Journal of Manufacturing Systems 59 (2021) 549–564

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

Maintenance costs and makespan minimization for assembly permutation flow shop scheduling by considering preventive and corrective maintenance

Zikai Zhang [a,b], Qiuhua Tang [a,b,*], Manuel Chica [c,d]

[a] Key Laboratory of Metallurgical Equipment and Control Technology of Ministry of Education, Wuhan University of Science and Technology, China
[b] Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Wuhan University of Science and Technology, China
[c] Andalusian Research Institute DaSCI "Data Science and Computational Intelligence", University of Granada, 18071, Granada, Spain
[d] School of Electrical Engineering and Computing, The University of Newcastle, Callaghan, NSW, 2308, Australia

ARTICLE INFO

Keywords:
Assembly permutation
Bi-objective flow shop
Preventive maintenance
Corrective maintenance

ABSTRACT

The joint optimization of production scheduling and maintenance planning has a significant influence on production continuity and machine reliability. However, limited research considers preventive maintenance (PM) and corrective maintenance (CM) in assembly permutation flow shop scheduling. This paper addresses the bi-objective joint optimization of both PM and CM costs in assembly permutation flow shop scheduling. We also propose a new mixed integer linear programming model for the minimization of the makespan and maintenance costs. Two lemmas are inferred to relax the expected number of failures and CM cost to make the model linear. A restarted iterated Pareto greedy (RIPG) algorithm is applied to solve the problem by including a new evaluation of the solutions, based on a PM strategy. The RIPG algorithm makes use of novel bi-objective-oriented greedy and referenced local search phases to find non-dominated solutions. Three types of experiments are conducted to evaluate the proposed MILP model and the performance of the RIPG algorithm. In the first experiment, the MILP model is solved with an epsilon-constraint method, showing the effectiveness of the MILP model in small-scale instances. In the remaining two experiments, the RIPG algorithm shows its superiority for all the instances with respect to four well-known multi-objective metaheuristics.

1. Introduction

In the current manufacturing environment, global competition and market demand force most enterprises to produce products by assembly flow shop models. The typical and successful applications of assembly flow shop include fire engines [1], computers [2], plastic products [3], clothes [4], automobiles [5,6], distributed database systems [7] and multi-page invoice printing systems [8]. The classical assembly flow shop scheduling mainly consists of two stages: fabrication and assembly. Recently, many researchers have explored this problem with a flow shop layout in the assembly stage. This new problem is named assembly permutation flow shop scheduling, and can be denoted as $DPm \rightarrow Fm$ according to the classification in Framinan, Perez-Gonzalez and Fernandez-Viagas [9]. In the fabrication stage of $DPm \rightarrow Fm$, a variety of products are produced by processing all their components. Then, in the assembly stage, these components are assembled in turn on a series of machines. $DPm \rightarrow Fm$ aims to find an optimal product sequence by optimizing one or more objective functions. The most common objectives in assembly flow shop scheduling include the maximum completion time (also known as makespan), the total completion times [10], the maximum tardiness, total tardiness, maximum lateness, and total lateness.

Although researchers have focused on the production scheduling in $DPm \rightarrow Fm$, their studies are based on the hypothesis that machines are always available and never break. However, machines are inevitably subject to some unavailable periods because of preventive maintenance (PM) as well as unexpected failures [11]. PM is a scheduled activity taken on the machines to keep them at the desired level of operation and decrease the probability of failure. Second, when unexpected failures occur, operators need to perform corrective maintenance (CM) activities to help to restore the failed machines to a productive state. Note that the

* Corresponding author at: Key Laboratory of Metallurgical Equipment and Control Technology of Ministry of Education, Wuhan University of Science and Technology, China.
E-mail addresses: zhangliping@wust.edu.cn (Z. Zhang), tangqiuhua@wust.edu.cn (Q. Tang), manuelchica@ugr.es (M. Chica).

https://doi.org/10.1016/j.jmsy.2021.03.020
Received 16 November 2020; Received in revised form 26 February 2021; Accepted 22 March 2021
Available online 13 April 2021

Nomenclature

Indices

j	Index of product, $j = 1, \ldots, n$
i	Index of position in the job sequence, $i = 1, \ldots, n$
k,k'	Index of machine, $k,k' = 1,\ldots,m_1 + m_2$

Parameters

n	Number of products
m_1	Number of machines at the first stage
m_2	Number of machines at the second stage
t_{jk}	Processing (fabrication or assembly) time of product j on k-th machine
tp_k	Preventive maintenance time of machine k
tr_k	Corrective maintenance time of machine k
cp_k	Preventive maintenance cost on machine k
cr_k	Corrective maintenance cost on machine k
β_k	Shape parameter of failure function of machine k
θ_k	Scale parameter of failure function of machine k
Tpm_k	Preventive maintenance interval of machine k
M	A number sufficiently large

Variables

C_{ik}	Complete time of the product in position i on machine k
C_{max}	Makespan
TMC	Total maintenance costs
ξ_{ik}	Expected number of k^{th} machine failures while processing the product in position i
a_{ik}	The age of machine k after processing the product in position i
b_{ik}	The age of machine k before processing the product in position i
X_{ji}	If product j is assigned in position i, $X_{ji} = 1$. Otherwise, 0
Y_{ik}	If PM activity is performed immediately before the start of the product j in position i on machine k, $Y_{ji} = 1$. Otherwise, 0

maintenance activities before the failures are named as PM activities, while those performed when a failure occurs are referred as CM activities. If these interruptions are ignored during production scheduling, they will bring production losses and will reduce the production efficiency and service life of the machines. Since the production scheduling and maintenance activities are interrelated, it is necessary to integrate PM and CM into *DPm→Fm*. Therefore, this paper focuses on solving the *DPm→Fm* scheduling problem by minimizing the makespan and a second objective which considers PM and CM costs.

Due to the NP-hardness nature of this new multi-criteria optimization problem, this paper proposes the design and application of a metaheuristic to obtain a Pareto set of solutions to the problem. The iterated greedy algorithm, as a local search-based algorithm, has been successfully applied to tackle combinational optimization problems [12]. Compared with the other metaheuristics, the iterated greedy algorithm has fewer control parameters and does not need to embed specific knowledge [13]. Besides, one of its multi-objective variants, the restarted iterated Pareto greedy (RIPG) algorithm, has been successfully applied to tackle the flow shop scheduling and assembly line balancing [14]. Hence, we apply and adapt the RPIG algorithm for the *DPm→Fm* problem with PM and CM costs. Apart from determining product sequence, this new problem needs to determine whether to perform a PM activity immediately before each product. Since the unexpected random failures are considered in this problem, CM is carried out when failures occur. Therefore, and in a nutshell, the contribution of this paper is twofold:

- The formulation of a new *DPm→Fm* problem with PM and CM costs. We do it by defining a multi-objective MILP model to minimize the makespan and maintenance costs. In this model, two lemmas are derived to linearize the unexpected number of machine failures.
- A RIPG algorithm to obtain the near-optimal Pareto set of solutions. We include in the RIPG algorithm a problem-specific solution evaluation to calculate the objective values. Additionally, a bi-objective-oriented greedy and referenced local search phases are extended to explore the neighborhood structure. Finally, a bi-objective-oriented acceptance criterion and a restart mechanism are embedded into the metaheuristic to avoid local optima.

We perform an extensive computational study based on 40 calibration instances and 800 test instances to determine the best parameter combination of RIPG and test the performance of the proposed model and RIPG. A diverse set of multi-objective performance indicators and attainment surfaces are included to validate the results, both quantitatively and qualitatively. The final experimental results suggest that the proposed model is effective in tackling small-scale instances and the proposed RIPG outperforms four well-known metaheuristics including MOPSO, MOSA, NSGA-II and NSGA-III in all instances.

The remainder of this paper is organized as follows. Section 2 presents a review of the related literature. A novel MILP model is formulated in Section 3 to define the *DPm→Fm* with PM and CM costs. Then, the RIPG algorithm is proposed in Section 4. Experimental results and discussion are reported in Section 5. Finally, main conclusions, some managerial insights, and future work are discussed in Section 6.

2. Literature review

Since this paper focuses on assembly permutation flow shop scheduling considering PM and CM, this section first reports the current state-of-the-art status on assembly permutation flow shop scheduling. Later, the second sub-section investigates similar scheduling problems having PM or CM activities.

2.1. State-of-the-art on assembly permutation flow shop scheduling

The assembly permutation flow shop scheduling has been explored in many research studies. Among them, most studies address that the flow shop in the assembly stage consists of two stages: transportation and assembly stages. The former aims to transport components to the assembly machines and the latter corresponds to the assembly process. This problem is first defined by Koulamas and J. Kyparisis [15] and denoted as *DPm→F2*. Koulamas and J. Kyparisis [15] also called this problem three-stage assembly flow shop scheduling. They designed several constructive heuristics to minimize the makespan. Since then, subsequent studies can be classified into two categories according to the number of objectives: single-objective and multi-objective.

For the single-objective optimization, Andrés and Hatami [16] considered the sequence-dependent setup times into *DPm→F2*, and proposed a MILP model to minimize the total completion times. Campos, Arroyo and Tavares [17] also addressed *DPm→F2* with setup times, and proposed VNS to minimize the tardiness. Komaki, et al. [18] improved the original cuckoo optimization algorithm to minimize the makespan of *DPm→F2*.

For the multi-objective optimization, there are two approaches in the current studies: the weighted approach and the Pareto front method. Regarding the former one, Hatami, et al. [19] addressed *DPm→F2* with the sequence-dependent setup times and proposed a MILP model, simulated annealing algorithm (SA) and tabu search to minimize the

Table 1
Publications about the *DPm→Fm* problem.

Reference	Problem	Constraints	Objectives	Model	Methods
Koulamas and J. Kyparisis [15]	*DPm→F2*	–	Makespan	–	Heuristics
Andrés and Hatami [16]	*DPm→F2*	Setup times	Total completion times	MILP	–
Campos, Arroyo and Tavares [17]	*DPm→F2*	Setup times	Tardiness	–	VNS
Komaki, et al. [18]	*DPm→F2*	–	Makespan	–	COA
Hatami, et al. [19]	*DPm→F2*	Setup times	Mean completion time and maximum tardiness	MILP	SA, TS
Maleki-Darounkolaei, et al. [20]	*DPm→F2*	Setup times and blocking	Weighted mean completion time and makespan	MILP	SA
Maleki and Seyedi [21]	*DPm→F2*	Setup times and blocking	Weighted mean completion time and makespan	–	VNS, SA
Wang, et al. [22]	*DPm→F2*	Batches	Average arrival time and total delivery cost	NLP	HGA-OVNS
Shoaardebili and Fattahi [23]	*DPm→F2*	Machine availability	Total weighted completion times, weighted tardiness, and earliness	NLP	NSGA-II, MOSA
Campos and Arroyo [24]	*DPm→F2*	Setup times	Total completion times and total tardiness	–	NSGA-II
Tajbakhsh, Fattahi and Behnamian [25]	*DPm→F2*	–	Makespan, earliness and tardiness costs	MILP	MOPSO-GA
Sheikh, et al. [26]	*DPm→Fm*	Setup times and release time	Makespan	MILP	VNS and GWO
Xiong, Xing and Wang [27]	*DPm→HF2*	–	Total completion times	MILP	HGA-VNS, HDDE-VNS and HEDA-VNS
This work	*DPm→Fm*	PM and CM	Makespan and maintenance costs	MILP	RIPG

mean completion time and maximum tardiness. Apart from setup times, Maleki-Darounkolaei, et al. [20] further addressed the blocking into *DPm→F2* problem. They developed a new MILP model and SA to optimize the weighted mean completion time and makespan. Based on the above research, Maleki and Seyedi [21] further proposed two meta-heuristics: a variable neighborhood search algorithm (VNS) and SA to minimize the same objectives. The final results indicated that VNS had a better performance than SA while SA needed less CPU time. Wang, et al. [22] studied *DPm→F2* with batch delivery. To optimize the weighted sum of average arrival time at the customer and total delivery cost, they presented two fast heuristics (SPT-based heuristic and LPT-based heuristic) and a new hybrid genetic algorithm (GA) with VNS and opposition-based learning.

Regarding the Pareto front method, Shoaardebili and Fattahi [23] considered the machine availability constraints in *DPm→F2* and decided to use a Pareto optimization instead of a weighting method. Concretely, they applied the non-dominated sorting genetic algorithm (NSGA-II) and multi-objective simulated annealing algorithm (MOSA). Campos and Arroyo [24] minimized the total completion times and total tardiness of *DPm→F2* with setup times via NSGA-II. Tajbakhsh, Fattahi and Behnamian [25] focused on minimizing the makespan and the sum of the earliness and tardiness costs. They first proposed a MILP model to formulate this problem and then designed a multi-objective combination algorithm (MOPSO-GA) mixing particle swarm optimization and genetic algorithm.

The above studies focus on *DPm→F2* where the second stage just has two operations. Besides, Sheikh, et al. [26] addressed the multi-stage assembly flow shop scheduling with setup times and release time. In this problem, after all the components have been finished in the fabrication stage, these components need to be assembled, transported, printed, and packaged. This new problem can be denoted as *DPm→Fm*. To tackle this problem, Sheikh, et al. [26] designed nine efficient heuristics to minimize the makespan, and further implemented general VNS and grey wolf optimizer (GWO) to improve the heuristic solutions. Xiong, Xing and Wang [27] studied the assembly flow shop scheduling with hybrid flow shop layout in the assembly stage to minimize the total completion times. They developed a MILP model, two fast heuristics (SPT-based heuristic and NEH-based heuristic), and three meta-heuristics (HGA-VNS, HDDE-VNS and HEDA-VNS).

