Multi-objective Particle Swarm Optimisation for Robust Dynamic Scheduling in a Permutation Flow Shop

Conference Paper *in* Advances in Intelligent Systems and Computing · February 2017

DOI: 10.1007/978-3-319-53480-0_49

CITATIONS
2

READS
46

3 authors:

Mohanad Al-Behadili
University of Basrah
12 PUBLICATIONS **4** CITATIONS
SEE PROFILE

Djamila Ouelhadj
University of Portsmouth
52 PUBLICATIONS **1,308** CITATIONS
SEE PROFILE

Dylan F Jones
University of Portsmouth
81 PUBLICATIONS **2,224** CITATIONS
SEE PROFILE

Multi-objective Particle Swarm Optimisation for Robust Dynamic Scheduling in a Permutation Flow Shop

Mohanad AL-Behadili[1,2], Djamila Ouelhadj[2], and Dylan Jones[2]

[1] University of Basra, Basra, Iraq
mohanad.saad@uobasrah.edu.iq
[2] Department of Mathematics, University of Portsmouth, Lion Gate Building, Portsmouth PO1 3HF, UK
{djamila.ouelhadj,dylan.jones}@port.ac.uk

Abstract. This paper proposes a multi-objective optimisation model and particle swarm optimisation solution method for the robust dynamic scheduling of permutation flow shop in the presence of uncertainties. The proposed optimisation model for robust scheduling considers utility, stability and robustness measures to generate robust schedules that minimise the effect of different real-time events on the planned schedule. The proposed solution method is based on a predictive-reactive approach that uses particle swarm optimisation to generate robust schedules in the presence of real-time events. The evaluation of both the optimisation model and solution method are conducted considering different types of disruptions including machine breakdown and new job arrival. The obtained results showed that the proposed model and solution method gives better results than a bi-objective model that considers only utility and stability measures [1] and the classical makespan model.

1 Introduction

Scheduling deals with the assignment of a set of jobs to a set of machines in a reasonable amount of time in order to optimise one or more objectives [2]. In the flow shop scheduling problem (FSP) all jobs have to follow the same route. FSP that do not allow sequence changes between machines is called permutation flow shop scheduling problem (PFSP). In the PFSP, the minimisation of the completion time or makespan is the most common criterion that has been studied in the literature. It has been proven that, the PFSP is an NP-hard problem [3]. The most traditional classifications of the FSP are static, stochastic and dynamic scheduling environments [4]. In dynamic scheduling environments, random disruptions may interrupt the system, which could change the scheduled production plans. Noticeable studies of dynamic scheduling are given by [5], [6] and [7], while a comprehensive survey is given by [8]. Most rescheduling strategies in the literature to handle the real-time events consider the following two goals [9]; make the schedule feasible again, and improve the efficiency of

the schedule in order to minimise the deviation of the current solution from the pre-planned one. To minimise the deviation between the performance measure values of the baseline and realised schedules, [10], [11] and [12] propose a bi-objective function and consider the robustness measure as a linear combination of utility and stability measures. Rahmani and Heydari [13] propose a multi-objective optimisation model that considers utility, stability and robustness for FSP with unexpected new job arrival and uncertain processing times. The model was designed for the case of n-jobs and 2-machines. An exact method was applied to obtain the solution for small instances. Most of approaches designed for rescheduling are mainly focused on machine breakdowns, new job arrivals, etc. However, the majority of the existing work addresses these disruptions only independently [14], [15], [16], [17], [18] and [19]. Katragjini et al. [1] introduce a novel benchmark for PFSP that considers different types of disruptions during the time horizon. The authors implement a predictive-reactive approach for PFSP with three different random disruptions that interrupt the initial schedule simultaneously. These disruptions are machine breakdown, new job arrival and job ready time variations. To keep the solution efficient and stable, Katragjini et al. [1] proposed a bi-objective optimisation model, which takes into account the makespan and instability measures. Particle swarm optimisation (PSO) is a stochastic optimisation technique that was introduced by Kennedy et al. [20]. The PSO algorithm has many advantages making it more preferable to be proposed for dynamic combinatorial optimisation problems. Some advantages are the dynamic nature of the algorithm and its simplicity of implementation [21], [22] and [23]. Other features are the use of self-information, individual best information and global best information to generate effective and optimal results, as well as the fast convergence speed of the swarm [24]. The main advantage of the PSO algorithm is that, it requires fewer parameters to be adjusted in comparison to other meta-heuristic methods [25]. The PSO algorithm was applied successfully to PFSP for example see in [26] and [27]. The contribution of this paper is to introduce a multi-objective optimisation model that considers utility, stability and robustness measures for robust dynamic PFSP under different types of real-time events and to apply an efficient predictive-reactive approach with PSO algorithm. The remainder of the paper is structured as follows; in section 2, the optimisation model for robust scheduling is presented. In section 3, the proposed predictive-reactive based PSO algorithm is developed. Section 4 shows numerical experiments and results that illustrate the performance of the proposed methodology, with a discussion of its results. Finally, conclusions and future work are presented in section 5.

