The International Journal of Advanced Manufacturing Technology (2018) 98:1817–1830
https://doi.org/10.1007/s00170-018-2171-y

ORIGINAL ARTICLE

New meta-heuristic for dynamic scheduling in permutation flowshop with new order arrival

Weibo Liu[1,2] · Yan Jin[2] · Mark Price[2]

Received: 24 February 2018 / Accepted: 13 May 2018 / Published online: 26 June 2018

Abstract
Permutation flowshop scheduling problem (PFSP) has attracted lots of attention from both academia and industry as it finds many applications especially for today's mass customization. The PFSP is proved NP-hard, and the dynamic uncertainties such as stochastic new order arrivals significantly increase the problem complexity and difficulty. Many enterprises often struggle to make decisions on accepting new orders and setting due dates for them due to lack of effective scheduling methods. To fill in the knowledge gap, this paper is to propose a new meta-heuristic algorithm which is based on a new enhanced destruction and construction method and a novel repair method while adopting the architecture of the iterated greedy algorithm. Statistical tests were conducted and results show that the new algorithm outperforms existing ones.

Keywords Dynamic scheduling · New order · Permutation flowshop · Iterated greedy algorithm · Order mixing

Notations

J_O	Job set of the old order
J_N	Job set of the new order
d_O	Due date of the old order
J_T	Tardy job set
J_E	Early job set
m	Number of machines
n_O	Number of jobs belonging to the old order
n_N	Number of jobs belonging to the new order
n	Number of jobs from both orders, $n=n_O+n_N$
i	Machine index, $i\in[1,m]$
j	Job index for the old order, $j\in[1,n_O]$
k	Job index for the new order, $k\in[1,n_N]$
$t_{j,\,i}$	Processing time of job j on machine i
s	Job sequence
$T_{\max}^{J_O}$	The maximum tardiness of the old order
$C_{\max}^{J_N}$	The maximum completion time of the new order

✉ Yan Jin
yjin@qub.ac.uk

1 Academy of Chinese Energy Strategy, China University of Petroleum-Beijing, Beijing 102249, China

2 School of Mechanical and Aerospace Engineering, Queen's University Belfast, Ashby Building, Belfast BT9 5AH, UK

1 Introduction

Customers nowadays demand more and more customized products at no extra cost comparing to that mass produced by dedicated production lines. Mass customization requires flow line production systems which can respond to the changing demand quickly and efficiently. A flow line, also known as a flowshop, refers to a production system in which machines are allocated on a line where one or more products are manufactured in a specified order, starting from the first to the last. Flow lines can be found in auto manufacturing [1], integrated circuit (IC) fabrication [2], photographic film production [3], and pharmaceutical and agro-food industries [4].

Flowshop scheduling is a challenging problem [5–9]. The flowshop scheduling problem has been proven to be non-deterministic polynomial-time hard (NP-hard) when the number of machines is larger than 2 [10]. Many studies [5, 11] focus on the static environment which simplifies the problem constraint. However, more attention should be paid to dynamic aspects that frequently occurred in reality such as new order arrival [12, 13], machine breakdown [14], and rush order [15]. It is typical in today's mass customization that new orders frequently arrive into factories. The randomness of new orders often leads to sheer complexity in scheduling due to the dynamic changes given various constraints of resources. The uncertainties in terms of job types, job numbers, arrival times, due dates, etc. have created significant challenges [12, 16], which increases the complexity of the scheduling problem significantly. This happens in warehouses as well, and the problem is defined online order batching problems [17–20], in which new orders arrive frequently and stochastically, and scheduling or rescheduling should respond immediately upon their arrival. Three key questions should be considered concerning new orders: (1) if the new order can be accepted or not, given the due date constraint of the old order; (2) how to complete the new order as soon as possible; and (3) how to maximize the number of accepted orders in order to maximize revenue.

The problem is a constraint optimization problem. It is NP-hard in the strong sense especially when the due date of the old order is loose as it can be reduced to the PFSP by including all jobs from new orders [13]. In this paper, a new meta-heuristic is derived from the framework of the iterated greedy (IG) algorithm [21] under which a new enhanced destruction and construction method and a novel repair method are proposed. The paper is organized as follows. Literature review is presented in section 2, followed by the problem statement in section 3. Section 4 presents the new meta-heuristic algorithm named eIG_Rep. Section 5 introduces the experimental validation of the new algorithm. Section 6 concludes this paper.

2 Related work

The approaches to handling uncertainties can be classified into three categories [22, 23]: (1) reactive approach; (2) proactive approach; and (3) predictive-reactive approach. The reactive methods such as the shortest processing time (SPT) rule, the first come first serve (FCFS) rule, and the longest processing time (LPT) rule can be easily used to schedule jobs in real time. Right shifting (RS) strategy and real-time (RT) strategy are also reactive methods for coping with newly arrived orders [12]. RS and RT are two strategies which attach new orders to the end of the existing schedule. It has been confirmed that rescheduling the remaining jobs of the old order can provide a higher quality schedule [13, 16, 24]. However, no reactive methods can provide high-quality solutions with order mixing, therefore missing the opportunity of optimized schedule [25]. Proactive scheduling techniques are developed based on anticipated disturbances [26, 27]. It is widely used to deal with uncertainties that are easy to predict and simulate. By assuming probabilistic processing times, an improved genetic algorithm is developed in order to maximize the probability of completing orders before expected due dates [28]. A two-phase simulation-based algorithm is developed for flowshop scheduling with stochastic processing times [29]. Though the proactive approaches are robust to uncertainties, it cannot be applied to the problem studied in this paper, because new order arrivals are difficult to be predicted and simulated precisely due to many uncertain parameters in terms of job types, number of jobs, arrival times, and due dates.

