Multi-objective Biased Randomised Iterated Greedy for Robust Permutation Flow Shop Scheduling Problem under Disturbances

Mohanad AL-Behadili[a] and Djamila Ouelhadj[b] and Dylan Jones[b]

[a]Department of Mathematics, College of Science, University of Basrah, Basrah, Iraq;
[b]Department of Mathematics, Faculty of Technology, University of Portsmouth, Lion Gate Building, Lion Terrace, Portsmouth, Hampshire, PO1 3HF, UK

ABSTRACT

Nowadays, scheduling problems under different disruptions are a key to become competitive in the global market of this century. Permutation flow shop scheduling problems are very important as they consider one of the important types of scheduling problems. In this paper, we consider a challenging scheduling problem of a permutation flow shop in the presence of different types of real-time events such as new job arrival and machine breakdown. A multi-objective optimisation model that takes into account multiple performance measures in order to minimise the effect of different real-time events is used in this paper. To solve this problem, we apply the proposed multi-objective model and adapt a predictive-reactive based Biased Randomised Iterated Greedy approach for the problem, which is hybridised a Biased Randomisation process and the Iterated Greedy algorithm. Furthermore, the proposed approach is compared against the predictive-reactive based Particle Swarm Optimisation method for the same problem. Additionally, to show the efficiency of the proposed model, we compare this model by testing the predictive-reactive based Biased Randomised Iterated Greedy approach to two other models: the bi-objective model that consider only two objectives and the classical single-objective model of minimising the makespan. Further statistical analysis is performed in this study by using an Analysis of Variance measure. The extensive experiments and statistical analysis demonstrate that the proposed multi-objective model is better than the other models in reducing the relative percentage deviation. Additionally, despite their simplicity, the Biased Randomised Iterated Greedy algorithm is shown to be State-of-the-art method that outperform the Particle Swarm Optimisation algorithm.

KEYWORDS

Permutation Flow Shop Scheduling; Multi-objective Optimisation Model; Predictive-Reactive Approach; Biased Randomised Iterated Greedy; Particle Swarm Optimisation

1. Introduction

Scheduling problems frequently arise in today's complex business and industrial environment. A general scheduling problem is defined as a decision-making process that is vital in many industrial and manufacturing services. It aims to assign a set of n-jobs to a set of m-machines in a period of time so as to optimise (minimise or max-

CONTACT Mohanad AL-Behadili Email: mohanad.saad@uobasrah.edu.iq

imise) one or more objectives (Pinedo, 2016). The permutation flow-shop scheduling problem (PFSP) is a well-known scheduling problem which is described 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 proceed with at most one job at a time, and job passing is not permitted, i.e. the processing sequence of the jobs is the same for all machines. The classical criterion is to determine a single list for executing the jobs so that a given goal is optimised. The most common objective applied in the literature is the minimisation of the maximum completion time, or makespan. In general, for scheduling problem there is not any procedure that solves the problem in a polynomial time, signifying that the PFSP is an NP-hard problem (Graham, Lawler, Lenstra, and Kan, 1979). Depending on the scheduling environment, scheduling problems are classified into three categories (Wojakowski and WarŻołek, 2014), namely; static, dynamic and stochastic scheduling. In dynamic scheduling, random real-time events may disrupt the scheduling system, which could change the scheduled plan performance. In this case, the revised schedule in the presence of random real-time events is termed as a dynamic schedule. The importance of taking into accounts of disruptions and uncertainties with developing of rescheduling approaches to cope any of disruptions and uncertainties have been mainly identified in the last decade. Vieira, Herrmann, and Lin (2003) reviewed the strategies, policies and approaches of rescheduling. Aytug, A. Lawley, McKay, Mohan, and Uzsoy (2005) categorised scheduling problems under uncertainties for three approaches; Reactive Scheduling, Robust (proactive) scheduling and Predictive-Reactive scheduling. The predictive-reactive approach is the most common approach that used in the literature of scheduling in the presence of uncertainties. Besides the predictive-reactive approach, there is another effective approach, entitled Robust Optimisation introduced by Bertsimas, B. Brown, and Caramanis (2011), to deal with uncertainties in practical production environment. The authors introduced a survey of Robust Optimisation for both theoretical and applied areas. Their study concentrated on the computational attractiveness of Robust Optimisation approaches and the modeling power. They also highlighted the applications of Robust Optimisation along a wide areas, such as; statistics, finance, learning, and different domains of engineering.

In recent years, there has been more studies considering the solution of scheduling problems in real floor shop. Ruiz and Stützle (2008) present the gap between real life and theoretical flow shop scheduling problems. They also introduced heuristic methods and mathematical models to solve the realistic flow shop problems. A comprehensive survey of dynamic scheduling is given by Ouelhadj and Petrovic (2008), while some modern papers of related research in this category of scheduling problems are given by Sabuncuoglu and Goren (2009) and Suwa and Sandoh (2012).

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. 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. There are several attempts to introduce mathematical models that consider more than one measure to preserve the optimisation and stability for dynamic scheduling problems.

Cowling and Johansson (2002) and Cowling, Ouelhadj, and Petrovic (2003) proposed a multi-objective optimisation model that defines utility and stability measures to minimise the deviation between the baseline (predictive) and actual (realised) schedules. More recently, for the flow shop scheduling problem of n-jobs and 2-machines, Rahmani and Heydari (2014) introduced a robust multi-objective optimisation model that takes into account the important performance measures of utility, stability and robustness. This model was applied for the dynamic PFSP in the presence of different types of real-time events (new job arrivals and uncertain processing times). The authors apply an exact method for this model to determine the optimal solution. However, they test the model of small size instances using the GAMS software. The Iterated Greedy algorithm (IG) used for the PFSP in the presence of different types of disruptions and compared against three different heuristic algorithms including; Schedule repair algorithm, Local search one single pass algorithm and Complete local search algorithm (Katragjini, Vallada, and Ruiz, 2013). The authors also applied a bi-objective model considering the minimisation of both of the maximal completion time and the instability measures. Juan, Lourenço, Mateo, Luo, and Castella (2014) applied a discrete teaching-learning-based optimisation to solve the flow shop rescheduling problem with the goal of minimising two objectives; the makespan and the instability performance. They consider the flow shop problem in the presence of five types of disruption simultaneously, which are; machine breakdown, new job arrival, cancellation of jobs, job processing variation and job release variation. A single machine rescheduling problem with unexpected machine unavailability has been investigated by Luo, Luo, Goebel, and Lin (2018). For this rescheduling problem, they used the objective function that minimising the sum of the total weighted completion time and the weighted maximum time deviation. In this problem, the maximum time deviation is considered as part of the objective function and as constraint. To solve this problem, the authors applied an exact algorithm which is pseudo-polynomial time method and a fully polynomial time approximation scheme. A proactivereactive approach has been proposed by Liu, Wang, Y., and Yue (2017) to optimise the robustness and stability in a Pareto optimisation manner, under new job arrival, job availability delay and stochastic machine breakdown.