2.2. Scheduling problems considering PM or CM activities

The studies of the previous section are based on the hypothesis that machines are always available and never break. However, machines in actual production are inevitably subject to some unavailable periods due to unexpected failure or PM with time elapsing [11]. In this situation and to approximate the actual production mode, it is necessary to consider PM and CM activities in the assembly permutation flow shop. Up to our knowledge, there is no research investigating PM and CM in *DPm→Fm*. The works of Seidgar, et al. [28] and Seidgar, Zandieh and Mahdavi [29] are the only ones considering PM in a two-stage assembly flow shop scheduling (*DPm→*1).

Regarding PM, two situations of consideration can be found in the literature [30]: machine unavailability constraints and joint production scheduling. For the first situation, it is assumed that PM is performed at intervals and production operations are executed within the periods between two consecutive PMs. The intervals may be fixed and known, or flexible [31–35]. For the second situation, the PM costs and frequency need to be determined as decision variables along with production scheduling. This case usually involves machine deterioration and hence, an appropriate PM plan can improve the service life of the machines. Ruiz, Carlos García-Díaz and Maroto [36] designed three maintenance policies to determine the PM intervals and employed six algorithms to minimize the makespan cited with the proposed policies. Wang and Liu [37] considered sequence-dependent set-up times and PM in the two-stage hybrid flow shop scheduling. Khamseh, Jolai and Babaei [38] investigated sequence-dependent setups and PM activities in flexible flow shop scheduling. With the minimization of makespan, they proposed a SA algorithm with a local search procedure and GA to solve the small- and large-scale instances. Yu and Seif [39] and Miyata, Nagano and Gupta [40] extended maintenance level to *m*-machine flow shop and no-wait flow shop scheduling, respectively. Sheikhalishahi, et al. [41] addressed the joint open shop scheduling with PM and human errors, and developed three metaheuristics, including NSGA-II, MOPSO and SPEA-II to find near-optimal Pareto front solutions. Yu and Han [42] aimed at the proportionate flow shop scheduling with PM and focused on examining the maximum lateness and the total completion time. Ghodratnama, et al. [43] designed a SA algorithm to solve the single-machine scheduling with maintenance activities. Hu, Jiang and Liao [44] studied the joint optimization of two-machine flow shop scheduling and maintenance plan. Wang and Liu [45] investigated the integration optimization of parallel machine scheduling and multi-resources preventive maintenance.

When an unexpected machine breakdown is considered, researchers usually employ a robustness method to arrange CM activities. Pan, Liao and Xi [46] studied single-machine scheduling where machine breakdown and PM activities with flexible time intervals are considered. Cui, et al. [47] dealt with the integration of flow shop scheduling and PM and CM to minimize the quality robustness and solution robustness. Since this integration involved failure uncertainty, they adopted a Monte Carlo sampling method to evaluate the solutions approximately. Boufellouh and Belkaid [30] employed NSGA-II and bi-objective adaptation of the particle swarm optimization (BOPSO) to minimize the makespan and total production costs of joint permutation flow shop and maintenance (PM and CM) under a global resource constraint. Ye, Wang and Liu [48] integrated adaptive PM and CM into a generic *m*-machine flow shop scheduling, and minimized the total tardiness cost, PM and CM costs.

To sum up, we have also summarized all the related literature about *DPm→Fm* in Table 1. From our study, we see that there is a lack of research on *DPm→Fm* by considering PM and CM. Hence, we propose here to minimize both makespan and maintenance costs through our novel MILP model and RIPG algorithm to obtain the near-optimal Pareto front solutions. We will describe both the model and metaheuristic-based algorithm in the next Sections 3 and 4, respectively.

3. Problem formulation

3.1. Mathematical variables and parameters

The main notation related to this new model is presetned at the beginning of this paper. According to the classification in Framinan, Perez-Gonzalez and Fernandez-Viagas [9], this new problem can be denoted as $DPm\rightarrow Fm|PM\&CM|(C_{max}, TMC)$. There are *n* products to be processed in the fabrication and assembly stages. Specifically, during the fabrication stage, each product *j* contains m_1 components that are produced on m_1 dedicated parallel machines respectively. Then in the assembly stage, these components are assembled on a set of m_2 assembly machines in flow shops. Each product *j* requires a fixed time t_{jk} on different fabrication or assembly machine *k*. At the same time, each product can be processed by one machine and each machine can only process one product. The product sequence in different machines is the same. Hence, the first decision variable is to determine the product sequence which is denoted as X_{ji}.

Each machine *k* is subject to failure and the time to failure follows a Weibull probability distribution with scale parameter θ_k and shape parameter β_k ($\beta_k > 1$ since the machine degenerates over time [47]). We also consider an age-based preventive maintenance policy. The initial age of each machine is set 0. Since unexpected failures reduce production capacity and cause production loss, PM is usually performed to improve machines' conditions. We assume that PM can restore the machine to the "as-good-as-new" state, i.e., the machine's age is reset to 0 after PM. Let Tpm_k be the preventive maintenance interval of machine *k*. To ensure the high reliability of each machine, the age of the machine is not allowed to exceed Tpm_k. Based on the optimal PM interval theory [36], Tpm_k can be calculated by maximizing the availability given in Eq. (1).

$$Tpm_k = \theta_k \cdot \left[\frac{tp_k}{tr_k(\beta_k - 1)}\right]^{1/\beta_k} \tag{1}$$

Although PM can reduce the probability of unexpected machine failures, it cannot completely eliminate failures. Hence, CM needs to be carried out once failures happen. CM restores the machine to an operating condition while the age of the machine does not change. When the machine is repaired, it continues to process the product without any additional time penalty. The PM and CM durations of machine *k* are set as tp_k and tr_k respectively, and their corresponding costs are cp_k and cr_k. In this situation, the start time and number of PM actions need to be decided. In that sense, the decision variable Y_{ik} determines whether a PM activity is performed immediately before the start of the product *j* in position *i* on machine *k*.

3.2. Objectives and constraints of the model

The main additional hypotheses of this model are as follows:

(1) Setup times are not considered.
(2) The buffers between fabrication and assembly stages are ignored.
(3) PM can restore machines to the "as-good-as-new" state.
(4) All machines are available at the beginning of the scheduling horizon.

The first minimization objective is the makespan, as done in traditional models, and is defined in Eq. (2). The second objective is to minimize the maintenance costs including both PM and CM costs, defined in Eq. (3).

$$\text{minimize } C_{max} = C_{n,m_1+m_2} \tag{2}$$

$$\text{minimize } TMC = \sum_{i=1}^{n}\sum_{k=1}^{m_1+m_2}(Y_{ik}\cdot cp_k + \xi_{ik}\cdot cr_k) \tag{3}$$

Constraints (4) and (5) limit the product sequence (i.e., each product is assigned in one position and each position just has one product). Constraints (6–10) restrict the completion times of all fabrication and assembly machines. Constraint (6) means that all the machines are available at the beginning. Constraint (7) requires that for all fabrication and assembly machines the product starts after the finishing of its previous product. In addition to the processing time, PM and CM times need to be added. Constraint (8) again limits the completion time on the first assembly machine ($k = m_1 + 1$). Each product is assembled on the first assembly machine only when all its components have been processed. Constraint (9) again limits the completion times on the other assembly machines. Each product starts to be assembled on assembly machine k ($k = m_1 + 2, \ldots, m_1 + m_2$) after it finishes the assembly in the previous assembly machine. Constraint (10) implies that the completion time is higher than 0.

$$\sum_{j=1}^{n} X_{ji} = 1,\ \forall i = 1, \ldots, n \tag{4}$$

$$\sum_{i=1}^{n} X_{ji} = 1,\ \forall j = 1, \ldots, n \tag{5}$$

$$C_{0k} = 0,\ \forall k = 1, \ldots, m_1 + m_2 \tag{6}$$

$$C_{ik} \geq C_{i-1,k} + \sum_{j=1}^{n} t_{jk}\cdot X_{ji} + Y_{ik}\cdot tp_k + \xi_{ik}\cdot tr_k,\ \forall i = 1, \ldots, n,\ k = 1, \ldots, m_1 + m_2 \tag{7}$$

$$C_{i,m_1+1} \geq C_{ik} + \sum_{j=1}^{n} t_{j,m_1+1}\cdot X_{ji} + \xi_{i,m_1+1}\cdot tr_{m_1+1},\ \forall i = 1, \ldots, n,\ k = 1, \ldots, m_1 \tag{8}$$

$$C_{ik} \geq C_{i,k-1} + \sum_{j=1}^{n} t_{jk}\cdot X_{ji} + \xi_{ik}\cdot tr_k,\ \forall i = 1, \ldots, n,\ k = m_1 + 2, \ldots, m_1 + m_2 \tag{9}$$

$$C_{ik} \geq 0,\ \forall i = 1, \ldots, n,\ k = 1, \ldots, m_1 + m_2 \tag{10}$$

Constraints (11)–(18) limit the age of the machines. Constraint (11) means that the initial age of each machine is set as 0. For each fabrication or assembly machine k, if PM is performed immediately before the start of the product j in position i, constraints (12) and (13) become active and the age a_{ik} is equal to $\sum_{j=1}^{n} t_{jk} \bullet X_{ji}$. Otherwise, constraints (14) and (15) become active and a_{ik} is equal to $a_{i-1,k} + \sum_{j=1}^{n} t_{jk} \bullet X_{ji}$. Constraint (16) defines the age of machine k before processing the product in position i. Constraints (17) and (18) state that machines' age is not allowed to exceed the PM interval. Finally, constraints (19) and (20) limit the decision variables.

$$a_{ok} = 0, \ \forall k = 1, \ \ldots, \ m_1 + m_2 \tag{11}$$

$$a_{ik} - \sum_{j=1}^{n} t_{jk} \bullet X_{ji} \leq M \bullet (1 - Y_{ik}), \ \forall i = 1, \ \ldots, \ n, \ k = 1, \ \ldots, \ m_1 + m_2 \tag{12}$$

$$a_{ik} - \sum_{j=1}^{n} t_{jk} \bullet X_{ji} \geq -M \bullet (1 - Y_{ik}), \ \forall i = 1, \ \ldots, \ n, \ k = 1, \ \ldots, \ m_1 + m_2 \tag{13}$$

$$a_{ik} - \left(a_{i-1,k} + \sum_{j=1}^{n} t_{jk} \bullet X_{ji} \right) \leq M \bullet Y_{ik}, \ \forall i = 1, \ \ldots, \ n, \ k = 1, \ \ldots, \ m_1 + m_2 \tag{14}$$

$$a_{ik} - \left(a_{i-1,k} + \sum_{j=1}^{n} t_{jk} \bullet X_{ji} \right) \geq -M \bullet Y_{ik}, \ \forall i = 1, \ \ldots, \ n, \ k = 1, \ \ldots, \ m_1 + m_2 \tag{15}$$

$$b_{ik} = a_{ik} - \sum_{j=1}^{n} t_{jk} \bullet X_{ji}, \ \forall i = 1, \ \ldots, \ n, \ k = 1, \ \ldots, \ m_1 + m_2 \tag{16}$$

$$a_{ik} \leq Tpm_k, \ \forall i = 1, \ \ldots, \ n, \ k = 1, \ \ldots, \ m_1 + m_2 \tag{17}$$

$$b_{ik} \leq Tpm_k, \ \forall i = 1, \ \ldots, \ n, \ k = 1, \ \ldots, \ m_1 + m_2 \tag{18}$$

$$X_{ji} = \{0, \ 1\}, \ \forall j = 1, \ \ldots, \ n, \ i = 1, \ \ldots, \ n \tag{19}$$

$$Y_{ik} = \{0, \ 1\}, \ \forall i = 1, \ \ldots, \ n, \ k = 1, \ \ldots, \ m_1 + m_2 \tag{20}$$

Besides and according to the maintenance theory, when the time to failure is governed by a Weibull probability distribution, the breakdown number ξ_{ik} will follow a Poisson probability distribution and $\Pr(\xi_{ik} = \eta) = \frac{(\lambda_{ik})^{\eta} \bullet e^{-\lambda_{ik}}}{\eta!}$, $\forall \eta \in [0, +\infty)$ where $\lambda_{ik} = \left(\frac{a_{ik}}{\theta_k}\right)^{\beta_k} - \left(\frac{b_{ik}}{\theta_k}\right)^{\beta_k}$. We set the expectation of ξ_{ik} as the failure number, i.e., $\xi_{ik} = E\left(\frac{(\lambda_{ik})^{\eta} \bullet e^{-\lambda_{ik}}}{\eta!}\right) = \lambda_{ik} = \left(\frac{a_{ik}}{\theta_k}\right)^{\beta_k} - \left(\frac{b_{ik}}{\theta_k}\right)^{\beta_k}$.

3.3. Linearization of the model

In the described model, the expected value of ξ_{ik} leads to a non-linear property. Therefore, the model cannot be directly solved by a commercial solver. Hence, we linearize the model by relaxing the value of ξ_{ik}. Since the age of the machines will not exceed the preventive maintenance interval Tpm_k, we can have the following lemma.

Lemma 1. The expected failure number during a PM period does not exceed a constant $\left(\frac{\mathrm{Tpm_k}}{\theta_k}\right)^{\beta_k}$.

