	14.892	13.779	15.607	11.886	11.877	13.181	10.254	8.526	12.528
W_6	**MSR**	12.413	15.153	16.107	8.975	15.659	14.977	11.218	10.327	11.395	10.483	10.298	8.357	12.114
	bi-obj	13.705	13.209	15.785	7.981	15.816	14.806	15.664	11.083	11.596	10.633	10.363	8.924	12.464
W_7	**MSR**	13.991	14.098	15.910	7.700	15.529	14.329	11.877	10.936	12.009	10.785	10.122	8.648	12.161
	bi-obj	13.174	9.829	16.120	9.703	16.143	14.618	14.243	11.922	13.056	9.972	10.506	8.977	12.355
W_8	**MSR**	12.201	13.426	16.411	8.182	14.278	14.685	12.758	11.593	11.382	10.513	10.065	8.101	11.966
	bi-obj	12.372	13.769	16.509	8.343	17.060	14.487	14.071	11.220	11.857	10.665	10.416	8.635	12.450
W_9	**MSR**	14.605	12.405	16.890	9.312	15.456	14.822	10.406	11.187	12.130	9.066	10.580	8.359	12.102
	bi-obj	11.260	14.270	17.149	8.939	16.306	14.254	11.794	10.885	12.551	16.297	10.672	8.825	12.767
W_{10}	**MSR**	13.018	14.388	16.075	9.301	15.502	14.581	12.480	11.547	12.361	11.164	10.130	8.417	12.414
	bi-obj	14.384	13.888	14.603	7.572	15.122	14.162	13.551	11.572	11.775	12.555	9.803	8.649	12.303

The results of the numerical experiments in this table revealed that the RPDs corresponding to the MSR model and the weight set $W_8 = (0.166, 0.166, 0.666)$ have lower values than the other RPDs corresponding to other weight sets. This emphasises the fact that giving higher priority for the robust term of the MSR model leads to less RPD, and hence produces better quality solution. Figure 6.4, shows the RPD values for all models (MSR, bi-obj and Utility models) with the weight $W_8 = (0.166, 0.166, 0.666)$. From this figure, it is obvious that the solutions corresponding to the MSR model show in general lower values of RPD when compared to the other models.

The results are statistically tested by the single factor mean of ANOVA. This statistical procedure is used to describe the impact of the proposed predictive-reactive based BRIG

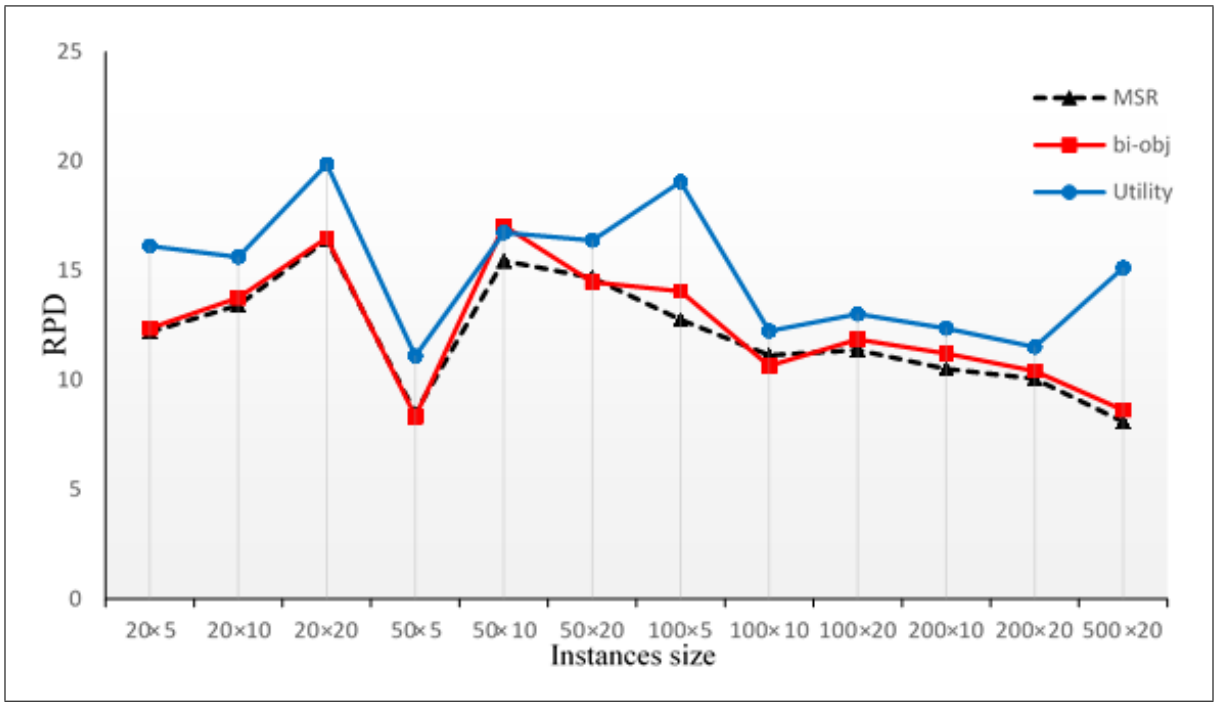


Fig. 6.4 RPD for all models with weight W_8 using the BRIG algorithm

and models on the dependent variable RPD. For our analysis with the three models the null hypotheses and its alternative are then as follows:

H_0: all means are same.

H_A: at least one mean is different.

The p-value and the F-ratio are shown in Table 6.2. It is clear that $p \leq 0.05$, this means there is a statistical significant difference in RPD between the factors of models at confidence level of 95%.

Table 6.2 ANOVA between models using the BRIG Algorithm

Groups	**Count**	**Sum**	**Average**	**Variance**
MSR	12	144.622	12.052	6.887
bi-obj	12	149.400	12.450	7.898
Utility	12	179.195	14.933	8.409

ANOVA						
Source of Variation	*SS*	*df*	*MS*	*F*	*P*-value	*F* crit
Between Groups	58.494	2	29.247	3.783	0.033	2.471
Within Groups	255.139	33 7.731				
Total	313.633	35				

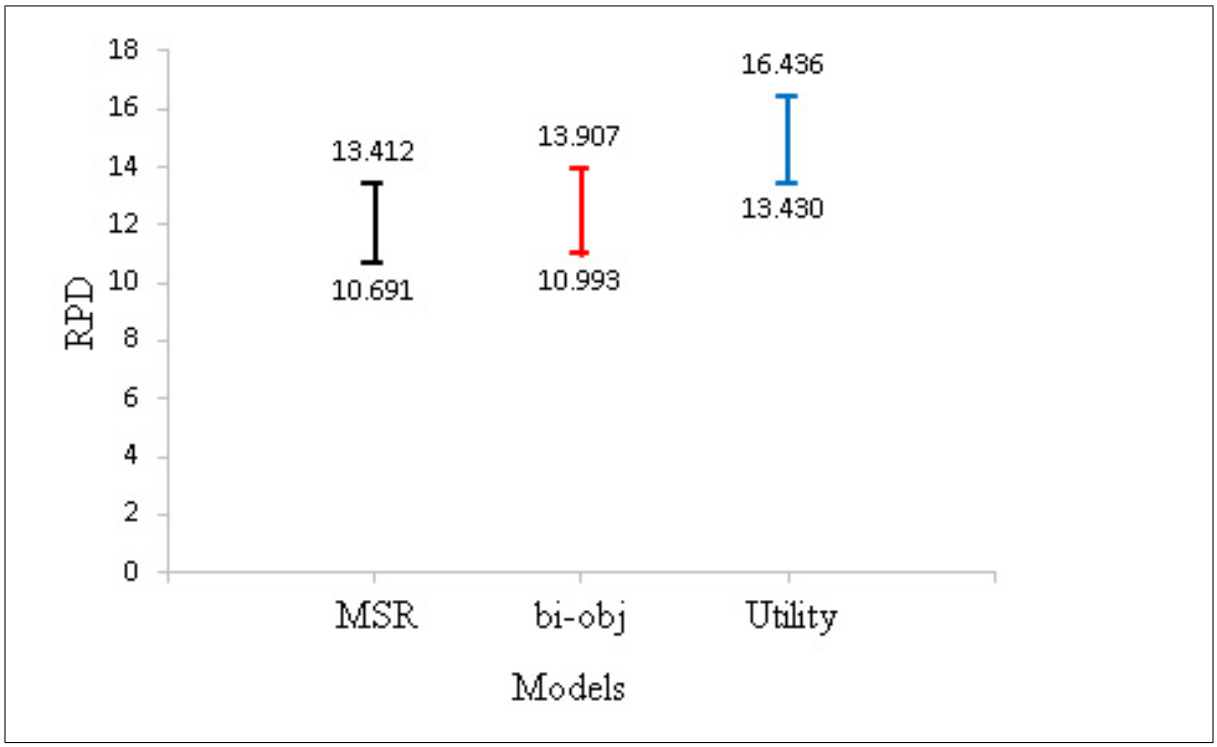


Fig. 6.5 95% Tukey confidence intervals for all models using the BRIG algorithm

The Tukey confidence 95% intervals are applied to assign the group that is statistically significant difference in the mean of RPD values. Figure 6.5 shows that the MSR model is significantly different when compared with the Utility model, while there is no significance difference between the bi-obj and the Utility Models.

6.4.1 Comparative study between PSO, IG and BRIG algorithms

The PSO, IG and BRIG algorithms are applied for the dynamic PFSP in the presence of machine breakdown and new job arrival. The MSR model is used to maintain the problem stability and robustness. Also, the predictive-reactive approach is used with the PSO, IG and BRIG algorithms at each disruption point. In this section, the proposed algorithms are compared against each other for the dynamic PFSP in the presence of different real-time events (machine breakdown and new job arrival), where the MSR model is used for this comparative study with the weight $W_8 = (0.166, 0.166, 0.666)$. In this experiment, each algorithm performed five runs independently. For all methods, the average of each instance solutions values are calculated. Figure 6.6 records the average RPD values obtained by using the PSO, IG and BRIG algorithms. From this figure, it is clear that the RPD corresponding to BRIG is always much lower than the RPD for PSO algorithm. The RPD values of BRIG are slightly (in general) lower than the solution of IG algorithm, this is true because the BRIG is the improved version of IG algorithm.

On the other hand, figure 6.7 clarifies the significant difference between the PSO, IG and BRIG algorithms.

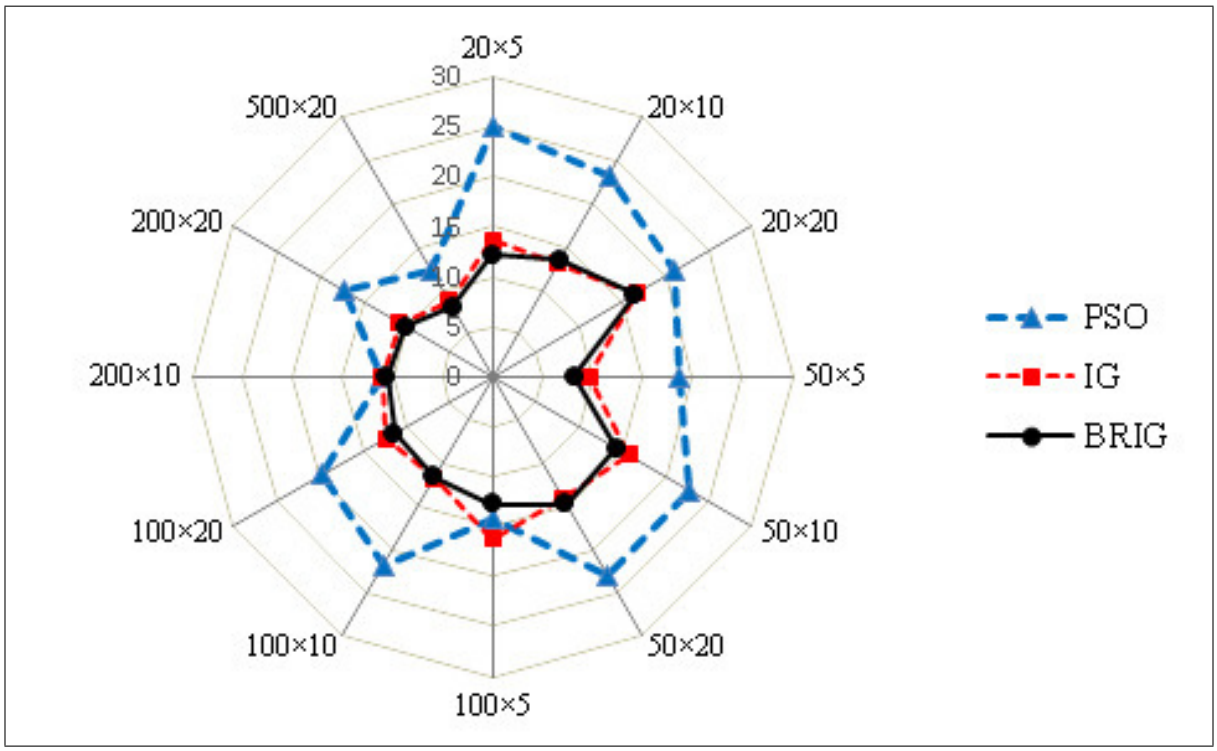


Fig. 6.6 The average RPD values obtained by using the PSO, IG and BRIG algorithms

This figure shows there is a significance difference between the BRIG and PSO algorithms, also between the IG and PSO algorithms. However, there is no significant difference between the BRIG and the IG algorithms. The reason of peak performance of the BRIG when compared to the PSO algorithm is that the BRIG algorithm used the BRNEH heuristic to generate an initial solution and also it applies the LS improvement step implicitly. Hence, this algorithm has the ability to start from good quality solution and try to improve it by exploring larger parts from the solution space. In addition, the BR technique produces a more efficient and natural way to select the next movement depending on the jobs priority list using the DDT probability distribution. Thus, this advantage shows the improvement of the BRIG performance when compared to the IG algorithm. This could explain the reasons for the better performance of the BRIG algorithm.

Finally, Table 6.3 shows the required computational time for all the aforementioned algorithms. It indicates that the computational time required by BRIG and IG are less than the time required by the PSO algorithm, where the computational time is calculated as the average of five independent runs in seconds.

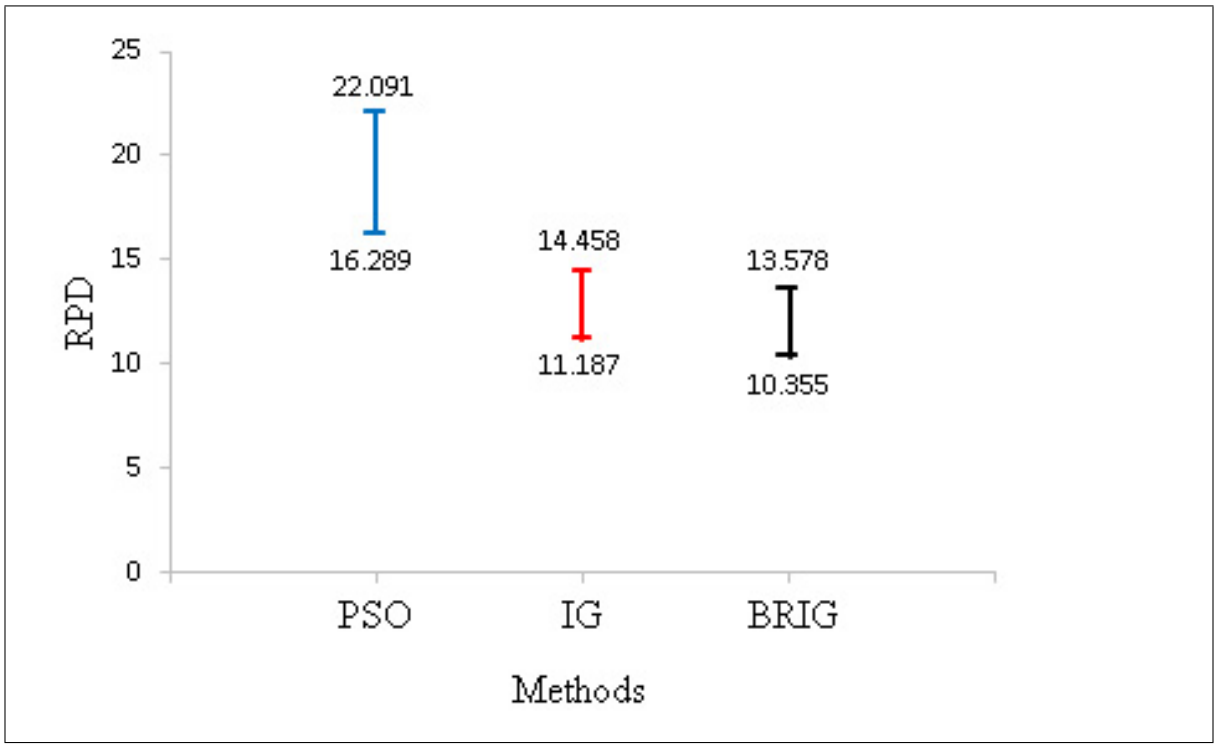


Fig. 6.7 95% Tukey confidence intervals for PSO, IG and BRIG algorithms

Table 6.3 Computational time of PSO, IG and BRIG algorithms in seconds

Problem	PSO (S.)	IG (S.)	BRIG (S.)
20×5	0.044	0.002	0.003
20×10	0.041	0.004	0.005
20×20	0.105	0.007	0.007
50×5	0.270	0.018	0.019
50×10	0.364	0.030	0.030
50×20	0.488	0.048	0.057
100×5	1.350	0.095	0.105
100×10	1.923	0.301	0.298
100×20	2.126	0.482	0.493
200×10	10.728	3.692	3.716
200×20	16.051	8.130	8.124
500×20	292.200	175.972	177.687
Average	27.141	15.732	15.879

As seen from the results in Table 6.3, the three algorithms running time are different for the same PFSP instances under the same environment; the running time of the PSO is the longest, followed by the BRIG then IG. The IG algorithm is the fastest in 12 groups for the test in general. In particular, we focus on comparing the PSO algorithm and IG with its randomised version in the all instances, for example, in the first instance of size 20×5 , we find that the running time of both of the IG and the BRIG algorithms are about 14% of the running time of the PSO algorithm. On the other hand, both the IG and the BRIG algorithm are essentially consume almost the same time. Therefore, from the above results we can get that consumption running time of both the IG and the BRIG algorithms on the dynamic PFSP under different real-time events are much smaller when, compared with the PSO algorithm. IG and BRIG need less computational time because their conceptual simplicity that both of IG and BRIG algorithms required fewer parameters, when compared to the PSO algorithm which is a population based method and it is a relatively more complex algorithm than the IG and BRIG algorithms.

6.5 Conclusion

In this chapter, we have analysed some key aspects, advantages, and experiment results related to the hybridisation of BR technique with NEH and IG heuristics. This hybridisation considers as a natural approach to develop probabilistic algorithms to solve dynamic PFSP in the presence of different types of real-time events. The predictive-reactive based on the probabilistic BRIG approach is used to solve this problem where the MSR model is also used to maintain the solution stability and robustness. The obtained results revealed that the BRIG algorithm is capable to cope successfully with the dynamic PFSP even for large size instances. The proposed BRIG algorithm hybridises the IG algorithm with the BR technique and use the LS technique implicitly. This gives the ability for strong seeking in the solution space in a relatively short time. A number of numerical experiments have been carried out to test the performance of the introduced MSR model and the BRIG algorithm. The results show that the proposed solution method with the MSR model outperforms the other models. This emphasises the importance of stability and robustness measures where these measures provide better robust and stable solutions. Moreover, the BRIG algorithm has been compared versus the IG and PSO algorithms, which have been already applied for the dynamic PFSP in the presence of machine breakdown and new job arrival. This comparative study shows that the BRIG algorithm outperforms both of the PSO and IG algorithms in reaching better quality solutions. On the other hand, the computational time spent on both of IG and BRIG algorithms to reach a good solution is much less than the time consumed by the PSO algorithm. Thus, the BRIG algorithm is flexible,

quite efficient, simple, and fast convergence, this is due to the advantages of the nature of IG algorithm and BR technique.

Part II

Stochastic PFSP under different real-time events

Chapter 7

Simulation Particle Swarm optimisation for Robust SPFSP

7.1 Introduction

The scheduling in a manufacturing environment has received a special attention for its wide real applications. In real-world scheduling systems, there are two main sources of uncertainties that lead to different scheduling environments, which are; dynamic and stochastic. When there are some variables that are considered as unknown and follow a probability distribution, the scheduling in this case is named as stochastic scheduling. The PFSP problems in a stochastic environment have received increasing interesting in the literature of scheduling, due to the nature of most real problems where the data and information cannot be known in advanced. Even though the literature today is filled with articles on a wide array of manufacturing scheduling problems, there are less studies discussing the case of stochastic and dynamic scheduling problems with efficient approaches. In this Part, we will consider the PFSP under stochastic processing time and different dynamic disruptions including; machine breakdowns and new job arrivals, where a multi-objective model (MSR) is used to preserve the problem stability and robustness. To our knowledge, no other study has been lead on a multi-objective scheduling under dynamic and stochastic environment (under different types of disruptions). Due to the dynamicity and stochasticity of such problem, the solutions are sensitive to different disruptions coming from different sources. The significance of models and methods that discuss unexpected disruptions in simulation and optimisation for complex COPs is evident by plenty of papers and books lately devoted to this topic. When optimising systems performances, considering or ignoring uncertainties may change the results completely, and hence, uncertainty management is a main problem in Sim-Opt. there are many applications for such methods in

almost any subject of human applications, in transportation, science, engineering, business and so on. Although there are wide applications of such methods, it is still unclear in how these methods are developed and applied. The usual reason for implementing simulation models is that we want insight or guidance on a decision. The word "decision" is somewhat synonymous with the word "optimisation". If the decisions can be represented as decision variables within a simulation model, then we have the option of performing some kind of simulation-optimisation, i.e., choosing the decision variables to try to opimise some performance measure that is estimated using simulation (Jian & Henderson, 2015). Despite the method of solving a related deterministic problem to solve the stochastic one not being a completely new idea, it has not previously been expand to solve the SPFSP under different uncertainties. Actually, the majority of the works so far have concentrated on the theoretical sides of the stochastic scheduling. On the other hand, the proposed approach provides a practical process to the solution, the following are some of important advantages:

1. The approach is flexible to use any probability distribution with a known mean, such as Normal, Log-Normal, Gamma, etc. because it employs simulation to cope with the stochastic attitude of jobs processing times. In any case it is possible to apply bootstrapping method to generate the random values. In the literature of SPFSP, it is not likely to suppose that the processing times are random variables and follow a Normal probability distribution or even an Exponential one, because real-life jobs processing times are related with continuous and non-negative values. On the other hand, random processing times must be modelled by applying any experimental or theoretical distribution, which provide values that are non-negative and asymmetries are produced by long right-hand tails such as Log-Normal, Gamma or Weibull distributions, as is frequently done in Reliability Analysis.

2. For complex SPFSP there is no efficient metaheuristic algorithm that have been developed yet. Thus, Sim-Opt approaches have been applied successfully in many COPs.

In this chapter, we adopt a predictive-reactive based Sim-PSO approach for the SPFSP under machine breakdown and new job arrival with applying of the MSR model, where the Sim-PSO approach based on the hybridisation of MCS approach and PSO algorithm. The remainder of the chapter is structured as follows; in section 7.2, we give a brief about the Sim-Opt. In section 7.3, the hybrid Sim-PSO framework for SPFSP under different disruptions is presented. Finally, the conclusions are given in section 7.4.

7.2 Simulation based Optimisation

Generally, solving stochastic COPs are harder than their deterministic counterparts. In the last few decades, computer simulations are implied for modeling of stochastic scheduling in manufacturing systems to evaluate the given objectives. It is important to select the best simulation parameters that could lead to improve operation. However, it is still not easy to configure these parameters well. Also, because of the complexity of the scheduling in manufacturing systems, the simulation techniques consume a long time before getting a reasonable stochastic optimal solution. On the other hand, optimisation algorithms (even stochastic version algorithms) do not guarantee a good optimal solution for stochastic manufacturing scheduling problems, even when some algorithms are able to find quasi-optimal solutions within a reasonable amount of time. Thus, there is a requirement for an efficient algorithm that integrate between the simulation and optimisation techniques is raises. The term "simulation" is short-hand for stochastic discrete-event simulation, meaning that the random nature of the system will be implicitly understood and the underlying models are discrete-event systems such as queuing networks. Actually, the central key of all of this study is the stochastic nature, and one main hypothesis of this chapter is that the currently implemented optimisation methods do not address this feature sufficiently. In the Sim-Opt approach, an optimisation algorithm is run for a stochastic COP so as to direct the exploration in the search space. The goal of this procedure is to find feasible and good local optimal solutions. When the iterative search procedure is running, the algorithm have to transact with the stochastic environment of the problem. One of most natural ways to deal with the stochastic nature of such problems is by employing the utility of the strengths simulation techniques that offer to manage randomness. In addition, there are other methods that have the ability to be used instead of simulation such as; fuzzy logic, dynamic programming, and so on. However, with the stochastic behavior under the presence of historical data, simulation provides the improvement of both flexible and accurate models. Particularly, it is possible to model the random behavior over a best-fit probability distribution without any extra constraints or assumptions. For this, simulation technique is frequently incorporated with the optimisation (heuristic or metaheuristic) method and it usually improves the final result by providing dynamic feedback to the searching procedure. In some sense, simulations have the ability to develop the existing highly quality heuristics/metaheuristics that were originally designed to solve the deterministic optimisation problems so these efficient algorithms also can be used to solve stochastic COPs. Clearly, one of main obstacle of combining simulation with heuristics/metaheuristics are that the obtained results may not be optimal anymore, since Sim-heuristics/metaheuristics are integrating two approximate approaches. Figure 7.1 illustrates the scheme of the Sim-Opt approach. Despite the fact that most of real life problems are too complicated and they are in general mathematically intractable (NP-hard). Thus,

Sim-heuristics/metaheuristics constitute very intriguing alternative for the majority of practical fields, because they are relatively flexible and simple approaches that have the ability to obtain good local optimal solutions for complicated real life COPs, also Sim-heuristics/metaheuristics consume reasonable amount of computational

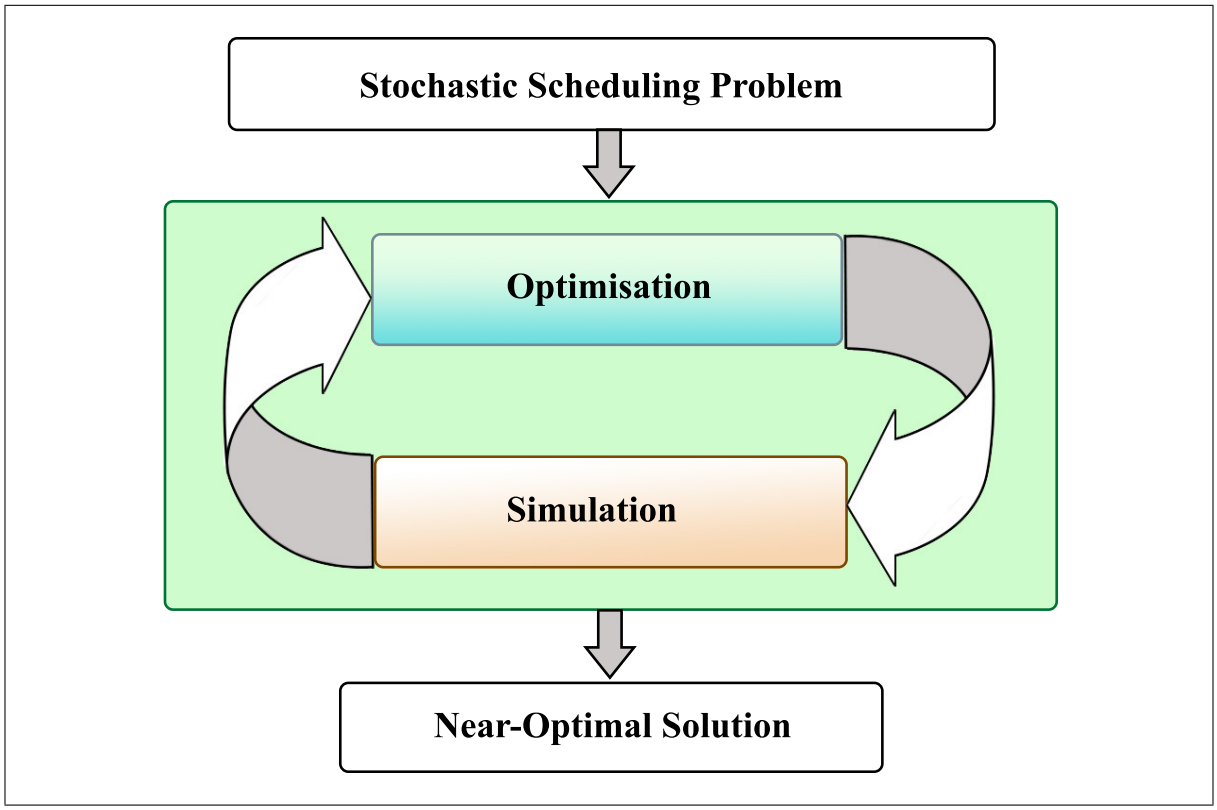


Fig. 7.1 Overview scheme of the Sim-Opt approach

7.3 The hybrid Sim-PSO framework for SPFSP under different disruptions

In this section, the framework of the hybrid MCS and PSO algorithm which is applied for the SPFSP under different real-time disruptions is explained in details. The basic idea of this algorithm starting with transforming the initial SPFSP into a dynamic problem, this transformation uses constant processing times as the expected values of the stochastic processing times. In spite of entering the inputs as deterministic values, these instances are designed for the PFSP in a dynamic environment where the problem is effected by different real-time events during the time horizon. Then, the predictive-reactive based PSO algorithm is running over this problem with the MSR model to generate a robust and stable local optimal schedule. The

obtained solution will then improve by applying a LS approach. It should be notice that, the Katragjini et al. (2013) benchmarks are based on known processing time values (not stochastic random variables). Thus, we assume the processing times as expected values for random variables follow the Log Normal distribution. At this stage, the MCS approach employs the Log Normal probability distribution to generate random variates of unknown processing times, then it estimates the final value of the stochastic makespan, which is consider as a local optimal stochastic solution for the SPFPS under different real-time events. The MCS technique has the following advantages:

- The application of MCS provides the ability to be naturally extended to take in account a various probability distributions to model the processing time of each job and even possible dependencies through these processing times.
- The MCS technique does not require huge running time as we run short simulations during the LS procedure for a reduced number of promising solutions. Thus, methodology of Sim-PSO can be used for large size SPFSP.