2 Multi-objective optimisation model

We propose a new multi-objective optimisation model based on the robust scheduling model introduced by [11] and [13]. This model has been extended for the case of n jobs and m machines for PFSP, considering three important measures: makespan measure (utility), stability and robustness. The new model

for PFSP is given in (1).

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

Where S^* refers to the new schedule after the time of disruption t_D and $\alpha + \beta + \gamma = 1$.
$U_n(S^*) = \sum_{\acute{j}} CR_{m\acute{j}}$ the real makespan in real scheduling.
$I_n(S^*) = \sum_i \sum_{\acute{j}} |CR_{i\acute{j}} - CP_{i\acute{j}}|$ the stability measure where $CR_{i\acute{j}}$ is the real completion time and $CP_{i\acute{j}}$ is the predicted completion time of job $\acute{j}$ on machine i according to the initial schedule.
$R_n(S^*) = |\sum_{\acute{j}} CR_{m\acute{j}} - \sum_{\acute{j}} CP_{m\acute{j}}|$ the robustness measure where $\sum_{\acute{j}} CP_{m\acute{j}}$ is the predictive makespan according to the initial schedule. n the number of jobs, m the number of machines, i index for machines $\{1, 2, ..., n\}$, $\acute{j}$ index of jobs that have not been executed on any machine yet and the newly arrived job at the time of disruption. We define the partial fixed sequence including the jobs that have already been processed or are in progress on the first machine before the time of disruption by π_{BD}. Similarly, the permutable subsequence containing the jobs after the time of disruption whose succession order can be modified is denote by π_{AD}. The three objective functions of model (1) are normalised to enable a reasonable comparison with the bi-objective model of [1] and the classical single objective makespan model. The normalised objective function (NMSR) in equation (1) is as follows:

$$NMSR = \alpha NU_n(S^*) + \beta NI_n(S^*) + \gamma NR_n(S^*) \tag{2}$$

Where $NU_n(S^*)$, $NI_n(S^*)$ and $NR_n(S^*)$ represent the normalised makespan, stability and robustness respectively. The calculations of normalised makespan $NU_n(S^*)$ and normalised stability $NI_n(S^*)$ are given in details in [1], while the normalised robustness function is defined as follows:

$$NR_n(S^*) = \frac{R_n(S^*) - Min(R_n)}{Max(R_n) - Min(R_n)} \tag{3}$$

Where $R_n(S^*)$ is the robustness after the time of disruption t_D. $Max(R_n)$ and $Min(R_n)$ are the upper and lower robustness bound, respectively. To obtain these bounds, we solve the problem three times where $\alpha = 1, \beta = 0, \gamma = 0$ for the first solution, $\alpha = 0, \beta = 1, \gamma = 0$ for the second solution and $\alpha = 0, \beta = 0, \gamma = 1$ for the last solution. The results are then given in the following 3×3 matrix form:

$$\begin{array}{c} \\ U_n^* \\ I_n^* \\ R_n^* \end{array} \begin{array}{c} \begin{array}{ccc} U_n & I_n & R_n \end{array} \\ \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} \end{array} \tag{4}$$

The values of $Max(R_n)$ and $Min(R_n)$ are then calculated as follows: $Max(R_n) = max\{a_{1,3}, a_{2,3}, a_{3,3}\}$ and $Min(R_n) = min\{a_{1,3}, a_{2,3}, a_{3,3}\}$.

3 Proposed solution methods

3.1 A robust predictive-reactive rescheduling approach

In this paper we consider two different real-time events that may occur at the same time or separately during the time horizon. These events are machine breakdown and new job arrival, also the number of disruptions could vary for each instance. A predictive reactive approach with an evolutionary PSO algorithm is applied to deal with large size PFSP in the presence of real-time. The predictive reactive approach starts with a robust predictive solution and then it applies rescheduling at the time of disruption. The rescheduling phase uses the PSO algorithm for the partial subsequence of jobs π_{AD} (the sequence of jobs that have not been executed on the first machine yet after the time of disruption) with the goal to minimise the partial makespan.