The following features should be considered for any algorithm to solve the PFSP: simplicity, efficiency, accuracy, flexibility and robustness. A method that has relatively these features is more likely to have the ability of solving large and complex problems. Simplicity avoids complexity in the design and implementation. Also, it averts time consuming fine-tuning processes. Efficiency is concerned about the quality of the generated results and the needed computational time. The accuracy is relevant to quality of the results, flexibility means the capability to handle different combinatorial optimisation problems under various scenarios. Finally, the robustness aims to generate schedules that are able to face real-time events without needing of new schedules. The aforementioned features are important to provide powerful algorithms that could deal efficiently with very complex and large PFSP. Such NP-hard complex and large problems require more attention when they considered under different environments of stochasticity and/or dynamicity. One of the most powerful algorithms for solving the PFSP is the Iterated Greedy algorithm (IG) that was introduced by Jacobs and Brusco (1995). The IG algorithm has improved and applied successfully for the PFSP by Ruiz and Stützle (2007). The authors showed that IG is a constructive method that relatively satisfied the features of simplicity, efficiency, accuracy, flexibility and robustness. Indeed, IG provides outstanding (STATE-of-the-art) results in terms of simplicity, accuracy and speed. Despite its relative simplicity, it is one of the most

efficient algorithms developed so far for the PFSP. Ruiz and Stützle (2007) tested IG against different approaches and the results showed that IG was the best performing approach. The authors showed in their work that IG is simpler and far superior to the hybrid Genetic algorithm proposed by Ruiz, Maroto, and Alcaraz (2006). Also, Pan, Ruiz, and Alfaro-Fernández (2017) used an IG algorithm for the hybrid flow shop scheduling problem 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 shows the best performance against all of the nine approaches. Ruiz, Pan, and Naderi (2019) applied an improved IG algorithm for the distributed permutation flowshop scheduling problem and compared it against five other recent algorithms. While the improved version of IG has maintaining most of its simplicity, it showed superior performance compared to all other presented methods. Arroyo, Leung, and Tavares (2019) proposed the IG algorithm for unrelated parallel batch machines with unequal job release and the objective of minimising the total flow time. The authors adapted three recent metaheuristic algorithms for the proposed problem and compared them with the IG algorithm. the IG algorithm significantly outperforms the other algorithms by a wide margin.

One of the solution methods applied to solve some of Combinatorial Optimisation Problems (COP) is probabilistic or randomised algorithms. They are often applied when there is some uncertainty or local optima involved. This kind of algorithms has been used widely for classes of COP such as: Scheduling Problems (Pinedo, 2016) and Vehicle Routing Problems (Laporte, 2009). Randomised techniques apply random variates or pseudo-random numbers in the constructive phase of the method. One of the advantage points of these techniques is that, the approach is 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. For more details about randomised approaches we referred the reader to (Collet and Rennard, 2007). In this work, we introduced a new approach based on the hybridisation of IG algorithm with a Biased Randomisation technique, which is the termed as Biased Randomised Iterated Greedy algorithm (BRIG). The Biased Randomised NEH heuristic is used to generate the initial solution for this approach. Moreover, the discretised decreasing triangular probability distribution (Kotz and Van Dorp, 2004) is used to generate random variates at the biased randomisation step. The BRIG method has been used successfully for the PFSP with the objective of minimising makespan (Juan et al., 2014), also it has been applied for the Routing Problems with a non-smooth objective function (Juan, Faulin, Ferrer, Lourenço, and Barrios, 2013).

The contribution of this paper is as follows: the BRIG based predictive-reactive approach is applied for the dynamic PFSP in the presence of different real-time events including machine breakdown and new job arrivals. To secure the stability and robustness of the problem, a multi-objective optimisation model of three measures of utility, stability and robustness (MSR) (Al-Behadili, Ouelhadj, and Jones, 2017) is applied along with the BRIG algorithm. This model is more realistic for the PFSP than only counting the objective of makespan, because in real manufacturing, the current schedule may deviate from its initial plan. In addition, the application of the BRIG based predictive-reactive approach for the PFSP under different disruptions is also tested with other two different models which are; the bi-objective optimisation model of Katragjini et al. (2013) and the single objective makespan model. Thus, the three models are compared to show the impact and efficiency of models. Moreover,

the BRIG based predictive-reactive approach is compared against the Particle Swarm Optimisation (PSO) based predictive-reactive approach (Al-Behadili et al., 2017) for same MSR model and same problem. The remainder of the paper is organised as follows; in section 2 a multi-objective optimisation model for the dynamic PFSP is presented. In section 3, the proposed solution methods are explained. In section 4, the real-time events and their combinations are explained. Section 5 shows experimental results that illustrate the proposed methodology and comparative study. Finally, section 6 describes the conclusions and future work related to this work.

2. A multi-objective optimisation model for robust dynamic PFSP

Scheduling problems are frequently executed in environments where uncertainty, stochastic and dynamic features are high. For this, consequence of a combination of different real-time events occurring will most likely to be a non-optimal performance. In this case, the goal of a robust scheduling approach is to get a sequence such that it keeps the current schedule robust despite the effect of the different real-time events by minimising the difference between the real schedule in real floor and the initial plan schedule (baseline). In this paper, a multi-objective optimisation model proposed by Al-Behadili et al. (2017) is presented. This model is designed for the PFSP of size $n \times m$ ($jobs \times machines$) in the presence of different real-time events. The model considers very important performance measures including utility, stability, and robustness. The proposed multi-objective optimisation model (MSR) is given as follows:

$$Min \quad MSR = \alpha U_n(S^*) + \beta I_n(S^*) + \gamma R_n(S^*) \tag{1}$$

where α, β and γ are the objective weights and $\alpha + \beta + \gamma = 1$, $U_n(S^*)$ is the real makespan, $I_n(S^*)$ is the stability objective and $R_n(S^*)$ is the robustness objective.
The first measure, utility, is defined as a classical makespan measure, which is used to indicate the degree of optimisation of the problem. It is required to define two makspans in this model, which are; the real and predictive makespans. To define these makespanes, for given processing times $p_{i\pi_j}$ for jobs $j = 1, 2, ..., n$ on machines $i = 1, 2, ..., m$, and a job permutation $\pi = \pi_1, \pi_2, ..., \pi_n$ that sequenced n jobs through m machines using the same permutation, the C_{ij} denotes the completion time of job π_j on machine i. The calculation of completion time for jobs j on machines i is given as follows:

$$C_{11} = P_{11}$$

$$C_{1j} = C_{1,j-1} + P_{1j}$$

$$C_{i1} = C_{i-1,1} + P_{i1}$$

$$C_{ij} = max\{C_{i,j-1}, C_{i-1,j}\} + P_{ij} \quad \forall i = 2, .., m, \quad j = 2, .., n$$

Where $P_{ij} = p_{i\pi_j}$. Then the makespan of the PFSP is defined as follows:

$$\sum_{j} C_{mj} \leq C_{mn}, \forall \pi \tag{2}$$

Hence, the first objective is defined as follows:

$$U_n(S^*) = \sum_{j'} CR_{mj'} \tag{3}$$

Where $\sum_j CR_{mj'}$ is the makespan in the real schedule.
m is total number of machines.
n' is the number of the jobs sequence that have not been processed yet and the job in progress on the first machine, including the newly arrived job at the disruption time t_D.
$j' = \{1, 2, ..., n'\}$ the index of n' jobs.
S^* refers to the new schedule for the partial subsequnce of jobs that have not been processed yet on the first machine at the time of disruption t_D. It should be noted that the S^* has n' number of jobs and n' depends on the disruption type, e.g., if there is new job arrived to the system then n' will be the number of not processed jobs, the job in process on the first machine and the new arrived job.
The stability measure is defined as the deviation of the completion time of each job in the baseline sequence and the new schedule. The stability performance is defined as follows:

$$I_n(S^*) = \sum_{i} \sum_{j} |CR_{ij'} - CP_{ij'}| \tag{4}$$

where $CR_{ij'}$ refers to the real completion time in the real schedule, and $CP_{ij'}$ represents the predicted completion time of a job j' on machine i according to the planned baseline solution.