7.3.1 Sim-PSO Approach

Following an analysis, we propose that the processing times follow the Log-Normal distribution. In this case, a random variable p_{ij} (of processing time) follows a Log Normal probability distribution where μ and σ are parameters if $logp_{ij}$ follows a Normal distribution $N(\mu,\sigma)$. Usually, the Log Normal probability distribution is employed to model the influence of uncontrolled environmental variables (Dauzére-Pérés et al., 2010). It is essential that the Sim-PSO framework starts to transform the SPFSP into a deterministic one. Then, the predictive-reactive based PSO algorithm is applied for the dynamic PFSP under machine breakdown and new job arrival in order to generate some good solutions. To transform the stochastic problem into dynamic version of deterministic processing times, we assume that the deterministic processing time of job i is the expected value of the probability distribution which characterised the unknown processing time of the same job. Since the feasible solutions for the deterministic benchmark are also possible solutions for the stochastic problem as well, where we are able to include them to the probabilistic scenario. After this, the expected values of makespan for the PFSP is estimated by employing the MCS technique. Thus, the current generated schedule will be simulate in the stochastic scenario. The MCS phase will be repeated running as many times as we require to obtain an enough reliable estimation. Thus, the steps of the Sim-PSO approach can be given as follows:

1. For stochastic FSP in a permutation scheduling, let us first consider stochastic processing times P_{ij} of jobs i and on machines j where the jobs number are n and the machines

number are m. Each stochastic processing time P_{ij} follows a probability distribution which is the Lognormal distribution with known mean $E[P_{ij}]$.

2. In the dynamic PFSP scheduling where the processing times are constant values, we consider the processing times p_{ij} as constant values given by $p_{ij} = E[P_{ij}]$.

3. For the dynamic PFSP under different real-time events, we generate an initial scheduling sequence (solution) by using the predictive-reactive based PSO algorithm (as we discussed in chapter 4). Also, the MSR model is employed to minimise the makespan, instability and robustness for this problem.

4. Improve the initial generated schedule by applying a classical LS algorithm, which is explained in the next section. Now, we consider the new improved solution as the new initial solution for the dynamic problem with constant processing times.

5. Apply a simulation for short runs, for example 250 iterations, to obtain the estimated expected stochastic solution associated with the sequence of jobs of the dynamic PFSP. After that, the stochastic solution is initilised by this estimated expected stochastic solution.

6. Employ the ILS technique (Lourenço et al., 2010) to improve the best of dynamic and stochastic solutions. Obtained so far. Thus, at each iteration, the ILS should complete the following two steps:

   (a) To create a new schedule, an ILS including the perturbation operator is used to the initial solution. This operator is called an Enhanced Swap Operator, which is performing as follows:

   - It picks out two different jobs randomly from the current obtained solution.
   - Swap the positions of both jobs in the permutation by moving each selected job to the other place of the other job.
   - Again apply a classical shift-to-left movement for both selecting jobs following a left-to-right order.

   The updating of the current solution to a new one is called deterioration. Sometimes this deterioration helps to prevent the solution from being trapped in bad local minimums in the solution space and it can be done from one of the following cases.

   (b) If the current base solution has a higher makespan value than the new obtained solution, then the baseline will be updated, also the new obtained solution is compared with the best dynamic schedule yet to decide if this best dynamic schedule

is required to be changed too. After each time of updating the dynamic schedule, the MCS technique with small number of runs (short simulation) is employed to evaluate the new expected makespan, and the best stochastic solution is updated if appropriate.

(c) Finally, if the makespan of the current baseline is less than the new obtained solution, then we apply an acceptance criterion to decide if we require to replace the baseline with the new solution or not.

7. Employ a simulation with long runs, e.g. 1000000 runs, to obtain good estimations for the expected stochastic makespans related to the best dynamic solution and the makespan related to the best stochastic solution. It should be noticed that, the stochastic makespan lower bound is related to the best dynamic solution and the stochastic makespan upper bound is related to the best stochastic solution this constitute the dynamic solution.

The Demon-like procedure (Talbi, 2009) is used as an acceptance criterion for the Sim-PSO approach. To do this, the criterion is simply that the baseline schedule will be deteriorated (modified with the new solution) even if the new solution was worse than the original one as long as; no consecutive deteriorations occur and the degradation does not overtake the value of the last improvement. For more details about the acceptance criterion and the perturbation process we refer to (Talbi, 2009).

The Sim-PSO approach then returns some information, which is given as follows:

- The best obtained dynamic solution and stochastic one so far and their related makespans.
- From stochastic makespans related to the best dynamic and stochastic solutions respectively, the method returns the sample stochastic makespans observations. The advantage from the sample observations is that they can be employed to determine more statistics, consequently, there is better understanding of their individual nature also to compare different solutions using criteria other than makespan.

Gourgand et al. (2003) and Dodin (1996) among others have proposed the methods of converting the SPFSP into an equivalent dynamic one, and then use an effective heuristic/metaheuristic to the transforming problem. Nevertheless, the proposed Sim-PSO method differs, by considering the following aspects:

- This Sim-based approach employs a more practical point of view by using the MCS approach, while the previous mentioned approaches are based on theoretical chance-constrained models.

- The proposed Sim-heuristic approach does not require to suppose a specific behaviour for the random variables which is used to model the processing times, while previous approaches require these assumptions. Thus, our proposed approach offers more flexibility for solving the SPFSP.

For this, as much as to our knowledge, the presented Sim-PSO provides unique features over other approaches in the literature for solving the SPFSP. Some of the strengths and advantages of our proposed algorithm are explained later in the section. The flowchart of Sim-PSO is given in figure 7.2 below.

The integration of simulation with PSO algorithm have been explained in the former sections. To understand this approach completely and also to implement it, we provide further relevant details in this section for the mentioned algorithm. The LS algorithm has been applied in different steps in the Sim-PSO approach, the proposed LS algorithm used in this research has been used by Ruiz & Stützle (2007) among others. The proposed LS algorithm is an iterative procedure, which has the following steps:

1. We select randomly a job with the position k from the current solution of sequence of jobs.

2. Insert the randomly selected job at the position k in each possible position located on the left side of job k. This procedure is called shift-to-left movement as it shown in figure 7.3. Then the makespan is calculated for each new permutation of jobs.

3. The last step is locating the job at the position that gives the minimum makespan.

These steps are iteratively repeated until all jobs locations have been visited or the improvement of makespan is attained. The Sim-PSO initialises the algorithm by using the PSO algorithm. The PSO algorithm is used for the dynamic PFSP under different real-time events, where the processing times are consider as constant values. After generating an initial solution and improve it by LS algorithm, the MCS is called into action. The integration of this approach with the mentioned metaheuristic is consider as one of the most important part of the approach. MCS is employed to estimate the expected makespan value that related to the given solution. This approach is shown in figure 7.4 and its steps are given in the following points:

a. Generate a random variate for all jobs processing times by employing the Log Normal probability distribution.

b. Depending on the jobs sequence obtained from the current solution, the random variates obtained from the proposed probability distribution are employed to generate a random stochastic makespan observations.

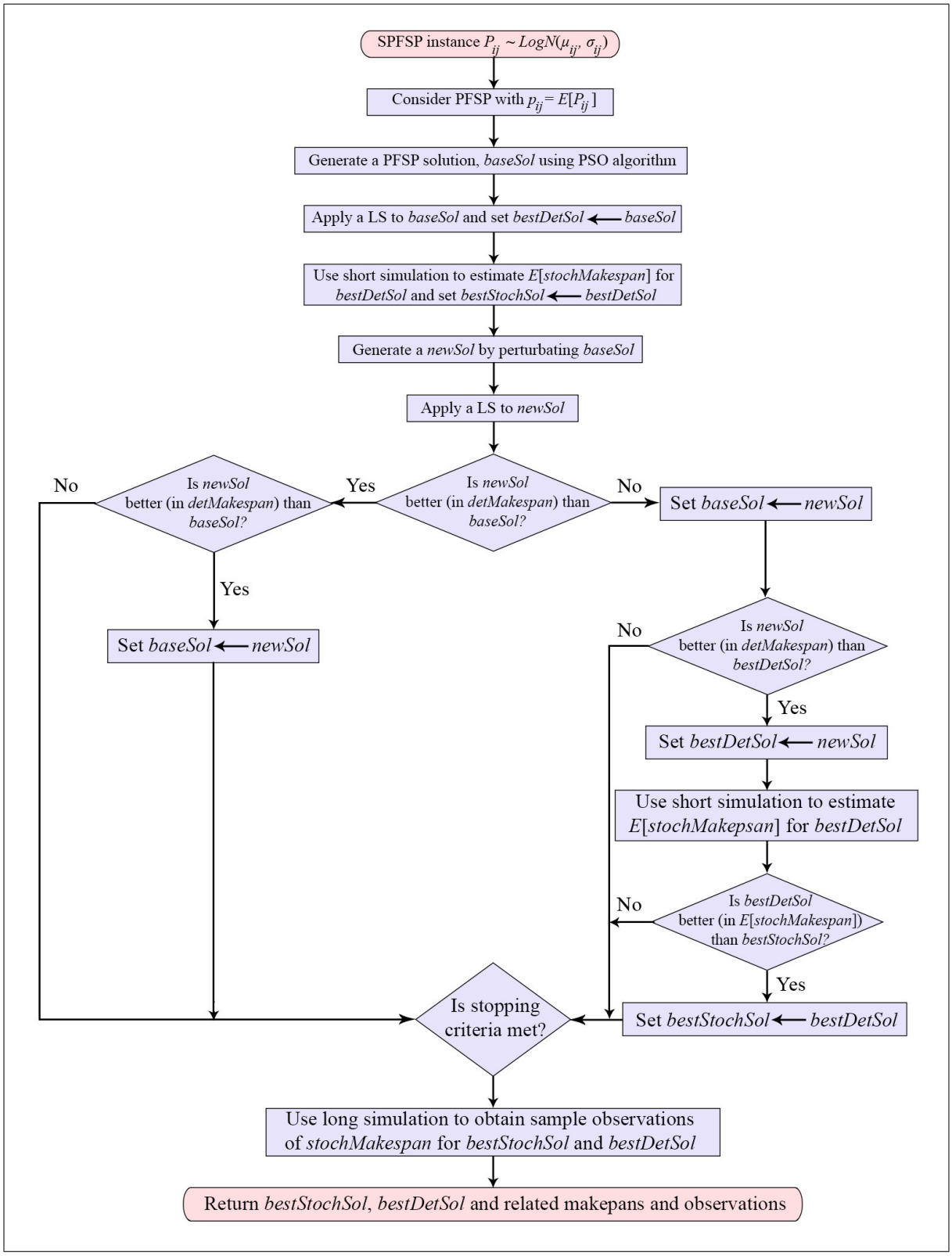


Fig. 7.2 Flowchart diagram of the Sim-PSO algorithm

c. These steps are iteratively repeated so as to build a random sample of makespan observations. Then these observations can be employed to estimate the expected makespan and interval estimates, also other probable statistics about the distribution of the stochastic makespan, such as extreme values, variance or quartiles.

d. The processes of the acceptance criterion and perturbation are used in this algorithm (Juan et al., 2014c). The acceptance criterion and the perturbation process were designed to prevent the solution from being trapped in a local minima during the algorithm process.

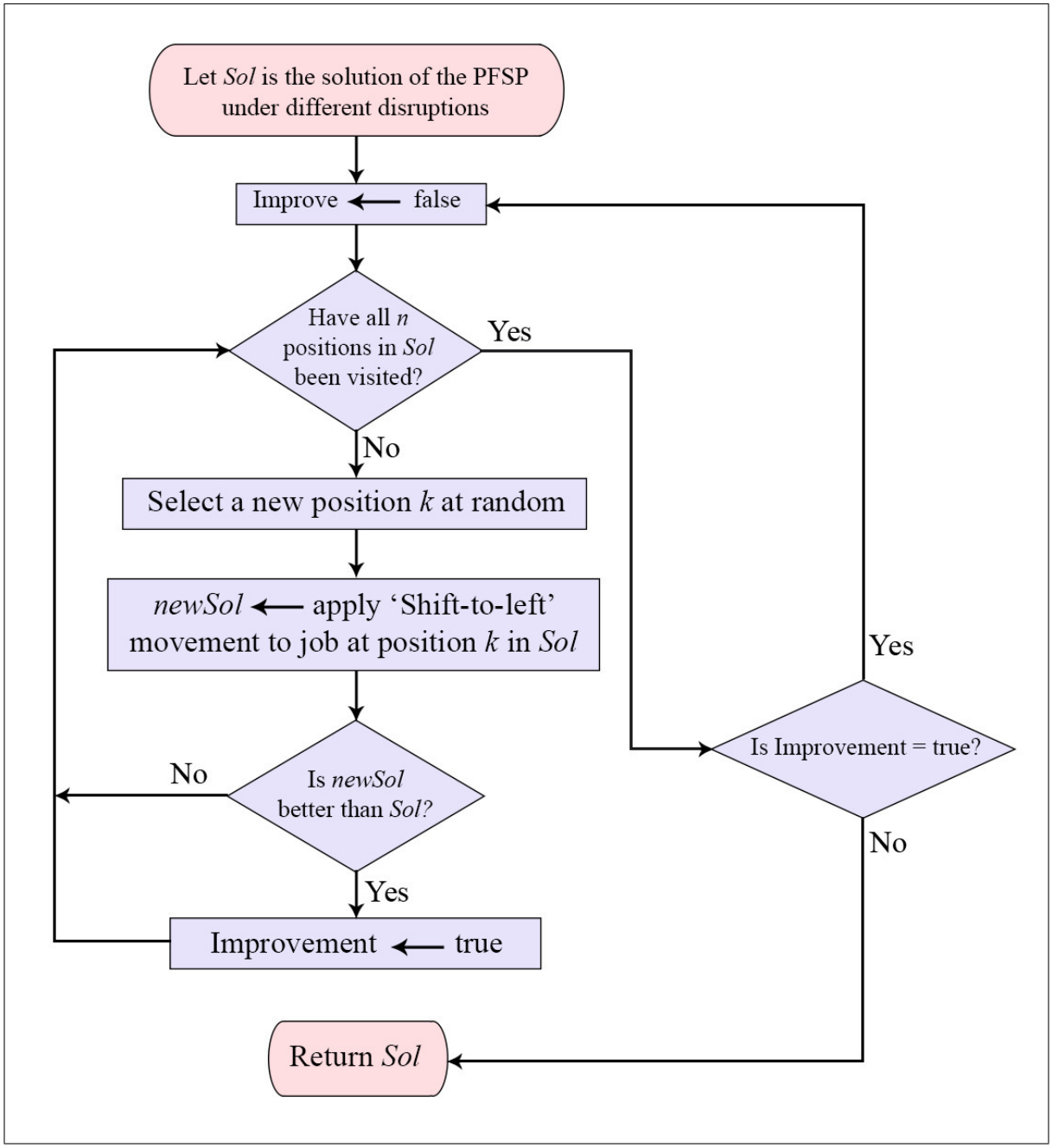


Fig. 7.3 Flowchart diagram of the LS algorithm

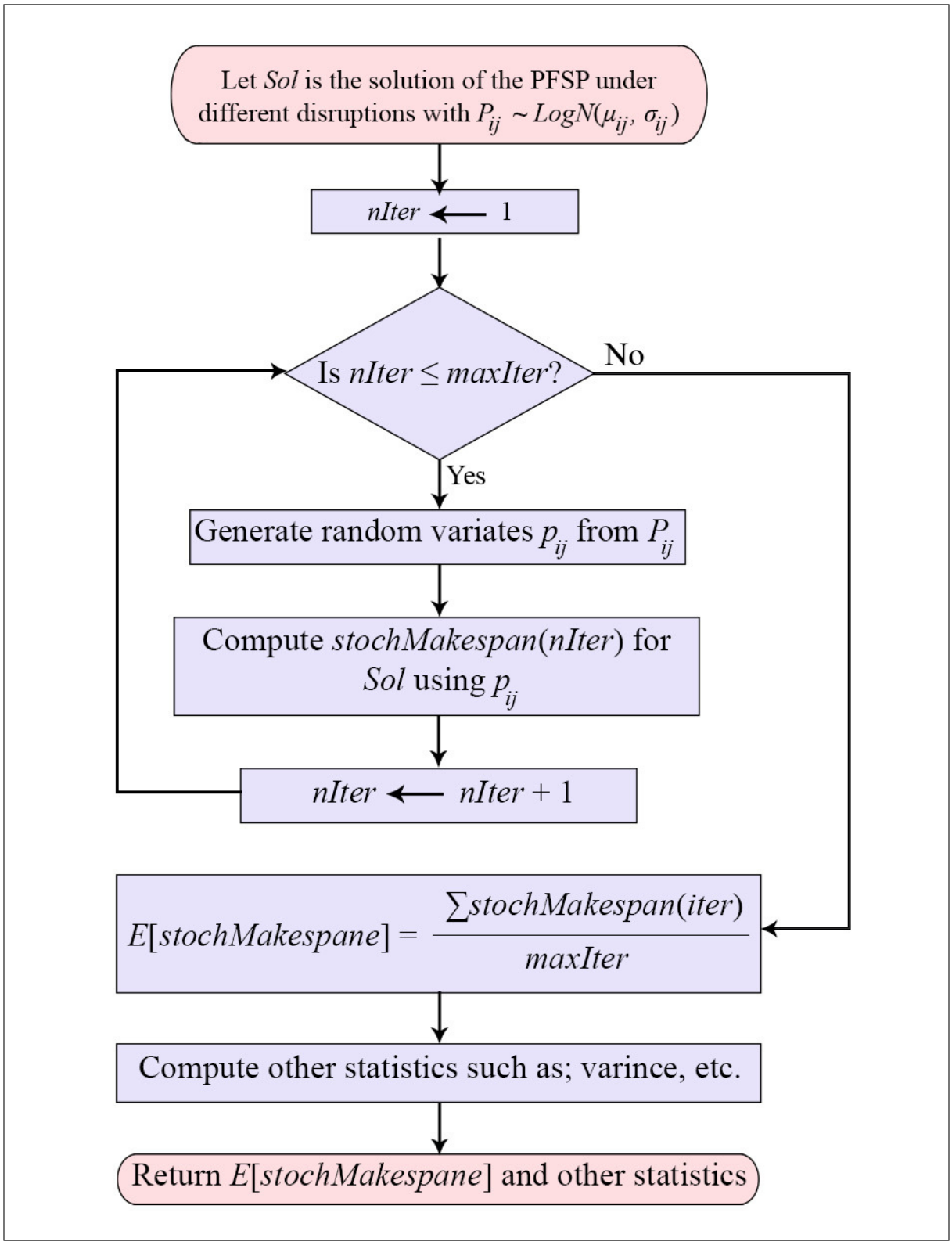


Fig. 7.4 Flowchart diagram of the MCS technique

7.4 Experiment Results

In this chapter, the experimental results and compression study for the approach is given and discussed. The predictive-reactive based PSO is hybridised with the MCS approach to solve the SPFSP under different real-time events, which are; machine breakdown and new job arrival. The MSR model is also used to keep the solution stable and robust. Java is used to implement all the experiments where it runs in eclipse platform and on a PC of Intel Cori5 2.6 GHz with 6GB of memory RAM. The proposed simulation based-optimisation approach starts with solving the dynamic part of the problem using the predictive-reactive based PSO algorithm with the MSR model, then, the MCS is applied to calculate the stochastic makespan. Since we apply the multi-objective optimisation mode (MSR model), the sensitivity analysis using a Practical Weighted Sensitivity approach of Jones (2011) is used, where thirteen distinct (α, β, γ) weight sets are generated (see chapter 3). The parameters used in this algorithm are the same that are used in chapter 4, hence, $TMax = 1$, $MaxLevel = 2$ and a sequential weight starting solution is set to be one.
The unity weights are applied to obtain the normalised model (3.2) as detailed in chapter 3. These three weights are;

$$(0.999, 0.001, 0.001),\ (0.001, 0.999, 0.001),\ (0.001, 0.001, 0.999)$$

While the sets of remaining ten different weights are using to test this experiment. These weights are given in Table 3.1 (see chapter 3).
There are no standard benchmarks in the literature of the SPFSP, even with the existence of the arguably exception benchmarks given by Baker & Trietsch (2011), which is a limited benchmark. This shows the existence of a lack in the knowledge of the SPFSP domain when compared to the PFSP under other environments. For this, for the instances introduced by Katragjini et al. (2013) are employed where random processing times are used instead of fixed ones in this experiment. Since the PFSP consider in this part is stochastic, which means the processing times are random variates and as we assume before, these processing times follow the Log Normal probability distribution. Thus, we suppose the processing times p_{ij} (of job i on machine j) proposed by Katragjini et al. (2013) (which based on the benchmark given by Taillard (1993)) as the expected processing times $p_{ij} = E[P_{ij}]$. The benchmark consists of 120 instances, categorised into12 problems of different size ranging from 20×5 to 500×20 ($jobs \times machines$). Each instances from the same size consists of 10 different problem. Katragjini et al. (2013) have reported the PFSP with different real-time disruptions including machine breakdown and new job arrivals. In this experiment, each problem have

been run for 5 independent times. In this experiment, we consider the limit $t_{max} = n \times m \times 0.03$ in seconds to stop the approach. This methodology provides some advantages:

A. A well-known instance of benchmarks which includes instances of different sizes are used in this methodology.

B. The values of all processing times are given in advance, hence, researches can use the instances, which have the same data information for benchmarking and verifying purposes.

C. In order to experience the proposed approach, we generalised the benchmark set of PFSP with different disruptions (Katragjini et al., 2013) to the stochastic environment.

Briefly, In the benchmark set of Katragjini et al. (2013), we change each deterministic processing times p_{ij} of job i on machine j to random variables P_{ij} following a well-known distribution where both mean and variance are given and $E[P_{ij}] = p_{ij}$. As we discussed before, since we employ the MCS, it is possible to use whatever probability distribution to model the jobs processing times, in other words, it is not require to suppose that the jobs processing times follow an Exponential or a Normal probability distribution. In this thesis, we choose a Log-Normal probability distribution to model the jobs processing times. The reason for using this distribution instead of the Normal distribution is that the Log-Normal distribution is a more normal option to model the non-negative processing times than the Normal distribution. The Log-Normal distribution has two parameters, namely; μ_{ij} and σ_{ij} parameters. These parameters are given in the equations below from the Log-Normal distribution properties.

$$\mu_{ij} = ln(E[P_{ij}]) - 0.5 \times ln\left(1 + \frac{V[P_{ij}]}{E[P_{ij}]^2}\right)$$

$$\sigma_{ij} = \left|\sqrt{ln\left(1 + \frac{V[P_{ij}]}{E[P_{ij}]^2}\right)}\right|$$

By using the parameter k, we assume that $Var[P_{ij}] = kE[P_{ij}]$. More specifically, we will consider the scenario where the levels of variances are relatively low levels of variances particularly $k = 0.1$ and $k = 0.5$, then medium level when $k = 2$ and finally at high level that is $k = 5$ (Juan et al., 2014a).

After each five independent runs, we register the average of the best found results of makespan solutions. Tables 7.1-7.3 show the results obtained in this experiment, for the SPFSP under different real-time events. Each Table includes the results corresponding to one of the ten weights W_i, $i = 1, 2, \ldots, 10$ where DRPD is the RPD for the dynamic solution and SRPD is the RPD for the stochastic solution. Each table shows the solutions corresponding to different variance levels where k is equal to $0.1, 0.5, 2$ and 5. Finally, each table have the following

information; dynamic solutions, stochastic solutions and the RPDs. From Tables 7.1 to 7.3, we can conclude the following two points:

- The RPD between the dynamic and stochastic solution of the SPFSP rises as the level of uncertainty increase. However, the RPDs are low for $k = 0.1$ comparing to the other k values.
- The expected RPDs corresponding to the best solution obtained from MCS are generally lower than the dynamic RPDs.
- The RPDs are variants typical to different weights, hence, the RPDs corresponding to weight $W_8 = (0.166, 0.166, 0.666)$ are generally lower than other RPDs of other weights. For this, we select the RPDs corresponding to the weight $W_8 = (0.166, 0.166, 0.666)$ for the next analysis study.