Proof. During each PM period on machine k, it is assumed that the operations (process or assembly) need to be completed in position s1, s1 + 1, …, s1+s2, and the corresponding expected failure numbers are $\xi_{s1,k}$, $\xi_{s1+1,k}$, …, $\xi_{s1+s2,k}$. The expected failure number during this PM period is equal to the sum of those of the operations $\sum_{i=s1}^{s2} \xi_{ik}$. Then:

$$\sum_{i=s1}^{s2} \xi_{ik} = \sum_{i=s1}^{s2} \left(\left(\frac{a_{ik}}{\theta_k}\right)^{\beta_k} - \left(\frac{b_{ik}}{\theta_k}\right)^{\beta_k} \right)$$

$$= \left(\frac{a_{s1,k}}{\theta_k}\right)^{\beta_k} - \left(\frac{b_{s1,k}}{\theta_k}\right)^{\beta_k} + \left(\frac{a_{s1+1,k}}{\theta_k}\right)^{\beta_k} - \left(\frac{b_{s1+1,k}}{\theta_k}\right)^{\beta_k} + \ldots + \left(\frac{a_{s1+s2,k}}{\theta_k}\right)^{\beta_k} - \left(\frac{b_{s1+s2,k}}{\theta_k}\right)^{\beta_k}$$

For the two adjacent operations, the subsequent operation's age before processing is equal to the previous operation's age after processing, i.e., $b_{i+1,k} = a_{i,k}$. Then, $\sum_{i=s1}^{s2} \xi_{ik} = \left(\frac{a_{s1+s2,k}}{\theta_k}\right)^{\beta_k} - \left(\frac{b_{s1,k}}{\theta_k}\right)^{\beta_k}$. Furthermore, since the initial machines' age is 0 and PM restore machines to the "as-good-as-new" state, $\left(\frac{b_{s1,k}}{\theta_k}\right)^{\beta_k} = 0$ and $\sum_{i=s1}^{s2} \xi_{ik} = \left(\frac{a_{s1+s2,k}}{\theta_k}\right)^{\beta_k}$. Finally, to ensure the high-reliability, the machine's age is not allowed to exceed Tpm_k, i.e., $a_{s1+s2,k} \leq Tpm_k$. Hence, we can conclude that the expected failure number during a PM period does not exceed a constant $\left(\frac{Tpm_k}{\theta_k}\right)^{\beta_k}$.

Based on the above lemma, we relax the expected failure number during a PM period as $\left(\frac{Tpm_k}{\theta_k}\right)^{\beta_k}$. Thus, the failure number per unit time is $\left(\frac{Tpm_k}{\theta_k}\right)^{\beta_k} \bullet \frac{1}{Tpm_k} = \frac{Tpm_k^{\beta_k - 1}}{\theta_k^{\beta_k}}$. In this situation, ξ_{ik} can be calculated by $\frac{Tpm_k^{\beta_k - 1}}{\theta_k^{\beta_k}} \bullet \sum_{j=1}^{n} t_{jk} \bullet X_{ji}$. After relaxing ξ_{ik}, we can get the second lemma:

Lemma 2. The relaxed CM cost is constant.

Proof. In the above non-linear model, the CM cost is equal to $\sum_{i=1}^{n} \sum_{k=1}^{m_1+m_2} (\xi_{ik} \bullet cr_k)$. We replace the expected failure number with the relaxed value $\frac{Tpm_k^{\beta_k - 1}}{\theta_k^{\beta_k}} \bullet \sum_{j=1}^{n} t_{jk} \bullet X_{ji}$. Then, the CM cost becomes:

$$\sum_{i=1}^{n} \sum_{k=1}^{m_1+m_2} (\xi_{ik} \bullet cr_k) = \sum_{i=1}^{n} \sum_{k=1}^{m_1+m_2} \left(\frac{Tpm_k^{\beta_k - 1}}{\theta_k^{\beta_k}} \bullet \sum_{j=1}^{n} t_{jk} \bullet X_{ji} \bullet cr_k \right)$$

$$= \sum_{k=1}^{m_1+m_2} cr_k \bullet \frac{Tpm_k^{\beta_k - 1}}{\theta_k^{\beta_k}} \bullet \sum_{i=1}^{n} \sum_{j=1}^{n} t_{jk} \bullet X_{ji}$$

$\sum_{i=1}^{n} \sum_{j=1}^{n} t_{jk} \bullet X_{ji}$ means that the total processing times of all products on machine k. It can be simplified as $\sum_{j=1}^{n} t_{jk}$. Correspondingly, the CM cost is equal to $\sum_{k=1}^{m_1+m_2} cr_k \bullet \frac{Tpm_k^{\beta_k - 1}}{\theta_k^{\beta_k}} \bullet \sum_{j=1}^{n} t_{jk}$. Therefore, the relaxed CM cost is constant.

The non-linear maintenance costs objective is translated into the linear form of Eq. (21).

$$\text{minimize } TMC = \sum_{i=1}^{n} \sum_{k=1}^{m_1+m_2} Y_{ik} \bullet cp_k + \sum_{k=1}^{m_1+m_2} cr_k \bullet \frac{Tpm_k^{\beta_k - 1}}{\theta_k^{\beta_k}} \bullet \sum_{j=1}^{n} t_{jk} \tag{21}$$

Meanwhile, the corresponding non-linear constraints (7)–(9) are turned into Eqs. (22)–(24).

$$C_{ik} \geq C_{i-1,k} + \sum_{j=1}^{n} t_{jk} \bullet X_{ji} + Y_{ik} \bullet tp_k + \frac{Tpm_k^{\beta_k - 1}}{\theta_k^{\beta_k}} \bullet tr_k \bullet \sum_{j=1}^{n} t_{jk} \bullet X_{ji}, \ \forall i = 1, \ \ldots, \ n, \ k = 1, \ \ldots, \ m_1 + m_2 \tag{22}$$

$$C_{i,m_1+1} \geq C_{ik} + \sum_{j=1}^{n} t_{j,m_1+1} \bullet X_{ji} + \frac{Tpm_{m_1+1}^{\beta_{m_1+1} - 1}}{\theta_{m_1+1}^{\beta_{m_1+1}}} \bullet tr_{m_1+1} \bullet \sum_{j=1}^{n} t_{j,m_1+1} \bullet X_{ji}, \ \forall i = 1, \ \ldots, \ n, \ k = 1, \ \ldots, \ m_1 \tag{23}$$

Table 2
The parameter values of the illustrative example.

Machine								Processing time t_{jk}									
Index k	Function	tp_k	tr_k	cp_k	cr_k	β_k	θ_k	1	2	3	4	5	6	7	8	9	10
1	Fabrication	4	8	10	16	3	30	4	5	6	10	3	7	2	5	5	7
2	Fabrication	3	7	8	15	4	38	6	2	4	7	4	4	8	5	9	6
3	Assembly	2	6	9	17	2	34	8	6	8	3	4	2	8	5	6	4
4	Assembly	4	7	9	15	3	32	2	4	5	7	6	8	10	4	3	4

Table 3
The relaxed expected CM times.

Machine k	Product j									
	1	2	3	4	5	6	7	8	9	10
1	0.42	0.53	0.63	1.06	0.32	0.74	0.21	0.53	0.53	0.74
2	0.26	0.09	0.17	0.30	0.17	0.17	0.34	0.21	0.39	0.26
3	0.82	0.61	0.82	0.31	0.41	0.20	0.82	0.51	0.61	0.41
4	0.19	0.38	0.47	0.66	0.57	0.76	0.95	0.38	0.28	0.38

Table 4
The completion times of all products on all machines.

Machine k	Product j									
	8	6	9	3	5	1	7	2	10	4
1	5.53	13.27	18.80	29.43	32.75	37.17	39.38	48.91	56.65	71.71
2	5.21	9.38	18.77	22.94	30.11	36.37	44.71	46.80	56.06	63.36
3	11.04	15.47	25.41	38.25	42.66	53.48	62.30	70.91	75.32	78.63
4	15.42	24.23	28.69	43.72	54.29	56.48	73.25	81.63	86.01	93.67

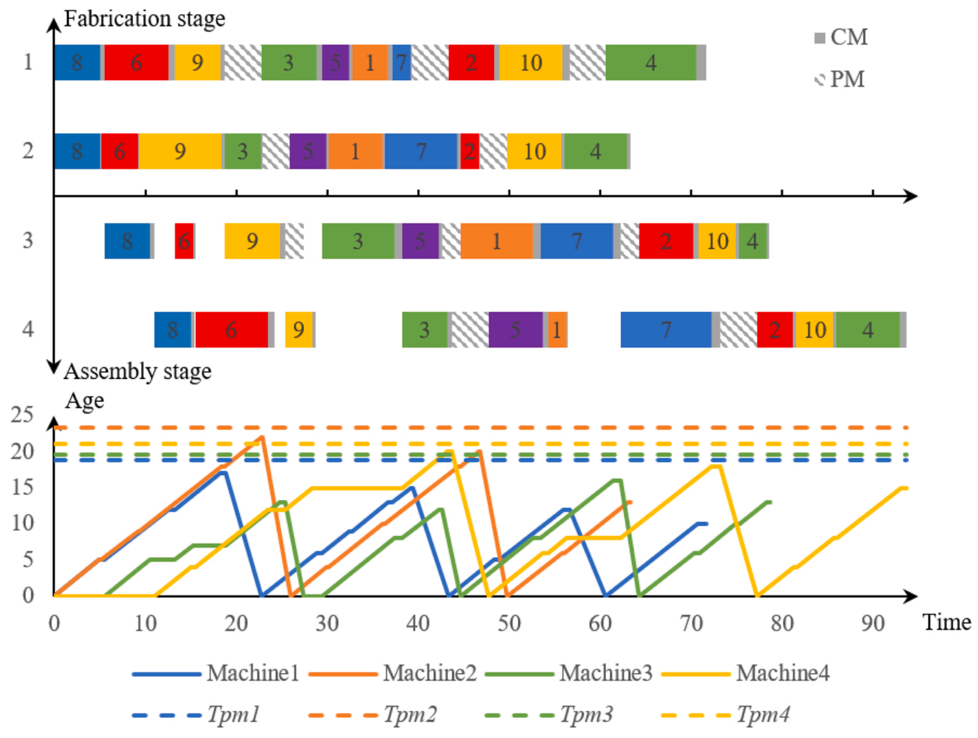

Fig. 1. Gantt chart and age curve with the solution to the scheduling example.

$$C_{ik} \geq C_{i,k-1} + \sum_{j=1}^{n} t_{jk} \cdot X_{ji} + \frac{Tpm_k^{\beta_k - 1}}{\theta_k^{\beta_k}} \cdot tr_k \cdot \sum_{j=1}^{n} t_{jk} \cdot X_{ji}, \ \forall i = 1, \ \ldots, \ n, \ k = m_1 + 2, \ \ldots, \ m_1 + m_2 \tag{24}$$

In a nutshell, a MILP model for this new problem is constructed with the objective functions (2) and (21) and constraints (4)–(6), (10)–(20), (22)–(24).

3.4. An illustrative example

We provide here an illustrative example with 10 products and 4 machines (2 fabrication machines and 2 assembly machines) for this new problem and the proposed MILP model. Table 2 presents the parameters' values. According to Eq. (1), the values of Tpm_k are equal to 18.90, 23.36, 19.63 and 21.08, respectively. Furthermore, since Section 3.2 relaxes the expected failure number ξ_{ik}, we can calculate the relaxed CM time for each product j on machine k through $\frac{Tpm_k^{\beta_k-1}}{\theta_k^{\beta_k}} \cdot t_{jk} \cdot tr_k$. The relaxed CM times are presented in Table 3. Under this scenario, we assume that a solution consists of a product sequence {8, 6, 9, 3, 5, 1, 7, 2, 10, 4} and a set of PM execution time points. Table 4 presents the completion times of all products on all machines. Specifically, PM operations are performed immediately before the start of products 3, 2, and 4 on machine 1; products 5 and 10 on machine 2; products 3, 1 and 2 on machine 3; products 5 and 2 on machine 4. Fig. 1 shows a Gantt chart and the age curve for the solution of the example. We can get three observations from the analysis of this figure:

(1) The completion time of each product on any machine needs to consider the relaxed CM time. Table 4 details the final completion times of all products on all machines.
(2) When a product is finished, a relaxed CM time is reserved and prepared for the failures. Once a failure happens, the completion time does not fluctuate much and it is still limited.
(3) The age of the machines does not exceed a threshold (preventive maintenance interval Tpm_k) since PM operations are performed to ensure the high reliability of all the machines. The makespan of this solution is 93.67. Since machines 1–4 have 3, 2, 3, and 2 PM operations respectively, the total PM cost is 91 ($3 \times 10 + 2 \times 8 + 3 \times 9 + 2 \times 9$). According to Lemma 2, the CM cost is 42.84 and therefore, the total maintenance cost is 133.84.

4. Restarted iterated Pareto greedy (RIPG) algorithm

The original iterated greedy algorithm is designed to solve the single-objective optimization problems. For our multi-objective optimization problem, this paper follows the research by Minella, Ruiz and Ciavotta [14] and Zhang, et al. [49], and extends a restarted iterated Pareto greedy algorithm (RIPG). In the proposed RIPG, a set of initial solutions is generated by two NEH heuristics and the bi-objective-oriented referenced local search in the initiation phase. At each iteration, the found non-dominated solutions are stored in an external Pareto Archive (POS) during the execution of the algorithm, being one non-dominated solution from the POS the incumbent solution. While the termination criterion is not reached, RIPG successively applies the bi-objective-oriented greedy search, the referenced local search, acceptance criterion, and restart mechanism on the incumbent solution. The details of these phases are introduced in the following sub-sections.