The third performance measure is robustness, which is defined as the difference of the total completion time between the new schedule and the baseline one and it is given by the following equation:

$$R_n(S^*) = |\sum_{j'} CR_{mj'} - \sum_{j'} CP_{mj'}| \tag{5}$$

where $\sum_{j'} CP_{mj'}$ is the predicted value of makespan according to the planned baseline solution.
To enable a fair comparison between models, the MSR model is normalised as follows:

$$NMSR = \alpha NU_n(S^*) + \beta NI_n(S^*) + \gamma NR_n(S^*) \tag{6}$$

Where $NU_n(S^*)$, $NI_n(S^*)$ and $NR_n(S^*)$ are the normalised objectives of makespan,

stability and robustness respectively. The calculation of the first two objectives are given in (Katragjini et al., 2013) while the normalised robustness measure is explained at the end of this section.
The normalised makespan is giving by the following equation:

$$NU_n(S^*) = \frac{U_n(S^*) - Min(U_n)}{Max(U_n) - Min(U_n)} \tag{7}$$

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

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

Then the lower bound of makespan is as follows:

$$Min(U_n) = U_n(\pi_{AD}) + \sum_{j'=1}^{n'}(P_{mj'}) \tag{8}$$

To calculate $Max(U_n)$, first we determine π_{AD} and π_{BD}. For every job $j's$ in π_{BD}, the job j' starts to be executed only after the previous job is terminated in the sequence. This calculation process is explained in example in figure (1).

From figure (1) 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.
The normalised Stability is given as follows (Katragjini et al., 2013):

$$NI_n(S^*) = \frac{I_n(S^*) - Min(I_n)}{Max(I_n) - Min(I_n)} \tag{9}$$

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^*. Thus, $I_n(S^*) = \sum_{i=1}^{m}\sum_{j=1}^{n} L_{ij}$.
Where:

$$L(ij) = \begin{cases} 1 & |q_{ij}^* s - q_{ij} s| > h \\ 0 & otherwise \end{cases} \tag{10}$$

In (10), the $q_{ij}^* s$ denotes job js starting time on machine i after the rescheduling,

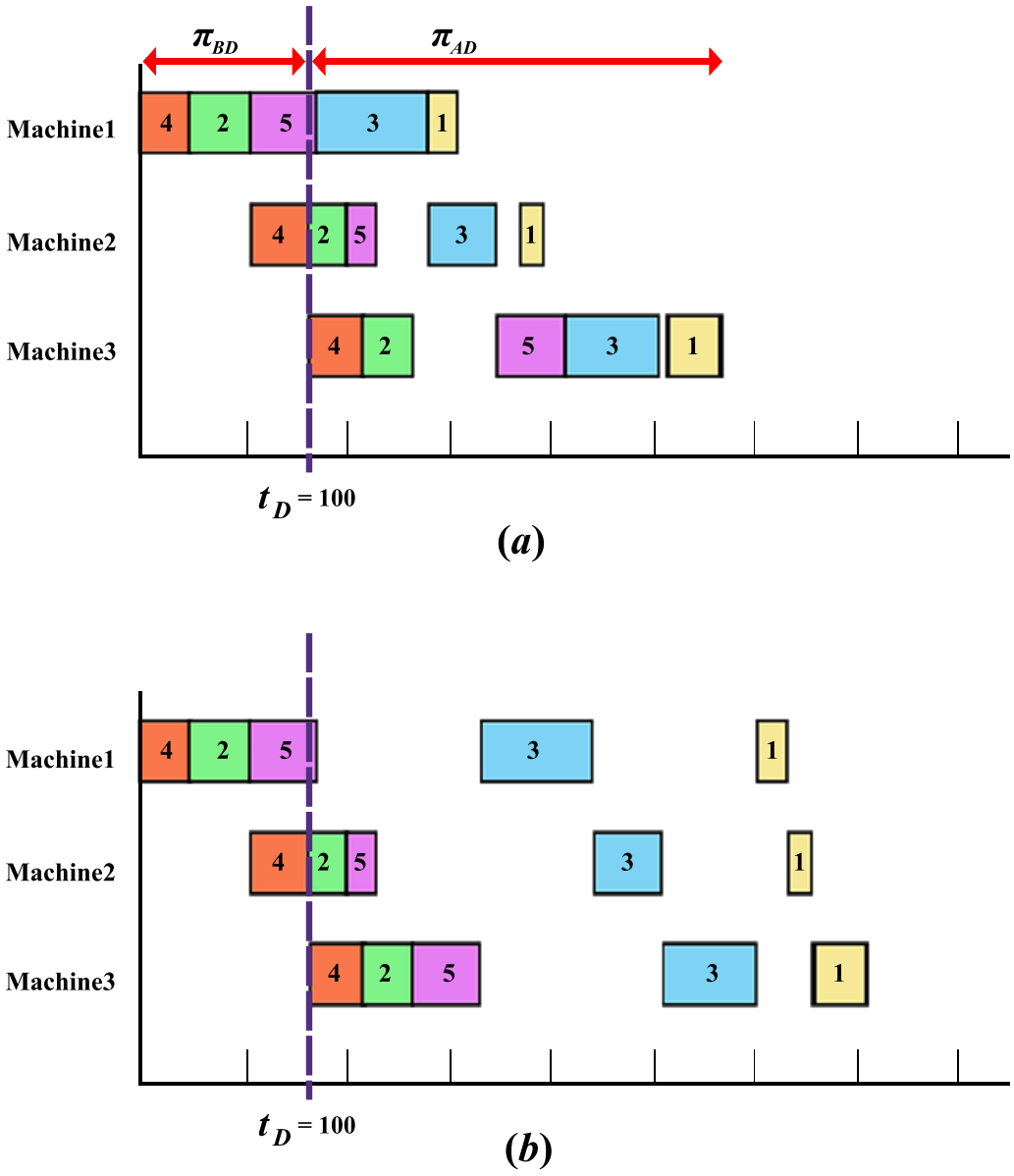

Figure 1. (a) The original PFSP, (b) Example of calculating $Max(U_n)$

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 operations 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.
$Min(I_n)$ is calculated for operations that have not been moved, thus, $Min(I_n) = 0, \forall t_D$. On the other hand, $Max(I_n)$ takes into account all operations starting times as being altered, for this $Max(I_n) = m \times n$.
The objective $NR_n(S^*)$ is calculated as follows:

$$NR_n(S^*) = \frac{R_n(S^*) - Min(R_n)}{Max(R_n) - Min(R_n)} \tag{11}$$

Where $R_n(S^*)$ represents the robustness measure after the disruption time t_D, $Max(R_n)$ is the robustness upper bound and $Min(R_n)$ is the robustness lower bound. We solve the model (1) three times to obtain these upper and lower bounds. At the first solution we set $\alpha = 1, \beta = 0, \gamma = 0$, for the second solution we set $\alpha = 0, \beta = 1, \gamma = 0$, and finally for the last solution we set $\alpha = 0, \beta = 0, \gamma = 1$. The final results are shown in 3×3 matrix form as follows:

$$\begin{array}{c} \\ U_n^* \\ I_n^* \\ R_n^* \end{array} \begin{array}{c} \begin{array}{ccc} U_n & I_n & R_n \end{array} \\ \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} \end{array} \tag{12}$$

In this matrix, the values $a_{k,l}$ where $k = 1, 2, 3$ and $l = 1, 2, 3$ represent the values of the objective function defined in (1) where the weight set (α, β, γ) is define as $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$, respectively. The values of $Min(R_n)$ and $Max(R_n)$ are then calculated as follows:

$$Min(R_n) = min\{a_{1,3}, a_{2,3}, a_{3,3}\} \tag{13}$$

$$Max(R_n) = max\{a_{1,3}, a_{2,3}, a_{3,3}\} \tag{14}$$

3. The proposed Solution method

To evaluate the performance of the MSR model for the dynamic PFSP under different real-time events, we propose a robust predictive-reactive rescheduling based BRIG technique. Figure 2 shows the application procedure of this approach in response to different real-time events. As shown in this figure, the approach starts with generating a predictive solution by using IG algorithm (Katragjini et al., 2013). All predictive solutions for different Taillard instances are stored as a benchmark, which is available in ⟨http://soa.iti.es/⟩. At the reactive step of this approach, the BRIG algorithm is applied to cope with the current real-time events. The BRIG algorithm requires an

initial solution to start the procedure, for this, the BR-NEH heuristic is used for this purpose. The BR-NEH and the BRIG are explained in the next two subsections.