Table 7.1 The average DRPD and SRPD for weights W_1-W_4 using the Sim-PSO

	$n \times m$	$k = 0.1$		$k = 0.5$		$k = 2$		$k = 5$	
		DRPD	SRPD	DRPD	SRPD	DRPD	SRPD	DRPD	SRPD
W_1	20×5	25.762	25.76	28.311	28	29.468	29.467	29.592	29.592
	20×10	28.201	28.2	31.451	31.4	31.792	31.7	34.669	34.669
	20×20	22.051	22.05	24.451	24.447	26.013	26	26.452	26.434
	50×5	19.024	19.02	21.558	21.556	23.648	23.64	24.198	23.982
	50×10	29.287	29.2	31.320	31.315	33.158	33.1	33.857	33.357
	50×20	25.781	25	27.027	27.027	28.202	28.002	29.421	29.421
	100×5	16.84	16.84	20.239	20.241	21.92	21.92	23.039	23.039
	100×10	26.755	26.75	28.508	26.432	29.391	29.391	30.206	30.206
	100×20	25.291	25	26.03	26.029	28.863	28.863	29.815	29.814
	200×10	18.433	18.4	19.036	19.03	19.67	19.6	20.353	20.353
	200×20	19.863	19.86	20.632	20.632	22.081	22.081	22.185	22.185
	500×20	16.352	16.3	16.98	16.98	17.223	17.2	19.185	19.004
W_2	20×5	27.455	26.03	29.767	29.7	32.356	32.356	32.866	32.866
	20×10	28.985	27.47	29.014	29.014	31.536	31	34.149	34.149
	20×20	22.397	22.098	25.317	25.316	25.866	25.83	26.96	26.9
	50×5	22.559	22.4	22.991	22.925	23.123	23.123	23.256	23.256
	50×10	27.409	26.55	32.299	32.299	35.841	35.841	36.76	36.76
	50×20	25.401	25.401	26.777	26.776	30.044	30	30.829	28.09
	100×5	20.894	20.40	21.129	21.127	21.553	21.552	25.142	25.141
	100×10	28.442	27.9	31.274	30.509	32.489	32.4	32.737	32.737
	100×20	24.382	24.343	25.642	25.6	27.621	27.621	28.799	28.7
	200×10	15.783	15.38	16.952	16.951	18.855	18.855	19.386	19.386
	200×20	19.212	19.2	21.631	21.63	21.993	21.993	22.932	22.932
	500×20	16.037	16.033	17.61	17.61	18.228	18.2	19.461	19.461
W_3	20×5	25.684	25.21	29.294	29.2	32.994	32.994	34.011	34.011
	20×10	29.603	29.54	30.356	30	32.399	32.399	33.563	33.563
	20×20	22.769	22.68	23.76	23.7	24.599	24.599	26.649	26.648
	50×5	21.845	21.61	22.753	22.715	24.596	24	25.802	25.802
	50×10	30.122	28.76	31.995	31.995	33.134	33.003	33.61	33.61
	50×20	25.085	25.085	26.769	26.769	29.519	26.941	30.073	30.073
	100×5	21.494	20.12	21.812	21.809	22.931	22.931	24.813	24.8
	100×10	30.464	29.08	32.747	32.746	33.313	33.3	33.984	33.984
	100×20	23.43	22.49	26.251	26.249	26.807	26.807	28.158	28.158
	200×10	16.049	16.046	16.145	16.145	18.985	18.985	19.441	19.441
	200×20	19.697	19.43	20.225	20.225	22.142	22.142	23.089	23.089
	500×20	17.004	17.004	17.578	17.578	18.689	18.689	18.918	18.918
W_4	20×5	27.64	27.05	31.529	31.5	34.571	34.5	35.578	35.578
	20×10	30.069	29.7	31.154	31.151	32.236	32.236	34.314	34.314
	20×20	22.057	22.057	23.896	23.893	24.916	24.916	25.982	25.982
	50×5	20.387	20.066	22.783	22.7	24.211	24.211	25.908	25.002
	50×10	30.873	30.08	33.54	33.536	34.987	34.986	35.583	35.583
	50×20	23.231	23.231	24.762	24.761	28.998	28.998	29.604	29.604
	100×5	22.221	22.038	22.935	22.935	24.101	24.101	25.003	25.002
	100×10	28.53	28.087	28.438	28.4	33.268	33.268	34.249	34.248
	100×20	25.752	25.752	26.431	26.43	26.881	26.881	27.368	27.368
	200×10	16.655	16.22	17.551	17.549	19.879	19.879	20.804	20.804
	200×20	19.77	19.53	21.083	21.085	22.699	22	23.613	23.613
	500×20	16.947	16.947	17.828	17.724	18.956	18.956	21.083	21.083

Table 7.2 The average DRPD and SRPD for weights W_5-W_8 using the Sim-PSO

	$n \times m$	$k = 0.1$		$k = 0.5$		$k = 2$		$k = 5$	
		DRPD	SRPD	DRPD	SRPD	DRPD	SRPD	DRPD	SRPD
W_5	20×5	27.475	27.46	29.472	29.47	31.628	31.628	33.543	33.542
	20×10	30.184	30.18	30.79	30.79	31.812	31.812	32.884	32.88
	20×20	23.167	23.1	23.207	23.2	26.225	26.224	27.261	27
	50×5	22.639	22.6	22.962	22.959	26.278	26.278	26.884	26.884
	50×10	28.583	28	29.614	29.6	32.614	32.6	33.451	33.45
	50×20	24.123	24.12	25.255	25.2	28.122	28.121	29.001	29.001
	100×5	18.889	18	22.84	22.839	24.061	24.061	25.199	25.199
	100×10	29.886	29.8	30.938	30	31.19	31.19	32.467	32.467
	100×20	23.479	23.47	25.417	25.416	26.27	26	27.223	27.223
	200×10	17.355	17.35	17.766	17.764	18.602	18.602	20.195	20.195
	200×20	20.332	20	21.065	21.065	22.747	22.747	23.304	23.304
	500×20	16.634	16.634	17.821	17.821	18.932	18.92	21.368	21.368
W_6	20×5	24.806	24.55	28.76	28	29.608	29.602	30.14	30.14
	20×10	26.324	26.082	30.821	30.819	32.331	32.331	34.951	34.951
	20×20	23.897	23.32	24.998	24.002	25.728	25.727	29.713	29.713
	50×5	22.2	22.2	24.101	23.89	25.738	25.738	26.353	26.353
	50×10	28.243	28.1	29.968	29.96	33.462	33.4	34.98	34
	50×20	22.897	22.25	23.532	23.531	25.178	25.178	26.985	26.985
	100×5	22.578	22.09	24.053	24.053	25.096	25.096	25.704	25.704
	100×10	27.579	27.5	28.737	28	30.754	30.75	31.316	31.316
	100×20	24.009	24.008	25.151	25.15	25.914	25.914	29.629	29.629
	200×10	17.98	17.34	18.856	18.856	20.048	20.048	20.537	20.537
	200×20	19.72	16.98	20.471	20.47	23.732	23.733	22.417	22.417
	500×20	16.76	16.76	17.218	16.927	17.892	17.892	19.284	19.2
W_7	20×5	25.692	24.72	26.492	26.49	27.717	27.717	29.761	29.7
	20×10	31.599	29.68	32.007	32.008	32.85	32.852	34.229	34.229
	20×20	22.864	22.7	25.744	25.745	26.397	26.397	26.921	26.921
	50×5	21.603	21.601	23.963	23.963	25.039	25.039	26.26	26.26
	50×10	29.07	29.07	30.908	30.908	33.731	33.732	34.225	34.225
	50×20	28.397	28.36	24.476	24.47	32.166	32.1	27.658	27.658
	100×5	20.07	20.07	23.352	23.352	24.135	24.135	26.87	26.87
	100×10	27.279	26.41	29.17	29.17	30.395	30.395	31.465	31.465
	100×20	24.996	24.07	25.105	25.105	26.002	26.002	26.315	26.315
	200×10	15.053	15.053	17.428	17.429	19.165	19.165	19.818	19.818
	200×20	19.293	19.293	21.059	21.059	23.574	23.574	23.925	23.925
	500×20	16.824	16.8	17.851	17.851	18.014	18.014	20.274	20.274
W_8	20×5	21.114	20.353	26.692	26.691	27.58	27.529	29.971	29.96
	20×10	27.157	27.157	28.528	28.528	30.173	30.001	32.62	32.62
	20×20	21.057	21.018	22.787	22.78	24.198	24.197	24.898	24.898
	50×5	23.385	23.11	24.021	24.025	26.878	26.878	27.741	27.741
	50×10	23.903	23.888	24.058	24.058	24.873	24.874	25.647	25.507
	50×20	24.525	24.525	25.645	25.645	27.86	27.86	29.587	29.587
	100×5	15.412	15.38	19.838	19.836	22.434	22.434	21.635	21.634
	100×10	24.62	24.586	26.431	26.275	28.931	28.931	30.284	30.283
	100×20	21.911	21.911	26.767	26.767	27.691	27.691	28.818	28.818
	200×10	14.786	14.64	15.625	15.625	18.07	18.07	19.011	19.011
	200×20	14.976	14.517	15.762	15.76	16.846	16.845	17.806	17.783
	500×20	16.021	16.02	16.772	16.772	17.472	17.4	18.052	18.052

Table 7.3 The average DRPD and SRPD for weights W_9-W_{10} using the Sim-PSO

	$n \times m$	$k = 0.1$		$k = 0.5$		$k = 2$		$k = 5$	
		DRPD	SRPD	DRPD	SRPD	DRPD	SRPD	DRPD	SRPD
W_9	20×5	30.539	30.54	31.823	31.824	32.768	32.768	33.598	33.598
	20×10	28.268	28.27	29.797	29.797	33.497	34.956	34.956	34.956
	20×20	23.085	23.09	23.887	23.887	27.139	27.138	27.539	27.538
	50×5	22.389	22.389	23.323	23.323	24.06	24.06	26.283	26.283
	50×10	31.431	31.43	32.359	32.358	34.024	34.024	36.061	36.061
	50×20	27.071	27.07	27.283	27.284	28.161	28.161	29.176	29.176
	100×5	18.365	18.36	21.252	21.252	22.442	22.442	23.106	23.106
	100×10	28.25	28.25	30.724	30.72	31.686	31.686	32.793	32.79
	100×20	23.98	23.98	25.706	25.706	26.146	26.145	29.248	29.24
	200×10	15.55	15.55	18.428	18.428	18.555	18.555	19.778	19.778
	200×20	20.471	20.47	21.368	21.368	23.391	23.39	24.82	24.82
	500×20	16.13	16.13	16.96	16.967	18.681	18.681	19.327	19.327
W_{10}	20×5	27.04	27.04	29.996	29.996	30.3	30.3	32.258	32.258
	20×10	29.364	29.364	31.67	31.67	31.971	31.97	34.268	34.267
	20×20	23.001	23	24.305	24.304	26.398	23.367	28.819	28.819
	50×5	21.042	21.01	23.031	23.029	24.512	24.512	24.89	24.89
	50×10	29.024	29.024	30.217	30.212	34.286	34.286	34.854	34.854
	50×20	26.231	26.03	28.439	28.4	30.583	30.583	31.654	31.654
	100×5	17.952	17.952	22.859	22.859	23.006	23.006	24.199	24.199
	100×10	29.645	29.096	31.912	31.912	33.511	33.511	33.907	33.907
	100×20	24.936	24.936	25.369	25.368	26.719	26.718	27.216	27.216
	200×10	16.208	16.208	16.964	16.964	18.739	18.739	19.554	19.554
	200×20	0.767	20.767	23.457	23.457	23.833	23.833	24.551	24.55
	500×20	17.34	17.34	17.982	17.88	18.702	18.702	20.532	20.007

7.4.1 Using reliability-based methods to compare different solutions

To compare the obtained solutions, the most utilised criterion in the literature is the makespan objective. Thus, in the last section, we use the expected values of makespan to compare the obtained solutions. However, the manager might be more concerned in taking into account more information about the makespan corresponding to the obtain solution. Due to the randomness of the expected makespan, decision-makers could be concerned with the following questions; which is the probability of this makespan being lower than a given threshold? and which is the variance and distribution of this makespan? This is where historical data or, alternatively, simulation can make a difference by providing random observations of the makespan associated with a given solution. Figures 7.5-7.7 illustrate boxplots for three different size instances 20×5, 50×10 and 200×20 (small, medium and large) with $k = 0.1, 5$ (low and high variations). These boxplots intended to show the comparing values of makespans corresponding to the best stochastic and best dynamic solutions where these solutions are used to find the SPFSP solution. Subsequently, a random makespan is obtained as an output when any of these solutions are used in a scenario under stochastic processing times. Hence, the boxplot shows the differences

between stochastic and dynamic solutions, and also the probability distribution and variance of each set of outputs (not only the expected makespan). From figures 7.5-7.7 we notice that the makespan related to the stochastic solution show (in general) not only a smaller expected makespan but also smaller quartiles and variance. Also, with high variance level, the expected makespan, quartiles and variance are higher than the solution associated to dynamic makespan.

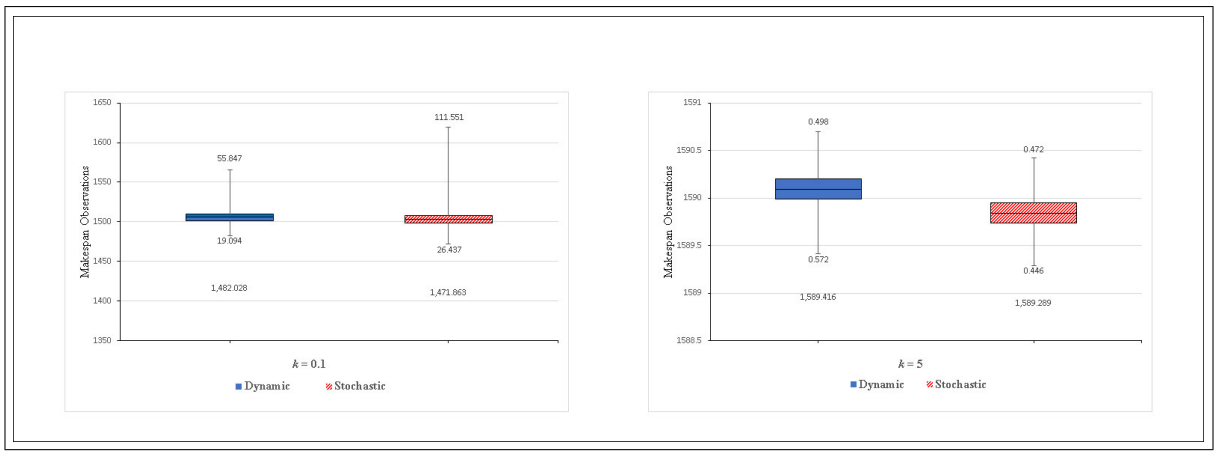


Fig. 7.5 Using MCS outputs to compare different solutions for problem of size 20×5 with $k = 0.1, 5$ and weight W_8

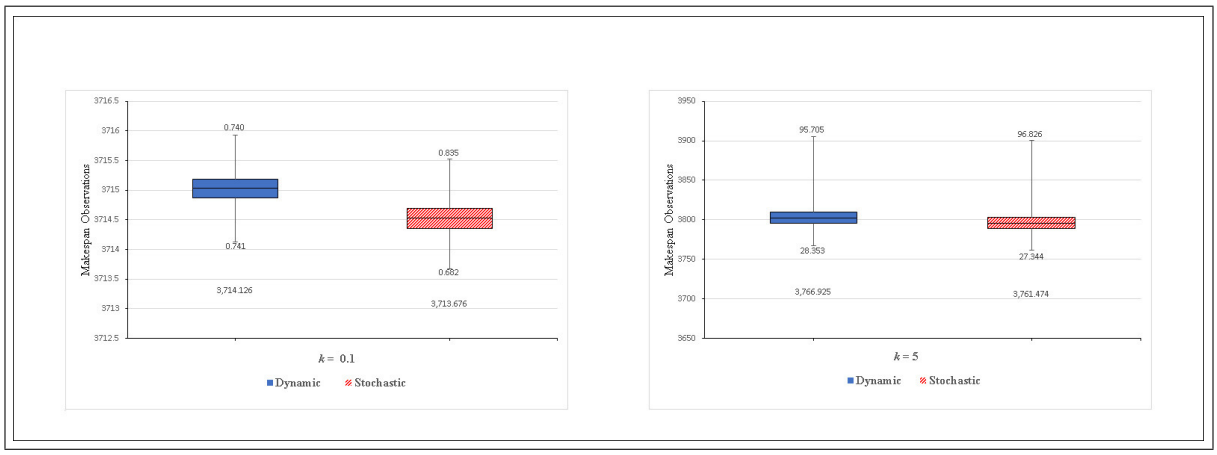


Fig. 7.6 Using MCS outputs to compare different solutions for problem of size 50×10 with $k = 0.1, 5$ and weight W_8

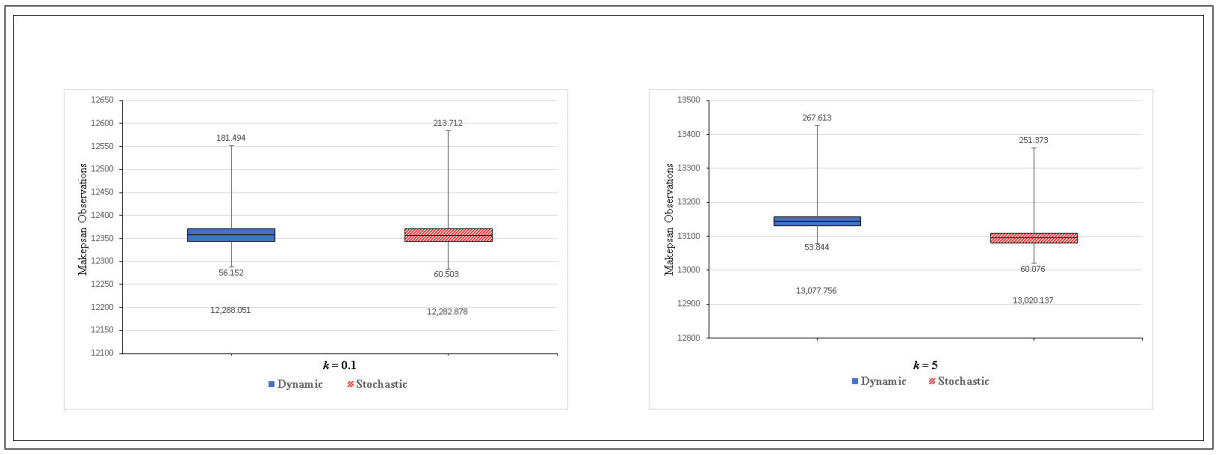


Fig. 7.7 Using MCS outputs to compare different solutions for problem of size 200×20 with $k = 0.1, 5$ and weight W_8

We also use the analysis of duration times (Modarres et al., 1999) to design an identification of makespans with reliability approaches. This is more important, as a makespan is the required time to finish all jobs. The reliability techniques might be more interesting than the classical statistical approaches, such as, distribution function plots or classical percentile. This could be true when facing historical data on random job duration, due to the fact that some observations could be incomplete data (censored). When censored data exist, the reliability approaches could help so as to provide very important data about the survival function corresponding with each solution. In 1958, a paper of Edward Kaplan and Paul Meier (Kaplan & Meier, 1958) introduced an efficient method for estimating patient survival rates, which is called Kaplan-Meier estimator. It is taking into account the fact that some patients may have died during a research trial while others will survive beyond the end of the trial. The Kaplan-Meier estimator based on a mathematical formula using information from those who have died and those who have survived to estimate the proportion of patients alive at any point during the trial. The estimator is plotted over time and the resulting curve is called the Kaplan-Meier curve, which is a series of horizontal steps of declining magnitude that, when a large enough sample is taken, approaches the true survival function for that population. In this section, the Kaplan–Meier estimator is used to calculate the survival function where it returns the inverse of the empirical distribution function. In reliability analysis, this is the most common non-parametric estimator used when dealing with non-censored data (Zio, 2007). Figures 7.8, 7.9 and 7.10 show the survival functions for different size instances selected previously with different levels of variances (k). The selected instances are of size 20×5, 50×10 and 200×20 with $k = 0.1, 5$, respectively. The survival functions are related for both of the best dynamic and the best stochastic solutions, respectively. The survival function produced from the observed stochastic makespans when employing the stochastic solution. This function corresponds to the

stochastic solution and is generally under the survival function corresponding to the dynamic solution. The dynamic solution is constructed from the observed stochastic makespans when the dynamic solution is used. It means in job schedule terms that the probability of keeping the job under operation will be generally lower when using the stochastic solution.

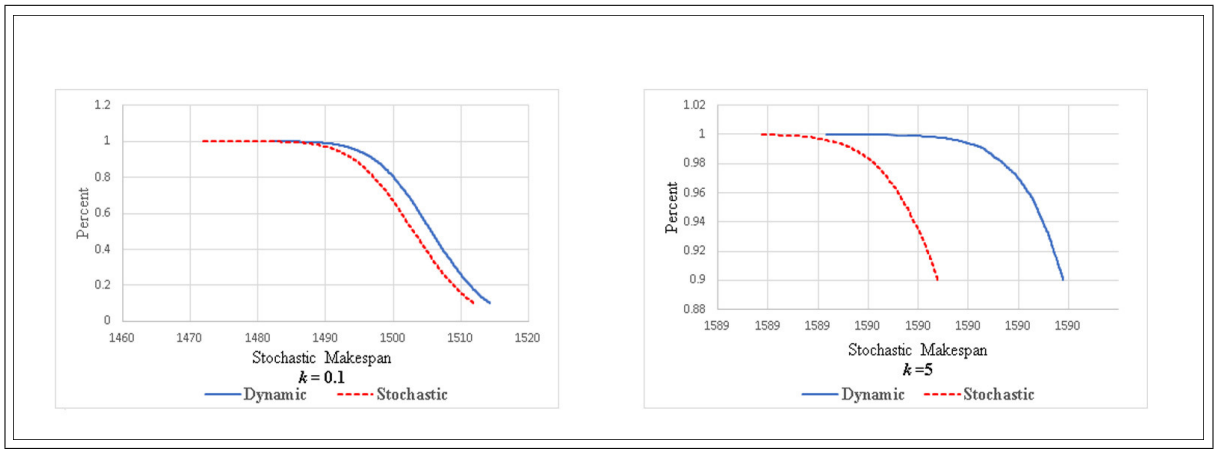


Fig. 7.8 Survival plot with intersecting solutions for problem 20×5 and $k = 0.1, 5$

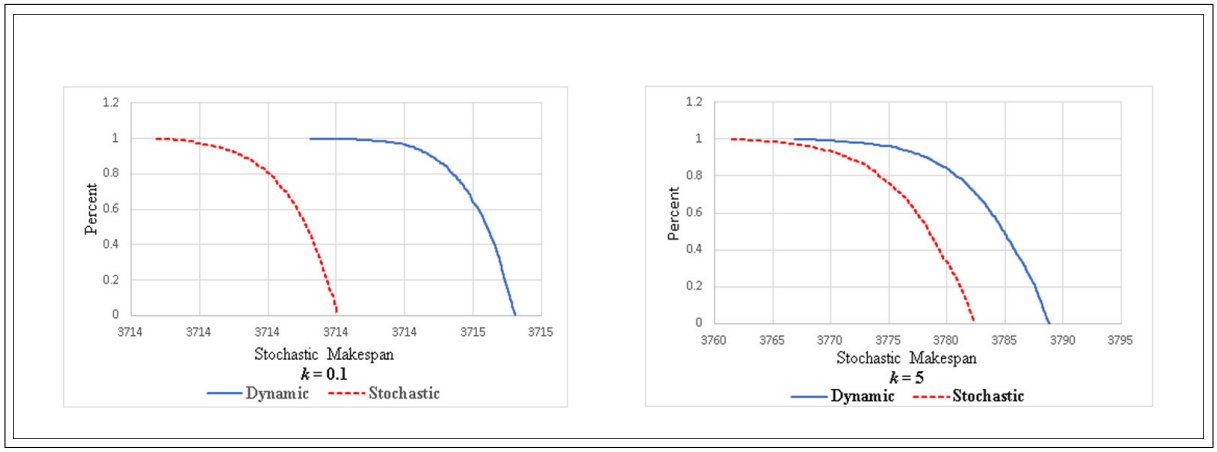


Fig. 7.9 Survival plot with intersecting solutions for problem 50×10 and $k = 0.1, 5$

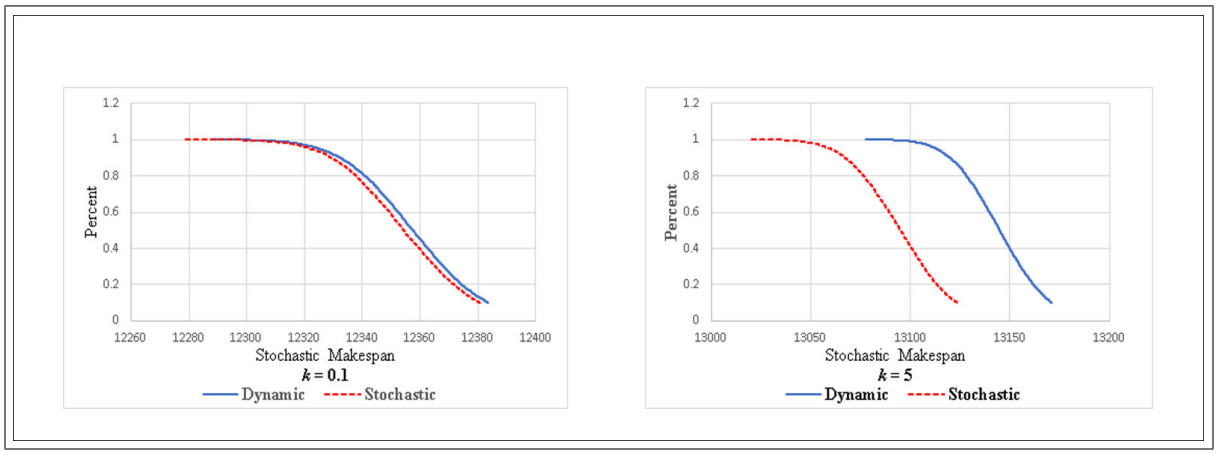


Fig. 7.10 Survival plot with intersecting solutions for problem 200×20 and $k = 0.1, 5$

7.5 Conclusion

In this chapter, we have presented the novel approach that hybridised the predictive-reactive based Sim-PSO, which hybridised the PSO algorithm with the MCS technique. This approach has been proposed with the MSR model to solve the SPFSP under different real-time events including machine breakdown and new job arrival. The main idea of the proposed method has the following points:

- Consider an initial PFSP with deterministic processing times instead of the stochastic one by replacing the expected values of the stochastic processing times with constant processing times values.
- Apply an efficient predictive-reactive based PSO algorithm and the MSR model to generate an initial good quality solutions for the dynamic PFSP under different real-time events.
- Consider each constant processing time as an expected value, then generating random variates using the MCS procedure to estimate the expected value of the stochastic makespan for each of the aforesaid solutions.

The PSO algorithm has the ability to deal with dynamic and stochastic problems successfully, hence, the contribution of this chapter is hybridising the evolutionary PSO algorithm with the MSC to solve the SPFSP under different uncertainties. It should be mentioned that, to our knowledge there is no other work to adapt the PSO with MCS as a framework to solve this problem. This hybrid methodology permits to use any theoretical or empirical probability distribution for solving the problem, where the probability distribution is employed to model

job-machine processing times. It does not matter if we use one probability distribution to model all job-machine processing times or we employ different probability distributions from one job-machine pair to another. Another contribution of this chapter is applying the MSR model to reduce instability and maintain robustness along with the Sim-PSO approach for the problem. Also, a set of experiments based on adapted instances from well-known benchmarks for the dynamic PFSP under different uncertainties are used to discuss the efficiency of the proposed approach. Moreover, this chapter indicates the analogy between work failure times and duration times and hence it provides the possibility of using procedures from Survival Analysis so as to better compare alternative solutions. Such analysis is important to compare different solutions, particularly when data of jobs is employed as real-life censored observations. Consequently, some realisations of the Sim-PSO may produce makespan values over the duration of the experiment, so providing incomplete or censored observations. In summary, this chapter clarifies some of the advantages which can be gained when hybridising the PSO algorithm with the MCS technique in solving the SPFSP under different uncertainties. The previous chapters discussed the application of each of the PSO, IG and BRIG algorithms and show the ability of these algorithms to handle the dynamic PFSP under different real-time events. Also, the BRIG algorithm outperform both of the PSO and IG algorithms in reaching better quality solutions. For this, we hybridise the BRIG with the MCS to solve the SPFSP under different real-time events as shown in the following chapter.