The predictive-reactive approaches are commonly used scheduling methods under uncertainties [30, 31]. Two main steps are included: (1) a predictive schedule is generated over the time horizon and (2) the resulted schedule is then modified to respond to uncertainties. To handle newly arrived orders, a genetic algorithm with the non-reshuffle and reshuffle strategies is proposed for job scheduling, and a match-up strategy is applied to determine the time window for rescheduling of flexible manufacturing systems [32]. A hybrid algorithm by integrating genetic algorithm and tabu search algorithm is developed as a rescheduling technique to cope with new orders and machine breakdowns simultaneously in jobshops [33]. An assignment model is built for generating the baseline schedule and an improved genetic algorithm is developed for rescheduling of new orders in single-machine layout [34]. A refreshing variable neighborhood search (RVNS) approach is presented for job rescheduling to set common due dates for newly arrived orders in permutation flowshops [13]. In summary, the predictive-reactive method is the most popular one and it can provide high-quality schedules as order mixing can be achieved in job rescheduling. However, existing work focuses on jobshop or flexible manufacturing system [33, 34]. The PFSP coping with new orders of multiple jobs has not been well addressed.

Iterated greedy (IG) algorithm is a predicative-reactive approach. In the IG algorithm framework, a greedy construction heuristic including destruction and construction phases is iterated until the stopping criterion is satisfied, and a local search can be applied to further improve the performance. Depending on the solution feasibility, a repair method could also be used in order to enhance the algorithm's effectiveness. IG has been employed to solve many scheduling problems [35–39], including the static permutation flowshop scheduling problems (PFSP) [40]. However, it tends to be trapped to local optimum. Many modifications have been developed for the IG algorithm [39–41]. For the destruction and construction phases, the majority of studies focus on the job insertion method. To avoid repeated job removing, tabu-based destruction methods are developed [39, 42]. Different heuristics or strategies are presented for choosing suitable jobs [43]. In [35], the prior and posterior jobs of the inserting job are reinserted in all possible positions in the construction phase. Some speed-up methods are presented for improving job insertion efficiency [38, 40]. Moreover, many studies concentrate on escaping from a local optimum with the destruction and construction phases [44, 45]. An insertion perturbation is designed to provide a different solution from the previous local optimum [44]. Some studies alter the number of perturbed jobs to change the strength of perturbation [36, 38]. Normally, it is a fixed and small value [45]. However, it has been confirmed that no single fixed value could ensure high-quality solutions [36]. It is difficult to escape from a local optimum if a small number of perturbed jobs is defined [35, 44, 46]. In [36], the number of perturbed jobs is randomly selected within a defined range. In [38], a differential evolution algorithm is modified and used to optimize the number of perturbed jobs within IG algorithm for no-idle permutation flowshop scheduling problems. However, extra computation time is needed for the optimization process.

Many variants or modifications [41, 43] have been developed by modifying the local search method for improving IG algorithm performance. A descent local search is added to the IG algorithm for complex flowshop problems with sequence-dependent setup times [41]. Three local search methods are revised based on jump moves, swap moves, and variable neighborhood descent approach for a scheduling problem with unrelated parallel machines [43]. For no-wait flowshop scheduling problems, a neighborhood search method based on insert, swap, and double-insert moves is developed [39]. A variant of the non-exhaustive descent algorithm is developed by swapping any two positions in the sequence and the influence of ties is also investigated during the local search [37]. It can be concluded that different problems need different local search methods. In this paper, a dedicated local search mechanism is proposed for skipping infeasible solutions which violate the due date constraints of old orders, in order to save computation time.

For highly constrained optimization problems, e.g., new order scheduling problem with due date constraints of old orders, solution feasibility may not be guaranteed during searching process, especially when using meta-heuristics [47]. It is important to determine whether the infeasible solutions are discarded or searched for finding feasible solutions. A repair algorithm is developed in GA algorithms for solving numerical optimization problems with nonlinear constraints where two separate populations are kept, one for marking "search points" which are to be repaired and evaluated, and the other for keeping "reference points" which are used for evaluation directly [47]. A repair algorithm for robot path planning problems is presented through designed genetic operators based on prior knowledge [48]. A genetic repair operator is designed in parallel GA algorithms for the traveling salesman problem and the graph partitioning problem by constructing a new feasible chromosome after identifying all gene loci and alleles [49]. There are two techniques to deal with the fixed infeasible solutions after repairing: returning it to populations or not. Some studies [50, 51] have taken the "never replacing" approach, i.e., the repaired is never returned to the population, while others [13] have taken the "always replacing" approach. Orvosh and Davis [52] evaluate the performance of both techniques in GA algorithms in terms of returning the repaired or the original chromosome to populations. In summary, there are no standard heuristics for designing a repair algorithm [53], and a repair method is usually problem-dependent [54].