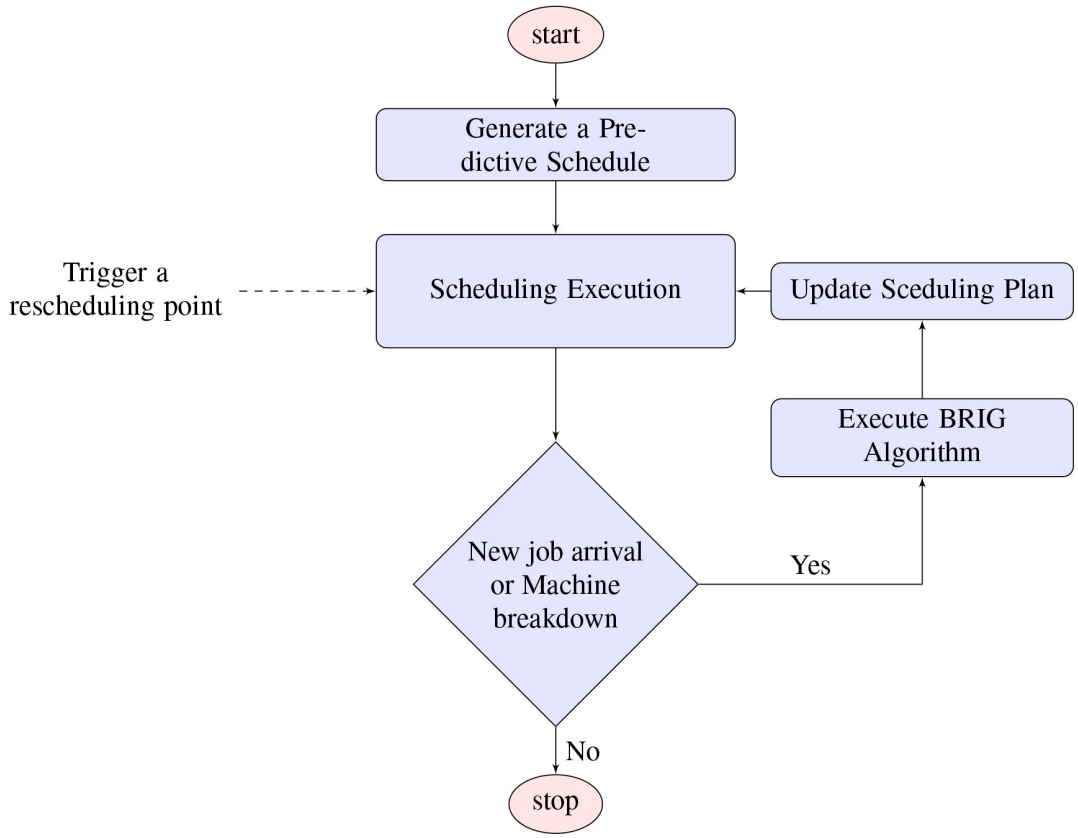

Figure 2. Predictive-Reactive approach

3.1. Generating an initial solution

In practice, generating heuristic solution instead of a uniformly random one is usually used to make the convergence of heuristic algorithms faster. However, the NEH heuristic (Nawaz, Enscore, and Ham, 1983) is commonly used to generate initial solutions for most algorithms designed to solve the PFSP. For this, the improve version of the NEH algorithm which is BR-NEH (Juan et al., 2014) is employed to obtain an initial solution for the BRIG algorithm in this paper. The BR-NEH algorithm is a combination of the classical NEH heuristic and the biased randomisation. The NEH heuristic is an iterative method that employs a sequence of jobs arranged by their total completion processing time on all the machines to construct a solution for the PFSP. It can be described by the following steps (Nawaz et al., 1983):

(1) For each job j, compute the S_j of its processing times on all machines as follows:

$$S_j = \sum p_{i\pi_j} \quad for \quad j = 1, ..., n$$

(2) Sort all jobs in descending order of S_j to form the sequence of jobs $\pi_1, ..., \pi_n$.

(3) Order the first two jobs π_1, π_2 in order to minimise the makespan of these two jobs.
(4) For each of the remaining jobs $\pi_3, ..., \pi_n$, successively: Insert the next job into the sequence and select the permutation that results minimum partial makespan among all possible insertion positions.

The biased randomisation assigns a different probability to each selected job from the sequence π_d. The BR-NEH algorithm arrange the jobs of π_d in descending order using their total completion time. Thus, the job with higher probability will be selected first and inserted into the list of the remaining jobs π_R. At this stage, the list with minimum partial makespan is selected, the procedure will continue until $\pi_d = \phi$. The discretised decreasing triangular distribution is used during the NEH construction phase to assign linearly diminishing probabilities for all jobs. The concept of biased randomisation versus the uniform one is shown in figure 3, and it is explained in detail by Juan et al. (2013). Algorithm 1 shows the main steps of BR-NEH algorithm to generate an initial solution for the BRIG algorithm.

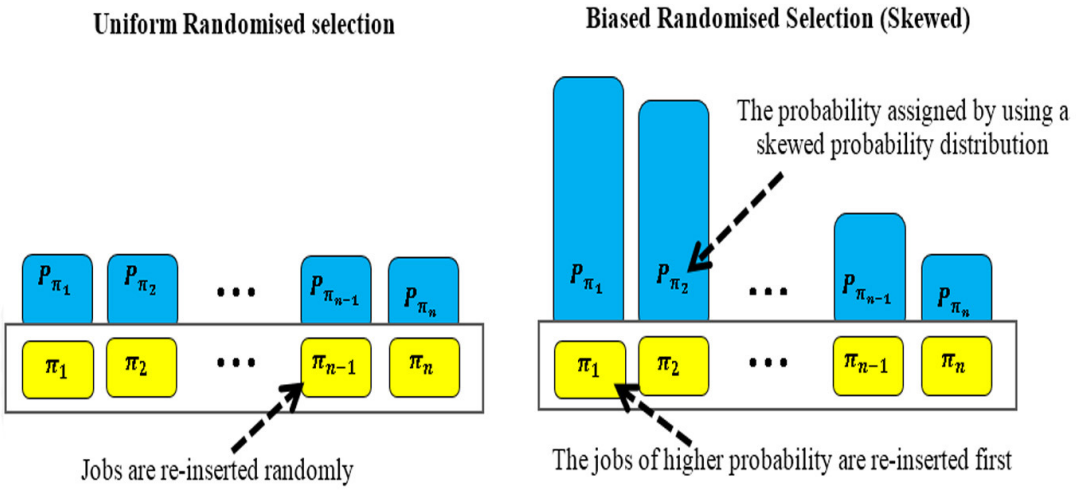

Figure 3. Biased randomness vs uniform

3.2. Hybridising IG with Biased Randomisation

This work proposed the BRIG algorithm that hybridises the IG algorithm with biased randomisation. To construct an initial solution in BRIG, we use the BR-NEH heuristic, which explained in the previous subsection. The next step is applying the hybrid BRIG algorithm, which combines biased randomisation with the IG algorithm. The application of IG algorithm for the PFSP has been proposed by Ruiz and Stützle (2007). This method has been used successfully for many types of scheduling problems since then. The main feature of the IG algorithm is its simplicity where it has very few parameters. Also, the IG algorithm has shown the best performance for different flow shop scheduling problems with different objectives. 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