Chapter 8

Sim-Biased Randomised Iterated Greedy for Robust SPFSP

8.1 Introduction

In part I, we proposed three efficient approaches for the PFSP under machine breakdown and new job arrivals, which are the PSO, IG and BRIG algorithms. The PSO shows the ability to deal with such complex dynamic problem successfully. However, the PSO requires huge computational time to obtain a good quality solution. For this reason, a simple and efficient BRIG heuristic algorithm was proposed, it shows an exceptional performance in reaching good local optimal solution in reasonable computational time. In this chapter, we propose the Sim-BRIG framework where the MCS is integrated with the BRIG algorithm. The proposed Sim-BRIG approach is designed to solve the SPFSP under different disruptions of real-time events. Also, the MSR model is defined for this problem to keep the solution efficient, stable and robust. In Sim-BRIG approach, the predictive-reactive based BRIG approach is proposed first to solve the dynamic PFSP, where the MSR model is applied. Then the MSC is performed to generate an expected makespan values. This chapter is organised as follows; Section 8.2 introduces the framework of the Sim-BRIG approach, and more details about this method are introduced in this section. Section 8.3 illustrates the experimental results including the comparative study between the Sim-BRIG and Sim-PSO approaches. Finally, the conclusions of this chapter are given in section 8.4.

8.2 The framework of Sim-BRIG approach for SPFSP under different disruptions

As we discussed in the previous chapter, the SPFSP is considered under dynamic environment where the processing times are random variables. Also, the Sim-PSO approach was used with the MSR model to solve this problem. The PSO required relatively hug computational time to obtain good solution. However, in this chapter, we apply the simple and effective Sim-BRIG with the MSR model to solve the SPFSP under machine breakdown and new job arrival. In this approach, the predictive-reactive based BRIG approach is used to solve the dynamic part of this problem and hence, the Sim-BRIG predicts an initial planned schedule by solving the dynamic problem using the BRIG algorithm. Then the MCS is applied to generate an expected value for the stochastic makespan. The hybridisation of the BRIG algorithm with MCS procedure is explained in the following subsection.

8.2.1 Integrated Simulation with the BRIG algorithm

In the integration of MCS and BRIG algorithm, the simulation is consider as a part of the solution procedure by estimating the expected values of stochastic makespan. The generation of random variates and estimation in the simulation are integrated with the solution of the BRIG algorithm that have been applied for the dynamic PFSP under different real-time events. This hybridisation starts from the sequence of jobs obtained from the predictive-reactive based BRIG algorithm. Then it assumes the processing times of jobs as expected values for random processing times which follow the Log Normal probability distribution (Juan et al., 2014a). At this point, the simulation implemented the MCS procedure to generate random variates for each expected processing time and hence calculate the expected stochastic makespan. The approach repeat this process for fixed number of iterations. For example 200 runs for short simulation and 10000 for long simulation, figure 8.1 explain the idea of the integration between MCS and BRIG algorithm.

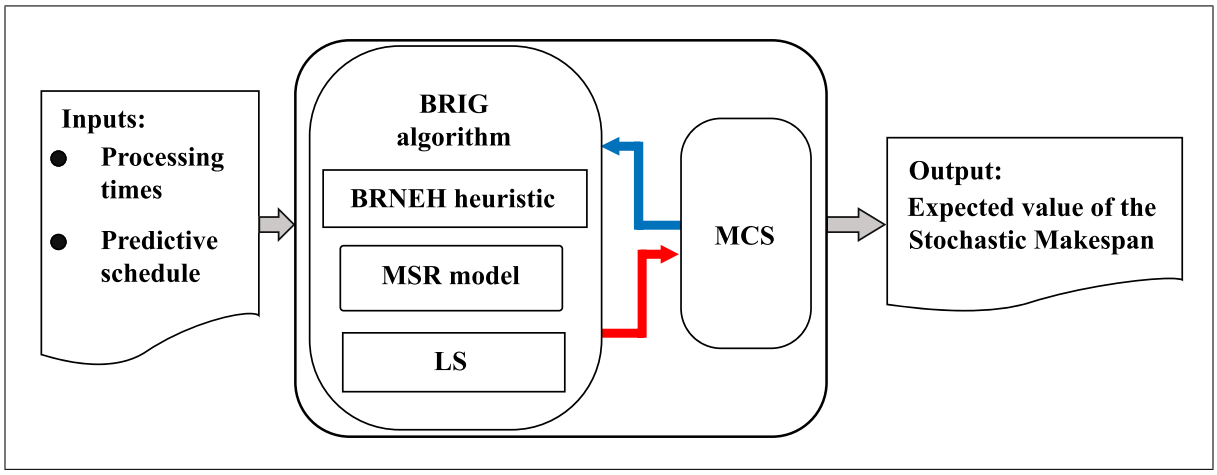


Fig. 8.1 The integrated MCS approach and BRIG algorithm

8.2.2 The Sim-BRIG algorithm

For the SPFSP under machine break down and new job arrival, we assume J is the set of jobs and M is the set of machines. In this problem, the processing times P_{ij} are considered as random variables following a Log Normal distribution where the mean $E[P_{ij}]$ is known for each job $i \in J$ on each machine $j \in M$. The steps of the Sim-BRIG can be summarised in the following points:

1. Solve the dynamic PFSP under machine breakdown and new job arrival where the sets J and M are defined. At this point, the processing times p_{ij} are considered as fixed values given by $p_{ij} = E[P_{ij}]$.

2. Determine the best sequence of jobs for the dynamic PFSP by using the predictive-reactive approach based BRIG, which starts by generating an initial solution for the problem using the BRNEH heuristic.

3. Improve the current solution by using a classical LS technique to the base solution as explained in section 7.3, then replace the current solution with the improved one as the new baseline and also the best dynamic solution so far.

4. Apply simulation with small number of iterations such as 300 runs (short MCS) to construct estimation of the expected makespan related to the sequence of jobs obtained from the best dynamic solution. Then, initialise the best stochastic solution as the best dynamic solution.

5. Apply the ILS technique (Lourenço et al., 2010) such that the best dynamic and stochastic solutions will be improved iteratively. Continue this technique until a stopping criteria is attained.

6. Employ the MCS such as 1E7 runs (long MCS) to generate more accurate estimations for the expected values of two different stochastic makespans, the one associated with the best stochastic solution and the one associated with the best dynamic solution. Notice that, when considering the parameters (not the estimates), the dynamic makespan associated with the best dynamic solution and the (stochastic) expected makespan associated with the best dynamic solution constitute, respectively, a lower and an upper bound for the (stochastic) expected makespan associated with the best stochastic solution.

8.2.3 More details about Sim-BRIG algorithm

More relevant details about the Sim-BRIG approach are given in this section to explain the whole idea of this approach. Some of these details are the LS and ILS procedures, which are used in different steps in the Sim-BRIG approach, we applied the same LS and ILS techniques that are used in chapter 7 to give a fair comparison between this approach and the Sim-PSO approach. The Sim-BRIG is initilised by using the BRIG algorithm as we explained before. The BRIG algorithm is used for the dynamic PFSP where the processing times are consider as constant values. After generating an initial solution and improve it by LS algorithm, the MCS is employed to estimate the expected stochastic makespan related to the given solution. The algorithm steps can be summarised as follows:

- Generate a random variate for all jobs processing times by employing the Log Normal probability distribution.
- According to the jobs sequence given by the current solution, these random variates are employed to construct a random observation of the stochastic makespan.
- These steps are iteratively repeated so as to get a random sample of makespan observations. These observations can then be employed to estimate the expected makespan and interval estimates, also other probable statistics about the distribution of the stochastic makespan, such as extreme values, variance or quartiles.
- The procedure of the acceptance criterion and the perturbation proposed by Juan et al. (2014a) are used in this algorithm.

In the Sim-BRIG approach, a very simple, fast, and efficient operator is also used at the perturbation process, which is the Enhanced Swap Operator, this operator is explained in details

in chapter 7. Finally, the Demon-like procedure is used as an acceptance criterion for the Sim-BRIG approach as explained in the previous chapter.
To some extent, the Sim-BRIG approach is reducing this complex problem into a more tractable SPFSP where fast, excellent and widely tested metaheuristics exist in the literature. Indeed, for our method, the basis employs one of these efficient metaheuristic algorithms, which is BRIG. This raises validity to the quality of the final solutions given to the decision-maker.

8.3 Experimental results

The Sim-BRIG approach is applied for the SPFSP under different real-time events. The method starts with generating a solution for the dynamic PFSP where a predictive-reactive approach is applied to accommodate the new disruptions. Then it employs the MCS approach to estimate the expected makespan. In this approach, we use a computational time limit $t_{max} = n \times m \times 0.03$ seconds to stop the procedure. This limited computational time is the same time used in the Sim-PSO approach. Tables 8.1, 8.2 and 8.3 shows the RPDs for dynamic (DRPD) and stochastic (SRPD) solutions corresponding to different weights. From these tables, we can conclude that the RPD between the stochastic solution and the dynamic one increases as the level of uncertainty increase. However, the RPDs are generally relatively low even for such complex NP-hard dynamic and stochastic problem. These RPDs ranging from 5.719% for $k = 0.1$ to 25.572% for $k = 5$. For this, our solution has relatively lower RPD in all tested problems for low variation. Also, the solution related to the weight $W_6 = (0.002, 0.498, 0.498)$ has in general lower RPD values when compared to the other weights. This emphasis the importance of stability and robustness measures. Figures 8.3, 8.4 and 8.5 illustrate boxplots for three different size instances 20×5, 50×10 and 200×20 (small, medium and large) with $k = 0.1, 0.5, 2, 5$ corresponding to the weight $W_6 = (0.002, 0.498, 0.498)$. These boxplots are intended to show the comparing values of makespans corresponding to the best stochastic and best dynamic solutions where these solutions are used to find the SPFSP solution. From figures 8.3-8.5 we notice that our approach shows in general a relatively smaller expected stochastic makespan, also, it shows smaller variance and quartiles. Moreover, with high variance level, the expected makespan, quartiles and variance are higher than the solution associated to dynamic makespan. We also use the reliability analysis by using the Kaplan–Meier estimator to compute the survival function. Figures 8.6, 8.7 and 8.8 show the survival functions for instances of small, medium and large size, 20×5, 50×10 and 200×20 ($jobs \times machines$) with $k = 0.1, 5$.
The survival functions obtained from the observed stochastic makespans when using the stochastic solution are generally lower than the survival functions of the dynamic solution. This

shows that the probability that the jobs are still in process will be generally lower when using the stochastic solution.

Table 8.1 The average DRPD and SRPD for weights W_1-W_4 using the Sim-BRIG

	$n \times m$	$k = 0.1$		$k = 0.5$		$k = 2$		$k = 5$	
		DRPD	SRPD	DRPD	SRPD	DRPD	SRPD	DRPD	SRPD
W_1	20×5	25.762	25.76	28.311	28	29.468	29.467	29.592	29.592
	20×5	13.121	13.121	15.612	15.588	16.539	16.119	17.649	17.649
	20×10	13.587	13.58	15.203	15.201	17.044	17.04	18.147	18.146
	20×20	16.103	15.686	16.82	16.82	18.508	18.16	20.523	19.843
	50×5	12.057	11.855	13.704	13.7	15.678	15.597	16.298	16.298
	50×10	18.038	18.038	18.349	18.13	19.829	19.829	20.44	20.359
	50×20	17.113	16.997	17.595	17.55	19.519	19.5	20.124	20.124
	100×5	8.058	8.049	9.492	9.49	11.944	11.94	12.973	12.973
	100×10	16.463	16.398	18.502	18.5	19.148	19.148	20.252	20.252
	100×20	14.598	14.597	15.703	15.7	17.341	17.341	17.907	17.907
	200×10	20.61	20.6	23.204	23.2	23.827	23.8	24.783	24.783
	200×20	18.498	18.4	20.939	20.9	22.278	22.278	23.434	23.434
	500×20	12.638	12.638	13.004	13.004	13.893	13.8	14.158	14.158
W_2	20×5	12.822	12.82	13.59	13.59	14.773	14.241	18.389	18.389
	20×10	13.668	13.429	15.037	15.03	17.168	17.168	17.382	17.277
	20×20	12.606	12.6	14.703	14.55	20.308	20.057	20.523	20.523
	50×5	10.389	10.385	12.281	12.28	14.607	14.6	16.158	16.158
	50×10	18.613	18.61	19.737	19. 6	21.127	21.113	22.403	22.403
	50×20	12.272	12.27	14.317	14.3	14.935	14.935	18.957	14.284
	100×5	7.694	7.692	8.696	8.69	9.503	9.502	10.543	10.507
	100×10	13.786	13.781	15.634	15.61	17.218	17.217	18.841	18.841
	10×20	12.285	12.28	13.764	13.76	15.104	15.103	20.049	19.753
	200×10	21.286	21.285	22.247	22.2	23.228	23.227	23.275	23.275
	200×20	17.692	17.69	19.723	19.72	20.353	20.353	21.806	21.806
	500×20	12.28	12.2	12.947	12.9	13.457	13.456	13.886	13.886
W_3	20×5	11.268	11.2	14.135	14.13	19.086	18.53	19.663	19.661
	20×10	10.542	10.54	14.046	14	13.103	13.038	15.896	15.895
	20×20	13.181	13.179	14.549	14.1	15.96	15.496	16.612	16.612
	50×5	11.334	11.33	12.514	12.51	13.009	13	14.3	14.096
	50×10	14.473	14.467	15.541	15.54	16.654	16.654	18.588	18.588
	50×20	12.063	12.058	18.24	18.24	21.087	21.087	22.576	22.576
	100×5	6.258	6.2	9.51	9.51	9.953	9.953	10.58	10.5
	100×10	16.789	16.16	16.462	16.46	17.181	17.181	17.915	17.915
	100×20	13.422	13.421	14.184	14.1	14.227	14.227	15.249	15.249
	200×10	20.006	20	23.66	23.64	24.523	24.523	25.572	25.501
	200×20	17.233	17.232	19.722	19.72	21.068	21	22.965	22.965
	500×20	12.371	12.37	12.781	12.781	13.327	13.327	13.983	13.983
W_4	20×5	12.127	11.652	13.599	13	14.585	14.585	15.275	15.275
	20×10	13.29	13.287	14.661	14.52	15.72	15.72	17.723	16.723
	20×20	13.163	13.022	13.734	13.73	14.609	14.6	17.269	16.027
	50×5	10.748	10.748	15.416	15.41	16.549	16.549	17.321	17.3
	50×10	14.532	14.502	16.262	16.26	17.959	17.702	18.187	18.187
	50×20	12.777	12.776	13.921	13.92	14.501	14.5	17.534	17.534
	100×5	8.705	8.7	9.91	9.91	10.433	10.433	11.965	11.102
	100×10	14.032	14.03	16.904	16.9	17.073	17.073	18.053	18
	100×20	12.264	12.258	13.357	13.049	14.273	14.27	15.061	15.061
	200×10	20.47	20.469	22.876	22.87	23.056	23.056	23.546	23.546
	200×20	17.089	17	19.514	19.5	20.682	20.681	21.343	21.343
	500×20	12.108	12	12.29	12.29	13.049	13	14.617	14.617

Table 8.2 The average DRPD and SRPD for weights W_5-W_8 using the Sim-BRIG

	$n \times m$	$k = 0.1$		$k = 0.5$		$k = 2$		$k = 5$	
		DRPD	SRPD	DRPD	SRPD	DRPD	SRPD	DRPD	SRPD
W_5	20×5	12.36	12.3	15.764	15.7	18.34	18.339	19.688	19.688
	20×10	15.165	15.153	16.101	16.1	17.907	17.907	19.063	19.063
	20×20	16.452	16.45	17.851	17.78	18.716	18.716	19.399	19.086
	50×5	12.124	11.811	13.839	13.8	16.437	16.437	17.02	17.02
	50×10	16.107	16.1	18.095	18.05	19.051	19.051	19.582	19.582
	50×205	17.655	17.66	17.966	17.9	18.64	18.64	19.639	19.639
	100×5	7.209	7.2	10.495	10.5	11.106	11.106	11.739	11.739
	100×10	17.029	17	17.997	17.74	18.421	18.421	19.848	19.848
	100×20	14.601	14.599	15.967	15.9	17.378	17.378	18.31	18.31
	200×10	20.549	20.547	21.845	21.82	22.868	22.868	23.71	23.655
	200×20	19.376	19.374	20.434	20.43	22.414	22.414	22.721	22.721
	500×20	12.371	12.37	12.781	12.781	13.327	13.327	13.983	13.983
W_6	20×5	8.949	8.351	12.23	12.13	12.482	12.48	14.5	14.48
	20×10	11.672	11.6	12.787	12.7	17.915	17.298	19.312	19.312
	20×20	11.253	10.159	12.922	12.92	13.971	13.002	14.187	14.187
	50×5	9.956	9.808	10.978	10.9	11.038	11.038	12.807	12.807
	50×10	12.337	12.229	13.984	13.98	14.207	14.207	14.703	14.69
	50×20	10.925	10.924	11.985	11.98	20.27	20.27	21.257	20.852
	100×5	5.721	5.719	6.876	6.8	11.052	11.015	12.12	12.12
	100×10	13.876	13	15.337	15.3	17.123	17.12	17.65	17.65
	100×20	16.434	16.434	17.504	17.5	17.955	17.955	18.609	18.609
	200×5	11.212	11.21	12.79	12.79	15.318	15.318	15.658	15.643
	200×10	14.116	14.064	17.8	17.77	18.259	18.259	18.611	18.56
	500×20	10.281	10.281	11.963	11.9	12.217	12.217	12.986	12.984
W_7	20×5	14.521	13.924	14.885	14.88	16.707	16.7	18.935	18.934
	20×10	13.507	13.5	14.638	14.04	15.994	15.606	16.968	16.896
	20×20	15.612	15.61	17.992	17.99	18.482	18.482	18.985	18.985
	50×5	12.273	12.272	12.915	12.9	15.584	15	16.29	16.02
	50×10	16.979	16.056	17.609	17.6	17.74	17.74	20.266	20.266
	50×20	16.675	16.67	17.417	17.4	17.582	17.582	20.399	20.399
	100×5	7.278	7.275	9.617	9.6	10.962	10.961	11.865	11.865
	100×10	15.74	15.712	18.76	18.76	19.263	19.26	19.552	19.552
	100×20	14.695	14.693	16.143	16.14	17.257	17.257	17.389	17.389
	200×10	21.157	21.156	21.755	21.75	23.973	23.973	24.588	24.201
	200×20	18.245	18.21	19.969	19.9	22.094	22.094	22.824	22.824
	500×20	11.894	11.89	12.384	12.384	13.207	13.2	16.638	16.638
W_8	20×5	12.313	12.299	14.559	14.5	18.592	18.59	15.714	15.626
	20×10	10.427	10.42	18.374	18.154	14.08	14	17.718	17.718
	20×20	11.399	11.002	14.023	14	20.29	19.847	20.336	20.3
	50×5	11.685	11.671	13.653	13.5	16.227	16.227	15.793	15
	50×10	18.017	17.939	18.903	18.9	22.992	22.99	19.836	19.836
	50×20	15.897	15.891	18.622	18.62	14.364	14.364	14.285	14.158
	100×5	8.677	8.673	11.31	11.31	12.299	12.299	12.363	12.363
	100×10	13.231	13.23	14.498	14.01	16.641	16.538	20.653	20.467
	100×20	16.195	16.19	16.736	16.73	18.692	18.598	14.753	14.05
	200×10	14.448	14.446	16.369	16.34	17.423	17.423	18.123	18.123
	200×20	15.579	15.576	17.965	17.9	20.115	20.115	21.108	21.1
	500×20	12.492	12.49	12.93	12.927	14.07	14.074	15.341	15.341

Table 8.3 The average DRPD and SRPD for weights W_9-W_{10} using the Sim-BRIG

	$n \times m$	$k = 0.1$		$k = 0.5$		$k = 2$		$k = 5$	
		DRPD	SRPD	DRPD	SRPD	DRPD	SRPD	DRPD	SRPD
W_9	20×5	12.547	12.558	14.81	15.25	17.08	17.081	18.72	18.409
	20×10	15.151	15.146	16.301	16.3	17.775	17.776	19.54	19.54
	20×20	15.565	15.52	17.087	17.078	17.904	17.989	18.742	18.63
	50×5	11.18	11.18	14.471	14.47	14.867	14.866	16.928	16.928
	50×10	16.003	16	18.43	18.43	20.33	20.329	20.449	20.449
	50×20	16.457	16.448	16.694	16.69	18.358	18.357	19.33	19.33
	100×5	8.247	8.243	8.903	8.9	9.072	9.072	9.582	9.582
	100×10	14.42	14.4	18.388	18.46	20.553	20.553	21.428	20.913
	100×20	13.741	13.7	16.477	16.48	17.675	17.675	18.06	18.06
	200×10	20.476	20.476	23.175	23.17	25.196	25.196	25.175	25.175
	200×20	18.721	18.719	21.446	21.45	22.247	22.246	22.897	22.94
	500×20	11.674	11.674	12.3	12.288	12.892	12.892	13.629	13.629
W_{10}	20×5	13.253	13.245	14.48	13.45	15.544	15.544	16.622	16.565
	20×10	15.971	15.967	16.68	16.68	19.658	19.175	20.726	20.678
	20×20	15.648	15.64	17.81	17.6	18.54	18.54	20.203	19.077
	50×5	13.308	13.282	14.57	14.57	16.365	16.364	18.651	17.651
	50×10	17.958	17.838	18.059	18	18.828	18.827	20.827	20.827
	50×20	14.12	14.12	17.082	17.06	18.138	18.138	21.743	19.743
	100×5	9.008	9.005	9.294	9.2	11.364	11.36	12.314	12.314
	100×10	15.932	15.871	17.628	17.6	18.335	18.335	19.962	19.962
	100×20	14.001	13.997	15.974	15.97	16.802	16.802	17.541	17.541
	200×10	20.335	20.334	23.256	23.25	23.551	23.551	25.339	25.339
	200×20	18.833	18.83	21.362	21.36	21.955	21.955	23.618	23.618
	500×20	12.581	12.581	13.915	13.9	14.054	14.05	14.796	14.796

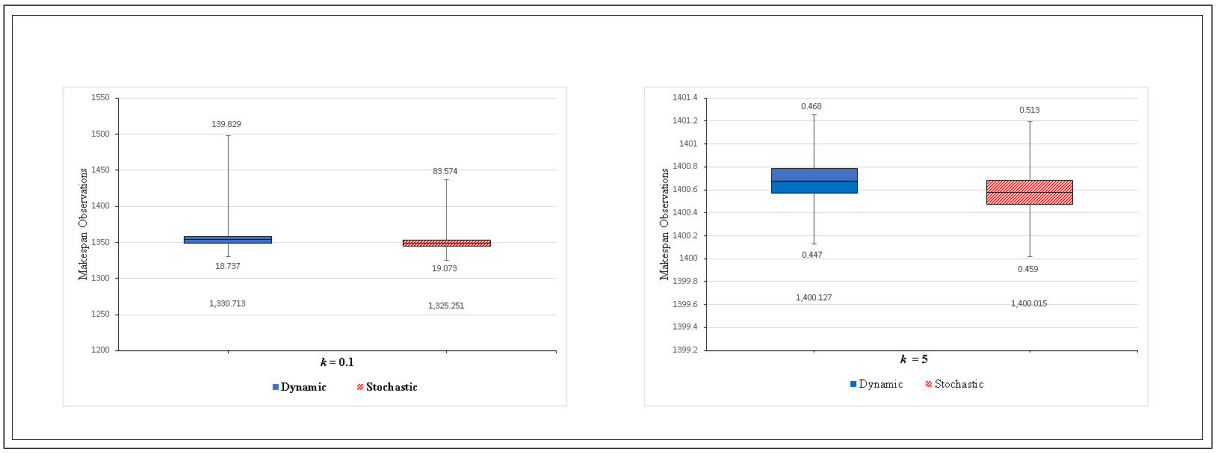


Fig. 8.2 Using MCS outputs to compare different solutions for problem of size 20×5 with $k = 0.1, 5$ and weight W_6

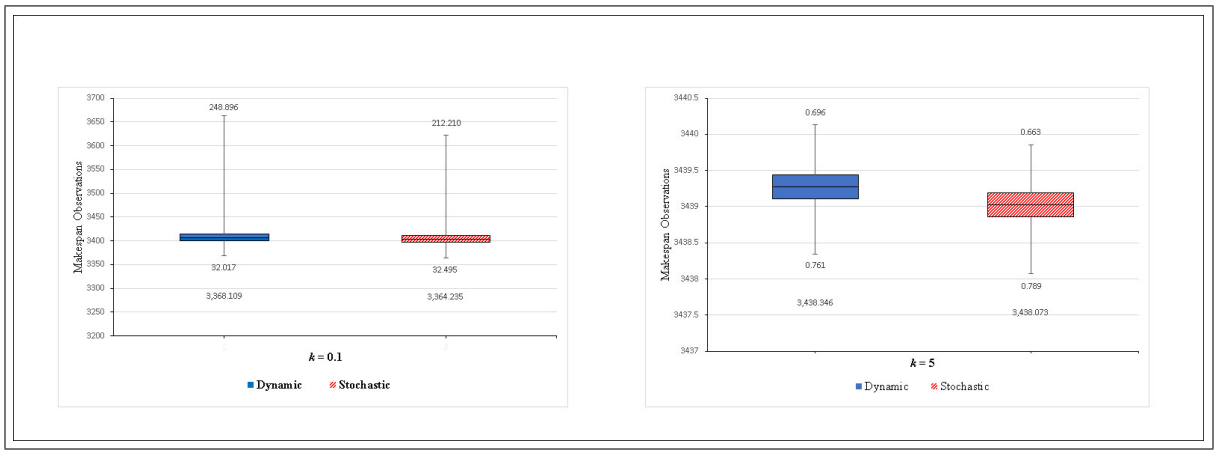


Fig. 8.3 Using MCS outputs to compare different solutions for problem of size 50×10 with $k = 0.1, 5$ and weight W_6

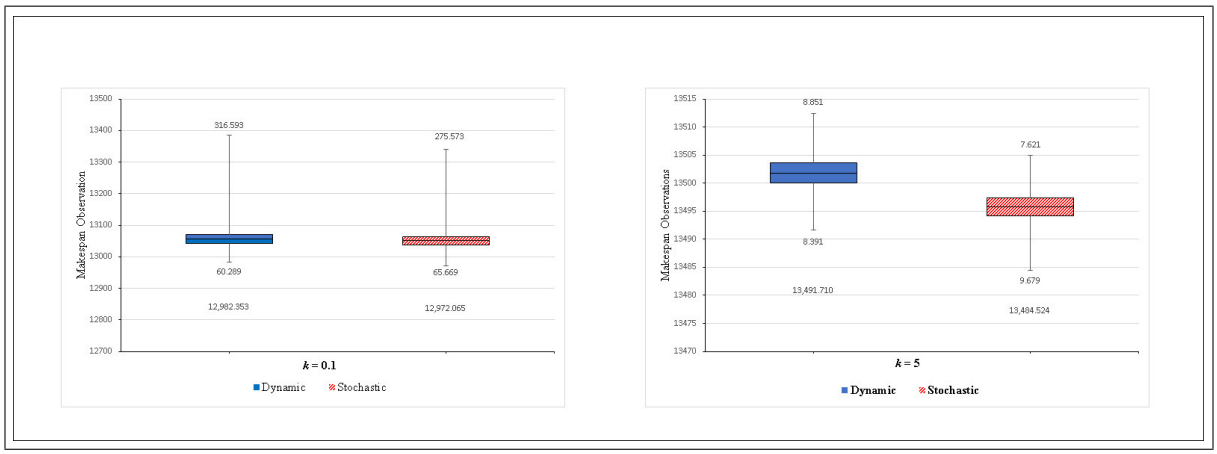


Fig. 8.4 Using MCS outputs to compare different solutions for problem of size 200×20 with $k = 0.1, 5$ and weight W_6