Various acceptance criteria are used for different problems. A criterion "replace if better" (RB) is adopted for dynamic parallel machine problems [55]. In solving blocking jobshop problems, two acceptance criteria are developed, i.e., random walk (RW) and simulated annealing-like (SA) [56]. When using RW, each feasible solution is accepted after the construction phase. While using SA, only the feasible solution with a better performance is accepted, or the candidate solution is accepted with a probability. In [43], RB and RW criteria are used for unrelated parallel machine problems. In [36], a simulated annealing-like acceptance is applied with a sinking temperature value in order to diversify the searching area for distributed permutation flowshops. In IG algorithms, a simulated annealing-like acceptance criterion is widely adopted [35, 41]. However, for the constrained optimization problem studied in this paper, e.g., new order scheduling problem with due date constraint of old orders, solution feasibility may not be guaranteed during searching process, especially in meta-heuristics [47]. It is important to determine whether the infeasible solutions are discarded or searched for finding feasible solutions. Three strategies are generally used to handle infeasible solutions: reject, penalize, and repair [57]. Reject strategy is a simple approach where only feasible solutions are kept. It is reasonable to use this strategy when computation time is to be saved. Penalizing strategy focuses on designing an appropriate penalty function for the infeasible ones. It is difficult to find a good compromise between the objective value and the penalty. Repair method is a constraints handling strategy. In many cases, the feasible region of the search space is very small, and feasible solutions may be surrounded by the infeasible ones. So it is necessary to guide the search towards feasible regions based on the infeasible solution information. In this paper, the acceptance criteria take infeasible solutions into consideration.

3 Problem statement

As shown in Fig. 1, the problem studied in this paper is a constraint optimization problem in which n_O jobs of the old order J_O have been scheduled but not fully completed yet when n_N jobs of the new order J_N come into the production system [16, 25]. All jobs of the old order have a common due date d_O which cannot be violated. Mathematically, it can be expressed as $T_{\max}^{J_O} = 0$. It is assumed that new orders will arrive continuously and randomly but only one at a time. So it is a dynamic problem but order selection is not considered in this paper. In order to set a tight due date for J_N and accept the maximum number of orders, the objective is defined as minimizing the maximum completion time of the new order $C_{\max}^{J_N}$ given the constraint of machine availability, a_k, following the convention of [24]. The problem can be denoted as $Fm \left| prmu, d_O, a_k \right| C_{\max}^{J_N} / T_{\max}^{J_O} = 0$ according to [58] where Fm represents a flowshop with m machines, $prmu$ stands for

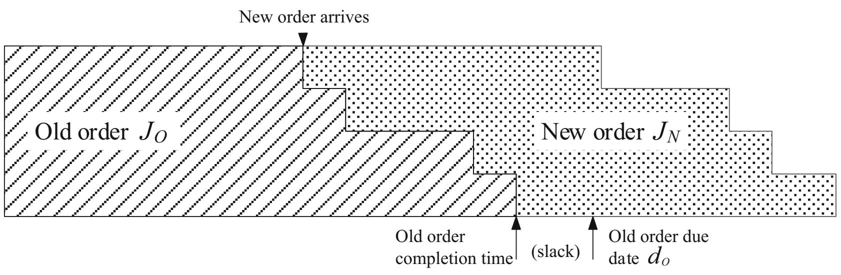

Fig. 1 Problem description

permutation, d_O indicates that all jobs of the old order have a common due date, a_k represents the availability on machine k and $C_{\max}^{J_N}/T_{\max}^{J_O}=0$ is a constrained objective. Therefore, the objective function and due date constraint can be expressed as

$$Min: F(s)=\boldsymbol{C}_{\max}^{J_N}, \tag{1}$$

$$T_{\max}^{J_O}=0. \tag{2}$$

where s represents a job sequence. The assumptions are defined as follows.

(1) All jobs should start as soon as possible;
(2) Processing times are known and deterministic;
(3) Setup time is included in the processing time for each operation;
(4) Machines are continuously available but cannot process two or more jobs simultaneously;
(5) Job pre-emption is not permitted;
(6) Buffers' capacity between machines is infinite;
(7) Only permutation schedules are allowed;
(8) Each order arrives randomly;
(9) Only one order arrives at a time;
(10) Each order may contain one or multiple jobs;
(11) Job information is known when the order arrives;
(12) All jobs in an order should be finished before its due date;
(13) Once a new order arrives, rescheduling is activated;
(14) If all existing jobs are finished and no new jobs arrive, all machines stay idle and available;
(15) No more than two orders can be mixed together, i.e., only the last old order can be rescheduled with the newly arrived order.

Note that when the new order arrives, the existing uncompleted jobs from the old order will be rescheduled with the new order. Old order jobs and partial jobs from the new order have to be scheduled before the due date. Therefore, the slack between the due date and its completion time of the old order has a direct impact on the resulted schedule. A relaxed due date of the old order can absorb more jobs from the new order and have a better chance to obtain a high-quality schedule. So, a reasonable and relaxed due date for the old order is assumed.

To solve the above problem, a new meta-heuristic is proposed based on the IG algorithm. Three types of problems including due date setting, order acceptance, and maximizing the number of accepted orders are studied.