Algorithm 1 BR-NEH algorithm

π' = Sort_Jobs_Using_NEH_Criterion
$Sol(\pi')$ = NEH_Algorithm(π') *// NEH solution*
$BaseSol = Sol(\pi')$
$Iter_n = 0$ *// Number of Iterations*
while $C(BaseSol \geq C(Sol(\pi'))$ and $Iter_n < n$ **do**
  $Iter_n = Iter_n + 1$
  π'' = Biased_Randomisation(π', Triangular)
  $New\_Sol(\pi'')$ = NEH_Algorithm(π'')
  if $C(New\_Sol(\pi'')) < C(BaseSol)$ **then**
    $BaseSol = New\_Sol(\pi'')$
  end if
end while
return $BaseSol$
end

reinserted into π_R and the sequence corresponding to the minimum partial makespan is chosen, this process continues until $\pi_d = \phi$. In this paper, the IG algorithm is transformed from a deterministic version into a probabilistic method by inducing biased randomness to perturbation behaviour (Juan et al., 2014). The idea of biased randomness 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 biased randomness 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 probability distribution, specifically the discretised decreasing triangular distribution as it applied for the BR-NEH algorithm, where jobs with higher probability are more likely to be inserted to the permutation list π_R than the jobs with lower probability. The discretised decreasing triangular distribution is preferred for its practicality where it provides good results and also it does not have relatively straight forward parameters to set. The asymmetric discretised decreasing triangular distribution can be generally defended as follows (Salomon, Purshouse, Giaghiozis, and Fleming, 2016):

$$f(x) = \begin{cases} \frac{2(h-x)}{h^2} & x = 0, 1, ..., h \\ 0 & x = \text{any other value.} \end{cases} \tag{15}$$

IG has one more optional step of applying a local search technique at the construction phase to improve the generated solutions. This local search step is based on the insertion neighborhood technique, which is efficient and common local search procedure for the PFSP (Ruiz and Stützle, 2007). For a permutation of jobs π, the insertion neighborhood is determined by taking in account all possible orders of pairs $r, s \in \{1, 2, ..., n\}$ of π, $r \neq s$ where the job at location r is removed and reinserted into locations s. Thus, the new list of jobs will be as follows: $\pi' = (\pi(1), ..., \pi(r-1), \pi(r+1), ..., \pi(s), \pi(r), \pi(s+1), ..., \pi(n))$ if $r < s$, or $\pi' = (\pi(1), ..., \pi(s-1), \pi(r), \pi(s), ..., \pi(r-1), \pi(r+1), ..., \pi(n))$ if $r > s$. The sequence of insertion moves I is determined as $\{(r, s) : r \neq s, 1 \leq r, s \leq n \wedge r \neq s-1, 1 \leq r \leq n, 2 \leq s \leq n\}$, also the insertion neighborhood of sequence π is defined as $V(I, \pi) = \{\pi_v : v \in I\}$. The optional local search phase is detailed in Algorithm 2 where $C(\pi)$ is the total completion time for the sequence of jobs π.

Algorithm 2 Iterative improvement of neighborhood Local search (Ruiz and Stützle, 2007)

```
improve := true;
while (improve = true) do
  improve := false;
  if i:=1 to n then
    remove a job s at random from π (without repetition)
    π' := best permutation obtained by inserting s in any possible positions of π;
    if C(π') ≤ C(π) then
      π = π'
      improve := true;
    end if
  end if
end while
return π
end
```

The next step is to decide whether to keep the incumbent solution or replace it with the new one, and to do this, we use an acceptance criterion based on the constant temperature Simulated Annealing-like criterion (Osman and Potts, 1989), which is basically calculates a constant temperature as follows:

$$Temperature = T\frac{\sum_{i=1}^{m}\sum_{j=1}^{n} p_{i\pi_j}}{m \times n \times 10} \tag{16}$$

where T is a further value to calibrate. The final proposed BRIG method is given in Algorithm 3.

Algorithm 3 BRIG Algorithm

```
Generate initial solution π';
Apply Local search to π', and put modified solution into π^s;
repeat
π^d = Destruction(π^s)
π^c = Construction(π^d)          // apply Biased_Randomisation
π^L = LocalSearch(π^c)
π^f = AcceptanceCriterion(π^s, π^L)
Until termination condition met
end
```

4. Real-time events

This paper considers the case where there are two different types of real-time events that interrupt the initial predictive schedule. These events are machine breakdowns and arrivals of new job. For every baseline BL generated in the initial predictive phase,

Katragjini et al. (2013) simulate baseline of shop floor execution by generating different real-time events randomly at time t where $0 \leq t \leq C(BL)$ and $C(BL)$ represents the makespan of the predictive baseline BL. They generates different real-time events till the end of the baseline time horison 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 real-time events generation would be unending if halted at the completion of each revised schedule. The second reason is that they aim to generate a confined benchmark of real-time events 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 real-time events 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 the results, similarly to a simulation process. The authors 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.

4.1. Machine breakdown

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

4.2. Arrival of New jobs

The scenario of a dynamic problem is considered by generating arrivals of new jobs randomly in the scheduling system. In other words, there is a probability of generating one new job arrival at every point t where $0 \leq t \leq C(BL)$. 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.

4.3. Combination between real-time events

It may have a machine breakdown and a new job arrival simultaneously during the time horison. At the beginning of every real-time events, reactive actions are used to cop the real-time events and to preserve a balance between schedule performance, stability and robustness. All the real-time events are saved as a rescheduling events benchmark, which can be found on ⟨http://soa.iti.es/⟩. It should notice that there does not exist any similar benchmark of real-time events in the literature, even for a single type of real-time event.

5. Computational results

5.1. Experimental design

The algorithms described in this paper are 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. In this section we present the results of numerical experiments designed for the PFSP in the presence of different real-time events including; machines breakdowns and new jobs arrival. The aim of this section can be explained as follows:

(1) We assess the efficiency of the presented MSR model for this problem by comparing this model against two other models, which are; the bi-objective model ($bi - obj$) that consider the makespan and instability objectives (Katragjini et al., 2013), and the single objective makespan model ($Utility$). All models have been applied for the PFSP under different real-time events using the predictive-reactive based BRIG approach.
(2) To show the efficiency of the BRIG algorithm, we compare the predictive-reactive based BRIG approach against the predictive-reactive based PSO approach. both approaches are applied for the PFSP under different real-time events using the MSR model.