3.2 Particle Swarm Optimisation (PSO)

PSO is a population-based evolutionary computational method [20]. The algorithm simulates a social behaviour such as swarm bird migration. The basic elements of the PSO algorithm are summarised below where $j = \{1, 2, ..., n\}$, $i = \{1, 2, ..., \rho\}$, ρ is the number of particles and n is the number of jobs:
Particle: The i^{th} particle X_i^t in the swarm at iteration t is defined as $X_i^t = \{x_{i1}^t, x_{i2}^t, ..., x_{in}^t\}$ where x_{ij}^t is the position value of particle i of job j at iteration t.
Particle velocity: The i^{th} particle velocity V_i^t at iteration t is defined as $V_i^t = \{v_{i1}^t, v_{i2}^t, ..., v_{in}^t\}$ where v_{ij}^t is the velocity of particle i of job j at iteration t.
Population: The set consisting of ρ particles in the swarm at iteration t is called a population po^t where $po^t = \{X_1^t, X_2^t, ..., X_\rho^t\}$.
Permutation: The permutation of job sequence implied by the particle X_i^t is defined as $\pi_i^t = \{\pi_{i1}^t, \pi_{i2}^t, ..., \pi_{in}^t\}$ 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 $P_i^t = \{p_{i1}^t, p_{i2}^t, ..., p_{in}^t\}$ 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} instead of $f(\pi_i^t)$.
Global Best: The global best among all the swarm of particles achieved so far is called global best G^t. It is defined as $G^t = \{g_1^t, g_2^t, ..., g_n^t\}$ 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 is terminated after a maximum number of iterations.

PSO particles are used to solve the PFSP as follows; for n jobs in the problem each particle $X_i^t = \{x_{i1}^t, x_{i2}^t, ..., x_{in}^t\}$ corresponds to a continuous position value. The vector of position values is transformed to the permutation of jobs according to the SPV rule [28]. The initial population of particles generated randomly and the initial position values for the particle are generated randomly as follows: $x_{ij}^0 = x_{min} + r_1(x_{max} - x_{min})$ where $[x_{min}, x_{max}] = [0, 4]$ and $r_1 \in (0, 1)$ is a uniform random number. Initial velocities are established similarly as follows: $v_{ij}^0 = v_{min} + r_2(v_{max} - v_{min})$ where $[v_{min}, v_{max}] = [-4, 4]$ and $r_2 \in (0, 1)$ is a uniform random number. The fitness function is given as $f(\pi_i^t) = NMSR$ and it is rewritten as f_i^t in short. The steps of the PSO algorithm are summarised as below:

Algorithm 1 Particle Swarm Optimisation Algorithm

1. Set initial iteration $t = 0$ and $\rho = 2 \times n$.
2. Generate ρ 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 ρ 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.

4 Computational results

We have conducted extensive experiments to evaluate the performance of the proposed optimisation model and solution methods. The proposed methodology has been implemented in Java on Windows7 64bit. An Intel Core i5, 2.30 GHz processor and 6GB of RAM was used. In PSO algorithm the population size is set to $2 \times n$, the initial inertia weight is set up to $w^0 = 0.9$ and no less than 0.4, for w the decrement factor β_{PSO} is taken as 0.975; the acceleration coefficients are set to $c_1 = c_2 = 2$ [28]. The computational experiments were carried out on the Taillard benchmark instances [29], which are grouped in 12 sets of 10 instances, ranging from 20 jobs and 5 machines to 500 jobs and 20 machines. All instances have been run independently ten times, and the average is computed. To evaluate the performance of the proposed model and the solution method, thirteen different set of weights (α, β, γ) have been chosen, these sets represent the relative importance of each objective in model (1). The selection of these weights are based on the practical weight sensitivity algorithm detailed in [30]. Three weights are used to normalise the model in equation (1) as described in section 2. These weights are as follows; $(0.999, 0.001, 0.001)$, $(0.001, 0.999, 0.001)$, $(0.001, 0.001, 0.999)$. The remaining 10 set of weights are examined for this experiment. These sets are arranged as follows: $W_1 = (0.333, 0.0.333, 0.333)$, $W_2 = (0.666, 0.166, 166)$, $W_3 = (0.499, 0.499, 0.002)$, $W_4 = (0.416, 0.416, 0.166)$, $W_5 = (0.166, 0.666, 0.166)$, $W_6 = (0.002, 0.499, 0.499)$, $W_7 = (0.166, 0.416, 0.416)$, $W_8 = (0.166, 0.166, 0.666)$, $W_9 = (0.499, 0.002, 0.499)$, $W_{10} = (0.416, 0.166, 0.416)$. On the other hand, the bi-objective model of [1] is examined for the first components of W_1 to W_{10}. These components are as follows:
$\alpha = 0.333, 0.666, 0.499, 0.416, 0.166, 0.002, 0.166, 0.166, 0.499, 0.416$.
Once the best value is found, we calculate the average relative percentage deviation (RPD) over 10 of Taillard instances with the same number of jobs and machines as follows:

$$RPD = \frac{M - Best_{Sol}}{Best_{Sol}} \times 100 \tag{5}$$

where M is the solution obtained by the proposed model and PSO algorithm. $Best_{Sol}$ is the average of 10 Taillard instances of lower bound solution from the same size. Table 1 shows the RPD for some Taillard instances according to the weights where **MSR** represents the solution obtained from our proposed model, while **bi-obj** is the solution obtained by bi-objective model [1].

In this table **Ta** represents the instance of n jobs and m machines. According to table 1, 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 on table 1, the weights W_6, W_7 and W_8 produce better solution in comparison with other weights. Furthermore, the solution corresponding to the weight W_8 has the lowest RPD values, this shows that giving more priority to the robustness measure produces better solutions. For this reason, the weight W_8 is selected in this experiment. Figure 1, shows

Table 1. RPD for MSR and bi-obj models.

	Ta	20×5	50×10	100×10	200×20	500×20	**Total**
W_1	**MSR**	29.446	22.872	23.259	22.559	17.737	**115.873**
	bi-obj	34.345	22.311	30.075	22.860	20.764	**130.355**
W_2	**MSR**	29.381	22.548	26.133	22.734	17.765	**118.561**
	bi-obj	31.384	22.241	30.012	21.896	23.592	**129.125**
W_3	**MSR**	29.062	23.539	26.151	22.518	17.960	**119.230**
	bi-obj	31.608	23.612	31.603	21.039	22.355	**130.217**
W_4	**MSR**	29.512	22.775	23.805	22.973	17.690	**116.755**
	bi-obj	31.945	24.781	32.556	21.367	20.482	**131.131**
W_5	**MSR**	30.796	23.369	24.082	22.600	17.440	**118.287**
	bi-obj	32.152	23.562	31.964	24.103	22.382	**134.163**
W_6	**MSR**	26.871	21.771	21.687	19.088	12.117	**101.534**
	bi-obj	31.278	25.360	34.158	23.554	20.359	**134.709**
W_7	**MSR**	28.163	22.358	20.920	19.082	13.794	**104.317**
	bi-obj	32.174	22.982	33.242	24.001	23.923	**136.322**
W_8	**MSR**	26.609	21.461	20.831	18.843	12.277	**100.021**
	bi-obj	31.769	23.375	32.380	23.909	24.521	**135.954**
W_9	**MSR**	29.422	22.184	23.648	22.889	17.177	**115.320**
	bi-obj	32.807	22.308	32.409	20.427	22.800	**130.751**
W_{10}	**MSR**	29.798	22.451	24.007	20.303	17.031	**113.590**
	bi-obj	32.611	23.332	33.679	22.830	22.763	**135.215**

that MSR has the best RPD, compared to bi-obj and Utility. It is clear that the solution obtained by MSR has the lowest RPD in general compared with the bi-objective and utility models.

To further investigate the impact of the proposed models on the dependent variable RPD, the single factor mean of an Analysis of Variance (ANOVA) is performed. The results of the MSR model corresponding to the weight W_8, bi-obj model with $\alpha = 0.166$ and Utility model are statistically tested by ANOVA. For this analysis, 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 2. 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 permits us 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. Figure 2 shows that MSR model has significance difference in comparison with the other two models.