4.1. Solution evaluation and problem-specific initialization

We encode each solution as a product sequence $\Pi = \{\pi_1, \ldots, \pi_i, \ \ldots, \ \pi_n\}$. π_i means the product assigned to position i, and each sequence contains n elements. This sequence represents the decision variable X_{ji} in the proposed model.

To decrease the times of PM, we perform the PM operations at the end of the optimal maintenance intervals. Before processing each product π_i on a machine k, there is a decision whether a PM activity is performed immediately or not. If the age of the machine after processing is larger than the optimal PM interval Tpm_k, a PM activity is performed. Otherwise, there is no PM activity. With this strategy, each PM activity is performed at a time close to the Tpm_k under the premise that constraints (17)-(18) are met. Based on this strategy, the specific procedure of the solution under evaluation is presented below:

Step 1: Calculate the relaxed CM time $\widetilde{tr}_{jk}$ for each product j on machine k by $\frac{Tpm_k^{\beta_k-1}}{\theta_k^{\beta_k}} \cdot t_{jk} \cdot tr_k$.

Step 2: Initialize the completion time and age of each machine. For each machine k, $C_{0k} = 0$ and $a_{0k} = 0$.

Step 3: Determine the PM execution time and calculate the completion time on the fabrication stage. For product π_i on machine k, if the age of machine k after processing this product is larger than Tpm_k (in other words, $a_{i-1,k} + t_{\pi_i,k} > Tpm_k$), a PM activity needs to be performed immediately before this position ($Y_{ik} = 1$). Meanwhile, $a_{ik} = t_{\pi_i,k}$ and $C_{ik} = C_{i-1,k} + t_{\pi_i,k} + \widetilde{tr}_{\pi_i,k} + tp_k$. Otherwise, $Y_{ik} = 0$, $a_{ik} = a_{i-1,k} + t_{\pi_i,k}$ and $C_{ik} = C_{i-1,k} + t_{\pi_i,k} + \widetilde{tr}_{\pi_i,k}$.

Step 4: Determine the PM execution time and calculate the completion time on the assembly stage. The judgment of PM execution time and the update of machines' age are the same as the method in Step 3; while the difference is the update of completion time. Apart from the completion time in the previous position $C_{i-1,k}$, the completion time in the first assembly machine ($k = m_1 + 1$) should also consider those in the fabrication stage, and that in the subsequent assembly machines ($k = m_1 + 2, \ \ldots, \ m_1 + m_2$) should consider that in the previous assembly machine. Therefore, for the first assembly machine, $C_{i,m_1+1} = max\{C_{ik'}(k' = 1, \ldots, m_1), \ C_{i-1,m_1+1} + Y_{i,m_1+1} \cdot tp_{m_1+1}\} + t_{\pi_i,m_1+1} + \widetilde{tr}_{\pi_i,m_1+1}$; for the subsequent machines, $C_{ik} = max\{C_{i,k-1}, \ C_{i-1,k} + Y_{ik} \cdot tp_k\} + t_{\pi_i,k} + \widetilde{tr}_{\pi_i,k}$.

Step 5: Calculate the objective values according to Eqs. (2) and (21).

A good initial solution can greatly improve the performance of the IG algorithm [14]. For a single-objective version, IG starts with an initial solution obtained by NEH heuristic for a specific objective criterion. However, for a bi-objective version, we need to find a good solution for both objectives. Hence, the proposed RIPG extends two NEH heuristics to provide initial solutions for makespan and maintenance costs, respectively. Then, the local search procedure is applied to the two initial solutions to generate a set of partial solutions. The obtained non-dominated solutions are included in the *POS*. Finally, a solution from the non-dominated set of solutions is randomly selected as the current solution for the next phases.

4.2. Bi-objective-oriented greedy phase

The greedy search of the RIPG algorithm consists of two key steps: destruction and construction. In the destruction phase, d products are randomly selected from the current sequence and put into the set P^r. These extracted products are also removed from the current sequence. Then, in the construction phase, the extracted products in P^r are iteratively reinserted into the current sequence one by one. Specifically, in the first iteration, the first product in P^r is inserted into all possible positions of the current sequence to generate a set of partial sequences. Since the two objectives are involved, the non-dominated sequences from the new generated sequences are selected. Then, in the second iteration, the second product in P^r is inserted into all possible positions of these non-dominated sequences. We repeat these steps until all the extracted products are reinserted. Note that the number of extracted products d has a great influence on the performance of the proposed RIPG. A high d value will add excessive diversification and result in a random walk, while a small d value makes it difficult to escape from local optima. Hence this value needs to be carefully calibrated as we will do in the experimentation of this study.

4.3. Bi-objective-oriented referenced local search

A bi-objective-oriented referenced local search is designed to improve the constructed solution in the greedy phase. Our local search procedure includes two improvements to the traditional local search. The first one is that the proposed method refers to the sequence of the solution from the *POS* to remove the products. The second one is to use Pareto dominance to update the temporary set. The procedure of the proposed local search is detailed as below:

Step 1: A solution is randomly selected from the *POS* and regarded as a reference solution $\Pi^r = \{\pi_1^r, \ldots, \pi_i^r, \ldots, \pi_n^r\}$. Set $i = 1$.

Step 2: Referring to the product π_i^r, this method removes the same product of the current solution and tests it in all possible positions. Accordingly, a set of solutions TS_i are generated.

Step 3: If there is at least one solution in TS_i updating the temporary set, go to Step 4 to update the temporary set; otherwise, terminate the process.

Step 4: The non-dominated solutions in TS_i are stored in the temporary set, and the remaining dominated solution in a temporary set is removed.

Step 5: Set $i = i + 1$. If $i < n$, return to Step 2 to continue with the improvement of the current solution. Otherwise, the algorithm terminates the process.

4.4. Bi-objective-oriented acceptance criterion

The proposed bi-objective-oriented acceptance criterion is similar to the method proposed by Zhang, et al. [49]. It mainly consists of two parts: the update of the *POS* and the acceptance judgment of the temporal set. For the update of the *POS*, we compare each solution in a temporal set with the solutions in the *POS*. If the solution is a non-dominated solution, it is placed into the *POS* and those solutions in *POS* dominated by the newly added solution are removed.

The acceptance judgment of the temporal set aims to decide whether any new generated solution can replace the incumbent solution for the next iteration. If any solution in the *POS* dominates the solutions in the temporal set, one solution is randomly selected from these new non-dominated solutions and accepted as the new incumbent one. Otherwise, each solution Π^{new} in the temporal set is accepted with two probabilities $exp((C_{max}(\Pi^{current}) - C_{max}(\Pi^{new}))/t)$ and $exp((TMC(\Pi^{current}) - TMC(\Pi^{new}))/t)$, where $t = T_0 \cdot \frac{\sum_{k=1}^{m_1+m_1} \sum_{j=1}^{n} t_{jk}}{10 \cdot n \cdot (m_1+m_2)}$. When more than one solution is accepted, the algorithm just selects one of them at random.

4.5. Restart mechanism

The RIPG employs a restart mechanism based on the crowding distance proposed by Minella, Ruiz and Ciavotta [14]. At each iteration, the algorithm counts the cumulative number of iterations without any improvement. If there are no non-dominated solutions generated at any given iteration, a counter *dn* is increased. When the counter value is higher than *DN*, the restart mechanism is applied with the final goal of selecting a non-dominated solution to replace the current solution for the subsequent iterations. The restart mechanism sets a *select_counter* to count the number of times a solution has been selected and uses the crowding distance [50] divided by *select_counter* to calculate a modified crowding distance to avoid selecting solutions repeatedly. The non-dominated solution with the larger value of modified crowding distance is selected.

5. Computational results and experimental discussion

This section carries out four sets of computational experiments. We describe in Section 5.1 the setup and performance indicators of the experimentation. The first set of experiments is conducted to calibrate the parameters of RIPG and determine the best parameter combination (Section 5.2). The subsequent sets of experiments aim to evaluate the performance of the MILP model and RIPG algorithm. Specifically, in the second set, we use the epsilon-constraint method to deal with the MILLP model and then solve it by CPLEX solver (Section 5.3). In the third set, we compare the RIPG with four well-known bi-objective metaheuristics (Section 5.4). In the fourth set, the differential empirical attainment function (Diff-EAF) is employed to show the differences between the empirical attainment functions (EAFs) obtained by the RIPG and other algorithms (Section 5.5).

5.1. Experimental setting and evaluation indicators

We first generate a set of benchmark instances to conduct the computational experiments. The benchmark consists of 10 replications of instances for different combinations of n, m_1 and m_2, where $n \in \{20, 40, 60, 80, 100\}$ and $m_1, m_2 \in \{2, 4, 6, 8\}$. Hence, a total of number 800 benchmark instances is obtained. By taking into account the parameter generation by Boufellouh and Belkaid [30], the processing times on fabrication and assembly machines are drawn from a uniform distribution [1, 100]. The PM time and cost are randomly generated between [1, 100] and [1, 200] respectively. tr_k and cr_k are distributed as U [tp_k+1, tp_k+400] and U [cp_k+1, cp_k+800], respectively. The shape parameter β_k is randomly selected from {2, 3, 4}, and the scale parameter θ_k is drawn from a uniform distribution [1000, 2000]. Besides, we also generate 40 calibration instances to calibrate the parameters of all the metaheuristics. We use IBM CPLEX solvers for the MILP model and code the RIPG algorithm in Visual Studio C++. The algorithms were run in a computer having an Intel^R Core™ i5 10210U processor running at 1.60 GHz with 16 GBytes of RAM.

Two unary and one binary multi-objective performance indicators are used in the experimentation: the hyper volume ratio (*HVR)* [51], the unary epsilon indicator (I_ε) [52] and the coverage indicator (C) [53]. For calculating the first two indicators we merge all the non-dominated solutions obtained by all the compared algorithms as the true Pareto front. *HVR,* calculated by Eq. (25), is the ratio between the hyper volume of the obtained Pareto set and that of the true Pareto set. Since this problem is not a theoretical one, we do not know the true Pareto set, so that we approximate it by merging all the obtained Pareto sets from the algorithms, as done in Chica, et al. [54]. In Eq. (25), n and m are the number of the obtained Pareto solutions and that of objectives respectively. v_i refers to the i^{th} hypercube, whose diagonal corners are the objective vector of solution i in the obtained Pareto set and that of reference point W. The reference point W is constructed as a vector of the worst possible objective values. The closer to 1 the HVR value of a Pareto set is, the better the approximation to the true frontier.

$$HVR = \frac{volume(\bigcup_{i=1}^{n} v_i)}{volume(\bigcup_{j=1}^{m} v_j)} \tag{25}$$

I_ε is calculated by Eq. (26) and measures the minimum distance between an obtained Pareto front set and the true frontier. S is an obtained Pareto front set and P is the true frontier. χ^1 and χ^2 are the solutions of S and P respectively, and f_j indicates the j^{th} objective function. The obtained Pareto front set with I_ε closer to 1 suggests that this front set is close to the true Pareto frontier.

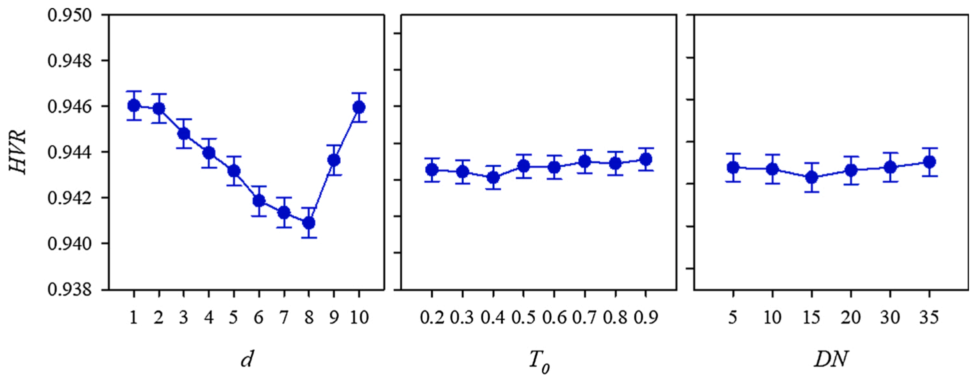

Fig. 2. Means plots of *HVR* with Tukey's Honest Significant Difference (HSD) 95 % confidence intervals for all the factors in the ANOVA calibration experiment for the proposed RIPG.

$$I_\varepsilon = I_\varepsilon(S,P) = \max_{\chi^2}\min_{\chi^1}\max_{j}\frac{f_j(\chi^1)}{f_j(\chi^2)} \tag{26}$$

Binary coverage C, calculated by Eq. (27), aims to measure the domination relation between two Pareto frontiers. $p \preccurlyeq q$ in Eq. (27) means that the solution p in Pareto frontier P weakly dominates the solution q in Pareto frontier Q. A $C(P,Q)$ value closer to 1 suggests that Pareto frontier Q is strongly dominated by Pareto frontier P. $C(P,Q) = 0$ means none of the solutions in Q are covered by the solutions of set P. Note that both $C(P,Q)$ and $C(Q,P)$ have to be considered, since $C(P,Q)$ is not necessarily equal to $1 - C(Q,P)$

$$C(P,Q) = \frac{|\{q \in Q;\ \exists p \in P : p \preccurlyeq q\}|}{|Q|} \tag{27}$$

5.2. Calibration of the proposed algorithms

This section employs the design of experiments technique coupled with multifactor analysis of variance (ANOVA) to determine the best parameter combination. The ANOVA is an important parametric statistical inference tool used to check the normality, homoscedasticity, and independence of the residuals. For the RIPG algorithm, three factors

Table 5
The Pareto solutions of the small-scale instances.