In order to reach these aims, we conduct a set of comprehensive numerical and statistical experiments for the benchmark set given by Katragjini et al. (2013). This benchmark set is based on 120 Taillard's instances (Taillard, 1993) where these instances are grouped into 12 sets of 10 problems, covering instance sizes (jobs and machine numbers) ranging from 20×5 to 500×20 ($jobs \times machines$). For each tested instance, five independent replications were run and the average is calculated for more reliable results. In this work, we apply the practical weight sensitivity technique given by Jones (2011) to test the versatility of the multi-objective MSR model in producing differing Pareto efficient solutions with respect to the three objectives. The Jones algorithm (Jones, 2011) produced thirteen distinct (α, β, γ) weight sets, each representing different levels of relative importance of the objectives in the MSR model. The parameters used in this algorithm are as follows: $TMax = 1$ which is defined to control the number of weights to be varied simultaneously in the the practical weight sensitivity method. Also, $MaxLevel = 2$, this parameter is used to control the maximum number of bi-sections of the line of direction between the initial estimate and maximum level. Finally, a sequential weight starting solution is set to be one. The unity weights are applied to obtain the normalised model (6) as detailed in section 2. 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 (1):

Table 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

The MSR model is compared against the $bi-obj$ model (Katragjini et al., 2013) and the $Utility$ model. This comparison is used to show the efficiency of the MSR model in providing better quality solution. All models are solved for the same aforementioned benchmark using the predictive-reactive based BRIG. For the $bi-obj$ model, the only the first α element of weights sets W_1 to W_{10} are used to evaluate this model. These elements are listed as follows:

$$\alpha = 0.333, 0.666, 0.499, 0.416, 0.166, 0.002, 0.166, 0.166, 0.499, 0.416$$

We first compare the MSR and bi-obj models for 10 different weights that given in Table 1. Then the weight corresponding to lower RPD values will be selected to compare with the solution obtained by the $Utility$ model. It should be noted that for the BRIG algorithm, the the parameter T in equation (15) is given the value 0.5 (Ruiz and Stützle, 2007). Once the best solution is found, the average relative percentage deviation (RPD) is calculated over 10 of Taillard problems of the same size ($n \times m$) and is given as follows:

$$RPD = \frac{M - Best_{Sol}}{Best_{Sol}} \times 100 \tag{17}$$

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 10's Taillard's instances that have the same number of jobs and machines. Table 2 illustrates the RPD for each instance corresponding to the weight set (α, β, γ). In this Table, the solution obtained from using the MSR model is represented as MSR. Similarly, the solution obtained from the $bi-obj$ model is termed as $bi-obj$. From Table 2, 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, the RPD increases in the bi-objective model, while it decreases in the MSR model.

Table 2. RPD for MSR and $bi-obj$ models

	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 revealed that the RPDs corresponding to the model MSR and the weight set W_8 have lower values than the other RPDs corresponding to other weight sets. In addition, higher weight for a robust term of the MSR model leads to less values of RPD, which produce better solution. Figure 4, shows the RPD values for all models (MSR, $bi-obj$ and $Utility$ models) with the weight W_8 where $P1, ..., P12$ represent the problems of sizes $20\times5, ..., 500\times20$, respectively. From this figure, it is obvious that the solution corresponding to the MSR model shows lower values of RPD in comparing with other solutions.

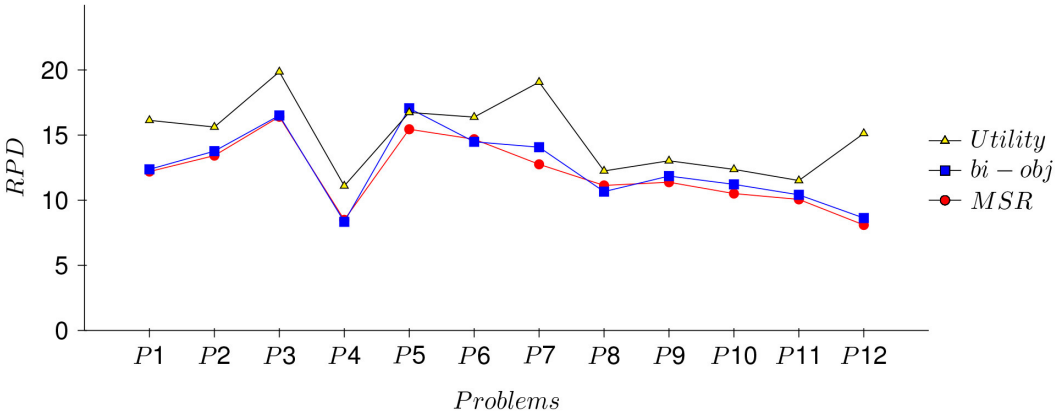

Figure 4. RPD for MSR, $bi-obj$ and $Utility$ models with weight W_8

The results are statistically tested by the single factor mean of ANOVA (Analysis of Variance). This statistical procedure used to describe the impact of the proposed 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-values and the F-ratios are shown in Table 3. 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 3. Analysis of Variance.

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	2.000	29.247	3.783	0.033	2.471
Within Groups	255.139	33.000	7.731			
Total	391.401	35				

Because the F-test in ANOVA allows the test to accept or reject the null hypothesis only, but it does not assign which one of the groups has different mean values, the Tukey confidence 95% intervals are applied to assign the group that is statistically significant difference in the mean of RPD values, Figure (5) show that model (1) is significantly different comparing with the $Utility$ model, while there is no significance difference between the $bi-obj$ and the $Utility$ Models.

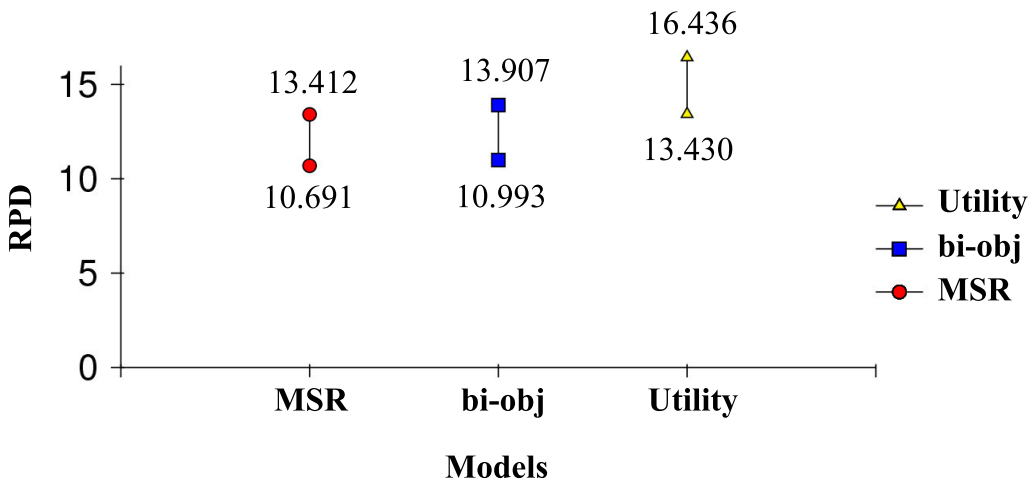

Figure 5. 95% Tukey confidence intervals for all models

5.2. Comparison between BRIG and PSO algorithms

The PSO algorithm is applied for the dynamic PFSP in the presence of machines breakdowns and new jobs arrivals in (Al-Behadili et al., 2017). The authors introduced the MSR model (1) and used a predictive-reactive based PSO approach. In this subsection, our proposed BRIG heuristic algorithm is compared against the PSO metaheuristic 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 weight $W_8 = (0.166, 0.166, 0.666)$. 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 $n \times m$. In this experiment, each algorithm were performed five runs independently. For both methods, the average of each instance solutions values is calculated. Figure 6 records the average RPD values obtained by using the BRIG and PSO algorithms. From this figure, it is clear that the RPD corresponding to BRIG is lower than the RPD for PSO algorithm.