4 New meta-heuristic: a modified iterated greedy algorithm (eIG_Rep)

In this section, a new meta-heuristic algorithm named the enhanced iterated greedy algorithm with repair method (eIG_Rep) is proposed. It is derived from the IG algorithm which is a single-point meta-heuristic that only one population is saved and searched in each iteration. It is fast but easy to be trapped in a local optimum even if an acceptance criterion is used. Therefore, an effective escape mechanism from a local optimum should be designed. Besides, the studied problem in this paper has a strong due date constraint that jobs from the old order should be completed by their due date. The best feasible solution may be surrounded by and evolved from infeasible solutions. So, an effective repair method is required to convert infeasible solutions into feasible ones.

During the searching process, the following two challenging issues should be solved: (1) how to escape from a local optimum and (2) how to repair infeasible solutions effectively. To address these issues, a new destruction and construction method and a new repair approach are proposed in this paper.

Figure 2 shows the flow chart of the new eIG_Rep algorithm. First, during initialization, the initial solution is generated, and the local search is applied to the initial solution. The initial parameters are given including δ, z, δ', and Ti. In the main body of the algorithm, destruction and construction, local search, repair method, and acceptance criterion are run in an iterated manner until the stopping criterion is satisfied. Note that in the new enhanced destruction and construction method, the number of perturbed jobs, δ, should be first decided before destruction and construction phases. The local search is then applied to the incumbent. If the solution resulted from the local search is infeasible, repair it. Otherwise, acceptance criterion is conducted directly. The details of the new algorithm are described below.

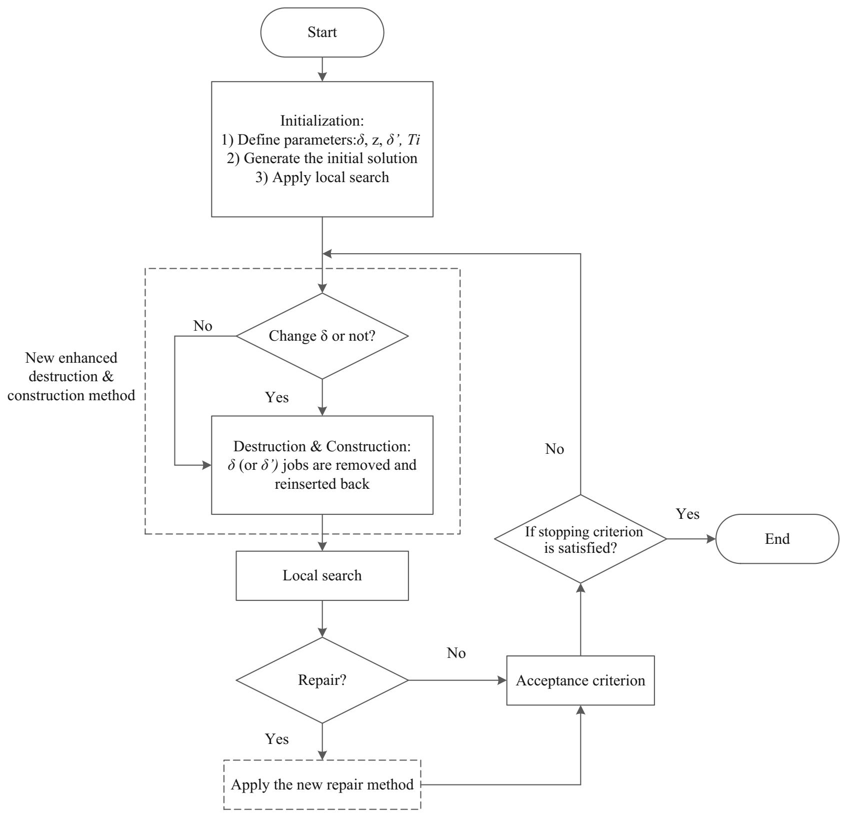
Start
Initialization:
1) Define parameters:δ, z, δ', Ti
2) Generate the initial solution
3) Apply local search
New enhanced destruction & construction method
Change δ or not?
No
Yes
Destruction & Construction:
δ (or δ') jobs are removed and reinserted back
Local search
Repair?
Yes
No
Apply the new repair method
Acceptance criterion
If stopping criterion is satisfied?
No
Yes
End

Fig. 2 Flow chart of the eIG_Rep algorithm (δ, the number of perturbed jobs; z, the number of iterations with δ unchanged; δ', the enlarged number of perturbed jobs; Ti, a parameter to adjust the temperature in acceptance criterion)

4.1 Initialization

An initial population is required for the eIG_Rep algorithm at the beginning. For this need, three initial solutions are generated: one is obtained by the NEH heuristic [59] that jobs from both the old and the new orders are mixed and scheduled without accounting for the due date constraint; the other two are generated by RS and RT strategies [12] respectively implemented with the NEH heuristic. Among these three solutions, the feasible one with the best objective value is chosen as the initial best solution S_{best}. So, a feasible solution is guaranteed in the initialization phase as a relaxed due date is assumed and determined by RT strategy implemented with the NEH heuristic. The local search method is then applied to S_{best}. If a solution S obtained through the local search is feasible and better than S_{best}, then S_{best} will be updated by S. Note that S is defined as the incumbent solution in each iteration. The local search method will be detailed in section 4.3.

Four key parameters δ, z, δ', and Ti are set during initialization. δ represents the number of perturbed jobs, δ' is the enlarged number of perturbed jobs in the destruction and construction method, z indicates the number of iterations allowing the performance of the incumbent to remain steady while not enlarging perturbation, and Ti represents a control parameter for the acceptance criterion.