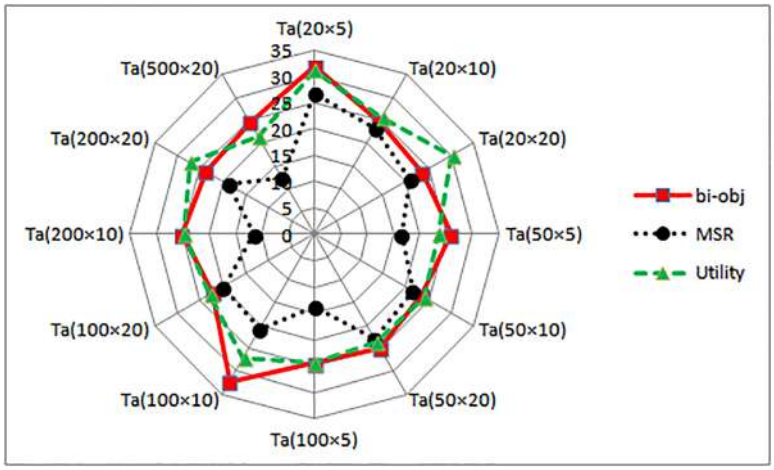

Fig. 1. RPD for models with the weight W_8.

Table 2. Analysis of Variance.

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				

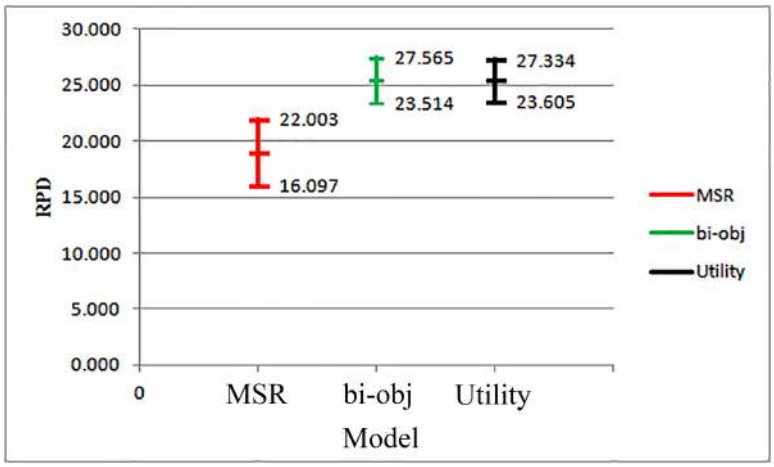

Fig. 2. 95% Tukey confidence intervals for all models.

4.1 Conclusion

This paper proposes a model that considers utility, stability and robustness to obtain robust and stable schedules of PFSP subject to different real-time events. Although we proposed a predictive-reactive approach based on generating robust schedule and reacting to every disturbance. Furthermore, PSO is applied for rescheduling, which shows the ability to deal with dynamic nature of this problem successfully. On the other hand, the proposed robust model has been compared with the bi-objective and the classical makespan model. Statistical ANOVA analysis have been conducted to compare the efficiency of these models. The ANOVA results revealed that the MSR model performed best. Also the sensitivity analysis indicated that giving higher priority to robustness leads to better solutions. For future work the current proposed model can be explored to handle other types of real-time events. Another interesting direction is to adapt the proposed model to other scheduling problems such as flexible flow shop and job shop scheduling.

References

1. Katragjini, K., Vallada, E., Ruiz, R.: Flow shop rescheduling under different types of disruption. Int. J. Prod. Res. vol. 51, no. 3, pp. 780–797 (2013)
2. Pinedo, M.: Scheduling: theory, algorithms, and systems, Fourth Edi. London: Springer New York Dordrecht Heidelberg (2012)
3. Graham, R. L., Lawer, E. L., Lenstra, J. K., Rinnooy Kan, A. H.: Optimisation and Approximation in Deterministic Sequencing and Scheduling: a Survey. Ann. Discret. Math. vol. Volume 5, pp. 287–326 (1979)
4. War, D.: The Classification of Scheduling Problems Under Production Uncertainty. Res. Logist. Prod. vol. 4, no. 3, pp. 245–255 (2014)
5. Sabuncuoglu I., Goren, S.: Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research. Int. J. Comput. Integr. Manuf. vol. 22, no. 2, pp. 37–41 (2009)
6. H., S., Sandoh, H.: Online Scheduling in Manufacturing A Cumulative Delay Approach. Japan: Springer-Verlag (2013)
7. Aytug, H., Lawley, M., McKay, K.: Executing production schedules in the face of uncertainties: A review and some future directions. Eur. J. vol. 161, pp. 86–110 (2005)
8. Ouelhadj, D., Petrovic, S.: A survey of dynamic scheduling in manufacturing systems. J. Sched. vol. 12, no. 4, pp. 417–431 (2008)
9. Vieira, G. E., Herrmann, J. W., Lin, E.: Rescheduling Manufacturing Systems: A Framework of Strategies, Policies, and Methods. pp. 39–62, (2003)
10. Cowling, P., Johansson, M.: Using real time information for effective dynamic scheduling. Eur. J. Oper. Res. vol. 139, no. 2, pp. 230–244 (2002)
11. Cowling, P., Ouelhadj, D., Petrovic, S.: A multi-agent architecture for dynamic scheduling of steel hot rolling. J. Intell. Manuf. vol. 14, pp. 457–470 (2003)
12. Cowling, P. I., Ouelhadj, D., Petrovic, S.: Dynamic scheduling of steel casting and milling using multi-agents, Prod. Plan. Control. vol. 15, no. 2, pp. 178–188 (2004)
13. Rahmani, D., Heydari, M.: Robust and stable flow shop scheduling with unexpected arrivals of new jobs and uncertain processing times. J. Manuf. Syst. vol.33, no.1, pp.84–92 (2014)