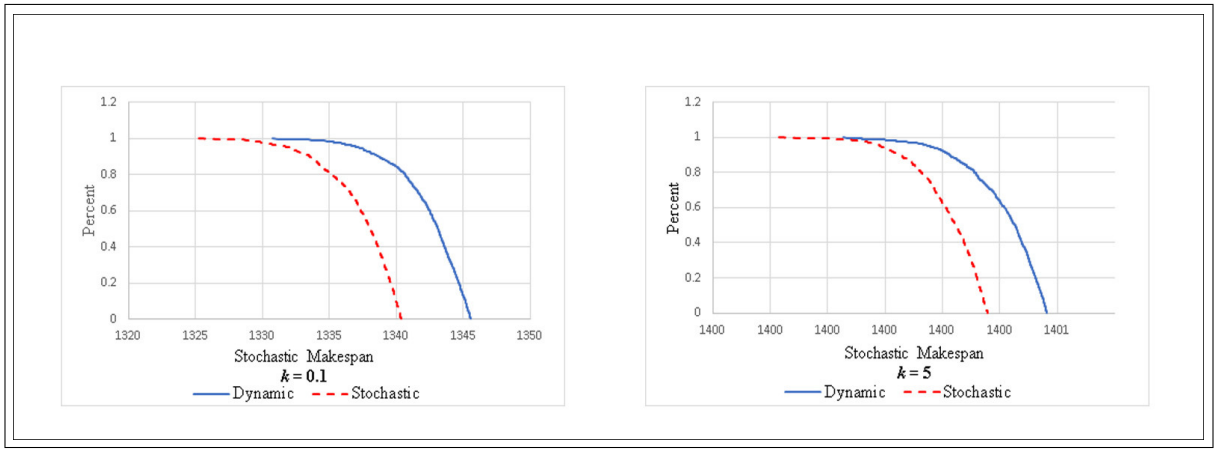


Fig. 8.5 Survival plot with intersecting solutions for problem 20×5 and $k = 0.1, 5$

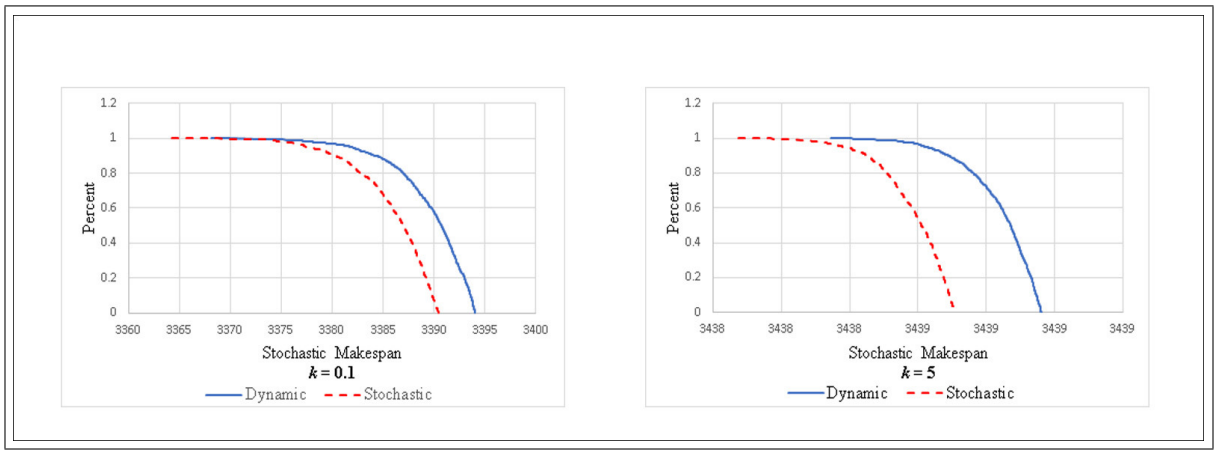


Fig. 8.6 Survival plot with intersecting solutions for problem 50×10 and $k = 0.1, 5$

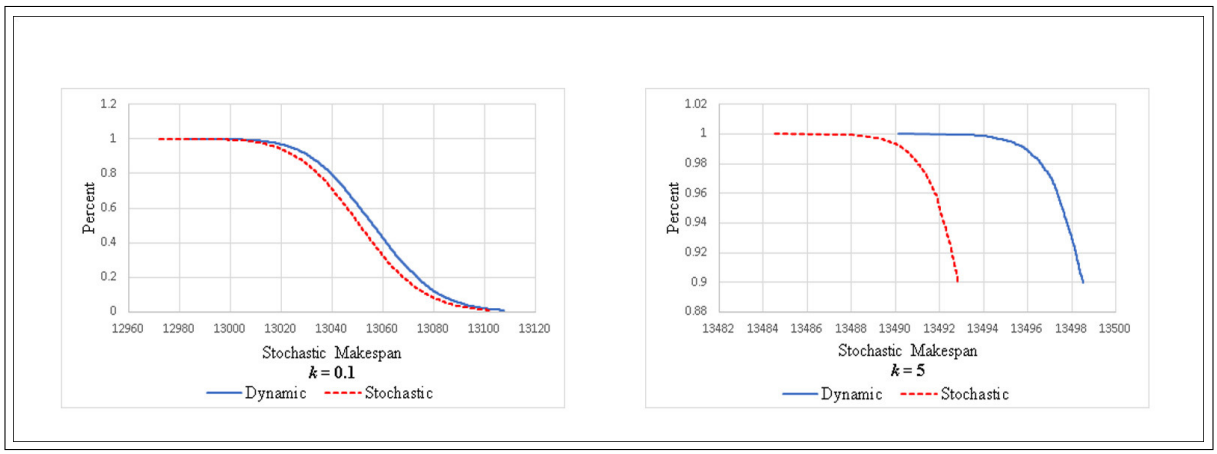


Fig. 8.7 Survival plot with intersecting solutions for problem 200×20 and $k = 0.1, 5$

8.3.1 Comparison between Sim-POS and Sim-BRG

The PSO algorithm is a stochastic evolutionary algorithm, which has the ability of dealing with stochastic COPs successfully. For this, we hybridised the MCS with the PSO algorithm and apply it to the SPFSP under different disruptions in the previous chapter (see chapter 7). On the other hand, the BRIG is an efficient and fast algorithm, and hence, we applied the Sim-BRIG for the SPFSP under different disruptions too. Tables 7.1-7.3, 8.1-8.3 reported the best RPD values over 5 replications found in their experiments for instances ranged from 20×5 to 500×20 ($jobs \times machines$). Also, figures 7.3-7.8, 8.3-8.7 show the boxplots and survival functions for problems of sizes 20×5 to 50×10 and 200×20 and $k = 0.1, 5$. Thus, we can conclude that the proposed Sim-PSO and Sim-BRIG approaches have the ability to handle different real-time events successfully and to generate good expected local optimal solutions. However, for equal running time for both methods (for a time limit $t_{max} = n \times m \times 0.03$ seconds), the

Sim-BRIG approach shows better performance than Sim-PSO, as we can see from Figure 8.8 which shows the RPD values for both approaches for low and high variations where $k = 0.1, 5$ and also with weights $W_6 = (0.002, 0.498, 0.498)$ and $W_8 = (0.166, 0.166, 0.666)$. This figure indicates that the RPDs obtained from the Sim-BRIG approach are always lower than the RPDs obtained from the Sim-PSO approach with different weights and different levels of k (low to high). This emphasises that the Sim-BRIG outperforms the Sim-PSO in reaching better quality solutions even for the stochastic case. Finally, from a statistical view the Sim-BRIG is significantly different with the Sim-PSO in low and high variations $k = 0.1, 5$ and weights $W_6 = (0.002, 0.498, 0.498)$ and $W_8 = (0.166, 0.166, 0.666)$ as shown in figure 8.9.

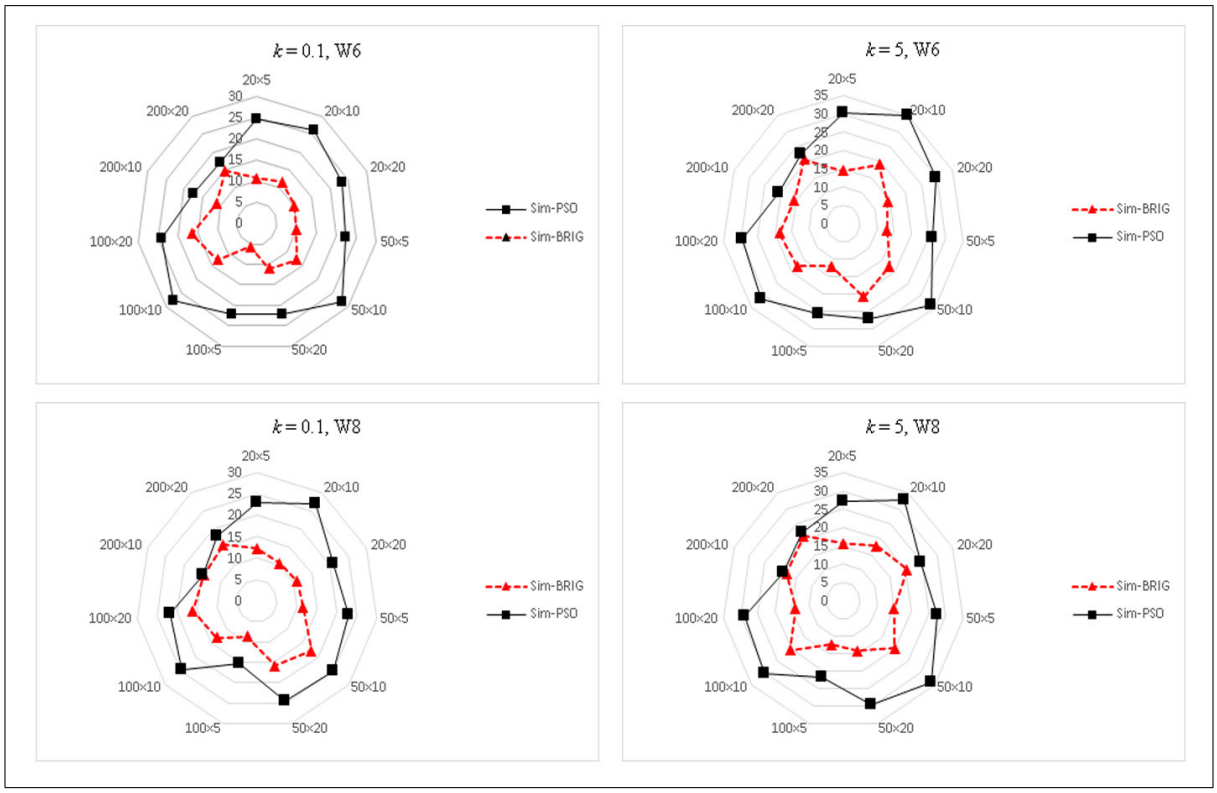


Fig. 8.8 RPD values for Sim-PSO and Sim-BRIG with $k = 0.1, 5$ and W_6, W_8

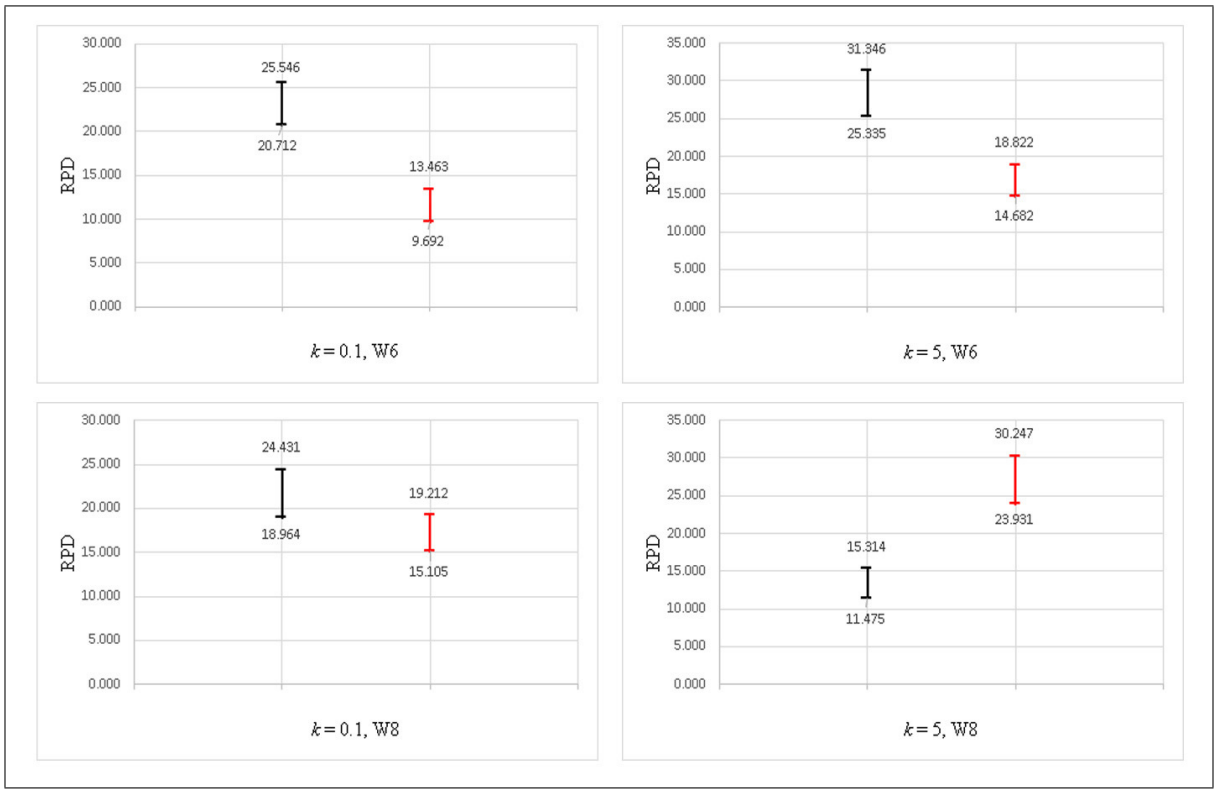


Fig. 8.9 95% Tukey confidence intervals for Sim-PSO and Sim-BRIG with $k = 0.1, 5$ and W_6, W_8

The Sim-BRIG has better performance than the Sim-PSO approach is due to the fact that the Sim-BRIG approach has the following advantages:

1. The Sim-BRIG requires fewer parameters, which makes the approach running faster, and hence reaching good quality solution in less computational time.
2. It uses the BRNEH heuristic to generate an initial solution while Sim-PSO starts with generating random solution.
3. The Sim-BRIG implements some efficient techniques implicitly to improve the quality of the obtained solution such as; LS procedure and BR technique.

8.4 Conclusions

The main methodology of Sim-BRIG algorithm can be given as follows: the algorithm starts to consider fixed values of processing times as the expected values of the stochastic processing times, and hence, the initial SPFSP problem is transformed into a dynamic deterministic PFSP under different real-time events; then apply an efficient predictive-reactive based BRIG

heuristic approach to generate good solutions for the dynamic problem. Finally, employ the MCS to determine the estimated expected value of the stochastic makespan for each of the above-mentioned solutions. By combining MCS with BRIG algorithm, it is possible to solve the problem by using any distribution to model the processing times of jobs, even in the situation where these probability distributions are different from one pair of job-machine to another pair. The performance quality of the Sim-BRIG algorithm is explained over a set of experiments based on adjusted instances from benchmarks for the PFSP under different disruptions with constant processing time values (Katragjini et al., 2013). Another contribution is considering the dynamic scenario for the SPFSP where the problem effected by different real-time events, which are machine breakdown and new job arrivals. This problem is very complex even for the dynamic case, so the dynamicity and stochasticity raise the complexity of this problem. It is possible to use approaches from survival analysis so as to have a better comparison with alternative solutions. Particularly, this has more attention when real-life censored observations related with the introduced jobs permutation, which are employed to compare different solutions. This scenario could happen, for example, when the random values of makespan related to the proposed solution have a high variability. Consequently, part of realisations of the given solution may perform in the values of makespan to override the period of the experiment, so providing incomplete or censored observations. As a result of this chapter, the integration between simulation and BRIG explains some of the advantages that can be gained at solving SPFSP with different real-time events. From the computational experiment of this chapter, the Sim-BRIG have been applied successfully for this problem. Different weights produced different relative percentages values, where the weight $W_6 = (0.002, 0.498, 0.498)$ showed the lowest RPDs in general. For this, the dynamic and stochastic solutions related to this weight are used in the survival analysis. The boxplots illustrated that the stochastic solutions have in general lower means and quartiles, which showed the higher quality of the obtained stochastic solutions. Also, the survival plots emphasised the higher performance of the stochastic solutions when compared with the dynamic one. On the other hand, the comparative study showed that the Sim-BRIG outperform the Sim-PSO approach in getting better dynamic and stochastic solution during the same computational time. This proves the fact that the hybrid MCS procedure and BRIG heuristic is much simple, fast and more accurate comparing to the hybrid MCS and PSO metaheuristic algorithm.

Chapter 9

Conclusion and future research

9.1 Conclusion

This thesis examined the PFSP under dynamic and stochastic environments, where the problem was effected by different uncertainties including; machine breakdowns, new jobs arrivals and/or stochastic processing times. This problem is very important and a present-day problem, impacting costs and productivity of manufacturing systems. The PFSP is an intensive study in which there is no uncertainty during the time horizon where the processing times of each job is known in advance and there are no disruptions interrupting the system. However, the research direction of both of the dynamic and stochastic PFSP under different disruptions are raised. This makes the problem more suitable for certain real-life cases in which there are different uncertainties such as; machine breakdown, new job arrivals and when the jobs processing times are not known in advance but can only be modelled by random variables. Such uncertainties could change the schedule performance, and hence, it is important to develop an optimisation model that consider different objectives to maintain the solution stability and robustness by minimising the measures of utility, instability and robustness, simultaneously. Also, it is significant to propose a rescheduling approach and efficient solution methods that are able to solve such a complicated problem even for large size instances. For this, our main contributions to scientific understanding dealt with several approaches for solving the dynamic and stochastic PFSP under different uncertainties. These approaches focus on five main axes: multi-objective optimisation model, predictive-reactive approach, metaheuristic, heuristic and BR heuristic. Beginning from Chapter 1 where a background is provided following by extensive literature review in chapter 2, which focused on describing the evolution of the main contribution from previous works. Continuing in chapter 3, a multi-objective optimisation model that consider three important measures were developed, this model shows better results when compared with two other models, namely; the bi-objective model that consider only utility and stability

measures, and the classical makespan model. The primary contribution of chapters 4, 5 and 6 are concentrated on the dynamic PFSP under different real-time events, where the framework of the solution methods have used the predictive-reactive based on three different algorithms, including; PSO, IG and BRIG along with the MSR model. The predictive-reactive approach was used to accommodate the machine breakdowns and new jobs arrivals in the partial subsequence of jobs after the time of disruption, where one of the aforementioned algorithms have been applied at the reactive stage. The PSO is a population evolutionary method that has the ability to deal with dynamic and stochastic problems. It starts with generating a population randomly, then it uses position and velocity to seek the best solution in the space. On the other hand, the IG algorithm and its randomised version are simple, efficient and consume a relatively small computational time. They produced a sequence of solutions by iterating over GC heuristics using two main phases iteratively: named destruction and construction. The main improvement in the BRIG algorithm is that it uses the BR technique which provide the ability to consider the jobs with higher probabilities to be selected in the construction phase. The solution of IG and BRIG methods are based on an initial solution that is generated by the well-known NEH and BRNEH heuristics, respectively. At the end of chapters 5 and 6, we demonstrated the effectiveness of our model when compared to the bi-objective model and the classical makespan model, where the MSR model showed the best performance among the other models. Also, we tested the efficiency of our approaches by showing the computational results and comparing with the results of the previous chapters. In this comparison, the experiments showed that the obtained solutions were compared between different efficient algorithms and hence the computational results obviously showed that the BRIG provides a good solution quality for almost all instances when compared with the IG and PSO algorithms. In addition, the computational time consumed by BRIG and IG algorithms were much less than the time of running the PSO algorithm. The main reasons of the high performance and speed are due to the nature of the BRIG where it has less parameters and apply LS improvement implicitly, which provide the algorithm with the ability of exploring more local areas in the solution space. More contributions of this thesis is to consider the SPFSP under different real-time events, where the hybridisation of predictive-reactive based PSO and BRIG algorithms with the MCS procedure along with the MSR model are discussed and examined. In these methods, the hybridised approaches provide near-optimal expected solutions and demonstrate their efficiency by obtaining high-quality solutions for the problem. In chapter 7, we adapted the Sim-PSO to solve the SPFSP under different real-time events, this is a novel approach in the PFSP area. Also, the Sim-BRIG approach to solve the stochastic case is given in chapter 8. There are two stages in order to solve SPFSP under different real-time events; in the first stage, the predictive-reactive approach based PSO (BRIG) algorithm is used along with the MSR model

in order to find a good quality solution for the dynamic PFSP under machine breakdowns and new jobs arrivals. The next stage applied the MCS to estimate good expected solutions. Both stages have been used on the well-known benchmark problem of Katragjini et al. (2013) where the processing times are consider as expected values for random variables that follow the Log Normal distribution, hence, this benchmark is designed for the SPFSP under different real-time events. Indeed, in all chapters, we designed experiments to measure performance with respect to the instances chosen from the literature proposed by Katragjini et al. (2013). As a summery, the results have shown that the Sim-BRIG outperform the Sim-PSO method in obtaining efficient solution in all 120 instances. In all chapters we used thirteen different weights derived from the weight sensitivity algorithm proposed by Jones (2011). Each weight representing different levels of relative importance of the objectives in the MSR model. There are a particular weights which are $W_8 = (0.166, 0.166, 0.666)$ and $W_6 = (0.002, 0.498, 0.498)$ that have showed better solutions where compared with other weights. The weights W_8 and W_6 show the importance of stability and robustness measures where giving higher priority to these measures produced better quality solution.

9.2 Extensions and future work

Whilst this thesis does show very interesting ideas of solving dynamic and stochastic PFSP under different real-time events and the computational results produced good quality solutions. It is clear from the computational effort that there are opportunities to implement different algorithm solutions in future work. More extensions and future works can be summarised as follows:

- In this thesis, we have shown how the model and the approaches can successfully be applied for dynamic and stochastic PFSP under different real-time events. One interesting future research is testing different models including stochastic models and taking into account different objectives with these approaches in order to improve and compare the solutions of this problem.

- Another research line is to expand the weight sensitivity analysis to include other goal programming variants and/or multiple criteria methods. Also considering different sensitivity analysis approaches for the proposed models and approaches.

- More research is suggested about how the multi-objective measures perform over a diverse set of weights could point towards a multi-objective performance measure.

- One direction for future research could be to use these proposed algorithms to develop efficient algorithms to solve similar kinds of problems with different/special characteristics. In addition, there is the opportunity to implement the model with the proposed algorithms to improve the solution quality.
- One potential area of interesting future research is to consider other biased (non-symmetric) probabilistic distributions to measure performance and its impact on results, while we proposed algorithms that are based on a BR selection of elements inside of heuristics.
- Another possibility of future work for improving the quality of the solutions achieved by the Sim-PSO and Sim-BRIG, is to apply efficient approaches to generate good quality initial solution, which could lead to an improvement in the solution from the beginning. Also, LS can be applied to improve the PSO algorithm.
- Also, considering different scheduling problems under different environments such as dynamic and stochastic FFSP and JSPs under different uncertainties. Also, applying different efficient approaches and solution methods. As well as, generating new benchmarks for such problems to enrich the scheduling area.
- Finally, the current proposed model and approaches can be explored to handle other types of uncertainties.

References

Abdollahpour, S., & Rezaeian, J. (2015). Minimizing makespan for flow shop scheduling problem with intermediate buffers by using hybrid approach of artificial immune system. *Applied Soft Computing*, *28*, 44 – 56. URL: http://www.sciencedirect.com/science/article/pii/S1568494614005845. doi:https://doi.org/10.1016/j.asoc.2014.11.022.

Adressi, A., Hassanpour, S. T., & Azizi, V. (2016). Solving group scheduling problem in no-wait flexible flowshop with random machinebreakdown, . *5*, 157–168. doi:10.5267/j.dsl.2015.7.001.

Akhshabi, M., Tavakkoli-Moghaddam, R., & Rahnamay-Roodposhti, F. (2014). A hybrid particle swarm optimization algorithm for a no-wait flow shop scheduling problem with the total flow time. *The International Journal of Advanced Manufacturing Technology*, *70*, 1181–1188. URL: https://doi.org/10.1007/s00170-013-5351-9. doi:10.1007/s00170-013-5351-9.

Al-Hinai, N., & Elmekkawy, T. Y. (2011). Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *International Journal of Production Economics*, *132*, 279 – 291. URL: http://www.sciencedirect.com/science/article/pii/S0925527311001952. doi:https://doi.org/10.1016/j.ijpe.2011.04.020.

Alcaide, D., Rodriguez-Gonzalez, A., & Sicilia, J. (2002). An approach to solve the minimum expected makespan flow-shop problem subject to breakdowns. *European Journal of Operational Research*, *140*, 384 – 398. URL: http://www.sciencedirect.com/science/article/pii/S0377221702000772. doi:https://doi.org/10.1016/S0377-2217(02)00077-2.

Ali, A., & John, M. (1998). Dual criteria scheduling on a two-machine flow-shop subject to random breakdowns. *International Transactions in Operational Research*, *5*, 317–324. doi:10.1016/S0969-6016(97)00042-7.

Allaoui, H., & Artiba, A. (2004). Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. *Computers Industrial Engineering*, *47*, 431 – 450. URL: http://www.sciencedirect.com/science/article/pii/S0360835204001299. doi:https://doi.org/10.1016/j.cie.2004.09.002.

Almeder, C., & Hartl, R. F. (2013). A metaheuristic optimization approach for a real-world stochastic flexible flow shop problem with limited buffer. *International Journal of Production Economics*, *145*, 88 – 95. URL: http://www.sciencedirect.com/science/article/pii/S0925527312004100. doi:https://doi.org/10.1016/j.ijpe.2012.09.014.

Amaran, S., Sahinidis, N. V., Sharda, B., & Bury, S. J. (2017). Simulation optimization: A review of algorithms and applications. *ArXiv e-prints*, . arXiv:1706.08591.

Amirian, H., & Sahraeian, R. (2016). A hybrid differential evolution for general multi-objective flow shop problem with a modified learning effect. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *230*, 2275–2285. URL: https://doi.org/10.1177/0954405416673094. doi:10.1177/0954405416673094. arXiv:https://doi.org/10.1177/0954405416673094.

Andradóttir, S. (1998). A review of simulation optimization techniques. In *Proceedings of the 1998 Mnrer Siinularion Conference* (pp. 151–158).

Aydilek, A., Aydilek, H., & Allahverdi, A. (2015). Production in a two-machine flowshop scheduling environment with uncertain processing and setup times to minimize makespan. *International Journal of Production Research*, *53*, 2803–2819. URL: http://dx.doi.org/10.1080/00207543.2014.997403. doi:10.1080/00207543.2014.997403. arXiv:http://dx.doi.org/10.1080/00207543.2014.997403.

Aytug, H., Lawley, M. A., McKay, K., Mohan, S., & Uzsoy, R. (2005). Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, *161*, 86 – 110. URL: http://www.sciencedirect.com/science/article/pii/S0377221703005307. doi:https://doi.org/10.1016/j.ejor.2003.08.027. IEPM: Focus on Scheduling.

Baker, K. R. (2014). Minimizing earliness and tardiness costs in stochastic scheduling. *European Journal of Operational Research*, *236*, 445 – 452. URL: http://www.sciencedirect.com/science/article/pii/S0377221713009867. doi:https://doi.org/10.1016/j.ejor.2013.12.011.

Baker, K. R., & Altheimer, D. (2012). Heuristic solution methods for the stochastic flow shop problem. *European Journal of Operational Research*, *216*, 172 – 177. URL: http://www.sciencedirect.com/science/article/pii/S0377221711006230. doi:https://doi.org/10.1016/j.ejor.2011.07.021.