Instance			Features of the Pareto set of solutions				Instance			Features of the Pareto set of solutions			
n	m_1	m_2	Cardinality	Status	C_{max}	*TMC*	n	m_1	m_2	Cardinality	Status	C_{max}	*TMC*
Instance 1							Instance 54						
20	2	2	1	Optimal	1251.96	274.246	20	4	4	1	Optimal	1428.85	868.0120
Instance 4							Instance 61						
20	2	2	1	Optimal	1536.00	527.618	20	4	6	1	Feasible	1537.97	1513.920
			2	Optimal	1478.66	629.618				2	Feasible	1542.38	1306.920
Instance 11										3	Feasible	1546.68	1231.920
20	2	4	1	Optimal	1505.62	996.774				4	Optimal	2211.00	1226.920
			2	Optimal	1811.00	967.774	Instance 64						
Instance 14							20	4	6	1	Feasible	1315.96	2256.030
20	2	4	1	Optimal	1331.74	452.964				2	Feasible	1331.11	1776.030
Instance 21										3	Feasible	2153.00	1280.030
20	2	6	1	Optimal	1469.75	1655.310	Instance 81						
			2	Feasible	1473.05	1521.310	20	6	2	1	Optimal	1369.18	440.272
			3	Optimal	2058.00	1468.310	Instance 84						
Instance 24							20	6	2	1	Optimal	1343.86	115.180
20	2	6	1	Optimal	1359.33	1158.890				2	Optimal	1673.00	114.180
			2	Feasible	1364.85	1102.890	Instance 91						
			3	Optimal	1850.00	1060.890	20	6	4	1	Optimal	1468.30	673.567
Instance 31							Instance 94						
20	2	8	1	Feasible	1617.47	2645.950	20	6	4	1	Optimal	1390.82	513.552
			2	Feasible	1619.78	1743.950	Instance 101						
			3	Feasible	1638.45	1570.950	20	6	6	1	Feasible	1609.91	2747.250
			4	Feasible	2376.00	1371.950				2	Feasible	1611.85	1787.250
Instance 34										3	Feasible	2134.00	1618.250
20	2	8	1	Optimal	1794.86	1207.070	Instance 121						
			2	Feasible	2314.00	1023.070	20	8	2	1	Optimal	1404.13	278.264
Instance 41							Instance 124						
20	4	2	1	Optimal	1244.89	232.0730	20	8	2	1	Optimal	1258.41	1097.900
Instance 44							Instance 131						
20	4	2	1	Optimal	1232.50	252.7620	20	8	4	1	Feasible	1376.59	1374.450
Instance 51							Instance 134						
20	4	4	1	Optimal	1764.55	894.8860	20	8	4	1	Optimal	1562.40	856.629

Table 6
HVR and I_ε of the epsilon-constraint method and RIPG in some small-scale instances.

Instances	Epsilon-constraint		RIPG		Instances	Epsilon-constraint		RIPG	
	HVR	I_ε	*HVR*	I_ε		*HVR*	I_ε	*HVR*	I_ε
Instance 1	1.00	1.00	0.32	2.16	Instance 61	1.00	1.00	0.29	1.59
Instance 4	1.00	1.00	0.28	2.11	Instance 64	1.00	1.00	0.64	3.73
Instance 11	1.00	1.00	0.32	2.27	Instance 81	1.00	1.00	0.29	11.66
Instance 14	1.00	1.00	0.24	2.59	Instance 84	1.00	1.00	0.18	3.07
Instance 21	1.00	1.00	0.51	1.37	Instance 91	1.00	1.00	0.25	2.61
Instance 24	1.00	1.00	0.45	1.35	Instance 94	1.00	1.00	0.24	1.92
Instance 31	1.00	1.00	0.71	1.41	Instance 101	1.00	1.00	0.41	5.47
Instance 34	1.00	1.00	0.36	1.68	Instance 121	1.00	1.00	0.20	2.41
Instance 41	1.00	1.00	0.20	4.04	Instance 124	1.00	1.00	0.32	2.33
Instance 44	1.00	1.00	0.28	5.02	Instance 131	1.00	1.00	0.25	3.34
Instance 51	1.00	1.00	0.24	2.68	Instance 134	1.00	1.00	0.22	1.59
Instance 54	1.00	1.00	0.30	1.90	Avg.	**1.00**	**1.00**	0.33	2.98

need to be calibrated: the extracted number of products in destruction phase *d*, the initial temperature in acceptance criterion T_0, and the number of iterations before restart (*DN)*. The levels of these parameters are listed below:

- The extracted number of products in destruction phase *d* at 10 levels: 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10.
- The initial temperature in acceptance criterion T_0 at 8 levels: 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9.
- The number of iterations before restart (*DN)* at 6 levels: 5, 10, 15, 20, 30 and 35.

Through the full factorial design, there is a total of $10 \times 8 \times 6 = 480$ parameter combinations. Each combination is run with all the 40 calibration instances. Each instance is solved 10 times to obtain 10 different Pareto front sets. Hence, a total of $480 \times 40 \times 10 = 192{,}000$ experiments is carried out. The stopping criterion for all the experiments is set as a CPU time limit of $n \times (m_1 + m_2)$ milliseconds. The average *HVR* is regarded as the response value. Fig. 2 presents the *HVR* means plots with Tukey's Honest Significant Difference (HSD) 95 % confidence intervals for all the factors. From the analysis, we can state that the best set of parameters is {d=8, T_0=0.4, $DN = 15$}.

5.3. Evaluation of the proposed MILP model by a CPLEX solver

This section employs 23 small-scale instances to evaluate the proposed bi-objective MILP model. These instances are selected from the 800 benchmark instances. They involve 20 products and different numbers of fabrication and assembly machines. Since two objectives are minimized in the proposed model, the epsilon-constraint method is used to obtain the Pareto solutions of each instance. The epsilon-constraint method restricts the optimization of one objective to different scopes. It facilitates using the CPLEX solver to optimize the proposed model with respect to each objective, in order to obtain a set of solutions optimizing all the objectives in conflict.

Specifically, and for each instance, this method first makes use of the CPLEX solver to minimize the C_{max} objective. The optimal value is regarded as the lower bound of the C_{max} objective and the obtained *TMC* value is set as the upper bound of the *TMC* objective. Then, the method minimizes the *TMC* objective to obtain the lower bound of the *TMC* objective and the upper bound of the C_{max} objective. From the lower bound to the upper bound of *TMC* objective, this method splits the range up into several sub-ranges and names the break points as epsilon values. Finally, and under the constraint that the *TMC* value is limited to each sub-range, the MILP model obtains the Pareto set of solutions having objective C_{max} solved.

Let us consider instance 1 as an example. The lower and upper bounds of C_{max} and *TMC* are respectively (1251.96, 1611.00) and (274.246, 459. 246). The epsilon values of *TMC* are set as {274, 284, 294, 304, 314, 324, 334, 344, 354, 364, 374, 384, 394, 404, 414, 424, 434, 444, 454, 464}. The MILP model with the objective C_{max} is solved under different sub-ranges of *TMC*. Hence, a total of 19 running times is involved, and only one solution is found. We set the maximum running time of the CPLEX solver to 1800 s. Table 5 presents the final Pareto solutions of the above 23 small-scale instances. In this table, the *Optimal* status means that the CPLEX solver can find the optimal solution in

Table 7
Average *HVR* and I_ε at CPU time $n \times (m_1 + m_2) \times 5$ milliseconds.

Instance		NSGA-II		NSGA-III		MOSA		MOPSO		RIPG	
		HVR	I_ε	*HVR*	I_ε	*HVR*	I_ε	*HVR*	I_ε	*HVR*	I_ε
n	20	0.944	1.020	0.955	1.016	0.899	1.037	0.868	1.042	0.980	1.009
	40	0.944	1.020	0.958	1.014	0.919	1.031	0.881	1.038	0.977	1.009
	60	0.949	1.018	0.960	1.013	0.932	1.026	0.890	1.034	0.975	1.008
	80	0.950	1.017	0.961	1.012	0.933	1.026	0.890	1.032	0.975	1.008
	100	0.951	1.017	0.961	1.012	0.934	1.025	0.895	1.031	0.978	1.007
m_1	2	0.948	1.019	0.960	1.013	0.919	1.031	0.882	1.036	0.977	1.009
	4	0.950	1.018	0.963	1.012	0.925	1.028	0.890	1.033	0.978	1.008
	6	0.946	1.019	0.958	1.014	0.926	1.029	0.885	1.036	0.976	1.009
	8	0.947	1.019	0.955	1.015	0.925	1.029	0.882	1.036	0.977	1.008
m_2	2	0.948	1.018	0.960	1.013	0.923	1.030	0.885	1.035	0.977	1.009
	4	0.949	1.018	0.960	1.013	0.922	1.029	0.884	1.035	0.978	1.008
	6	0.947	1.019	0.958	1.014	0.923	1.030	0.884	1.036	0.976	1.009
	8	0.948	1.018	0.958	1.014	0.927	1.028	0.886	1.035	0.977	1.008
Avg.		0.948	1.018	0.959	1.014	0.924	1.029	0.885	1.035	**0.977**	**1.008**

Table 8
Average *HVR and* I_ε at CPU time $n \times (m_1 + m_2) \times 10$ milliseconds.

Instance		NSGA-II		NSGA-III		MOSA		MOPSO		RIPG	
		HVR	I_ε	*HVR*	I_ε	*HVR*	I_ε	*HVR*	I_ε	*HVR*	I_ε
n	20	0.959	1.015	0.965	1.012	0.895	1.038	0.856	1.045	0.981	1.008
	40	0.954	1.016	0.963	1.012	0.914	1.032	0.873	1.040	0.980	1.008
	60	0.954	1.016	0.964	1.011	0.927	1.028	0.885	1.035	0.979	1.008
	80	0.957	1.015	0.967	1.009	0.931	1.027	0.889	1.032	0.980	1.007
	100	0.957	1.015	0.968	1.009	0.929	1.026	0.889	1.032	0.980	1.006
m₁	2	0.958	1.015	0.967	1.010	0.916	1.031	0.876	1.038	0.979	1.008
	4	0.959	1.014	0.969	1.010	0.922	1.029	0.885	1.035	0.982	1.007
	6	0.952	1.017	0.963	1.012	0.919	1.031	0.877	1.038	0.979	1.008
	8	0.955	1.015	0.963	1.011	0.920	1.030	0.876	1.037	0.980	1.007
m_2	2	0.955	1.015	0.965	1.011	0.918	1.031	0.879	1.037	0.979	1.008
	4	0.958	1.015	0.966	1.011	0.917	1.030	0.878	1.037	0.981	1.007
	6	0.955	1.016	0.965	1.011	0.919	1.031	0.877	1.038	0.981	1.007
	8	0.956	1.015	0.965	1.011	0.923	1.029	0.879	1.037	0.979	1.007
Avg.		0.956	1.015	0.965	1.011	0.919	1.030	0.878	1.037	**0.980**	**1.007**

Table 9
Average *HVR and* I_ε at CPU time $n \times (m_1 + m_2) \times 20$ milliseconds.

Instance		NSGA-II		NSGA-III		MOSA		MOPSO		RIPG	
		HVR	I_ε	*HVR*	I_ε	*HVR*	I_ε	*HVR*	I_ε	*HVR*	I_ε
n	20	0.966	1.012	0.969	1.010	0.896	1.037	0.854	1.046	0.984	1.007
	40	0.959	1.015	0.966	1.011	0.910	1.033	0.866	1.042	0.982	1.007
	60	0.959	1.014	0.967	1.010	0.924	1.028	0.881	1.036	0.982	1.007
	80	0.961	1.013	0.969	1.009	0.927	1.028	0.881	1.034	0.982	1.006
	100	0.963	1.012	0.970	1.009	0.926	1.027	0.884	1.033	0.982	1.006
m₁	2	0.962	1.013	0.968	1.010	0.913	1.032	0.870	1.039	0.981	1.007
	4	0.965	1.013	0.972	1.008	0.919	1.029	0.880	1.036	0.984	1.006
	6	0.959	1.015	0.966	1.011	0.916	1.031	0.871	1.040	0.982	1.007
	8	0.960	1.013	0.965	1.010	0.918	1.030	0.872	1.038	0.982	1.007
m_2	2	0.962	1.013	0.969	1.009	0.917	1.031	0.874	1.038	0.982	1.007
	4	0.961	1.013	0.967	1.010	0.913	1.031	0.873	1.038	0.982	1.006
	6	0.962	1.013	0.968	1.010	0.917	1.031	0.873	1.039	0.983	1.007
	8	0.961	1.013	0.968	1.010	0.919	1.029	0.873	1.038	0.981	1.007
Avg.		0.961	1.013	0.968	1.010	0.917	1.031	0.873	1.038	**0.982**	**1.007**

Table 10
Average coverage for RIPG and metaheuristics at $n \times (m_1 + m_2) \times 5$ milliseconds.