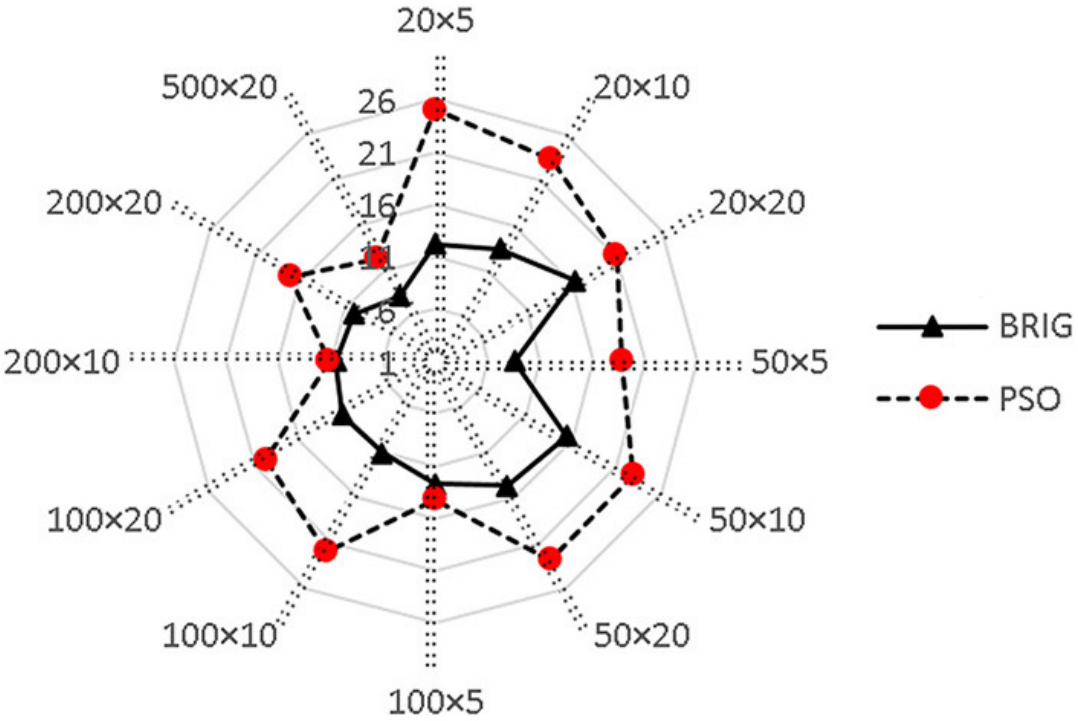

Figure 6. *RPD* for BRIG and PSO algorithms

On the other hand, figure 7 clarifies the significant difference between the PSO and BRIG algorithms. This figure shows there is a significance difference between these algorithms.

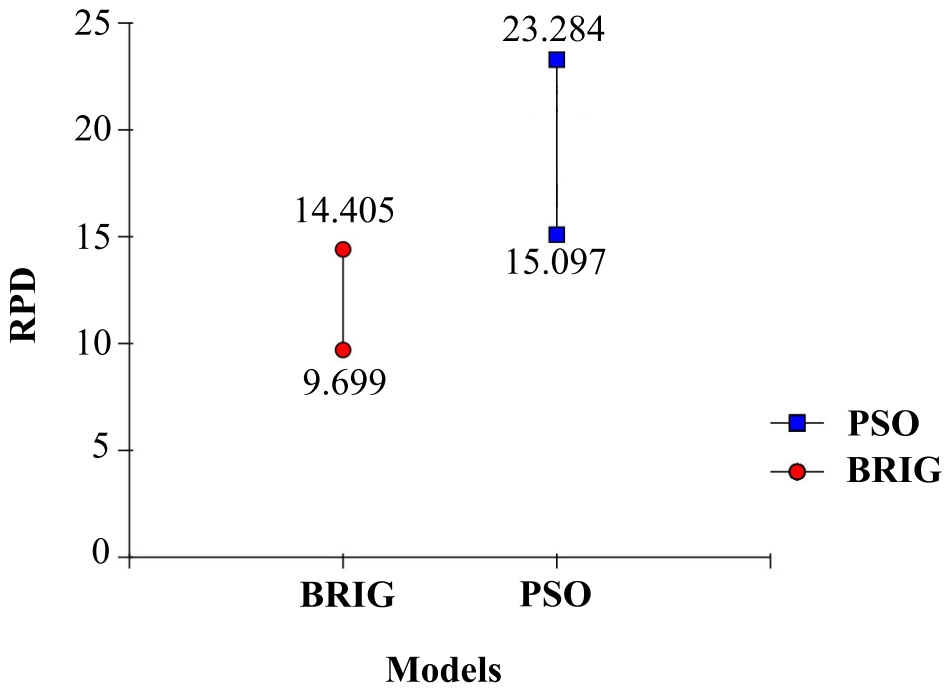

Figure 7. 99% Tukey confidence intervals for BRIG and PSO algorithms with weight W_8

Finally, Table 4 indicates that the computational time required by BRIG is less than the time required by the PSO algorithm.

Table 4. Computational time of BRIG and PSO algorithms in seconds.

Problem	BRIG(S.)	PSO(S.)
20×5	0.0025	0.0446
20×10	0.005	0.0411
20×20	0.0074	0.1056
50×5	0.0188	0.2706
50×10	0.0302	0.3644
50×20	0.0579	0.4888
100×5	0.1053	1.3500
100×10	0.2989	1.9237
100×20	0.4939	2.1264
200×10	3.7169	10.7286
200×20	8.1249	16.051
500×20	177.6876	292.2008

In this table, the computational time is calculated from the average of five independent runs in seconds. Thus, the BRIG algorithm outperforms the PSO algorithm as in obtains better results in less computational time.

6. Conclusion

This paper has discussed the dynamic PFSP in the presence of different types of real-time events, in this problem, the different real-time events are new jobs arrival and machine breakdowns. A a multi-objective model that aims to minimise the makespan, stability and robustness simultaneously is proposed for this problem. Also, we introduced a solution method based on a predictive-reactive approach that uses the probabilistic BRIG algorithm in the reactive stage. The proposed BRIG algorithm hybridises the IG algorithm with the biased randomisation technique and use the local search technique implicitly, so that the algorithm explores a greater portion of the solution space. This approach provides the ability of generating and rating a huge number of local optima during a short amount of computational time. A number of numerical experiments have been carried out to test the performance of the introduced multi-objective model and the BRIG algorithm. The results show that the proposed model (1) 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 and stable solutions. Moreover, the BRIG algorithm has been tested versus PSO algorithm that has been already applied for the dynamic PFSP in the presence of machine breakdown and new job arrival (Al-Behadili et al., 2017). This comparative study shows that the BRIG algorithm outperforms the PSO algorithm, also the computational time used by the BRIG algorithm to reach good quality solution is much less than the time consumed by the PSO algorithm. This shows the fast convergence of the BRIG comparing with the well-known evolutionary PSO algorithm. In respect to this paper, there are some main limitations, which can be described as follows: First, the application of the predictive-reactive approach with the BRIG algorithm to the dynamic PFSP in the presence of different two types of real-time events is studied. For this, research work is required to find out potential applications of the BRIG to other dynamic scheduling problems in the presence of different real-time events. Secondly, in the numerical experiments, we have used a multi-objective optimisation model and tested different weights to determine a fair solution from this model. However, for future work, more research in the field of multi-objective and goal programming are required for robust optimisation scheduling problems. Another future study is to test the competitiveness of the proposed approach against other state-of-the-art multi objective algorithms in the literature such as NSGA-III (Jain and Deb, 2014) and MOEA/D (Li and Zhang, 2009). Another direction is considering the PFSP in the presence of different types of real-time events including stochastic processing times and proposed different approaches to solve this stochastic problem. Other relevant extension of this research is to extend the application of the BRIG to solve other scheduling problems in the presence of different types of real-time events, e.g. the scheduling problem of job shop or fixable flow shop scheduling problems.