Baker, K. R., & Trietsch, D. (2011). Three heuristic procedures for the stochastic, two-machine flow shop problem. *Journal of Scheduling*, *14*, 445–454. URL: https://doi.org/10.1007/s10951-010-0219-4. doi:10.1007/s10951-010-0219-4.

Bargaoui, H., & Driss, O. B. (2014). Multi-agent model based on tabu search for the permutation flow shop scheduling problem. In S. Omatu, H. Bersini, J. M. Corchado, S. Rodríguez, P. Pawlewski, & E. Bucciarelli (Eds.), *Distributed Computing and Artificial Intelligence, 11th International Conference* (pp. 519–527). Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-319-07593-8_60. doi:10.1007/978-3-319-07593-8_60.

Behnamian, J. (2014). Particle swarm optimization-based algorithm for fuzzy parallel machine scheduling. *The International Journal of Advanced Manufacturing Technology*, *75*, 883–895. URL: https://doi.org/10.1007/s00170-014-6181-0. doi:10.1007/s00170-014-6181-0.

Belloso, J., Juan, A. A., Martinez, E., & Faulin, J. (2017). A biased-randomized metaheuristic for the vehicle routing problem with clustered and mixed backhauls. *Networks*, *69*, 241–255. URL: http://dx.doi.org/10.1002/net.21734. doi:10.1002/net.21734.

Bertsekas, D. P., & Castanon, D. A. (1999). Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, *5*, 89–108. URL: https://doi.org/10.1023/A:1009634810396. doi:10.1023/A:1009634810396.

Bessedik, M., Benbouzid-Si Tayeb, F., Cheurfi, H., & Blizak, A. (2016). An immunity-based hybrid genetic algorithms for permutation flowshop scheduling problems. *The International Journal of Advanced Manufacturing Technology*, *85*, 2459–2469. URL: https://doi.org/10.1007/s00170-015-8052-8. doi:10.1007/s00170-015-8052-8.

Blackwell, T. (2007). Particle swarm optimization in dynamic environments. In S. Yang, Y.-S. Ong, & Y. Jin (Eds.), *Evolutionary Computation in Dynamic and Uncertain Environments* (pp. 29–49). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-540-49774-5_2. doi:10.1007/978-3-540-49774-5_2.

Blum, C., & Merkle, D. (2008). *Swarm Intelligence Introduction and Applications* volume 1. doi:10.1017/CBO9781107415324.004.

Bonney, M. C., & Gundry, S. W. (1976). Solutions to the constrained flowshop sequencing problem. *Journal of the Operational Research Society*, *27*, 869–883. URL: https://doi.org/10.1057/jors.1976.176. doi:10.1057/jors.1976.176.

Bożejko, W., & Wodecki, M. (2004). Parallel genetic algorithm for minimizing total weighted completion time. In L. Rutkowski, J. H. Siekmann, R. Tadeusiewicz, & L. A. Zadeh (Eds.), *Artificial Intelligence and Soft Computing - ICAISC 2004: 7th International Conference, Zakopane, Poland, June 7-11, 2004. Proceedings* (pp. 400–405). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-540-24844-6_58. doi:10.1007/978-3-540-24844-6_58.

Cáceres-Cruz, J., Juan, A. A., Bektas, T., Grasman, S. E., & Faulin, J. (2012). Combining monte carlo simulation with heuristics for solving the inventory routing problem with stochastic demands. In *Proceedings of the Winter Simulation Conference* WSC '12 (pp. 274:1–274:9). Winter Simulation Conference. URL: http://dl.acm.org/citation.cfm?id=2429759.2430129.

Cai, X. Q., Wu, X., & Zhou, X. (2014). *Optimal Stochastic Scheduling*. International Series in Operations Research & Management Science. Springer US. URL: https://books.google.co.uk/books?id=RLouBAAAQBAJ.

Campbell, H. G., Dudek, R. A., & Smith, M. L. (1970). A Heuristic Algorithm for The *n* Job, *m* Machine Sequencing Problem. *Management science*, *16*, B630– B637. doi:10.1287/mnsc.16.10.B630.

Campos, S. C., & Arroyo, J. E. C. (2014). Nsga-ii with iterated greedy for a bi-objective three-stage assembly flowshop scheduling problem. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation* GECCO '14 (pp. 429–436). New York, NY, USA: ACM. URL: http://doi.acm.org/10.1145/2576768.2598324. doi:10.1145/2576768.2598324.

Carlier, J., & Rebaï, I. (1996). Two branch and bound algorithms for the permutation flow shop problem. *European Journal of Operational Research*, *90*, 238 – 251. URL: http://www.sciencedirect.com/science/article/pii/0377221795003525. doi:https://doi.org/10.1016/0377-2217(95)00352-5.

Chang, P.-c., Hsieh, J.-c., & Lin, S.-g. (2002). The development of gradual-priority weighting approach for the multi-objective flow-shop scheduling problem. *Int. J. Production Economics 79*, *79*, 171–183.

Chen, C.-L., Huang, S.-Y., Tzeng, Y.-R., & Chen, C.-L. (2014). A revised discrete particle swarm optimization algorithm for permutation flow-shop scheduling problem. *Soft Computing*, *18*, 2271–2282. URL: https://doi.org/10.1007/s00500-013-1199-z. doi:10.1007/s00500-013-1199-z.

Chen, C.-L., Vempati, V. S., & Aljaber, N. (1995). An application of genetic algorithms for flow shop problems. *European Journal of Operational Research*, *80*, 389–396.

Chryssolouris, G. (2006). *Manufacturing Systems: Theory and Practice*. (2nd ed.). Springer Science-i-Business Media, Inc. doi:10.1007/b22134. arXiv:arXiv:1011.1669v3.

Ciavotta, M., Minella, G., & Ruiz, R. (2013). Multi-objective sequence dependent setup times permutation flowshop: A new algorithm and a comprehensive study. *European Journal of Operational Research*, *227*, 301 – 313. URL: http://www.sciencedirect.com/science/article/pii/S0377221713000052. doi:https://doi.org/10.1016/j.ejor.2012.12.031.

Coello, C. A. C., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *8*, 256–279. doi:10.1109/TEVC.2004.826067.

Collet, P., & Rennard, J.-P. (2007). Stochastic Optimization Algorithms. *Handbook of Research on Nature-Inspired Computing for Economics and Management*, *1*, 28–44. URL: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-59140-984-7.ch003. doi:10.4018/978-1-59140-984-7.ch003.

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., & Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, *53*, 512–522. URL: https://doi.org/10.1057/palgrave.jors.2601319. doi:10.1057/palgrave.jors.2601319.

Cowling, P. I., Ouelhadj, D., & Petrovic, S. (2003). A multi-agent architecture for dynamic scheduling of steel hot rolling. *Journal of Intelligent Manufacturing*, *14*, 457–470. URL: https://doi.org/10.1023/A:1025701325275. doi:10.1023/A:1025701325275.

Cowling, P. I., Ouelhadj, D., & Petrovic, S. (2004). Dynamic scheduling of steel casting and milling using multi-agents. *Production Planning & Control*, *15*, 178–188. URL: http://dx.doi.org/10.1080/09537280410001662466. doi:10.1080/09537280410001662466. arXiv:http://dx.doi.org/10.1080/09537280410001662466.

Cui, Z., & Gu, X. (2014). A Discrete Group Search Optimizer for Hybrid Flowshop Scheduling Problem with Random Breakdown. *Mathematical Problems in Engineering*, *2014*. doi:http://dx.doi.org/10.1155/2014/621393.

Cunningham, A. A., & Dutta, S. K. (1973). Scheduling jobs, with exponentially distributed processing times, on two machines of a flow shop. *Naval Research Logistics Quarterly*, *20*, 69–81. URL: http://dx.doi.org/10.1002/nav.3800200107. doi:10.1002/nav.3800200107.

Dahal, K. P., Galloway, S. J., Burt, G. M., McDonald, J. R., & Hopkins, I. (2005). A case study of process facility optimization using discrete event simulation and genetic algorithm. URL: http://portal.acm.org/citation.cfm?doid=1068009.1068372. doi:10.1145/1068009.1068372.

Dahal, K. P., Tan, K. C., & Cowling, P. I. (2007). *Evolutionary Scheduling*. Springer-Verlag Berlin Heidelberg.

Damodaran, P., Rao, A. G., & Mestry, S. (2013). Particle swarm optimization for scheduling batch processing machines in a permutation flowshop. *The International Journal of Advanced Manufacturing Technology*, *64*, 989–1000. URL: https://doi.org/10.1007/s00170-012-4037-z. doi:10.1007/s00170-012-4037-z.

Danilovic, M., & Ilic, O. (2016). A generalized constructive algorithm using insertion-based heuristics. *Computers Operations Research*, *66*, 29 – 43. URL: http://www.sciencedirect.com/science/article/pii/S0305054815001768. doi:https://doi.org/10.1016/j.cor.2015.07.009.

Dannenbring, D. G. (1977). An Evaluation of Flow Shop Sequencing Heuristics. *Management Science*, *23*, 1174–1182.

Dauzére-Pérés, S., Castagliola, P., & Lahlou, C. (2010). Service level in scheduling. In *Flexibility and Robustness in Scheduling* (pp. 99–121). ISTE. URL: http://dx.doi.org/10.1002/9780470611432.ch5. doi:10.1002/9780470611432.ch5.

Deng, G., & Gu, X. (2014). An iterated greedy algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times. *International Journal of Systems Science*, *45*, 351–362. URL: http://dx.doi.org/10.1080/00207721.2012.723054. doi:10.1080/00207721.2012.723054. arXiv:http://dx.doi.org/10.1080/00207721.2012.723054.

Deng, J., & Wang, L. (2017). A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem. *Swarm and Evolutionary Computation*, *32*, 121 – 131. URL: http://www.sciencedirect.com/science/article/pii/S2210650216300281. doi:https://doi.org/10.1016/j.swevo.2016.06.002.

Ding, J.-Y., Song, S., N.D. Gupta, J., Zhang, R., Chiong, R., & Wu, C. (2015). An improved iterated greedy algorithm with a tabu-based reconstruction strategy for the no-wait flowshop scheduling problem. *Applied Soft Computing*, *30*, 604 – 613. URL: http://www.sciencedirect.com/science/article/pii/S1568494615000964. doi:https://doi.org/10.1016/j.asoc.2015.02.006.

Ding, J.-Y., Song, S., & Wu, C. (2016). Carbon-efficient scheduling of flow shops by multi-objective optimization. *European Journal of Operational Research*, *248*, 758 – 771. URL: http://www.sciencedirect.com/science/article/pii/S0377221715004099. doi:https://doi.org/10.1016/j.ejor.2015.05.019.

Dodin, B. (1996). Determining the optimal sequences and the distributional properties of their completion times in stochastic flow shops. *Computers Operations Research*, *23*, 829 – 843. URL: http://www.sciencedirect.com/science/article/pii/0305054895000836. doi:https://doi.org/10.1016/0305-0548(95)00083-6.

Dong, X., Huang, H., & Chen, P. (2008). An improved neh-based heuristic for the permutation flowshop problem. *Computers Operations Research*, *35*, 3962 – 3968. URL: http://www.sciencedirect.com/science/article/pii/S0305054807001116. doi:https://doi.org/10.1016/j.cor.2007.05.005. Part Special Issue: Telecommunications Network Engineering.

Dong, X., Nowak, M., Chen, P., & Lin, Y. (2015). Self-adaptive perturbation and multi-neighborhood search for iterated local search on the permutation flow shop problem. *Computers Industrial Engineering*, *87*, 176 – 185. URL: http://www.sciencedirect.com/science/article/pii/S0360835215002089. doi:https://doi.org/10.1016/j.cie.2015.04.030.

Dongdong, L., Kai, L., Zhengping, Z., Bo, H., & Yan, Z. (2015). Discrete Particle Swarm Optimization Algorithm in FlexibleHybrid Flow Shop Scheduling. *International Journal of Hybrid Information Technology*, *8*, 299–310. doi:http://dx.doi.org/10.14257/ijhit.2015.8.10.27.

Dubois-Lacoste, J., Pagnozzi, F., & Stützle, T. (2017). An iterated greedy algorithm with optimization of partial solutions for the makespan permutation flowshop problem. *Computers Operations Research*, *81*, 160 – 166. URL: http://www.sciencedirect.com/science/article/pii/S030505481630329X. doi:https://doi.org/10.1016/j.cor.2016.12.021.

Dudek, R. A., & Teuton, O. F. (1964). Development of *M*-Stage Decision Rule for Scheduling *n* Jobs through *m* Machines. *Operations Research*, *12*, 471–497.

Dutton, J. (1964). Production scheduling: a behaviour model. *International Journal of Production Research*, (pp. 3–27). URL: http://search.ebscohost.com/login.aspx?direct=true{&}db=bth{&}AN=5553140{&}site=ehost-live.

Dutton, J. M. (1962). Simulation of an Actual Production Scheduling and Work Flow Control System. *International Journal of Production Research*, *1*, 421–441. URL: http://search.ebscohost.com/login.aspx?direct=true{&}db=bth{&}AN=5553140{&}site=ehost-live. doi:10.1080/00207546108943095.

Ebrahimi, M., Ghomi, S. M. T. F., & Karimi, B. (2014). Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates. *Applied Mathematical Modelling*, *38*, 2490 – 2504. URL: http://www.sciencedirect.com/science/article/pii/S0307904X13007282. doi:https://doi.org/10.1016/j.apm.2013.10.061.

El-bouri, A. (2013). The effect of inserted idle time on the performance of dispatching rules in a flowshop. In *International Conference on Industrial Engineering and Operations Management*.

Elias C. Arroyo, J., Joséand Y.-T. Leung (2017). An effective iterated greedy algorithm for scheduling unrelated parallel batch machines with non-identical capacities and unequal ready times. *Computers Industrial Engineering*, *105*, 84 – 100. URL: http://www.sciencedirect.com/science/article/pii/S0360835216305162. doi:https://doi.org/10.1016/j.cie.2016.12.038.

Elyasi, A., & Salmasi, N. (2013). Stochastic scheduling with minimizing the number of tardy jobs using chance constrained programming. *Mathematical and Computer Modelling*, *57*, 1154 – 1164. URL: http://www.sciencedirect.com/science/article/pii/S0895717712002786. doi:https://doi.org/10.1016/j.mcm.2012.10.017.

Emmons, H., & Vairaktarakis, G. (2012). *Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications*. International Series in Operations Research & Management Science. Springer US. URL: https://books.google.co.uk/books?id=4UWMlwrescgC.

Entezari, S., & Gholami, S. (2015). Multi-objective flexible flow shop scheduling with unexpected arrivals of new jobs, . *3*, 172–181.

Fazayeli, M., Aleagha, M.-R., Bashirzadeh, R., & Shafaei, R. (2016). A hybrid meta-heuristic algorithm for flowshop robust scheduling under machine breakdown uncertainty. *International Journal of Computer Integrated Manufacturing*, *29*, 709–719. URL: http://dx.doi.org/10.1080/0951192X.2015.1067907. doi:10.1080/0951192X.2015.1067907. arXiv:http://dx.doi.org/10.1080/0951192X.2015.1067907.

Framinan, J., Leisten, R., & García, R. R. (2014). *Manufacturing Scheduling Systems: An Integrated View on Models, Methods and Tools*. SpringerLink : Bücher. Springer London. URL: https://books.google.co.uk/books?id=9DO3BAAAQBAJ.

Framinan, J. M., Gupta, J. N. D., & Leisten, R. (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society*, *55*, 1243–1255. URL: https://doi.org/10.1057/palgrave.jors.2601784. doi:10.1057/palgrave.jors.2601784.

Framinan, J. M., & Leisten, R. (2008). A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria. *OR Spectrum*, *30*, 787–804. URL: https://doi.org/10.1007/s00291-007-0098-z. doi:10.1007/s00291-007-0098-z.

Framinan, J. M., Leisten, R., & Rajendran, C. (2003). Different initial sequences for the heuristic of nawaz, enscore and ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem. *International Journal of Production Research*, *41*, 121–148. URL: https://doi.org/10.1080/00207540210161650. doi:10.1080/00207540210161650.

Framinan, J. M., Leisten, R., & Ruiz-Usano, R. (2002). Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation. *European Journal of Operational Research*, *141*, 559 – 569. URL: http://www.sciencedirect.com/science/article/pii/S0377221701002788. doi:https://doi.org/10.1016/S0377-2217(01)00278-8.

Framinan, J. M., & Perez-Gonzalez, P. (2015). On heuristic solutions for the stochastic flowshop scheduling problem. *European Journal of Operational Research*, *246*, 413 – 420. URL: http://www.sciencedirect.com/science/article/pii/S0377221715003781. doi:https://doi.org/10.1016/j.ejor.2015.05.006.

Frazzon, E. M., Albrecht, A., & Andrea, H. P. (2016). Simulation-based optimization for the integrated scheduling of production and logistic systems. *IFAC-PapersOnLine*, *49*, 1050 – 1055. URL: http://www.sciencedirect.com/science/article/pii/S2405896316308394. doi:https://doi.org/10.1016/j.ifacol.2016.07.581. 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016.

Fu, M. C. (1994). Optimization via simulation A review. *Annals of Operations Research*, *53*, 199–247.

Fu, M. C., Glover, F. W., & April, J. (2005). Simulation optimization: a review, new developments, and applications. In *Proceedings of the Winter Simulation Conference, 2005.* (pp. 13 pp.–). doi:10.1109/WSC.2005.1574242.

Gao, K. Z., Suganthan, P. N., Pan, Q. K., Tasgetiren, M. F., & Sadollah, A. (2016). Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion. *Knowledge-Based Systems*, *109*, 1 – 16. URL: http://www.sciencedirect.com/science/article/pii/S0950705116301794. doi:https://doi.org/10.1016/j.knosys.2016.06.014.

Gaspero, L. D., Schaerf, A., & Cadoli, M. (2003). *Local Search Techniques for Scheduling Problems: Algorithms and Software Tools*. Ph.D. thesis a degli Studi di Udine. URL: http://www.diegm.uniud.it/schaerf/SAS/articoli/PhDThesisLucaDiGaspero.pdf.

Geiger, M. J. (2006). On the distribution of pareto optimal solutions in alternative space – the investigation of multi objective permutation flow shop scheduling problems. *Ukio Technologinis ir Ekonominis Vystymas*, *12*, 23–29. URL: http://www.tandfonline.com/doi/abs/10.1080/13928619.2006.9637718. doi:10.1080/13928619.2006.9637718. arXiv:http://www.tandfonline.com/doi/pdf/10.1080/13928619.2006.9637718.

Geiger, M. J. (2008). Foundations of the pareto iterated local search metaheuristic. In *Proceedings of the 18th International Conference on Multiple Criteria Decision Making, Chania, Greece, June 19-23, 2006*. volume abs/0809.0406. URL: http://arxiv.org/abs/0809.0406.

Gholami, M., Zandieh, M., & Alem-Tabriz, A. (2009). Scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. *The International Journal of Advanced Manufacturing Technology*, *42*, 189–201. URL: https://doi.org/10.1007/s00170-008-1577-3. doi:10.1007/s00170-008-1577-3.

González-Neira, E. M., García-Cáceres, R. G., Pablo, C.-V. J., Molina-Sánchez, L. P., Montoya-Torres, & Montoya-Torres, J. R. (2016). Stochastic flexible flow shop scheduling problem under quantitative and qualitative decision criteria. *Computers Industrial Engineering*, *101*, 128 – 144. URL: http://www.sciencedirect.com/science/article/pii/S0360835216303308. doi:https://doi.org/10.1016/j.cie.2016.08.026.

González-Neira, E. M., Montoya-Torres, J. R., & Barrera, D. (2017). Flow-shop scheduling problem under uncertainties Review and trends. *International Journal of Industrial Engineering Computations*, *8*, 1–28. doi:10.5267/j.ijiec.2017.2.001.

Gourgand, M., Grangeon, N., & Norre, S. (2003). A contribution to the stochastic flow shop scheduling problem. *European Journal of Operational Research*, *151*, 415 – 433. URL: http://www.sciencedirect.com/science/article/pii/S0377221702008354. doi:https://doi.org/10.1016/S0377-2217(02)00835-4. Meta-heuristics in combinatorial optimization.

Gourgand, M., Grangeon, N., & Norre, S. (2010). Metaheuristics and performance evaluation models for the stochastic permutation flow-shop scheduling problem. In *Flexibility and Robustness in Scheduling* (pp. 143–170). ISTE. URL: http://dx.doi.org/10.1002/9780470611432.ch7. doi:10.1002/9780470611432.ch7.

Graham, R., Lawler, E., Lenstra, J., & Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In P. Hammer, E. Johnson, & B. Korte (Eds.), *Discrete Optimization II* (pp. 287 – 326). Elsevier volume 5 of *Annals of Discrete Mathematics*. URL: http://www.sciencedirect.com/science/article/pii/S016750600870356X. doi:https://doi.org/10.1016/S0167-5060(08)70356-X.

Grasas, A., Juan, A. A., & Lourenço, H. R. (2016). Simils: a simulation-based extension of the iterated local search metaheuristic for stochastic combinatorial optimization. *Journal of Simulation*, *10*, 69–77. URL: https://doi.org/10.1057/jos.2014.25. doi:10.1057/jos.2014.25.

Gu, J., Gu, X., & Jiao, B. (2008). A quantum genetic based scheduling algorithm for stochastic flow shop scheduling problem with random breakdown. *IFAC Proceedings Volumes*, *41*, 63 – 68. URL: http://www.sciencedirect.com/science/article/pii/S1474667016389261. doi:https://doi.org/10.3182/20080706-5-KR-1001.00010. 17th IFAC World Congress.

Guo, G., Wu, B., & Yang, S. (2011). A job-insertion heuristic for minimizing the mean flowtime in dynamic flowshops. *Frontiers of Mechanical Engineering*, *6*, 197–202. URL: https://doi.org/10.1007/s11465-011-0211-5. doi:10.1007/s11465-011-0211-5.

Gupta, J. N. D. (1971). A functional heuristic algorithm for the flowshop scheduling problem. *Journal of the Operational Research Society*, *22*, 39–47. URL: https://doi.org/10.1057/jors.1971.18. doi:10.1057/jors.1971.18.

Hariri, A., & Potts, C. (1989). A branch and bound algorithm to minimize the number of late jobs in a permutation flow-shop. *European Journal of Operational Research*, *38*, 228 – 237. URL: http://www.sciencedirect.com/science/article/pii/0377221789901082. doi:https://doi.org/10.1016/0377-2217(89)90108-2.

Hassanzadeh, A., Rasti-Barzoki, M., & Khosroshahi, H. (2016). Two new meta-heuristics for a bi-objective supply chain scheduling problem in flow-shop environment. *Applied Soft Computing*, *49*, 335 – 351. URL: http://www.sciencedirect.com/science/article/pii/S1568494616304136. doi:https://doi.org/10.1016/j.asoc.2016.08.019.

Hejazi, R. S., & Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: a review. *International Journal of Production Research*, *43*, 2895–2929. URL: http://dx.doi.org/10.1080/00207540500056417. doi:10.1080/00207540500056417. arXiv:http://dx.doi.org/10.1080/00207540500056417.

Hentenryck, P. V., & Bent, R. (2006). *Online Stochastic Combinatorial Optimization*. London, England: Massachusetts Institute of Technology.

Ho, J. C., & Chang, Y.-L. (1991). A new heuristic for the *n*-job, *m*-machine flow-shop problem. *European Journal of Operational Research*, *52*, 194 – 202. URL: http://www.sciencedirect.com/science/article/pii/037722179190080F. doi:https://doi.org/10.1016/0377-2217(91)90080-F.

Hundal, T. S., & Rajgopal, J. (1988). An extension of palmer's heuristic for the flow shop scheduling problem. *International Journal of Production Research*, *26*, 1119–1124. URL: http://dx.doi.org/10.1080/00207548808947922. doi:10.1080/00207548808947922. arXiv:http://dx.doi.org/10.1080/00207548808947922.

Ignall, E., & Schrage, L. (1965). Application of the branch and bound technique to some flow-shop scheduling problems. *Oper. Res.*, *13*, 400–412. URL: http://dx.doi.org/10.1287/opre.13.3.400. doi:10.1287/opre.13.3.400.

Imed, K., & Hans, K. (2016). Semi-online scheduling on a single machine with unexpected breakdown. *Theoretical Computer Science*, *646*, 40 – 48. URL: http://www.sciencedirect.com/science/article/pii/S0304397516303371. doi:https://doi.org/10.1016/j.tcs.2016.07.014.

Jackson, S., Wilson, J. R., & MacCarthy, B. L. (2004). A new model of scheduling in manufacturing: Tasks, roles, and monitoring. *Human Factors*, *46*, 533–550. URL: https://doi.org/10.1518/hfes.46.3.533.50393. doi:10.1518/hfes.46.3.533.50393. arXiv:https://doi.org/10.1518/hfes.46.3.533.50393. PMID: 15573550.

Jacobs, L. W., & Brusco, M. J. (1995). Note: A local-search heuristic for large set-covering problems. *Naval Research Logistics (NRL)*, *42*, 1129–1140. URL: http://dx.doi.org/10.1002/1520-6750(199510)42:7<1129::AID-NAV3220420711>3.0.CO;2-M. doi:10.1002/1520-6750(199510)42:7<1129::AID-NAV3220420711>3.0.CO;2-M.

James, E. D., & Michael, P. H. (1970). Review of sequencing research. *Naval Research Logistics*, *17*, 11–39.

Jarboui, B., Siarry, P., & Teghem, J. (2013). *Metaheuristics for Production Scheduling*. London, England: ISTE Ltd and John Wiley & Sons, Inc.

Jensen, M. T. (2003). Generating robust and flexible job shop schedules using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, *7*, 275–288. doi:10.1109/TEVC.2003.810067.

Jeong, S. J., Lim, S. J., & Kim, K. S. (2006). Hybrid approach to production scheduling using genetic algorithm and simulation. *The International Journal of Advanced Manufacturing Technology*, *28*, 129–136. URL: https://doi.org/10.1007/s00170-004-2345-7. doi:10.1007/s00170-004-2345-7.

Jia, Y., Qu, J., & Wang, L. (2016). A novel particle swarm optimization algorithm for permutation flow-shop scheduling problem. In Q. Zu, & B. Hu (Eds.), *Human Centered Computing: Second International Conference, HCC 2016, Colombo, Sri Lanka, January 7-9, 2016, Revised Selected Papers* (pp. 676–682). Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-319-31854-7_62. doi:10.1007/978-3-319-31854-7_62.

Jian, N., & Henderson, S. G. (2015). An introduction to simulation optimization. In *Proceedings of the 2015 Winter Simulation Conference* WSC '15 (pp. 1780–1794). Piscataway, NJ, USA: IEEE Press. URL: http://dl.acm.org/citation.cfm?id=2888619.2888819.

Jiao, L., & Wang, L. (2000). A novel genetic algorithm based on immunity. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *30*, 552–561. doi:10.1109/3468.867862.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, *1*, 61–68. URL: http://dx.doi.org/10.1002/nav.3800010110. doi:10.1002/nav.3800010110.

Joines, J. A., Gupta, D., Gokce, R. E., M. A. andKing, & Kay, M. G. (2002). Supply chain multi-objective simulation optimization. In *Proceedings of the Winter Simulation Conference* (pp. 1306–1314 vol.2). volume 2. doi:10.1109/WSC.2002.1166395.

Jones, D. (2011). A practical weight sensitivity algorithm for goal and multiple objective programming. *European Journal of Operational Research*, *213*, 238 – 245. URL: http://www.sciencedirect.com/science/article/pii/S0377221711002232. doi:https://doi.org/10.1016/j.ejor.2011.03.012.

Jones, D., & Tamiz, M. (2010). *Practical Goal Programming*. Springer, New York.