Instance		*C*(RIPG, MOPSO)	*C*(RIPG, MOSA)	*C*(RIPG, NSGA-II)	*C*(RIPG, NSGA-III)
n	20	0.281	0.356	0.274	0.228
	40	0.498	0.550	0.458	0.341
	60	0.559	0.562	0.498	0.408
	80	0.694	0.713	0.632	0.487
	100	0.724	0.714	0.672	0.578
m₁	2	0.494	0.537	0.462	0.388
	4	0.578	0.599	0.515	0.397
	6	0.623	0.640	0.572	0.450
	8	0.509	0.540	0.479	0.398
m_2	2	0.553	0.579	0.520	0.423
	4	0.523	0.568	0.479	0.383
	6	0.577	0.582	0.500	0.404
	8	0.551	0.586	0.528	0.424
Avg.		0.551	0.579	0.507	0.408

Table 11
Average coverage for RIPG and metaheuristics at $n \times (m_1 + m_2) \times 10$ milliseconds.

Instance		*C*(RIPG, MOPSO)	*C*(RIPG, MOSA)	*C*(RIPG, NSGA-II)	*C*(RIPG, NSGA-III)
n	20	0.291	0.361	0.222	0.163
	40	0.489	0.537	0.370	0.256
	60	0.575	0.574	0.430	0.314
	80	0.704	0.717	0.589	0.416
	100	0.726	0.720	0.635	0.505
m₁	2	0.502	0.544	0.414	0.318
	4	0.578	0.600	0.456	0.322
	6	0.626	0.635	0.507	0.368
	8	0.520	0.548	0.420	0.315
m_2	2	0.564	0.582	0.471	0.359
	4	0.527	0.578	0.420	0.311
	6	0.565	0.579	0.439	0.313
	8	0.571	0.589	0.467	0.341
Avg.		0.557	0.582	0.449	0.331

1800 s. *Feasible* status means the solver can get a feasible solution but not the optimal one. From this table we can observe that, when the product number is larger than 20 or the assembly machine numbers are larger than 4, it is difficult for the CPLEX solver to find the optimal Pareto solutions at the given time. As the instance scale increases, the performance of the solver is decreasing.

Besides, this section also compares the results obtained by the epsilon-constraint method and the proposed RIPG and evaluates their performance through the indicators *HVR and* I_ε. The compared results are presented in Table 6. It can be observed that the epsilon-constraint method outperforms the RIPG in small-scale instances. However, the former takes more CPU time than the latter. The RPIG algorithm has a stopping criterion of $n \times (m_1 + m_2) \times 20$ milliseconds. We also see that when the scale of the instances increases, it is difficult to find feasible solutions by the epsilon-constraint method.

Therefore, the main conclusion is that the proposed model can be solved by a mathematical solver when the number of products and assembly machines do not exceed 20 and 4, respectively. Besides, for large-scale instances, we should use a metaheuristic method to obtain the Pareto set of solutions. Hence, the subsequent sections discuss the

Table 12
Average coverage for RIPG and metaheuristics at $n \times (m_1 + m_2) \times 20$ milliseconds.

Instance		C(RIPG, MOPSO)	C(RIPG, MOSA)	C(RIPG, NSGA-II)	C(RIPG, NSGA-III)
	20	0.307	0.360	0.183	0.124
	40	0.493	0.536	0.315	0.198
n	60	0.573	0.588	0.404	0.268
	80	0.713	0.723	0.523	0.345
	100	0.738	0.719	0.584	0.423
	2	0.527	0.557	0.372	0.262
m_1	4	0.586	0.600	0.404	0.259
	6	0.620	0.634	0.446	0.297
	8	0.526	0.551	0.385	0.269
	2	0.572	0.590	0.423	0.288
m_2	4	0.537	0.582	0.363	0.242
	6	0.582	0.574	0.394	0.270
	8	0.569	0.595	0.428	0.287
Avg.		0.565	0.585	0.402	0.272

Table 13
Average coverage for metaheuristics and RIPG at $n \times (m_1 + m_2) \times 5$ milliseconds.

Instance		C(MOPSO, RIPG)	C(MOSA, RIPG)	C(NSGA-II, RIPG)	C(NSGA-III, RIPG)
	20	0.000	0.000	0.000	0.005
	40	0.000	0.000	0.001	0.013
n	60	0.001	0.000	0.000	0.008
	80	0.000	0.000	0.001	0.017
	100	0.000	0.000	0.002	0.014
	2	0.000	0.000	0.001	0.013
m_1	4	0.000	0.000	0.001	0.014
	6	0.000	0.000	0.000	0.011
	8	0.000	0.000	0.001	0.008
	2	0.000	0.000	0.000	0.009
m_2	4	0.000	0.000	0.002	0.010
	6	0.000	0.000	0.000	0.012
	8	0.000	0.000	0.001	0.015
Avg.		0.000	0.000	0.001	0.011

Table 14
Average coverage for metaheuristics and RIPG at $n \times (m_1 + m_2) \times 10$ milliseconds.

Instance		C(MOPSO, RIPG)	C(MOSA, RIPG)	C(NSGA-II, RIPG)	C(NSGA-III, RIPG)
	20	0.000	0.000	0.000	0.007
	40	0.000	0.000	0.001	0.010
n	60	0.000	0.000	0.001	0.019
	80	0.000	0.000	0.000	0.024
	100	0.001	0.000	0.003	0.029
	2	0.000	0.000	0.001	0.013
m_1	4	0.000	0.000	0.001	0.023
	6	0.000	0.000	0.000	0.017
	8	0.000	0.000	0.001	0.018
	2	0.000	0.000	0.001	0.016
m_2	4	0.000	0.000	0.001	0.022
	6	0.000	0.000	0.001	0.013
	8	0.000	0.000	0.001	0.021
Avg.		0.000	0.000	0.001	0.018

performance of the proposed RIPG algorithm to solve the scheduling problem under larger and more realistic instances.

5.4. Performance comparison of RIPG and other multi-objective metaheuristics

This section compares the performance of the RIPG algorithm with respect to four well-known multi-objective metaheuristics: NSGA-II [23], NSGA-III [55], MOSA [23] and MOPSO [55]. We have selected these specific metaheuristics due to their successful application in two-stage or three-stage assembly flow shop scheduling. To apply these metaheuristics into $DPm \rightarrow Fm|PM\&CM|(C_{max}, TMC)$, the proposed solution evaluation is used to determine the product sequence and PM execution time points and calculate the objective values. The stopping criteria are set to the CPU time limit (in milliseconds). This CPU time limit depends on the value of p, which can be 5, 10 and 20, and is calculated by $n \times (m_1 + m_2) \times p$. 800 benchmark instances are solved by these metaheuristics. Each metaheuristic is run 10 independent times to obtain 10 Pareto sets of solutions.

Tables 7–9 respectively report the average HVR and I_ε values of these metaheuristics for different CPU time limits (three p values). From these tables, we can see how the HVR and I_ε values of RIPG are close to 1 for all

Table 15
Average coverage for metaheuristics and RIPG at $n \times (m_1 + m_2) \times 20$ milliseconds.

Instance		C(MOPSO, RIPG)	C(MOSA, RIPG)	C(NSGA-II, RIPG)	C(NSGA-III, RIPG)
	20	0.000	0.000	0.005	0.011
	40	0.000	0.000	0.003	0.018
n	60	0.000	0.000	0.002	0.030
	80	0.000	0.000	0.003	0.030
	100	0.000	0.000	0.001	0.032
	2	0.000	0.000	0.002	0.016
m_1	4	0.000	0.000	0.003	0.032
	6	0.000	0.000	0.003	0.029
	8	0.000	0.000	0.003	0.018
	2	0.000	0.000	0.005	0.028
m_2	4	0.000	0.000	0.001	0.020
	6	0.000	0.000	0.002	0.018
	8	0.000	0.000	0.004	0.030
Avg.		0.000	0.000	0.003	0.024

Table 16
ANOVA results for the metaheuristic types.

Sources	df	Type III sum of squares	Mean square	F-ratio	P-value
$n \times (m_1 + m_2) \times 5$					
HVR					
Metaheuristic types	4	40.75	10.1879	4601.99	<0.001
Error	39995	88.54	0.0022		
Total	39999	129.29			
I_ε					
Metaheuristic types	4	3.939	0.984808	3881.87	<0.001
Error	39995	10.147	0.000254		
Total	39999	14.086			
$n \times (m_1 + m_2) \times 10$					
HVR					
Metaheuristic types	4	53.83	13.4569	5720.03	<0.001
Error	39995	94.09	0.0024		
Total	39999	147.92			
I_ε					
Metaheuristic types	4	5.223	1.30585	4864.60	<0.001
Error	39995	10.736	0.00027		
Total	39999	15.960			
$n \times (m_1 + m_2) \times 20$					
HVR					
Metaheuristic types	4	64.26	16.0661	6598.49	<0.001
Error	39995	97.38	0.0024		
Total	39999	161.64			
I_ε					
Metaheuristic types	4	6.205	1.55126	5542.42	<0.001
Error	39995	11.194	0.00028		
Total	39999	17.399			

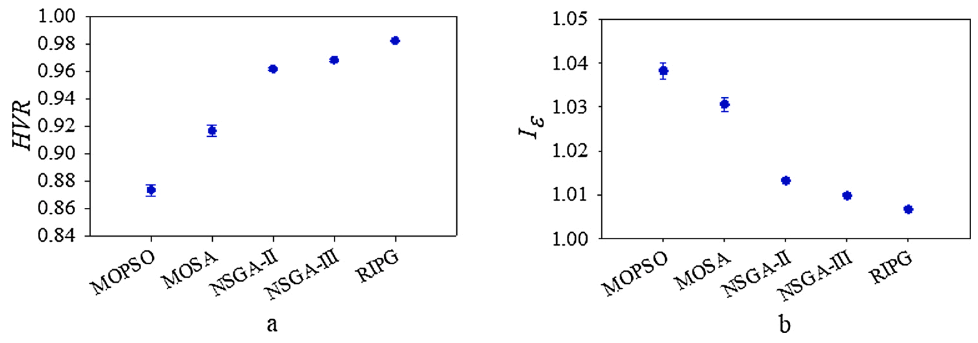

Fig. 3. Means plots of *HVR* (a) and I_ε (b) with Tukey's Honest Significant Difference (HSD) 95 % confidence intervals for all the metaheuristics at CPU time $n \times (m_1 + m_2) \times 20$ milliseconds.

the instances and stopping criteria. This observation indicates that the RIPG outperforms other multi-objective metaheuristics when tackling $DPm{\rightarrow}Fm|PM\&CM|(C_{max}, TMC)$ problem. Specifically, and regarding *HVR*, the average values of RIPG under three stopping criteria are 0.977, 0.980, and 0.982, respectively. These values are better than those from NSGA-II, NSGA-III, MOPSO and MOSA. Apart from RIPG, the NSGA-III and NSGA-II outperform the other two algorithms (i.e., MOPSO and MOSA).

With respect to the I_ε values, the proposed RIPG again obtains the best values for the three stopping criteria limits. RIPG ranks the first one, followed by NSGA-III, NSGA-II, MOSA, and MOPSO. Therefore, we can conclude that the proposed RIPG has the best convergence and diversity for all the analyzed instances and stopping criteria values.

Tables 10–15 report the average *C*(RIPG, metaheuristic) and *C* (metaheuristic, RIPG) values for different CPU time limits (three *p* values). Taking, as an example, the CPU time limits of $n \times (m_1 + m_2) \times 5$, the average values of *C*(RIPG, MOPSO), *C*(RIPG, MOSA), *C*(RIPG, NSGA-II) and *C*(RIPG, NSGA-III) are 0.551, 0.579, 0.507 and 0.408. These results mean that almost half of the solutions obtained by MOPSO, MOSA, NSGA-II and NSGA-III, are dominated by the solutions of the