References

Al-Behadili M, Ouelhadj D, Jones D (2017) Multi-objective Particle Swarm Optimisation for Robust Dynamic Scheduling in a Permutation Flow Shop, Springer International Publishing, Cham, pp 498–507. , URL https://doi.org/10.1007/978-3-319-53480-0_49

Arroyo JEC, Leung JYT, Tavares RG (2019) An iterated greedy algorithm for total flow time minimization in unrelated parallel batch machines with unequal job release times. Engineer-

ing Applications of Artificial Intelligence 77:239 – 254, , URL http://www.sciencedirect.com/science/article/pii/S0952197618302240

Aytug H, A Lawley M, 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(1):86 – 110, , URL http://www.sciencedirect.com/science/article/pii/S0377221703005307, iEPM: Focus on Scheduling

Bertsimas D, B Brown D, Caramanis C (2011) Theory and applications of robust optimization. SIAM Review 53(3):464–501, , URL https://doi.org/10.1137/080734510, https://doi.org/10.1137/080734510

Collet P, Rennard J (2007) Stochastic optimization algorithms. CoRR abs/0704.3780, URL http://arxiv.org/abs/0704.3780

Cowling P, Johansson M (2002) Using real time information for effective dynamic scheduling. European Journal of Operational Research 139(2):230 – 244, , URL http://www.sciencedirect.com/science/article/pii/S0377221701003551, eURO XVI: O.R. for Innovation and Quality of Life

Cowling PI, Ouelhadj D, Petrovic S (2003) A multi-agent architecture for dynamic scheduling of steel hot rolling. Journal of Intelligent Manufacturing 14(5):457–470, , URL https://doi.org/10.1023/A:1025701325275

Graham R, Lawler E, Lenstra J, Kan A (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Hammer P, Johnson E, Korte B (eds) Discrete Optimization II, Annals of Discrete Mathematics, vol 5, Elsevier, pp 287 – 326, , URL http://www.sciencedirect.com/science/article/pii/S016750600870356X

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

Jain H, Deb K (2014) An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach. IEEE Transactions on Evolutionary Computation 18(4):602–622,

Jones D (2011) A practical weight sensitivity algorithm for goal and multiple objective programming. European Journal of Operational Research 213(1):238 – 245, , URL http://www.sciencedirect.com/science/article/pii/S0377221711002232

Juan AA, Faulin J, Ferrer A, Lourenço HR, Barrios B (2013) Mirha: multi-start biased randomization of heuristics with adaptive local search for solving non-smooth routing problems. TOP 21(1):109–132, , URL https://doi.org/10.1007/s11750-011-0245-1

Juan AA, Lourenço HR, Mateo M, Luo R, Castella Q (2014) Using iterated local search for solving the flow-shop problem: Parallelization, parametrization, and randomization issues. International Transactions in Operational Research 21(1):103–126, , URL http://dx.doi.org/10.1111/itor.12028

Katragjini K, Vallada E, Ruiz R (2013) Flow shop rescheduling under different types of disruption. International Journal of Production Research 51(3):780–797, , URL http://dx.doi.org/10.1080/00207543.2012.666856, http://dx.doi.org/10.1080/00207543.2012.666856

Kotz S, Van Dorp J (2004) Beyond Beta: Other Continuous Families of Distributions with Bounded Support and Applications. World Scientific, URL https://books.google.co.id/books?id=NpK1xSB_SE4C

Laporte G (2009) Fifty years of vehicle routing. Transportation Science 43(4):408–416, , URL https://doi.org/10.1287/trsc.1090.0301, https://doi.org/10.1287/trsc.1090.0301

Li H, Zhang Q (2009) Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. IEEE Transactions on Evolutionary Computation 13(2):284–302,

Liu F, Wang S, Y H, Yue Y (2017) On the robust and stable flowshop scheduling under stochastic and dynamic disruptions. IEEE Transactions on Engineering Management 64(4):539–553,

Luo W, Luo T, Goebel R, Lin G (2018) Rescheduling due to machine disruption to minimize the total weighted completion time. Journal of Scheduling 21(5):565–578, , URL https://doi.org/10.1007/s10951-018-0575-z

Nawaz M, Enscore EE, Ham I (1983) A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11(1):91 – 95, , URL http://www.sciencedirect.com/science/article/pii/0305048383900889

Osman I, Potts C (1989) Simulated annealing for permutation flow-shop scheduling. Omega 17(6):551 – 557, , URL http://www.sciencedirect.com/science/article/pii/0305048389900595

Ouelhadj D, Petrovic S (2008) A survey of dynamic scheduling in manufacturing systems. Journal of Scheduling 12(4):417, , URL https://doi.org/10.1007/s10951-008-0090-8

Pan QK, Ruiz R, Alfaro-Fernández P (2017) Iterated search methods for earliness and tardiness minimization in hybrid flowshops with due windows. Computers Operations Research 80(Supplement C):50 – 60, , URL http://www.sciencedirect.com/science/article/pii/S0305054816302969

Pinedo M (2016) Scheduling: Theory, Algorithms, and Systems, 5th edn. Springer International Publishing, URL https://books.google.co.uk/books?id=M4-RCwAAQBAJ

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(1):84 – 92, , URL http://www.sciencedirect.com/science/article/pii/S0278612513000332

Ruiz R, Stützle T (2007) A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 177(3):2033 – 2049, , URL http://www.sciencedirect.com/science/article/pii/S0377221705008507

Ruiz R, Stützle T (2008) An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. European Journal of Operational Research 187(3):1143 – 1159, , URL http://www.sciencedirect.com/science/article/pii/S0377221706008277

Ruiz R, Maroto C, Alcaraz J (2006) Two new robust genetic algorithms for the flowshop scheduling problem. Omega 34(5):461 – 476, , URL http://www.sciencedirect.com/science/article/pii/S0305048305000174

Ruiz R, Pan QK, Naderi B (2019) Iterated greedy methods for the distributed permutation flowshop scheduling problem. Omega 83:213 – 222, , URL http://www.sciencedirect.com/science/article/pii/S0305048317306990

Sabuncuoglu I, Goren S (2009) Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research. International Journal of Computer Integrated Manufacturing 22(2):138–157, , URL http://dx.doi.org/10.1080/09511920802209033, http://dx.doi.org/10.1080/09511920802209033

Salomon S, Purshouse RC, Giaghiozis I, Fleming PJ (2016) A toolkit for generating scalable stochastic multiobjective test problems. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016, ACM, New York, NY, USA, GECCO '16, pp 597–604, , URL http://doi.acm.org/10.1145/2908812.2908873

Suwa H, Sandoh H (2012) Online Scheduling in Manufacturing: A Cumulative Delay Approach. SpringerLink : Bücher, Springer London, URL https://books.google.co.uk/books?id=N04JjAaKXIwC

Taillard E (1993) Benchmarks for basic scheduling problems. European Journal of Operational Research 64(2):278 – 285, , URL http://www.sciencedirect.com/science/article/pii/037722179390182M, project Management and Scheduling

Vieira GE, Herrmann JW, Lin E (2003) Rescheduling manufacturing systems: A framework of strategies, policies, and methods. Journal of Scheduling 6(1):39–62, , URL https://doi.org/10.1023/A:1022235519958

Wojakowski P, Warołek D (2014) The classification of scheduling problems under production uncertainty. Research in Logistics & Production 4(3):245 – 255