4.2 New destruction and construction method

In the herein paper, a new destruction and construction method with a steered variation approach for δ is designed in order to escape efficiently from a local optimum while maximizing the computational efficiency. The new method is presented as follows.

In the destruction phase, δ jobs are randomly selected, half from the old order and half from the new order. So δ is defined as an even number. When the incumbent is trapped in a local optimum or its objective value stays unchanged for z iterations, the perturbation will be intensified by replacing δ with a larger value δ' ($\delta' > \delta$). The number of removed and reinserted jobs increases to δ', so as to diversify the population and escape from the local optimum. Once the objective value of the incumbent is improved, δ is changed back to a smaller value in order to save computation time.

For example, 4 ($\delta = 4$) jobs are evenly selected from both the old and new orders in destruction phase. If the objective value does not improve after 2 ($z = 2$) iterations, 6 jobs ($\delta' = 6$) are then selected to enlarge the perturbation. If the objective value is improved, δ is changed back to maintain algorithm efficiency.

In the construction phase, for our problem, many infeasible solutions will be generated if all positions are investigated when inserting jobs of the old order. The due date constraint will be violated if the jobs of the old order are reinserted after the due date. To avoid infeasible solutions, the idea of general local search [13] is used. If the job belongs to the new order, it is reinserted into each possible position. For the job from the old order, it is reinserted into each possible and feasible position given the due date constraint. That is, if the maximum completion time of the reinserted job exceeds the due date, the subsequent positions are then neglected. After construction, if a better and feasible solution is generated, then S_{best} is updated.

4.3 Local search

In the IG algorithm, a local search can be applied after the construction phase to improve its performance [21]. In this paper, the general local search method [13] is modified to avoid infeasible solutions and save computational time.

Flow chart of the local search is shown in Fig. 3. For a given solution, choose one job and insert it into each possible and feasible position sequentially. When inserting jobs by using Taillard improvement [60], jobs from the old order are inserted given the due date constraint. That is, when the maximum completion time of the inserted job exceeds the due

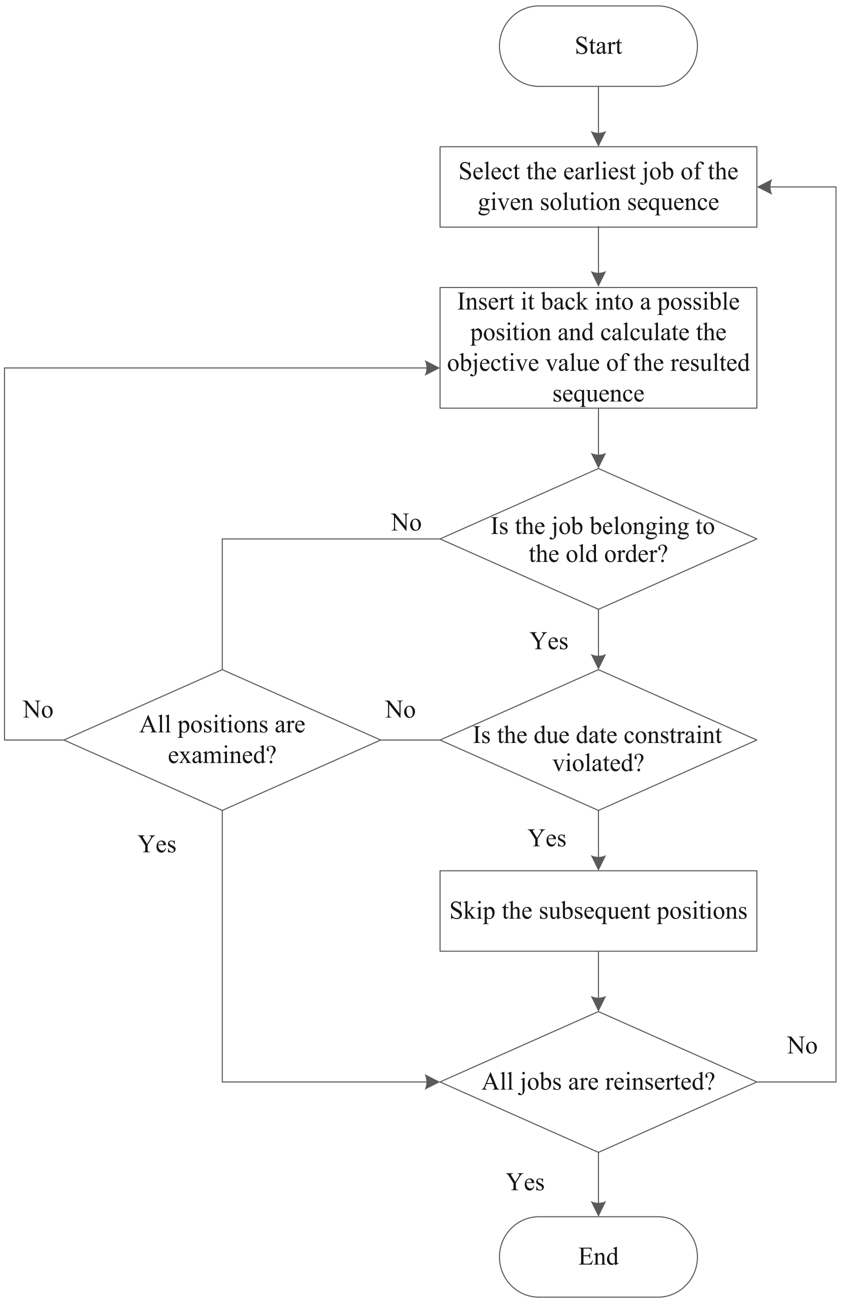
Start
Select the earliest job of the given solution sequence
Insert it back into a possible position and calculate the objective value of the resulted sequence
Is the job belonging to the old order?
No
Yes
Is the due date constraint violated?
No
Yes
All positions are examined?
No
Yes
Skip the subsequent positions
All jobs are reinserted?
No
Yes
End