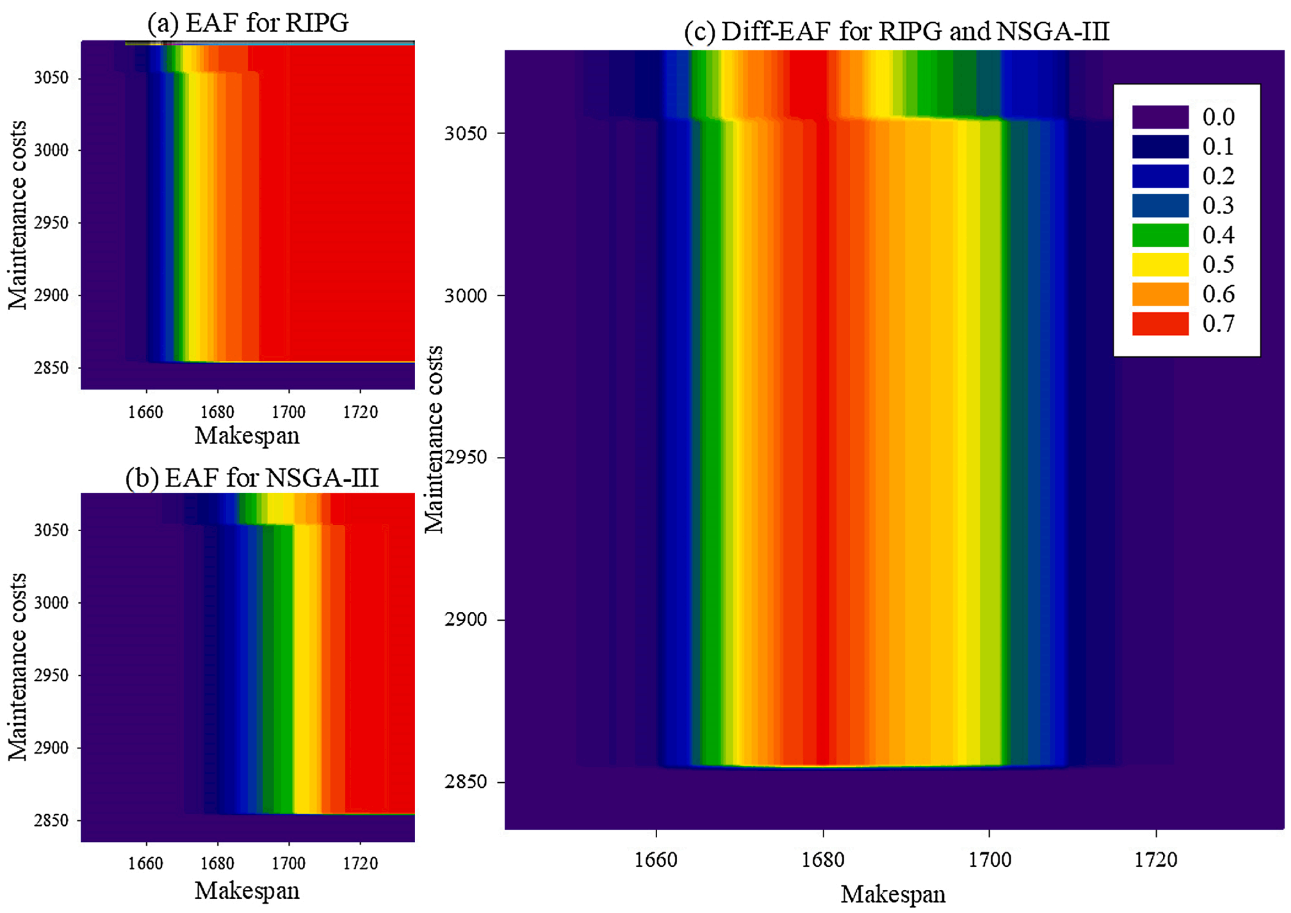

Fig. 4. Empirical Attainment Functions and the Differences between Empirical Attainment Function for RIPG and NSGA-III for instance 146. (For interpretation of the references to colour in this figure text, the reader is referred to the web version of this article.)

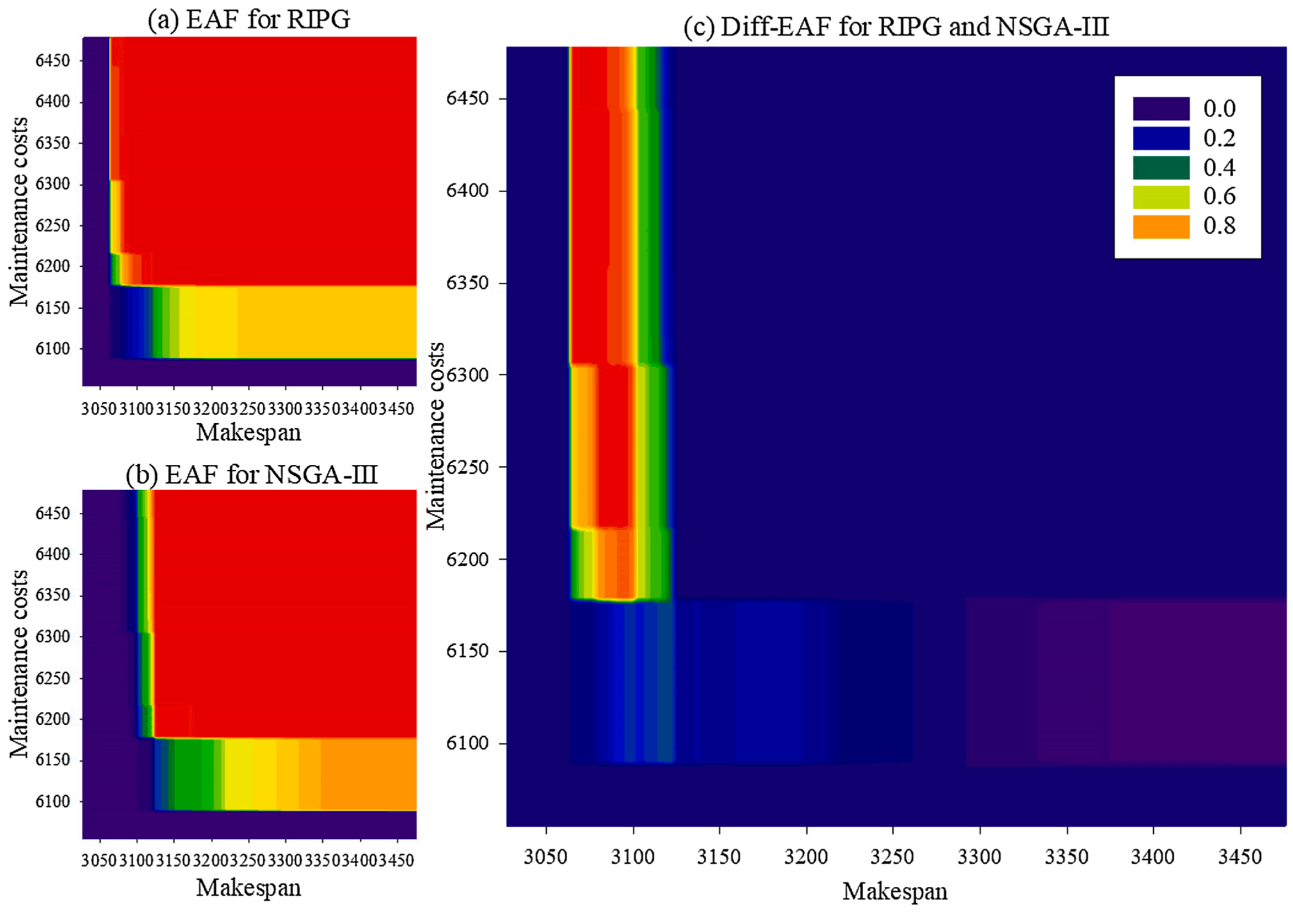

Fig. 5. Empirical Attainment Functions and the Differences between Empirical Attainment Function for RIPG and NSGA-III for instance 261. (For interpretation of the references to colour in this figure text, the reader is referred to the web version of this article.)

Pareto set obtained by the RIPG algorithm. The average values of C (MOPSO, RIPG), C(MOSA, RIPG), C(NSGA-II, RIPG) and C(NSGA-III, RIPG) are 0, 0, 0.001, and 0.011, respectively. These results suggest that the Pareto front solutions, generated by MOPSO, MOSA, NSGA-II and NSGA-III, hardly dominate those generated by the RIPG algorithm. The RIPG algorithm is therefore superior to MOPSO, MOSA, NSGA-II and NSGA-III in tackling $Pm \rightarrow Fm|PM\&CM|(C_{max}, TMC)$ problem.

Additionally, we use the ANOVA test to statistically confirm the significant differences between the analyzed multi-objective metaheuristics. After conducting the normality test and equal variance test, the HVR and I_ε are regarded as the response variables. The five types of metaheuristics (RIPG, MOPSO, MOSA, NSGA-II, NSGA-III) are regarded as the controlling factor. Table 16 shows the final ANOVA results. Fig. 3 presents means plots of HVR (a) and I_ε (b) with Tukey's Honest Significant Difference (HSD) 95 % confidence intervals for all the metaheuristics with a CPU time limit of $n \times (m_1 + m_2) \times 20$ milliseconds. By analyzing the values of Table 16 we can see that, under the three stopping criteria and two indicators, the p-values of all metaheuristic are less than 0.001. This fact means that all the metaheuristics have a significant effect on the performance of $DPm \rightarrow Fm|PM\&CM|(C_{max}, TMC)$ problem. To sum up, we can reinforce, from Fig. 3, the previous conclusion: the proposed RIPG outperforms NSGA-III, NSGA-II, MOSA, and MOPSO, with statistical significance.

5.5. Differential empirical attainment functions

In this section we qualitatively explore the obtained Pareto sets from the metaheuristics using empirical attainment function (EAF) and differential empirical attainment function (Diff-EAF) for the RIPG algorithm and the second-best metaheuristic, the NSGA-II. We employ this methodology for the benchmark instances 146, 261, and 352. To do so, the RIPG algorithm and NSGA-III are run for 100 times to obtain 100 Pareto front sets for each instance. Figs. 4–6 show the EAFs and Diff-EAFs of RIPG and NSGA-III and Diff-EAF of RIPG and NSGA-III, following the approach of Grunert da Fonseca, Fonseca and Hall [56]. EAF plots are placed in Figs. 4–6(a–b) where values close to 1 (i.e., area colored in red and orange) mean a high dominance of the corresponding algorithm. In contrast, those areas with light colors, such as blue and purple, mean low or null dominance of the corresponding algorithm. Diff-EAF plots are placed in Figs. 4–6(c), where different colors indicate different dominant probability by RIPG over NSGA-III (e.g., values close to 1 and colored in red mean RIPG totally dominates NSGA-III).

As an example, take instance 261 from Fig. 5(a–b). We can observe that those areas dominated by RIPG are larger than those by NSGA-III. From Fig. 5(c), we see that most of the borders around the makespan objective are colored in blue. This means that, in these areas, most of the Pareto front solutions can be found by RIPG, while NSGA-III can only get a few Pareto solutions. In the boundaries around the maintenance costs objective, most of them are colored in red, meaning the dominance probability of these zones by RIPG is 1.0. This observation indicates that almost all the Pareto front solutions are obtained by the RIPG algorithm. This qualitative analysis ends to the same previous conclusion: the proposed RIPG outperforms NSGA-III for both the makespan and maintenance costs objective spaces.

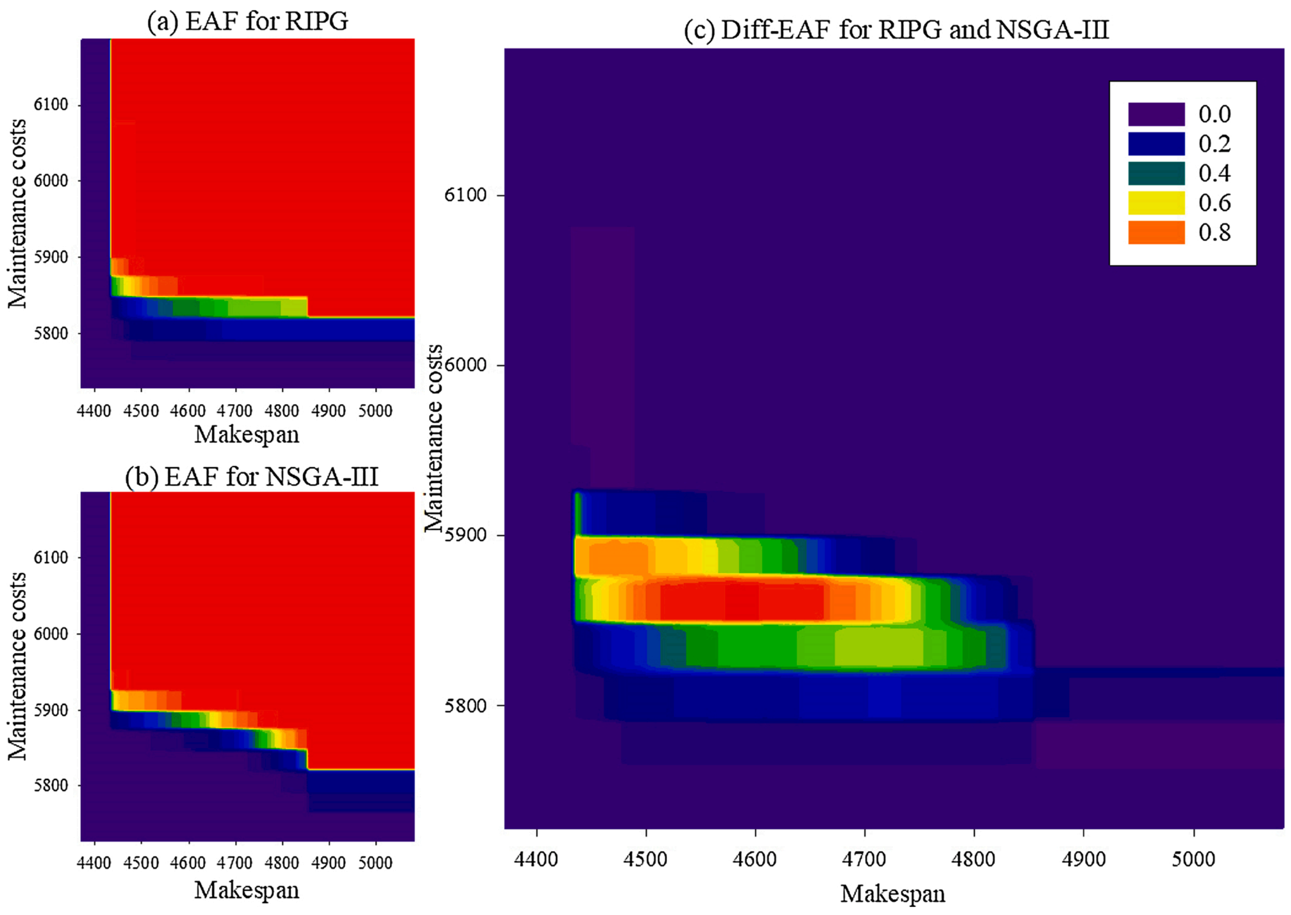

Fig. 6. Empirical Attainment Functions and the Differences between Empirical Attainment Function for RIPG and NSGA-III for instance 352. (For interpretation of the references to colour in this figure text, the reader is referred to the web version of this article.)