Joo, B. J., Choi, Y. C., & Xirouchakis, P. (2013). Dispatching rule-based algorithms for a dynamic flexible flow shop scheduling problem with time-dependent process defect rate and quality feedback. *Procedia CIRP*, *7*, 163 – 168. URL: http://www.sciencedirect.com/science/article/pii/S2212827113002357. doi:https://doi.org/10.1016/j.procir.2013.05.028. Forty Sixth CIRP Conference on Manufacturing Systems 2013.

Juan, A., Faulin, J., Grasman, S., Riera, D., Marull, J., & Mendez, C. (2011). Using safety stocks and simulation to solve the vehicle routing problem with stochastic demands. *Transportation Research Part C: Emerging Technologies*, *19*, 751 – 765. URL: http://www.sciencedirect.com/science/article/pii/S0968090X10001439. doi:https://doi.org/10.1016/j.trc.2010.09.007. Freight Transportation and Logistics (selected papers from ODYSSEUS 2009 - the 4th International Workshop on Freight Transportation and Logistics).

Juan, A. a., Barrios, B. B., Vallada, E., Riera, D., & Jorba, J. (2014a). A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times. *Simulation Modelling Practice and Theory*, *46*, 101 – 117. URL: http://www.sciencedirect.com/science/article/pii/S1569190X1400029X. doi:https://doi.org/10.1016/j.simpat.2014.02.005. Simulation-Optimization of Complex Systems: Methods and Applications.

Juan, A. A., Cáceres-Cruz, J., González-Martín, S., Riera, D., & Barrios, B. B. (2014b). Biased randomization of classical heuristics. In *Encyclopedia of Business Analytics and Optimization* (pp. 304–314). IGI Global.

Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M., & Figueira, G. (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, *2*, 62 – 72. URL: http://www.sciencedirect.com/science/article/pii/S221471601500007X. doi:https://doi.org/10.1016/j.orp.2015.03.001.

Juan, A. A., Lourenço, H. R., Mateo, M., Luo, R., & Castella, Q. (2014c). Using iterated local search for solving the flow-shop problem: Parallelization, parametrization, and randomization issues. *International Transactions in Operational Research*, *21*, 103–126. URL: http://dx.doi.org/10.1111/itor.12028. doi:10.1111/itor.12028.

Kalczynski, P. J., & Kamburowski, J. (2006). A heuristic for minimizing the expected makespan in two-machine flow shops with consistent coefficients of variation. *European Journal of Operational Research*, *169*, 742 – 750. URL: http://www.sciencedirect.com/science/article/pii/S0377221705001402. doi:https://doi.org/10.1016/j.ejor.2004.08.045.

Kalczynski, P. J., & Kamburowski, J. (2008). An improved neh heuristic to minimize makespan in permutation flow shops. *Computers Operations Research*, *35*, 3001 – 3008. URL: http://www.sciencedirect.com/science/article/pii/S0305054807000172. doi:https://doi.org/10.1016/j.cor.2007.01.020. Part Special Issue: Bio-inspired Methods in Combinatorial Optimization.

Kamble, S. V., & Kadam, K. S. (2012). A Particle Swarm Optimization –Based Heuristic for Scheduling in FMS Review. *International Journal on Advanced Computer Theory and Engineering (IJACTE) problem*, (pp. 92–96).

Kamble, S. V., Mane, S. U., & Umbarkar, A. J. (2015). Hybrid Multi-Objective Particle Swarm Optimization for Flexible Job Shop Scheduling Problem. *I.J. Intelligent Systems and Applications*, *4*, 54–61. doi:10.5815/ijisa.2015.04.08.

Kamburowski, J. (2000). On three-machine flow shops with random job processing times. *European Journal of Operational Research*, *125*, 440 – 448. URL: http://www.sciencedirect.com/science/article/pii/S0377221799002222. doi:https://doi.org/10.1016/S0377-2217(99)00222-2.

Kan, A. H. G. R. (1976). *Machine Scheduling Problems Classification, complexity and computations*. Boston, MA: Springer US. URL: http://www.springerlink.com/index/10.1007/978-1-4613-4383-7. doi:10.1007/978-1-4613-4383-7.

Kang, Q., He, H., & Wei, J. (2013). An effective iterated greedy algorithm for reliability-oriented task allocation in distributed computing systems. *Journal of Parallel and Distributed Computing*, *73*, 1106 – 1115. URL: http://www.sciencedirect.com/science/article/pii/S074373151300052X. doi:https://doi.org/10.1016/j.jpdc.2013.03.008.

Kaplan, E., & Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, *53*, 457 – 481. doi:10.2307/2281868.

Kaplan, S., & Rabadi, G. (2012). Exact and heuristic algorithms for the aerial refueling parallel machine scheduling problem with due date-to-deadline window and ready times. *Computers Industrial Engineering*, *62*, 276 – 285. URL: http://www.sciencedirect.com/science/article/pii/S0360835211002841. doi:https://doi.org/10.1016/j.cie.2011.09.015.

Kaplan, S., & Rabadi, G. (2013). Simulated annealing and metaheuristic for randomized priority search algorithms for the aerial refuelling parallel machine scheduling problem with due date-to-deadline windows and release times. *Engineering Optimization*, *45*, 67–87. URL: http://dx.doi.org/10.1080/0305215X.2012.658783. doi:10.1080/0305215X.2012.658783. arXiv:http://dx.doi.org/10.1080/0305215X.2012.658783.

Kaplan, S., & Rabadi, G. (2015). Minimising the total weighted tardiness and instability for the parallel machine re-scheduling problem with deadlines and ready times. *Int. J. Planning and Scheduling*, *2*, 87–109.

Kaplanoğlu, V. (2014). Multi-agent based approach for single machine scheduling with sequence-dependent setup times and machine maintenance. *Applied Soft Computing*, *23*, 165 – 179. URL: http://www.sciencedirect.com/science/article/pii/S1568494614002944. doi:https://doi.org/10.1016/j.asoc.2014.06.020.

Kasap, N., Aytug, H., & Paul, A. (2006). Minimizing makespan on a single machine subject to random breakdowns. *Operations Research Letters*, *34*, 29 – 36. URL: http://www.sciencedirect.com/science/article/pii/S0167637705000313. doi:https://doi.org/10.1016/j.orl.2005.02.002.

Katragjini, K., Vallada, E., & Ruiz, R. (2013). Flow shop rescheduling under different types of disruption. *International Journal of Production Research*, *51*, 780–797. URL: http://dx.doi.org/10.1080/00207543.2012.666856. doi:10.1080/00207543.2012.666856. arXiv:http://dx.doi.org/10.1080/00207543.2012.666856.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on* (pp. 1942–1948 vol.4). volume 4. doi:10.1109/ICNN.1995.488968.

Konak, A., & Kulturel-Konak, S. (2005). Simulation optimization using tabu search: an empirical study. In *Proceedings of the Winter Simulation Conference, 2005.* (pp. 7 pp.–). doi:10.1109/WSC.2005.1574571.

Koole, G. (2000). Stochastic scheduling with event-based dynamic programming. *Mathematical Methods of Operations Research*, *51*, 249–261. URL: https://doi.org/10.1007/s001860050087. doi:10.1007/s001860050087.

Koulamas, C. (1998). A new constructive heuristic for the flowshop scheduling problem. *European Journal of Operational Research*, *105*, 66 – 71. URL: http://www.sciencedirect.com/science/article/pii/S0377221797000271. doi:https://doi.org/10.1016/S0377-2217(97)00027-1.

Kouvelis, P., Daniels, R. L., & Vairaktarakis, G. (2000). Robust scheduling of a two-machine flow shop with uncertain processing times. *IIE Transactions*, *32*, 421–432. URL: https://doi.org/10.1023/A:1007640726040. doi:10.1023/A:1007640726040.

Koyama, A., Barolli, L., Matsumoto, K., & Apduhan, B. O. (2004). A ga-based multi-purpose optimization algorithm for qos routing. In *18th International Conference on Advanced Information Networking and Applications, 2004. AINA 2004.* (pp. 23–28 Vol.1). volume 1. doi:10.1109/AINA.2004.1283882.

Ku, P.-S., & Niu, S.-C. (1986). On johnson's two-machine flow shop with random processing times. *Operations Research*, *34*, 130–136. URL: https://doi.org/10.1287/opre.34.1.130. doi:10.1287/opre.34.1.130. arXiv:https://doi.org/10.1287/opre.34.1.130.

Kuo, I.-H., Horng, S.-J., Kao, T.-W., Lin, T.-L., Lee, C.-L., Terano, T., & Pan, Y. (2009). An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model. *Expert Systems with Applications*, *36*, 7027 – 7032. URL: http://www.sciencedirect.com/science/article/pii/S0957417408006015. doi:https://doi.org/10.1016/j.eswa.2008.08.054.

Lei, D. (2008). Multi-objective production scheduling: a survey. *The International Journal of Advanced Manufacturing Technology*, *43*, 926. URL: https://doi.org/10.1007/s00170-008-1770-4. doi:10.1007/s00170-008-1770-4.

Leisten, R., & Rajendran, C. (2015). Variability of completion time differences in permutation flow shop scheduling. *Computers Operations Research*, *54*, 155 – 167. URL: http://www.sciencedirect.com/science/article/pii/S0305054814002251. doi:https://doi.org/10.1016/j.cor.2014.08.015.

Li, J.-q., Pan, Q.-k., & Mao, K. (2014). Hybrid particle swarm optimization for hybrid flowshop scheduling problem with maintenance activities. *TheScientificWorldJournal*, *2014*, 596850. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=4032694{&}tool=pmcentrez{&}rendertype=abstract. doi:10.1155/2014/596850.

Li, J.-q., Pan, Q.-k., & Mao, K. (2015). A discrete teaching-learning-based optimisation algorithm for realistic flowshop rescheduling problems. *Engineering Applications of Artificial Intelligence*, *37*, 279 – 292. URL: http://www.sciencedirect.com/science/article/pii/S0952197614002358. doi:https://doi.org/10.1016/j.engappai.2014.09.015.

Li, J. Q., Pan, Q. K., & Mao, K. (2016). A hybrid fruit fly optimization algorithm for the realistic hybrid flowshop rescheduling problem in steelmaking systems. *IEEE Transactions on Automation Science and Engineering*, *13*, 932–949.

Li, X., Branke, J., & Blackwell, T. (2006). Particle swarm with speciation and adaptation in a dynamic environment. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* GECCO '06 (pp. 51–58). New York, NY, USA: ACM. URL: http://doi.acm.org/10.1145/1143997.1144005. doi:10.1145/1143997.1144005.

Li, X., & Li, M. (2015). Multiobjective local search algorithm-based decomposition for multiobjective permutation flow shop scheduling problem. *IEEE Transactions on Engineering Management*, *62*, 544–557. doi:10.1109/TEM.2015.2453264.

Li, X., Lin, J., Li, J., & Jin, B. (2012). *Computational Intelligence and Intelligent Systems* volume 575. Springer-Verlag Berlin Heidelberg. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-84957936675{&}partnerID=tZOtx3y1. doi:10.1007/978-981-10-0356-1.

Li, X., & Ma, S. (2016). Multi-objective memetic search algorithm for multi-objective permutation flow shop scheduling problem. *IEEE Access*, *4*, 2154–2165. doi:10.1109/ACCESS.2016.2565622.

Li, X., & Yin, M. (2013). An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. *Advances in Engineering Software*, *55*, 10 – 31. URL: http://www.sciencedirect.com/science/article/pii/S0965997812001305. doi:https://doi.org/10.1016/j.advengsoft.2012.09.003.

Li, Z., Tang, Q., & Zhang, L. P. (2017). Two-sided assembly line balancing problem of type i: Improvements, a simple algorithm and a comprehensive study. *Computers Operations Research*, *79*, 78 – 93. URL: http://www.sciencedirect.com/science/article/pii/S0305054816302544. doi:https://doi.org/10.1016/j.cor.2016.10.006.

Lian, Z., Gu, X., & Jiao, B. (2008). A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan. *Chaos, Solitons & Fractals*, *35*, 851 – 861. URL: http://www.sciencedirect.com/science/article/pii/S0960077906005388. doi:https://doi.org/10.1016/j.chaos.2006.05.082.

Lian, Z., Jiao, B., & Gu, X. (2006). A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan. *Applied Mathematics and Computation*, *183*, 1008 – 1017. URL: http://www.sciencedirect.com/science/article/pii/S0096300306006369. doi:https://doi.org/10.1016/j.amc.2006.05.168.

Liangliang, J., Zhang, C., & Shao, X., Xinyuand Yang (2017). A study on the impact of periodic and event-driven rescheduling on a manufacturing system: An integrated process planning and scheduling case. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *231*, 490–504. URL: https://doi.org/10.1177/0954405416629585. doi:10.1177/0954405416629585. arXiv:https://doi.org/10.1177/0954405416629585.

Liao, C.-J., Tjandradjaja, E., & Chung, T.-P. (2012). An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem. *Applied Soft Computing*, *12*, 1755 – 1764. URL: http://www.sciencedirect.com/science/article/pii/S1568494612000373. doi:https://doi.org/10.1016/j.asoc.2012.01.011.

Lin, S.-W., Lee, Z.-J., Ying, K.-C., & Lu, C.-C. (2011). Minimization of maximum lateness on parallel machines with sequence-dependent setup times and job release dates. *Computers Operations Research*, *38*, 809 – 815. URL: http://www.sciencedirect.com/science/article/pii/S0305054810002157. doi:https://doi.org/10.1016/j.cor.2010.09.020.

Lin, S.-W., Ying, K.-C., & Huang, C.-Y. (2013). Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm. *International Journal of Production Research*, *51*, 5029–5038. URL: http://dx.doi.org/10.1080/00207543.2013.790571. doi:10.1080/00207543.2013.790571. arXiv:http://dx.doi.org/10.1080/00207543.2013.790571.

Liu, B., Wang, L., & Jin, Y. H. (2007). An effective pso-based memetic algorithm for flow shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *37*, 18–27. doi:10.1109/TSMCB.2006.883272.

Liu, B., Wang, L., & Jin, Y.-H. (2008a). An effective hybrid pso-based algorithm for flow shop scheduling with limited buffers. *Computers Operations Research*, *35*, 2791 – 2806. URL: http://www.sciencedirect.com/science/article/pii/S0305054806003169. doi:https://doi.org/10.1016/j.cor.2006.12.013. Part Special Issue: Bio-inspired Methods in Combinatorial Optimization.

Liu, B., Wang, L., Qian, B., & Jin, Y. (2008b). Hybrid particle swarm optimization for stochastic flow shop scheduling with no-wait constraint. *IFAC Proceedings Volumes*, *41*, 15855 – 15860. URL: http://www.sciencedirect.com/science/article/pii/S1474667016415442. doi:https://doi.org/10.3182/20080706-5-KR-1001.02680. 17th IFAC World Congress.

Liu, H., Gao, L., & Pan, Q. (2011). A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem. *Expert Systems with Applications*, *38*, 4348 – 4360. URL: http://www.sciencedirect.com/science/article/pii/S0957417410010614. doi:https://doi.org/10.1016/j.eswa.2010.09.104.

Liu, W., Jin, Y., & Price, M. (2016a). A new nawaz–enscore–ham-based heuristic for permutation flow-shop problems with bicriteria of makespan and machine idle time. *Engineering Optimization*, *48*, 1808–1822. URL: http://dx.doi.org/10.1080/0305215X.2016.1141202. doi:10.1080/0305215X.2016.1141202. arXiv:http://dx.doi.org/10.1080/0305215X.2016.1141202.

Liu, W., Jin, Y., & Price, M. (2017). New scheduling algorithms and digital tool for dynamic permutation flowshop with newly arrived order. *International Journal of Production Research*, *55*, 3234–3248. URL: http://dx.doi.org/10.1080/00207543.2017.1285077. doi:10.1080/00207543.2017.1285077. arXiv:http://dx.doi.org/10.1080/00207543.2017.1285077.

Liu, X.-p., Liu, F., & Wang, J.-j. (2016b). An enhanced memetic algorithm for combinational disruption management in sequence-dependent permutation flowshop. In D.-S. Huang, V. Bevilacqua, & P. Premaratne (Eds.), *Intelligent Computing Theories and Application: 12th International Conference, ICIC 2016, Lanzhou, China, August 2-5, 2016, Proceedings, Part I* (pp. 548–559). Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-319-42291-6_55. doi:10.1007/978-3-319-42291-6_55.

Liu, Y.-F., & Liu, S.-Y. (2013). A hybrid discrete artificial bee colony algorithm for permutation flowshop scheduling problem. *Applied Soft Computing*, *13*, 1459 – 1463. URL: http://www.sciencedirect.com/science/article/pii/S1568494611004364. doi:https://doi.org/10.1016/j.asoc.2011.10.024. Hybrid evolutionary systems for manufacturing processes.

Liu, Y.-H., Huang, H.-P., & Lin, Y.-S. (2005). Dynamic scheduling of flexible manufacturing system using support vector machines. In *IEEE International Conference on Automation Science and Engineering, 2005.* (pp. 387–392). doi:10.1109/COASE.2005.1506800.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. In F. Glover, & G. A. Kochenberger (Eds.), *Handbook of Metaheuristics* (pp. 320–353). Boston, MA: Springer US. URL: https://doi.org/10.1007/0-306-48056-5_11. doi:10.1007/0-306-48056-5_11.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2010). Iterated local search: Framework and applications. In M. Gendreau, & J.-Y. Potvin (Eds.), *Handbook of Metaheuristics* (pp. 363–397). Boston, MA: Springer US. URL: https://doi.org/10.1007/978-1-4419-1665-5_12. doi:10.1007/978-1-4419-1665-5_12.

Lu, C., Xiao, S., Li, X., & Gao, L. (2016). An effective multi-objective discrete grey wolf optimizer for a real-world scheduling problem in welding production. *Advances in Engineering Software*, *99*, 161 – 176. URL: http://www.sciencedirect.com/science/article/pii/S0965997816301260. doi:https://doi.org/10.1016/j.advengsoft.2016.06.004.

Maccarthy, B. L., & Liu, J. (1993). Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*, *31*, 59–79. URL: http://dx.doi.org/10.1080/00207549308956713. doi:10.1080/00207549308956713. arXiv:http://dx.doi.org/10.1080/00207549308956713.

Madureira, A., Pereira, I., & Falcão, D. (2013). Dynamic adaptation for scheduling under rush manufacturing orders with case-based reasoning. In *International Conference on Algebraic and Symbolic Computation (SymComp 2013), At Lisbon, Portugal* September (pp. 330–344).

Manz, E., Haddock, J., & Mittenthal, J. (1989). Optimization Of An Automated Manufacturing System Simulation Model Using Simulated Annealing. *1989 Winter Simulation Conference Proceedings*, (pp. 390–395). doi:10.1109/WSC.1989.718704.

Marichelvam, M. K., Prabaharan, T., & Yang, X. S. (2014). Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan. *Applied Soft Computing*, *19*, 93 – 101. URL: http://www.sciencedirect.com/science/article/pii/S1568494614000738. doi:https://doi.org/10.1016/j.asoc.2014.02.005.

Marinakis, Y., & Marinaki, M. (2013). Particle swarm optimization with expanding neighborhood topology for the permutation flowshop scheduling problem. *Soft Computing*, *17*, 1159–1173. URL: https://doi.org/10.1007/s00500-013-0992-z. doi:10.1007/s00500-013-0992-z.

Mason, S. J., Fowler, J. W., & Matthew Carlyle, W. (2002). A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops. *Journal of Scheduling*, *5*, 247–262. URL: http://dx.doi.org/10.1002/jos.102. doi:10.1002/jos.102.

Mehta, S. V., & Uzsoy, R. M. (1998). Predictable scheduling of a job shop subject to breakdowns. *IEEE Transactions on Robotics and Automation*, *14*, 365–378. doi:10.1109/70.678447.

Minella, G., Ruiz, R., & Ciavotta, M. (2008). A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. *INFORMS Journal on Computing*, *20*, 451–471. URL: https://doi.org/10.1287/ijoc.1070.0258. doi:10.1287/ijoc.1070.0258. arXiv:https://doi.org/10.1287/ijoc.1070.0258.

Minella, G., Ruiz, R., & Ciavotta, M. (2011). Restarted iterated pareto greedy algorithm for multi-objective flowshop scheduling problems. *Computers Operations Research*, *38*, 1521 – 1533. URL: http://www.sciencedirect.com/science/article/pii/S0305054811000220. doi:https://doi.org/10.1016/j.cor.2011.01.010.

Mirabi, M. (2014). A novel hybrid genetic algorithm to solve the sequence-dependent permutation flow-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, *71*, 429–437. URL: https://doi.org/10.1007/s00170-013-5489-5. doi:10.1007/s00170-013-5489-5.

Modarres, M., Kaminskiy, M., & Krivtsov, V. (1999). *Reliability Engineering and Risk Analysis: A Practical Guide*. New York: Marcel Dekker Inc.

Möhring, R. H., Radermacher, F. J., & Weiss, G. (1984). Stochastic scheduling problems i — general strategies. *Zeitschrift für Operations Research*, *28*, 193–260. URL: https://doi.org/10.1007/BF01919323. doi:10.1007/BF01919323.

Mokotoff, E. (2009). Multi-objective simulated annealing for permutation flow shop problems. In U. K. Chakraborty (Ed.), *Computational Intelligence in Flow Shop and Job Shop Scheduling* (pp. 101–150). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-642-02836-6_4. doi:10.1007/978-3-642-02836-6_4.

Molina-Sánchez, L. P., & González-Neira, E. M. (2016). GRASP to minimize total weighted tardiness in a permutation flow shop environment. *International Journal of Industrial Engineering Computations*, *7*, 161–176. doi:10.5267/j.ijiec.2015.6.004.

Murata, T., Ishibuchi, H., & Tanaka, H. (1996). Genetic algorithms for flowshop scheduling problems. *Computers Industrial Engineering*, *30*, 1061 – 1071. URL: http://www.sciencedirect.com/science/article/pii/0360835296000538. doi:https://doi.org/10.1016/0360-8352(96)00053-8.

Nagano, M. S., & Moccellin, J. V. (2002). A high quality solution constructive heuristic for flow shop sequencing. *Journal of the Operational Research Society*, *53*, 1374–1379. URL: https://doi.org/10.1057/palgrave.jors.2601466. doi:10.1057/palgrave.jors.2601466.

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem. *Omega*, *11*, 91 – 95. URL: http://www.sciencedirect.com/science/article/pii/0305048383900889. doi:https://doi.org/10.1016/0305-0483(83)90088-9.

Noura, A.-D., Jebali, A., & Diabat, A. (2016). A simulation-based genetic algorithm approach for the quay crane scheduling under uncertainty. *Simulation Modelling Practice and Theory*, *66*, 122 – 138. URL: http://www.sciencedirect.com/science/article/pii/S1569190X16000162. doi:https://doi.org/10.1016/j.simpat.2016.01.009.

Osman, I., & Potts, C. (1989). Simulated annealing for permutation flow-shop scheduling. *Omega*, *17*, 551 – 557. URL: http://www.sciencedirect.com/science/article/pii/0305048389900595. doi:https://doi.org/10.1016/0305-0483(89)90059-5.

Osyczka, A. (1985). 7– multicriteria optimization for engineering design. In J. S. Gero (Ed.), *Design Optimization* (pp. 193 – 227). Academic Press. URL: http://www.sciencedirect.com/science/article/pii/B978012280910150012X. doi:https://doi.org/10.1016/B978-0-12-280910-1.50012-X.

Ouelhadj, D., & Petrovic, S. (2008). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, *12*, 417. URL: https://doi.org/10.1007/s10951-008-0090-8. doi:10.1007/s10951-008-0090-8.

Ovacikt, I. M., & Uzsoy, R. (1994). Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times. *International Journal of Production Research*, *32*, 1243–1263. URL: http://dx.doi.org/10.1080/00207549408956998. doi:10.1080/00207549408956998. arXiv:http://dx.doi.org/10.1080/00207549408956998.

Palmer, D. S. (1965). Sequencing jobs through a multi-stage process in the minimum total time—a quick method of obtaining a near optimum. *Journal of the Operational Research Society*, *16*, 101–107. URL: https://doi.org/10.1057/jors.1965.8. doi:10.1057/jors.1965.8.

Pan, J. C.-H., & Chen, J.-S. (2005). Mixed binary integer programming formulations for the reentrant job shop scheduling problem. *Computers Operations Research*, *32*, 1197 – 1212. URL: http://www.sciencedirect.com/science/article/pii/S0305054803003174. doi:https://doi.org/10.1016/j.cor.2003.10.004.

Pan, Q.-K., Gao, L., Li, X.-Y., & Gao, K.-Z. (2017). Effective metaheuristics for scheduling a hybrid flowshop with sequence-dependent setup times. *Applied Mathematics and Computation*, *303*, 89 – 112. URL: http://www.sciencedirect.com/science/article/pii/S0096300317300127. doi:https://doi.org/10.1016/j.amc.2017.01.004.

Pan, Q.-K., Tasgetiren, M. F., & Liang, Y.-C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers Operations Research*, *35*, 2807 – 2839. URL: http://www.sciencedirect.com/science/article/pii/S0305054806003170. doi:https://doi.org/10.1016/j.cor.2006.12.030. Part Special Issue: Bio-inspired Methods in Combinatorial Optimization.

Parajuli, A. (2010). *Scheduling to Optimize Due Date Performance under Uncertainty of Processing Times*. Ph.D. thesis University of CALGARY.

Park, J., Mei, Y., Nguyen, S., Chen, G., & Zhang, M. (2017). Investigating the generality of genetic programming based hyper-heuristic approach to dynamic job shop scheduling with machine breakdown. In M. Wagner, X. Li, & T. Hendtlass (Eds.), *Artificial Life and Computational Intelligence: Third Australasian Conference, ACALCI 2017, Geelong, VIC, Australia, January 31 – February 2, 2017, Proceedings* (pp. 301–313). Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-319-51691-2_26. doi:10.1007/978-3-319-51691-2_26.

Pei, J., Liu, X., Fan, W., Pardalos, P. M., Migdalas, A., Goldengorin, B., & Yang, S. (2016). Minimizing the makespan for a serial-batching scheduling problem with arbitrary machine breakdown and dynamic job arrival. *The International Journal of Advanced Manufacturing Technology*, *86*, 3315–3331. URL: https://doi.org/10.1007/s00170-016-8408-8. doi:10.1007/s00170-016-8408-8.

Pereira, L. F. d. J. V. (2016). *Optimization and Simulation of Manufacturing Systems*. Ph.D. thesis Faculdade de Engenharia da Universidade do Porto.

Perelson, A. S. (1989). Immune network theory. *Immunological Reviews*, *110*, 5–36. URL: http://dx.doi.org/10.1111/j.1600-065X.1989.tb00025.x. doi:10.1111/j.1600-065X.1989.tb00025.x.

Persson, G. H. N. A. L. T. E. J. F. S., A., & Stablum, P. (2006). Simulation-based multi-objective optimization of real-world scheduling problem. In L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, & R. M. Fujimoto. (Eds.), *Winter Simulation Conference* (pp. 1757–1764). IEEE.

Pickardt, C. (2013). *Evolutionary Methods for the Design of Dispatching Rules for Complex and Dynamic Scheduling Problems*. Ph.D. thesis.

Pinedo, M. L. (2016). *Scheduling: Theory, Algorithms, and Systems*. (5th ed.). Springer Publishing Company, Incorporated.

Pour, H. D. (2001). A new heuristic for the *n*-job, *m*-machine flow-shop problem. *Production Planning & Control*, *12*, 648–653. URL: http://dx.doi.org/10.1080/09537280152582995. doi:10.1080/09537280152582995. arXiv:http://dx.doi.org/10.1080/09537280152582995.

Prabhaharan, G., Khan, B. S. H., & Rakesh, L. (2006). Implementation of grasp in flow shop scheduling. *The International Journal of Advanced Manufacturing Technology*, *30*, 1126–1131. URL: https://doi.org/10.1007/s00170-005-0134-6. doi:10.1007/s00170-005-0134-6.

Pugazhenthi, R., & Saravanan, R. (2015). A heuristic to minimise makespan time with breakdown nature flowshop, . (pp. 27–29).