Fig. 3 Flow chart of the local search

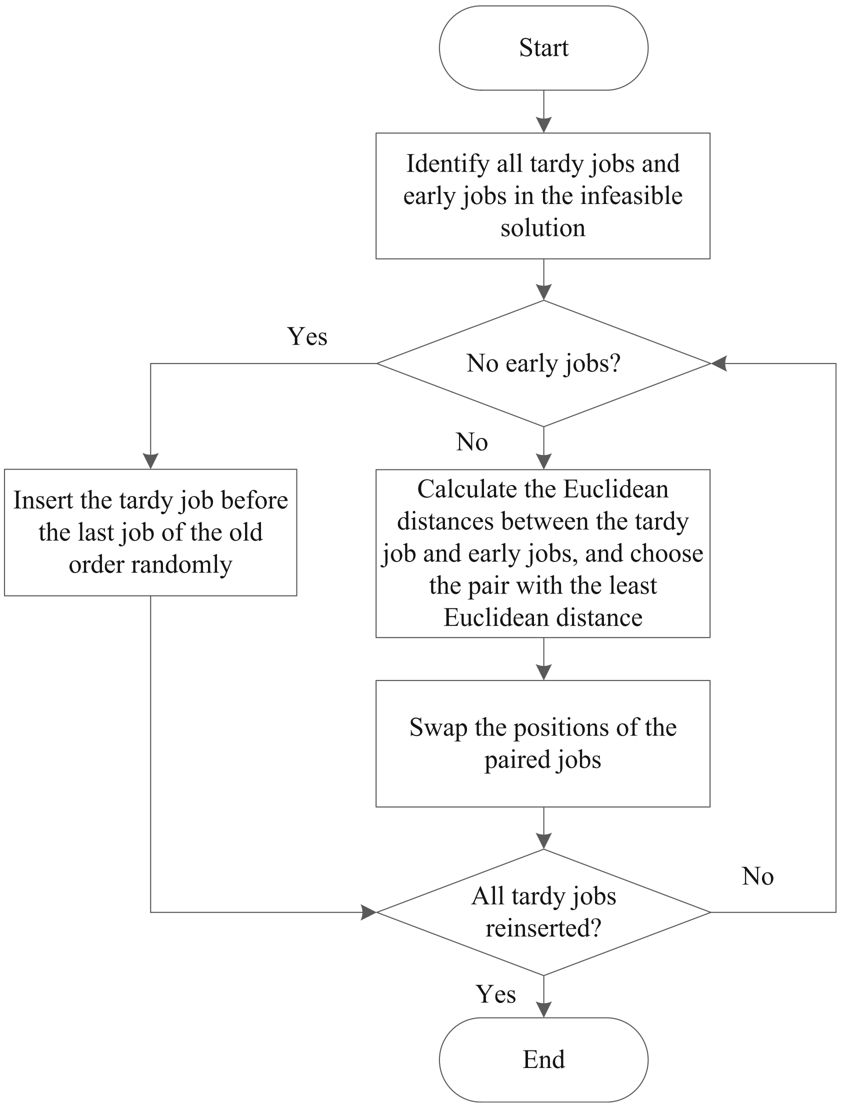
Start
Identify all tardy jobs and early jobs in the infeasible solution
No early jobs?
Yes
No
Insert the tardy job before the last job of the old order randomly
Calculate the Euclidean distances between the tardy job and early jobs, and choose the pair with the least Euclidean distance
Swap the positions of the paired jobs
All tardy jobs reinserted?
No
Yes
End