6. Conclusions, managerial insights, and future work

Given the importance of production scheduling and maintenance planning we studied in this paper a multicriteria assembly permutation flow shop scheduling problem. The MILP model considers the makespan as the first objective and a combination of PM and CM costs as the second objective. This work has also presented a new RIPG algorithm to deal with the bi-objective scheduling problem $DPm \rightarrow Fm|PM\&CM|(C_{max}, TMC)$. In our proposed model, the unexpected failure times lead to the non-linear property and the model cannot be directly solved by a commercial solver. Accordingly, two lemmas are inferred to relax the expected failure numbers and CM cost, and therefore, to make the model linearized.

The RIPG algorithm includes a new solution evaluation to determine the maintenance planning and calculate the objective values. Four improvements are proposed to enhance the performance of the RIPG algorithm: bi-objective-oriented greedy search procedure, local search, bi-objective-oriented acceptance criterion, and a restart mechanism. Through comprehensive experimentation, the ANOVA method is used to determine the best parameter combination of RIPG, and three types of experiments are conducted to evaluate the proposed MILP model and RIPG. The computational results and statistical analysis lead to the following main three conclusions:

(1) Through the epsilon-constraint method, the MILP model can be used to tackle the bi-objective problem, and it is effective when solving small-scale instances.
(2) The proposed RIPG outperforms four well-known multi-objective metaheuristics (namely, NSGA-II, NSGA-III, MOSA and MOPSO) for all the instances. The results are statistically significant.
(3) According to the empirical attainment functions, the proposed RIPG has a superior performance when jointly considering maintenance costs and makespan in this bi-objective scheduling problem.

We can obtain some managerial insights from the model and results of our work. First, the implementation of $DPm \rightarrow Fm|PM\&CM|(C_{max}, TMC)$ can improve production efficiency and reduce maintenance costs. When implementing the proposed method in real production, managers first refer to the historical maintenance data to determine the scale and shape parameters of all machines. Later, they determine the preventive maintenance interval and expected numbers of machine failures. When all the production and maintenance parameters are obtained, managers could use the epsilon-constraint method to solve the proposed model to obtain equally-preferred solutions (i.e., those from the Pareto set of solutions) in the case of small-scale instances (up to 20 products and 4 assembly machines). In the case of middle and large-scale instances, managers could use the proposed RIPG algorithm. Finally and according to the manager's importance level for the different objectives, the decision-makers can select one solution from the Pareto set of solutions (including both a product sequence and PM plans).

Future research can consider setup times or release time in $DPm \rightarrow Fm|PM\&CM|(C_{max}, TMC)$. From the methodology point of view, researchers can extend a Monte Carlo sampling method to deal with unexpected failure times. Other bio-inspired multi-objective metaheuristics can be employed to obtain better Pareto front approximations.

Declaration of Competing Interest

The authors report no declarations of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 51875421). M. Chica is jointly supported by the Spanish Ministry of Science, Andalusian Government, the National Agency for Research Funding AEI, and ERDF under grants EXASOCO (PGC2018-101216-B-I00), SIMARK (P18-TP-4475) and RYC-2016-19800.

References

[1] Lee C-Y, Cheng TCE, Lin BMT. Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem. Manage Sci 1993;39(5):616–25.
[2] Potts CN, Sevast'janov SV, Strusevich VA, Van Wassenhove LN, Zwaneveld CM. The two-stage assembly scheduling problem: complexity and approximation. Oper Res 1995;43(2):346–55.
[3] Allahverdi A, Aydilek H. The two stage assembly flowshop scheduling problem to minimize total tardiness. J Intell Manuf 2013;26(2):225–37.
[4] Yokoyama M. Scheduling for two-stage production system with setup and assembly operations. Comput Oper Res 2004;31(12):2063–78.
[5] Fattahi P, Hosseini SMH, Jolai F. A mathematical model and extension algorithm for assembly flexible flow shop scheduling problem. Int J Adv Manuf Technol 2013;65(5–8):787–802.
[6] Liao C-J, Lee C-H, Lee H-C. An efficient heuristic for a two-stage assembly scheduling problem with batch setup times to minimize makespan. Comput Ind Eng 2015;88:317–25.
[7] Allahverdi A, Al-Anzi FS. The two-stage assembly scheduling problem to minimize total completion time with setup times. Comput Oper Res 2009;36(10):2740–7.
[8] Zhang Y, Zhou Z, Liu J. The production scheduling problem in a multi-page invoice printing system. Comput Oper Res 2010;37(10):1814–21.
[9] Framinan JM, Perez-Gonzalez P, Fernandez-Viagas V. Deterministic assembly scheduling problems: a review and classification of concurrent-type scheduling models and solution procedures. Eur J Oper Res 2019;273(2):401–17.
[10] Al-Anzi FS, Allahverdi A. An artificial immune system heuristic for two-stage multi-machine assembly scheduling problem to minimize total completion time. J Manuf Syst 2013;32(4):825–30.
[11] Moghaddam KS. Multi-objective preventive maintenance and replacement scheduling in a manufacturing system using goal programming. Int J Prod Econ 2013;146(2):704–16.
[12] Tasgetiren MF, Kizilay D, Pan QK, Suganthan PN. Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. Comput Oper Res 2017;77:111–26.
[13] Ruiz R, Stutzle T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur J Oper Res 2007;177(3):2033–49.
[14] Minella G, Ruiz R, Ciavotta M. Restarted Iterated Pareto Greedy algorithm for multi-objective flowshop scheduling problems. Comput Oper Res 2011;38(11):1521–33.
[15] Koulamas C, Kyparisis GJ. The three-stage assembly flowshop scheduling problem. Comput Oper Res 2001;28(7):689–704.
[16] Andrés C, Hatami S. The three stage assembly permutation flowshop scheduling problem. V international conference on industrial engineering and industrial management 2011:867–75.
[17] Campos SC, Arroyo JEC, Tavares RG. A general VNS heuristic for a three-stage assembly flow shop scheduling problem. International Conference on Intelligent Systems Design and Applications 2016:955–64.
[18] Komaki GM, Teymourian E, Kayvanfar V, Booyavi Z. Improved discrete cuckoo optimization algorithm for the three-stage assembly flowshop scheduling problem. Comput Ind Eng 2017;105:158–73.
[19] Hatami S, Ebrahimnejad S, Tavakkoli-Moghaddam R, Maboudian Y. Two meta-heuristics for three-stage assembly flowshop scheduling with sequence-dependent setup times. Int J Adv Manuf Technol 2010;50(9–12):1153–64.
[20] Maleki-Darounkolaei A, Modiri M, Tavakkoli-Moghaddam R, Seyyedi I. A three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times. J Ind Eng Int 2012;8(1):26.
[21] Maleki A, Seyedi I. Taguchi method for three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times. J Eng Sci Technol 2013;8(5):603–22.
[22] Wang K, Ma WQ, Luo H, Qin H. Coordinated scheduling of production and transportation in a two-stage assembly flowshop. Int J Prod Res 2016;54(22):6891–911.
[23] Shoaardebili N, Fattahi P. Multi-objective meta-heuristics to solve three-stage assembly flow shop scheduling problem with machine availability constraints. Int J Prod Res 2014;53(3):944–68.
[24] Campos S.C., Arroyo J.E.C. 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* (pp. 429-436).
[25] Tajbakhsh Z, Fattahi P, Behnamian J. Multi-objective assembly permutation flow shop scheduling problem: a mathematical model and a meta-heuristic algorithm. J Oper Res Soc 2014;65(10):1580–92.
[26] Sheikh S, Komaki GM, Kayvanfar V, Teymourian E. Multi-Stage assembly flow shop with setup time and release time. Oper Res Perspect 2019;6.
[27] Xiong F, Xing K, Wang F. Scheduling a hybrid assembly-differentiation flowshop to minimize total flow time. Eur J Oper Res 2015;240(2):338–54.
[28] Seidgar H, Zandieh M, Fazlollahtabar H, Mahdavi I. Simulated imperialist competitive algorithm in two-stage assembly flow shop with machine breakdowns and preventive maintenance. Proc Inst Mech Eng Part B-J Eng Manuf 2016;230(5):934–53.
[29] Seidgar H, Zandieh M, Mahdavi I. Bi-objective optimization for integrating production and preventive maintenance scheduling in two-stage assembly flow shop problem. J Ind Prod Eng 2016;33(6):404–25.
[30] Boufellouh R, Belkaid F. Bi-objective optimization algorithms for joint production and maintenance scheduling under a global resource constraint: application to the permutation flow shop problem. Comput Oper Res 2020;122.
[31] Ma Y, Chu C, Zuo C. A survey of scheduling with deterministic machine availability constraints. Comput Ind Eng 2010;58(2):199–211.
[32] Hadda H. A polynomial-time approximation scheme for the two machine flow shop problem with several availability constraints. Optim Lett 2011;6(3):559–69.
[33] Gara-Ali A, Espinouse M-L. Erratum to: "Simultaneously scheduling n jobs and the preventive maintenance on the two-machine flow shop to minimize the makespan" [Int. J. Prod. Econ. 112 (2008) 161–167]. Int J Prod Econ 2014;153:361–3.
[34] Vahedi-Nouri B, Fattahi P, Tavakkoli-Moghaddam R, Ramezanian R. A general flow shop scheduling problem with consideration of position-based learning effect and multiple availability constraints. Int J Adv Manuf Technol 2014;73(5–8):601–11.
[35] Hadda H. A note on "Simultaneously scheduling n jobs and the preventive maintenance on the two-machine flow shop to minimize the makespan". Int J Prod Econ 2015;159:221–2.
[36] Ruiz R, Carlos García-Díaz J, Maroto C. Considering scheduling and preventive maintenance in the flowshop sequencing problem. Comput Oper Res 2007;34(11):3314–30.
[37] Wang S, Liu M. Two-stage hybrid flow shop scheduling with preventive maintenance using multi-objective tabu search method. Int J Prod Res 2014;52(5):1495–508.
[38] Khamseh A, Jolai F, Babaei M. Integrating sequence-dependent group scheduling problem and preventive maintenance in flexible flow shops. Int J Adv Manuf Technol 2015;77(1–4):173–85.
[39] Yu AJ, Seif J. Minimizing tardiness and maintenance costs in flow shop scheduling by a lower-bound-based GA. Comput Ind Eng 2016;97:26–40.
[40] Miyata HH, Nagano MS, Gupta JND. Integrating preventive maintenance activities to the no-wait flow shop scheduling problem with dependent-sequence setup times and makespan minimization. Comput Ind Eng 2019;135:79–104.
[41] Sheikhalishahi M, Eskandari N, Mashayekhi A, Azadeh A. Multi-objective open shop scheduling by considering human error and preventive maintenance. Appl Math Model 2019;67:573–87.
[42] Yu T-S, Han J-H. Scheduling proportionate flow shops with preventive machine maintenance. Int J Prod Econ 2021;231:107874.
[43] Ghodratnama A, Rabbani M, Tavakkoli-Moghaddam R, Baboli A. Solving a single-machine scheduling problem with maintenance, job deterioration and learning effect by simulated annealing. J Manuf Syst 2010;29(1):1–9.
[44] Hu J, Jiang Z, Liao H. Joint optimization of job scheduling and maintenance planning for a two-machine flow shop considering job-dependent operating condition. J Manuf Syst 2020;57:231–41.
[45] Wang SJ, Liu M. Multi-objective optimization of parallel machine scheduling integrated with multi-resources preventive maintenance planning. J Manuf Syst 2015;37:182–92.
[46] Pan E, Liao W, Xi L. Single-machine-based production scheduling model integrated preventive maintenance planning. Int J Adv Manuf Technol 2010;50(1–4):365–75.
[47] Cui WW, Lu ZQ, Li C, Han XL. A proactive approach to solve integrated production scheduling and maintenance planning problem in flow shops. Comput Ind Eng 2018;115:342–53.
[48] Ye H, Wang X, Liu K. Adaptive preventive maintenance for flow shop scheduling with resumable processing. IEEE Trans Autom Sci Eng 2020:1–8.
[49] Zhang ZK, Tang QH, Ruiz R, Zhang LP. Ergonomic risk and cycle time minimization for the U-shaped worker assignment assembly line balancing problem: a multi-objective approach. Comput Oper Res 2020;118.
[50] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. Ieee Trans Evol Comput 2002;6(2):182–97.
[51] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. Ieee Trans Evol Comput 1999;3(4):257–71.
[52] Knowles JD, Thiele L, Zitzler E. A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK-Report. 2006. 214.
[53] Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 2000;8(2):173–95.
[54] Chica M, Cordón Ó, Damas S, Bautista J. Multiobjective constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and random greedy search. Inf Sci 2010;180(18):3465–87.
[55] Sheikh S, Komaki GM, Kayvanfar V. Multi objective two-stage assembly flow shop with release time. Comput Ind Eng 2018;124:276–92.
[56] Grunert da Fonseca V, Fonseca CM, Hall AO. Inferential performance assessment of stochastic optimisers and the attainment function. 1993. 2001. p. 213–25.