Qian, B., Wang, L., Huang, D.-x., Wang, W.-l., & Wang, X. (2009). An effective hybrid de-based algorithm for multi-objective flow shop scheduling with limited buffers. *Computers Operations Research*, *36*, 209 – 233. URL: http://www.sciencedirect.com/science/article/pii/S0305054807001542. doi:https://doi.org/10.1016/j.cor.2007.08.007. Part Special Issue: Operations Research Approaches for Disaster Recovery Planning.

Qian, B., Wang, L., Huang, D.-X., & Wang, X. (2006). Multi-objective flow shop scheduling using differential evolution. In D.-S. Huang, K. Li, & G. W. Irwin (Eds.), *Intelligent Computing in Signal Processing and Pattern Recognition: International Conference on Intelligent Computing, ICIC 2006 Kunming, China, August 16–19, 2006* (pp. 1125–1136). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-540-37258-5_146. doi:10.1007/978-3-540-37258-5_146.

Rahimi-Vahed, A., & Mirzaei, A. H. (2008). Solving a bi-criteria permutation flow-shop problem using shuffled frog-leaping algorithm. *Soft Computing*, *12*, 435–452. URL: https://doi.org/10.1007/s00500-007-0210-y. doi:10.1007/s00500-007-0210-y.

Rahimi-Vahed, A. R., & Mirghorbani, S. M. (2007). A multi-objective particle swarm for a flow shop scheduling problem. *Journal of Combinatorial Optimization*, *13*, 79–102. URL: https://doi.org/10.1007/s10878-006-9015-7. doi:10.1007/s10878-006-9015-7.

Rahman, H., Sarker, R., & Essam, D. (2013). Permutation flow shop scheduling with dynamic job order arrival. In *2013 IEEE Conference on Cybernetics and Intelligent Systems (CIS)* (pp. 30–35). doi:10.1109/ICCIS.2013.6751574.

Rahman, H. F., Sarker, R., & Essam, D. (2015). A real-time order acceptance and scheduling approach for permutation flow shop problems. *European Journal of Operational Research*, *247*, 488 – 503. URL: http://www.sciencedirect.com/science/article/pii/S0377221715005329. doi:https://doi.org/10.1016/j.ejor.2015.06.018.

Rahmani, D., & Heydari, M. (2014). Robust and stable flow shop scheduling with unexpected arrivals of new jobs and uncertain processing times. *Journal of Manufacturing Systems*, *33*, 84 – 92. URL: http://www.sciencedirect.com/science/article/pii/S0278612513000332. doi:https://doi.org/10.1016/j.jmsy.2013.03.004.

Rahmani, D., Ramezanian, R., & Mehrabad, M. S. (2014). Multi-objective flow shop scheduling problem with stochastic parameters fuzzy goal programming approach. *International Journal of Operational Research*, *21*, 322–340. doi:10.1504/IJOR.2014.065411.

Rajendran, C., & Ziegler, H. (2004). Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research*, *155*, 426 – 438. URL: http://www.sciencedirect.com/science/article/pii/S0377221702009086. doi:https://doi.org/10.1016/S0377-2217(02)00908-6. Financial Risk in Open Economies.

Ramanan, T., Iqbal, M., & Umarali, K. (2014). A particle swarm optimization approach for permutation flow shop scheduling problem. *International Journal for Simulation and Multidisciplinary Design Optimization*, *5*, A20.

Rangsaritratsamee, R., Ferrell, W. G., & Kurz, M. B. (2004). Dynamic rescheduling that simultaneously considers efficiency and stability. *Computers Industrial Engineering*, *46*, 1 – 15. URL: http://www.sciencedirect.com/science/article/pii/S0360835203000950. doi:https://doi.org/10.1016/j.cie.2003.09.007.

Rao, K. V., & Ranga Janardhana, G. (2014). The effect of rescheduling on operating performance of the supply chain under disruption - a literature review. In *Dynamics of Machines and Mechanisms, Industrial Research* (pp. 2704–2710). Trans Tech Publications volume 592 of *Applied Mechanics and Materials*. doi:10.4028/www.scientific.net/AMM.592-594.2704.

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computers Operations Research*, *22*, 5–13. URL: http://www.sciencedirect.com/science/article/pii/0305054893E0014K. doi:https://doi.org/10.1016/0305-0548(93)E0014-K. Genetic Algorithms.

Reisman, A., Kumar, A., & Motwani, J. (1997). Flowshop scheduling/sequencing research: a statistical review of the literature, 1952-1994. *IEEE Transactions on Engineering Management*, *44*, 316–329. doi:10.1109/17.618173.

Ribas, I., Companys, R., & Tort-Martorell, X. (2015). An efficient discrete artificial bee colony algorithm for the blocking flow shop problem with total flowtime minimization. *Expert Systems with Applications*, *42*, 6155 – 6167. URL: http://www.sciencedirect.com/science/article/pii/S0957417415002201. doi:https://doi.org/10.1016/j.eswa.2015.03.026.

Ribas, I., Companys, R., & Tort-Martorell, X. (2017). Efficient heuristics for the parallel blocking flow shop scheduling problem. *Expert Systems with Applications*, *74*, 41 – 54. URL: http://www.sciencedirect.com/science/article/pii/S0957417417300064. doi:https://doi.org/10.1016/j.eswa.2017.01.006.

Ribas, I., & Mateo, M. (2010). Improvement tools for neh based heuristics on permutation and blocking flow shop scheduling problems. In B. Vallespir, & T. Alix (Eds.), *Advances in Production Management Systems. New Challenges, New Approaches: IFIP WG 5.7 International Conference, APMS 2009, Bordeaux, France, September 21-23, 2009, Revised Selected Papers* (pp. 33–40). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-642-16358-6_5. doi:10.1007/978-3-642-16358-6_5.

Robert, R. B. J., & Kumar, R. R. (2016). A Hybrid Algorithm for Minimizing Makespan in the Permutation Flow Shop Scheduling Environment. *Asian Journal of Research in Social Sciences and Humanities*, *6*, 1239–1242. doi:10.5958/2249-7315.2016.00867.4.

Rossi, F. L., Nagano, M. S., & Neto, R. F. T. (2016). Evaluation of high performance constructive heuristics for the flow shop with makespan minimization. *The International Journal of Advanced Manufacturing Technology*, *87*, 125–136. URL: https://doi.org/10.1007/s00170-016-8484-9. doi:10.1007/s00170-016-8484-9.

Rossi, F. L., Nagano, M. S., & Sagawa, J. K. (2017). An effective constructive heuristic for permutation flow shop scheduling problem with total flow time criterion. *The International Journal of Advanced Manufacturing Technology*, *90*, 93–107. URL: https://doi.org/10.1007/s00170-016-9347-0. doi:10.1007/s00170-016-9347-0.

Rothkopf, M. H. (1966). Scheduling with random service times. *Management Science*, *12*, 707–713. URL: https://doi.org/10.1287/mnsc.12.9.707. doi:10.1287/mnsc.12.9.707. arXiv:https://doi.org/10.1287/mnsc.12.9.707.

Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, *165*, 479 – 494. URL: http://www.sciencedirect.com/science/article/pii/S0377221704002553. doi:https://doi.org/10.1016/j.ejor.2004.04.017. Project Management and Scheduling.

Ruiz, R., Maroto, C., & Alcaraz, J. (2006). Two new robust genetic algorithms for the flowshop scheduling problem. *Omega*, *34*, 461 – 476. URL: http://www.sciencedirect.com/science/article/pii/S0305048305000174. doi:https://doi.org/10.1016/j.omega.2004.12.006.

Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, *177*, 2033 – 2049. URL: http://www.sciencedirect.com/science/article/pii/S0377221705008507. doi:https://doi.org/10.1016/j.ejor.2005.12.009.

Saaty, T. L. (1981). *The Analytic Hierarchy Process*. McGraw-Hill, Inc.

Sabuncuoglu, I., & Kizilisik, B. O. (2003). Reactive scheduling in a dynamic and stochastic fms environment. *International Journal of Production Research*, *41*, 4211–4231. URL: http://dx.doi.org/10.1080/0020754031000149202. doi:10.1080/0020754031000149202. arXiv:http://dx.doi.org/10.1080/0020754031000149202.

Sahin, C., Demirtas, M., Erol, R., Baykasoğlu, A., & Kaplanoğlu, V. (2015). A multi-agent based approach to dynamic scheduling with flexible processing capabilities. *Journal of Intelligent Manufacturing*, . URL: https://doi.org/10.1007/s10845-015-1069-x. doi:10.1007/s10845-015-1069-x.

Salch, A., Gayon, J. P., & Lemaire, P. (2013). Optimal static priority rules for stochastic scheduling with impatience. *Operations Research Letters*, *41*, 81 – 85. URL: http://www.sciencedirect.com/science/article/pii/S0167637712001447. doi:https://doi.org/10.1016/j.orl.2012.11.008.

Saravanan, R., & Pugazhenthi, R. (2015). Computation of makespan time associated with stochastic natured jobs in a flow shop. *Vels Journal Of Mechanical Engineering*, *2*, 43–46.

Sha, D., & Hsu, C.-Y. (2008). A new particle swarm optimization for the open shop scheduling problem. *Computers Operations Research*, *35*, 3243 – 3261. URL: http://www.sciencedirect.com/science/article/pii/S030505480700055X. doi:https://doi.org/10.1016/j.cor.2007.02.019. Part Special Issue: Search-based Software Engineering.

Sha, D. Y., & Hung Lin, H. (2009). A particle swarm optimization for multi-objective flowshop scheduling. *The International Journal of Advanced Manufacturing Technology*, *45*, 749–758. URL: https://doi.org/10.1007/s00170-009-1970-6. doi:10.1007/s00170-009-1970-6.

Shao, W., & Pi, D. (2016). A self-guided differential evolution with neighborhood search for permutation flow shop scheduling. *Expert Systems with Applications*, *51*, 161 – 176. URL: http://www.sciencedirect.com/science/article/pii/S0957417415007927. doi:https://doi.org/10.1016/j.eswa.2015.12.001.

Shao, W., Pi, D., & Shao, Z. (2017). Memetic algorithm with node and edge histogram for no-idle flow shop scheduling problem to minimize the makespan criterion. *Applied Soft Computing*, *54*, 164 – 182. URL: http://www.sciencedirect.com/science/article/pii/S1568494617300327. doi:https://doi.org/10.1016/j.asoc.2017.01.017.

Sharma, M., Soni, R., Chaudhary, A., & Goar, V. (2016). Comparison of approximation heuristics with minimizing make-span in permutation flow shop scheduling environment. In *2016 Second International Conference on Computational Intelligence Communication Technology (CICT)* (pp. 539–542). doi:10.1109/CICT.2016.112.

Shen, J. N., Wang, L., & Wang, S. Y. (2015). A bi-population {EDA} for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion. *Knowledge-Based Systems*, *74*, 167 – 175. URL: http://www.sciencedirect.com/science/article/pii/S095070511400416X. doi:https://doi.org/10.1016/j.knosys.2014.11.016.

Sioud, A., Gagné, C., & Dort, J. (2015). A gismoo algorithm for a multi-objective permutation flowshop with sequence-dependent setup times. In *2015 7th International Joint Conference on Computational Intelligence (IJCCI)* (pp. 116–121). volume 1.

de Siqueira, E. C., Diana, R. O. M., Souza, M. J. F., & de Souza, S. R. (2016). A study concerning the application of genetic algorithms for solving the multi-objective hybrid flowshop scheduling. *XIII Encontro Nacional de Inteligencia Artificial e Computacional*, (pp. 301–312).

Skutella, M., & Uetz, M. (2005). Stochastic machine scheduling with precedence constraints. *SIAM Journal on Computing*, *34*, 788–802. URL: https://doi.org/10.1137/S0097539702415007. doi:10.1137/S0097539702415007. arXiv:https://doi.org/10.1137/S0097539702415007.

Solimanpur, M., Vrat, P., & Shankar, R. (2004). A neuro-tabu search heuristic for the flow shop scheduling problem. *Computers Operations Research*, *31*, 2151 – 2164. URL: http://www.sciencedirect.com/science/article/pii/S0305054803001692. doi:https://doi.org/10.1016/S0305-0548(03)00169-2.

Steele, D. C. (1975). The nervous mrp system: how to do battle. *Production and Inventory Management*, *16*, 83–89.

Stützle, T. (1998). *Applying Iterated Local Search to the Permutation Flow Shop Problem*. Technical Report AIDA-98-04, FG Intellektik, TU Darmstadt.

Subashini, G., & Bhuvaneswari, M. C. (2011). Non Dominated Particle Swarm Optimization For Scheduling Independent Tasks On Heterogeneous Distributed Environments. *International Journal of Advances in Soft Computing and its Applications*, *3*, 1–17.

Suliman, S. (2000). A two-phase heuristic approach to the permutation flow-shop scheduling problem. *International Journal of Production Economics*, *64*, 143 – 152. URL: http://www.sciencedirect.com/science/article/pii/S0925527399000535. doi:https://doi.org/10.1016/S0925-5273(99)00053-5.

Sun, Y., Zhang, C., Gao, L., & Wang, X. (2011). Multi-objective optimization algorithms for flow shop scheduling problem: a review and prospects. *The International Journal of Advanced Manufacturing Technology*, *55*, 723–739. URL: https://doi.org/10.1007/s00170-010-3094-4. doi:10.1007/s00170-010-3094-4.

Suwa, H., & Sandoh, H. (2013). *Online Scheduling in Manufacturing A Cumulative Delay Approach*. London: Springer-Verlag. doi:10.1201/9781420072747-c3.

Taillard, E. (1990). Theory and methodology some efficient heuristic methods for the flow shop sequencing problem. *European Journal Of Operational Research*, *47*, 65–74.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*, 278 – 285. URL: http://www.sciencedirect.com/science/article/pii/037722179390182M. doi:https://doi.org/10.1016/0377-2217(93)90182-M. Project Management anf Scheduling.

Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation*. Hoboken, New Jersey Published simultaneously in Canada: JohnWiley & Sons, Inc. All.

Tasgetiren, M. F., Kizilay, D., Pan, Q.-K., & Suganthan, P. N. (2017). Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. *Computers Operations Research*, *77*, 111 – 126. URL: http://www.sciencedirect.com/science/article/pii/S030505481630171X. doi:https://doi.org/10.1016/j.cor.2016.07.002.

Tasgetiren, M. F., Liang, Y.-C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research*, *177*, 1930 – 1947. URL: http://www.sciencedirect.com/science/article/pii/S0377221705008453. doi:https://doi.org/10.1016/j.ejor.2005.12.024.

Tasgetiren, M. F., Pan, Q.-K., Suganthan, P., & Buyukdagli, O. (2013). A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem. *Computers Operations Research*, *40*, 1729 – 1743. URL: http://www.sciencedirect.com/science/article/pii/S0305054813000130. doi:https://doi.org/10.1016/j.cor.2013.01.005.

Tasgetiren, M. F., Sevkli, M., Liang, Y.-C., & Gencyilmaz, G. (2004). Particle swarm optimization algorithm for permutation flowshop sequencing problem. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, & T. Stützle (Eds.), *Ant Colony Optimization and Swarm Intelligence: 4th International Workshop, ANTS 2004, Brussels, Belgium, September 5-8, 2004. Proceedings* (pp. 382–389). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-540-28646-2_38. doi:10.1007/978-3-540-28646-2_38.

Tekin, E., & Sabuncuoglu, I. (2004). Simulation optimization: A comprehensive review on theory and applications. *IIE Transactions*, *36*, 1067–1081. URL: http://dx.doi.org/10.1080/07408170490500654. doi:10.1080/07408170490500654. arXiv:http://dx.doi.org/10.1080/07408170490500654.

Tiwari, A., Chang, P.-C., Tiwari, M., & Kollanoor, N. J. (2015). A pareto block-based estimation and distribution algorithm for multi-objective permutation flow shop scheduling problem. *International Journal of Production Research*, *53*, 793–834. URL:

http://dx.doi.org/10.1080/00207543.2014.933273. doi:10.1080/00207543.2014.933273. arXiv:http://dx.doi.org/10.1080/00207543.2014.933273.

Toptal, A., & Sabuncuoglu, I. (2010). Distributed scheduling: a review of concepts and applications. *International Journal of Production Research*, *48*, 5235–5262. URL: http://dx.doi.org/10.1080/00207540903121065. doi:10.1080/00207540903121065. arXiv:http://dx.doi.org/10.1080/00207540903121065.

Turkcan, A., Akturk, M. S., & Storer, R. H. (2009). Predictive/reactive scheduling with controllable processing times and earliness-tardiness penalties. *IIE Transactions*, *41*, 1080–1095. URL: http://dx.doi.org/10.1080/07408170902905995. doi:10.1080/07408170902905995. arXiv:http://dx.doi.org/10.1080/07408170902905995.

Turner, S., & Booth, D. (1987). Comparison of heuristics for flow shop sequencing. *Omega*, *15*, 75 – 78. URL: http://www.sciencedirect.com/science/article/pii/0305048387900545. doi:https://doi.org/10.1016/0305-0483(87)90054-5.

Vallada, E., Ruiz, R., & Framinan, J. M. (2015). New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*, *240*, 666 – 677. URL: http://www.sciencedirect.com/science/article/pii/S0377221714005992. doi:https://doi.org/10.1016/j.ejor.2014.07.033.

Vasiljevic, D., & Danilovic, M. (2015). Handling ties in heuristics for the permutation flow shop scheduling problem. *Journal of Manufacturing Systems*, *35*, 1 – 9. URL: http://www.sciencedirect.com/science/article/pii/S027861251400140X. doi:https://doi.org/10.1016/j.jmsy.2014.11.011.

Vieira, G. E., Herrmann, J. W., & Lin, E. (2003). Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling*, *6*, 39–62. URL: https://doi.org/10.1023/A:1022235519958. doi:10.1023/A:1022235519958.

Vijay chakaravarthy, G., Marimuthu, S., & Naveen Sait, A. (2013). Performance evaluation of proposed differential evolution and particle swarm optimization algorithms for scheduling *m*-machine flow shops with lot streaming. *Journal of Intelligent Manufacturing*, *24*, 175–191. URL: https://doi.org/10.1007/s10845-011-0552-2. doi:10.1007/s10845-011-0552-2.

Wang, B., & Yang, Z. (2007). A particle swarm optimization algorithm for robust flowshop scheduling with fuzzy processing times. In *2007 IEEE International Conference on Automation and Logistics* (pp. 824–828). doi:10.1109/ICAL.2007.4338678.

Wang, D.-J., Liu, F., Wang, Y.-z., & Jin, Y. (2015a). A knowledge-based evolutionary proactive scheduling approach in the presence of machine breakdown and deterioration effect. *Knowledge-Based Systems*, *90*, 70 – 80. URL: http://www.sciencedirect.com/science/article/pii/S0950705115003718. doi:https://doi.org/10.1016/j.knosys.2015.09.032.

Wang, K., & Choi, S. (2014). A holonic approach to flexible flow shop scheduling under stochastic processing times. *Computers Operations Research*, *43*, 157 – 168. URL: http://www.sciencedirect.com/science/article/pii/S0305054813002839. doi:https://doi.org/10.1016/j.cor.2013.09.013.

Wang, K., Choi, S., & Lu, H. (2015b). A hybrid estimation of distribution algorithm for simulation-based scheduling in a stochastic permutation flowshop. *Computers Industrial Engineering*, *90*, 186 – 196. URL: http://www.sciencedirect.com/science/article/pii/S0360835215003873. doi:https://doi.org/10.1016/j.cie.2015.09.007.

Wang, K., Huang, Y., & Qin, H. (2016). A fuzzy logic-based hybrid estimation of distribution algorithm for distributed permutation flowshop scheduling problems under machine breakdown. *Journal of the Operational Research Society*, *67*, 68–82. URL: https://doi.org/10.1057/jors.2015.50. doi:10.1057/jors.2015.50.

Wang, L., Zhang, L., & Zheng, D.-Z. (2005). A class of hypothesis-test-based genetic algorithms for flow shop scheduling with stochastic processing time. *The International Journal of Advanced Manufacturing Technology*, *25*, 1157–1163. URL: https://doi.org/10.1007/s00170-003-1961-y. doi:10.1007/s00170-003-1961-y.

Wang, X.-j., Zhang, C.-y., Gao, L., & Li, P.-g. (2008). A survey and future trend of study on multi-objective scheduling. In *2008 Fourth International Conference on Natural Computation* (pp. 382–391). volume 6. doi:10.1109/ICNC.2008.817.

Weiss, G. (1991). Approximation results in parallel machines stochastic scheduling. *Ann. Oper. Res.*, *26*, 195–242. URL: http://dx.doi.org/10.1007/BF02248591. doi:10.1007/BF02248591.

Weiss, G. (1992). Turnpike optimality of smith's rule in parallel machines stochastic scheduling. *Mathematics of Operations Research*, *17*, 255–270. URL: https://doi.org/10.1287/moor.17.2.255. doi:10.1287/moor.17.2.255. arXiv:https://doi.org/10.1287/moor.17.2.255.

Weng, W., & Fujimura, S. (2009). Distributed feedback mechanism for just-in-time scheduling problem, . (pp. 15–20). doi:10.1109/ICIS.2009.12.

Weng, W., & Fujimura, S. (2010). Flexible flow shop scheduling by intelligent multi-agents. In *2010 Eighth ACIS International Conference on Software Engineering Research, Management and Applications* (pp. 113–120). doi:10.1109/SERA.2010.24.

Xia, H., Li, X., & Gao, L. (2016). A hybrid genetic algorithm with variable neighborhood search for dynamic integrated process planning and scheduling. *Computers Industrial Engineering*, *102*, 99 – 112. URL: http://www.sciencedirect.com/science/article/pii/S0360835216303849. doi:https://doi.org/10.1016/j.cie.2016.10.015.

Xingbao, H. A. N., Bing, W., & Yabing, N. I. E. (2015). Predictive scheduling for permutation flow shop subject to new arrival job. In *2015 34th Chinese Control Conference (CCC)* (pp. 2732–2737). doi:10.1109/ChiCC.2015.7260056.

Xu, J., Wu, C.-C., Yin, Y., & Lin, W.-C. (2017). An iterated local search for the multi-objective permutation flowshop scheduling problem with sequence-dependent setup times. *Applied Soft Computing*, *52*, 39 – 47. URL: http://www.sciencedirect.com/science/article/pii/S1568494616306007. doi:https://doi.org/10.1016/j.asoc.2016.11.031.

Yamamoto, M., & Nof, S. Y. (1985). Scheduling/rescheduling in the manufacturing operating system environment. *International Journal of Production Research*, *23*, 705–722. URL: http://dx.doi.org/10.1080/00207548508904739. doi:10.1080/00207548508904739. arXiv:http://dx.doi.org/10.1080/00207548508904739.

Yang, T., Kuo, Y., & Chang, I. (2004). Tabu-search simulation optimization approach for flow-shop scheduling with multiple processors — a case study. *International Journal of Production Research*, *42*, 4015–4030. URL: http://dx.doi.org/10.1080/00207540410001699381. doi:10.1080/00207540410001699381. arXiv:http://dx.doi.org/10.1080/00207540410001699381.

Yenisey, M. M., & Yagmahan, B. (2014). Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega*, *45*, 119 – 135. URL: http://www.sciencedirect.com/science/article/pii/S0305048313000832. doi:https://doi.org/10.1016/j.omega.2013.07.004.

Ying, K.-C. (2007). Solving non-permutation flowshop scheduling problems by an effective iterated greedy heuristic. *The International Journal of Advanced Manufacturing Technology*, *38*, 348. URL: https://doi.org/10.1007/s00170-007-1104-y. doi:10.1007/s00170-007-1104-y.

Ying, K.-C. (2009). An iterated greedy heuristic for multistage hybrid flowshop scheduling problems with multiprocessor tasks. *Journal of the Operational Research Society*, *60*, 810–817. URL: https://doi.org/10.1057/palgrave.jors.2602625. doi:10.1057/palgrave.jors.2602625.

Ying, K.-C. (2012). Scheduling identical wafer sorting parallel machines with sequence-dependent setup times using an iterated greedy heuristic. *International Journal of Production Research*, *50*, 2710–2719. URL: http://dx.doi.org/10.1080/00207543.2011.588617. doi:10.1080/00207543.2011.588617. arXiv:http://dx.doi.org/10.1080/00207543.2011.588617.

Ying, K.-C., & Cheng, H.-M. (2010). Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic. *Expert Systems with Applications*, *37*, 2848 – 2852. URL: http://www.sciencedirect.com/science/article/pii/S0957417409007878. doi:https://doi.org/10.1016/j.eswa.2009.09.006.

Ying, K.-C., Lin, S.-W., & Huang, C.-Y. (2009). Sequencing single-machine tardiness problems with sequence dependent setup times using an iterated greedy heuristic. *Expert Systems with Applications*, *36*, 7087 – 7092. URL: http://www.sciencedirect.com/science/article/pii/S0957417408005873. doi:https://doi.org/10.1016/j.eswa.2008.08.033.

Ying, K.-C., Lin, S.-W., & Wan, S.-Y. (2014). Bi-objective reentrant hybrid flowshop scheduling: an iterated pareto greedy algorithm. *International Journal of Production Research*, *52*, 5735–5747. URL: http://dx.doi.org/10.1080/00207543.2014.910627. doi:10.1080/00207543.2014.910627. arXiv:http://dx.doi.org/10.1080/00207543.2014.910627.

Yuan, B., Jiang, Z., & Wang, L. (2016). Dynamic parallel machine scheduling with random breakdowns using the learning agent. *International Journal of Services Operations and Informatics*, *8*, 1741–5403.

Zadeh, L. (1963). Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, *8*, 59–60. doi:10.1109/TAC.1963.1105511.

Zangari, M., Mendiburu, A., Santana, R., & Pozo, A. (2017). Multiobjective decomposition-based mallows models estimation of distribution algorithm. a case of study for permutation flowshop scheduling problem. *Information Sciences*, *397*, 137 – 154. URL: http://www.sciencedirect.com/science/article/pii/S0020025517305352. doi:https://doi.org/10.1016/j.ins.2017.02.034.

Zhang, C., Ning, J., & Ouyang, D. (2010). A hybrid alternate two phases particle swarm optimization algorithm for flow shop scheduling problem. *Computers Industrial Engineering*, *58*, 1 – 11. URL: http://www.sciencedirect.com/science/article/pii/S0360835209000357. doi:https://doi.org/10.1016/j.cie.2009.01.016.

Zhang, C., Sun, J., Zhu, X., & Yang, Q. (2008). An improved particle swarm optimization algorithm for flowshop scheduling problem. *Information Processing Letters*, *108*, 204 – 209. URL: http://www.sciencedirect.com/science/article/pii/S0020019008001567. doi:https://doi.org/10.1016/j.ipl.2008.05.010.

Zhang, L., Li, X., Gao, L., & Zhang, G. (2016). Dynamic rescheduling in fms that is simultaneously considering energy consumption and schedule efficiency. *The International Journal of Advanced Manufacturing Technology*, *87*, 1387–1399. URL: https://doi.org/10.1007/s00170-013-4867-3. doi:10.1007/s00170-013-4867-3.

Zhang, L., & Wu, J. (2014). A PSO-based hybrid metaheuristic for permutation flowshop scheduling problems. *TheScientificWorldJournal*, *2014*, 902950. doi:10.1155/2014/902950.

Zhang, Y., Wang, S., & Ji, G. (2015). A comprehensive survey on particle swarm optimization algorithm and its applications. *Hindawi Publishing Corporation Mathematical Problems in Engineering*, *2015*, 38. arXiv:931256.

Zheng, D.-Z., & Wang, L. (2003). An effective hybrid heuristic for flow shop scheduling. *The International Journal of Advanced Manufacturing Technology*, *21*, 38–44. URL: https://doi.org/10.1007/s001700300005. doi:10.1007/s001700300005.

Zio, E. (2007). *An Introduction to the Basics of Reliability and Risk Analysis*. World Scientific Publishing Co. Re. Ltd.