Fig. 4 Flow chart of new repair method

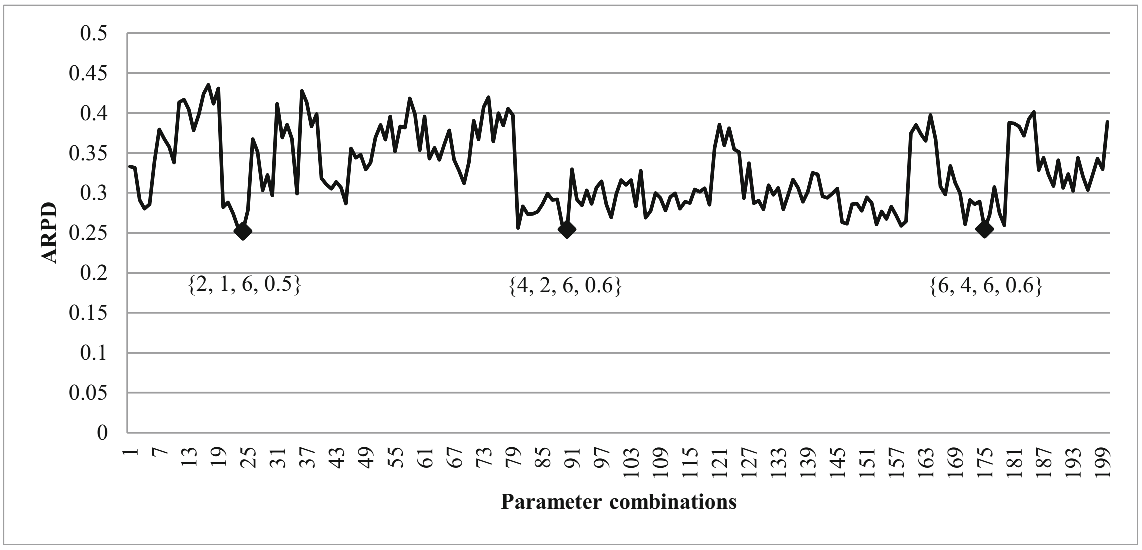

Fig. 5 ARPD values of different parameter combinations on small instances of VRF benchmark

Table 1 ARPD values of each algorithm on Taillard benchmark

Problem	RS_NEH	RT_NEH	MR_WWD	RS_IG	RT_IG	RVNS	eIG_Rep
20 × 5	19.95	10.35	6.51	17.86	9.36	3.21	2.30
20 × 10	19.36	11.14	9.46	19.99	8.73	3.03	1.35
20 × 20	18.09	10.86	3.73	17.27	8.32	1.18	0.11
50 × 5	11.83	5.28	2.78	10.78	4.36	1.28	0.45
50 × 10	22.90	11.09	8.24	18.83	7.07	5.66	2.52
50 × 20	23.00	12.10	9.19	19.84	8.56	6.89	3.29
100 × 5	10.45	3.77	1.71	8.26	2.92	1.08	0.28
100 × 10	15.30	7.25	4.61	12.66	5.36	3.90	1.58
100 × 20	20.16	10.98	8.01	17.17	7.35	7.23	4.04
200 × 10	12.43	4.11	3.05	10.80	3.04	2.55	1.16
200 × 20	18.24	8.94	7.03	16.31	6.95	7.01	4.74
500 × 20	12.83	4.94	4.60	11.66	4.30	4.00	3.37
AVG	17.05	8.40	5.74	15.12	6.36	3.92	2.10

Table 2 ARPD values of each algorithm on VRF benchmark

Problems		RS_NEH		RT_NEH		MR_WWD		RS_IG		RT_IG		RVNS		eIG_Rep	
S	L	S	L	S	L	S	L	S	L	S	L	S	L	S	L
10 × 5	100 × 20	14.25	20.82	8.09	10.75	2.06	5.63	12.84	16.98	7.38	6.95	*0.00*	4.75	*0.00*	*2.93*
10 × 10	100 × 40	10.93	18.07	5.54	9.97	1.63	5.48	10.31	15.83	4.44	6.22	*− 0.27*	4.68	*− 0.27*	*3.00*
10 × 15	100 × 60	12.94	17.00	7.43	9.52	1.53	4.76	12.25	13.90	6.69	6.13	*0.00*	4.39	*0.00*	*2.55*
10 × 20	200 × 20	13.44	17.29	6.71	8.37	1.99	4.17	12.65	15.04	5.89	6.08	*0.00*	3.47	*0.00*	*2.76*
20 × 5	200 × 40	18.43	17.61	7.65	9.40	1.53	4.74	15.48	15.10	5.56	6.80	0.37	4.21	*0.03*	*3.24*
20 × 10	200 × 60	22.85	16.05	12.95	8.74	4.82	4.40	20.71	13.75	10.38	6.23	1.54	4.17	*0.16*	*3.27*
20 × 15	300 × 20	19.32	14.52	11.26	6.79	4.29	3.03	18.44	13.51	8.29	5.53	1.27	2.55	*0.13*	*2.14*
20 × 20	300 × 40	18.16	16.20	10.49	8.07	4.12	4.02	16.24	14.39	7.86	6.73	1.35	3.69	*0.11*	*3.25*
30 × 5	300 × 60	14.82	15.30	5.16	8.01	1.64	3.98	9.69	13.62	3.04	6.47	0.54	3.72	*0.11*	*3.16*
30 × 10	400 × 20	22.82	13.34	12.33	5.77	5.26	2.48	19.77	12.12	8.50	4.63	2.62	2.16	*1.09*	*1.84*
30 × 15	400 × 40	25.97	14.76	14.67	7.20	5.73	3.54	23.33	13.41	10.19	6.12	2.98	3.36	*1.07*	*3.09*
30 × 20	400 × 60	19.73	14.51	11.67	7.12	5.41	3.62	17.74	13.13	7.13	6.16	3.00	3.38	*0.91*	*2.98*
40 × 5	500 × 20	13.49	12.34	5.42	4.86	1.20	2.05	10.63	11.15	3.54	4.20	0.20	1.92	*0.13*	*1.77*
40 × 10	500 × 40	22.31	13.93	12.41	6.66	4.97	3.09	18.51	12.94	7.24	5.82	2.87	3.00	*1.21*	*2.70*
40 × 15	500 × 60	22.64	14.03	12.13	6.88	6.08	3.28	17.98	13.01	7.77	6.15	3.65	3.04	*1.51*	*2.71*
40 × 20	600 × 20	21.88	10.97	13.13	4.27	5.16	1.72	18.14	10.18	8.19	3.61	3.78	1.29	*1.73*	*1.16*
50 × 5	600 × 40	12.14	13.51	4.38	6.05	1.05	3.07	11.10	12.31	3.12	5.53	0.22	2.97	*0.04*	*2.77*
50 × 10	600 × 60	20.09	12.99	10.40	6.43	4.45	2.88	16.20	12.12	6.59	5.88	2.60	2.79	*1.02*	*2.61*
50 × 15	700 × 20	22.77	10.39	12.52	4.09	6.38	1.38	18.68	9.80	7.82	3.65	4.83	1.23	*2.17*	*1.05*
50 × 20	700 × 40	21.56	12.81	11.97	5.96	5.91	2.71	18.66	12.08	7.27	5.39	4.37	2.64	*2.23*	*2.44*
60 × 5	700 × 60	12.35	12.78	3.59	6.19	1.51	2.71	9.78	12.06	2.63	5.61	0.12	2.54	*0.03*	*2.41*
60 × 10	800 × 20	18.74	9.49	10.25	3.79	3.81	1.29	16.54	8.89	6.78	3.33	2.82	1.08	*1.18*	*0.95*
60 × 15	800 × 40	22.73	11.93	12.12	5.32	5.97	2.60	18.04	11.30	7.49	4.89	4.39	2.31	*2.15*	*2.15*
60 × 20	800 × 60	20.52	12.13	12.06	5.76	6.44	2.71	18.33	11.62	8.02	5.34	4.82	2.62	*2.64*	*2.41*
AVG(S)	AVG(L)	18.54	14.28	9.76	6.92	3.87	3.31	15.92	12.84	6.74	5.56	2.00	3.00	*0.81*	*2.47*
AVG		16.41		8.34		3.59		14.38		6.15		2.50		*1.64*	