14. CHURCH, L. K., UZSOY, R.: Analysis of periodic and event-driven rescheduling policies in dynamic shops. Int. J. Comput. Integr. Manuf. vol. 5, no.3, pp. 153–163 (1992)
15. O'Donovan, R., Uzsoy, R., McKay, K. N.: Predictable scheduling of a single machine with breakdowns and sensitive jobs. Int. J. Prod. Res. vol. 37, no. 18, pp. 4217–4233 (1999)
16. Vieira, G. E., Herrmann, J. W., Lin, E.: Predicting the Performance of Rescheduling Strategies for Parallel Machine Systems. vol. 19, no. 4, pp. 256–266 (2000)
17. Hall N. G., Potts, C. N.: Rescheduling for New Orders. Oper. Res. vol. 3, no. 52, pp. 440–453 (2004)
18. Rangsaritratsamee, R., Ferrell, W. G., Kurz, M. B.: Dynamic rescheduling that simultaneously considers efficiency and stability. Comput. Ind. Eng. vol. 46, no. 1, pp. 1–15 (2004)
19. Kopanos, G. M., Capo, E., Espun, A., Puigjaner, L.: Costs for Rescheduling Actions: A Critical Issue for Reducing the Gap between Scheduling Theory and Practice. Ind. Eng. Chem. Res. vol. 47, no. 22, 8785–8795 (2008)
20. Kennedy, J., Eberhart, R.: Particle swarm optimisation. Proc. ICNN95-Int. Conf. Neural Networks. vol. 4, pp. 1942–1948 (1995)
21. Blackwell, T.: Particle Swarm Optimisation in Dynamic Environments. Evol. Comput. Dyn. Uncertain Environ. vol. 51, pp. 29–49 (2007)
22. Li, X., Branke, J., Blackwell, T.: Particle Swarm with Speciation and Adaptation in a Dynamic Environment. In: 8th GECCO, 51 (2006)
23. Zhang, Y., Wang, S., Ji, G.: A comprehensive survey on particle swarm optimisation algorithm and its applications. vol. 2015, p. 38, (2015)
24. Blum C., Merkle, D.: Swarm Intelligence Introduction and Applications. Berlin: Springer-Verlag Berlin Heidelberg (2008)
25. Eslami, M., Shareef, H., Khajehzadeh, M., Mohamed, A.: An Effective Particle Swarm Optimisation for Global Optimisation. Comput. Intell. Syst. vol. 316, pp. 267–274 (2012)
26. Lian, Z., Gu, X., Jiao, B.: A novel particle swarm optimisation algorithm for permutation flow-shop scheduling to minimise makespan. Chao. So. Fr. vol.35, no.4, pp.851–861 (2008)
27. Ramanan, T., Iqbal, M., Umarali, K.: A particle swarm optimisation approach for permutation flow shop scheduling problem. Int. J. Simul. Multidiscip. Des. Optim. vol. 5, p. A20 (2014)
28. Tasgetiren, M.F., Sevkli, M., Liang, Y.-C., Gencyilmaz, G.: Particle Swarm Optimisation Algorithm for Permutation Flowshop Sequencing Problem. The series Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg. Vol. 3172, pp. 382–389 (2004)
29. Taillard, E.: Benchmarks for basic scheduling problems. Eur. J. Oper. Res. vol. 64, pp. 278–285 (1993)
30. Jones, D.: A practical weight sensitivity algorithm for goal and multiple objective programming. Eur. J. Oper. Res. vol. 213, no. 1, pp. 238–245 (2011)