The italic number is the best ARPD value for each problem

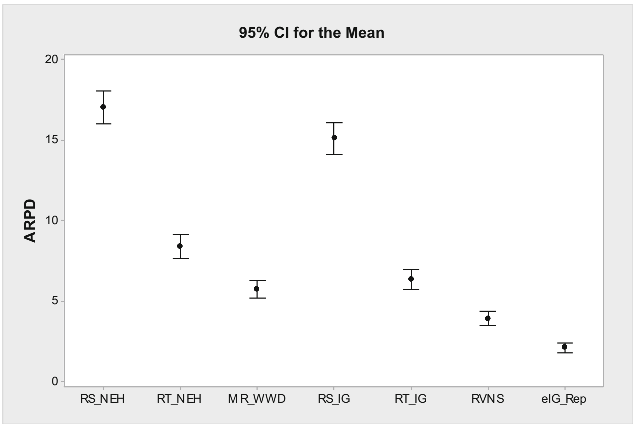

Fig. 6 Means and 95% HSD intervals for RS_NEH, RT_NEH, MR_WWD, RS_IG, RT_IG, RVNS, and eIG_Rep on Taillard benchmark

date, the subsequent positions are not examined. All jobs are selected and inserted one by one and the best solution is saved. After local search, if the generated solution is feasible and better than S_{best}, then S_{best} is updated. Otherwise, a repair method is applied.

4.4 New repair method

Although infeasible solutions whose completion times are beyond the due date constraint are largely avoided, many infeasible solutions with better objective values would be generated after the local search. Therefore, a repair method is required to convert these infeasible solutions into feasible ones during the search. In this paper, a novel repair mechanism is proposed.

Figure 4 shows the flow chart of the new repair method. The jobs of the old order J_T which are completed beyond their due date are selected and defined as tardy jobs. The jobs of the new order J_E which are scheduled before the due date are deemed as early jobs. In order to fix the infeasible solutions and keep their excellent performance, the tardy jobs and early jobs with similar processing times are paired and swapped one by one until all tardy jobs are scheduled before the due date. Herein, the least Euclidean distance is introduced as a measure of similarity of tardy and early jobs, in order to maintain the overall performance of the schedule after swapping. The Euclidean distance between two jobs is defined as

$$Ed = \sqrt{\sum_{i=1}^{m}\left(t_{j,i}-t_{k,i}\right)^2}, i\in[1,m], j\in[1,n_O], k\in[1,n_N]. \qquad (3)$$

If the number of tardy jobs J_T is larger than that of J_E, the rest jobs in J_T after pairing will be randomly inserted before the last job position of the old order. After repairing, if a feasible and better solution is found, S_{best} is then updated.

4.5 Acceptance criterion

In the IG algorithm, a simulated annealing-like acceptance criterion is normally used to improve population diversification and escape from a local optimum. During searching, the incumbent does not have to be feasible. If the solution S' resulted from the local search or the repair method has a better objective value, it is then used as the new incumbent S. If the solution S' is worse than previous incumbent, it can also be accepted as the new incumbent with a probability of $e^{-\left(C_{\max}\left(S'\right)-C_{\max}(S)\right)}$ /Temp, where $\text{Temp} = T_i \times \frac{sumt}{\left(n_1'+n_2\right)*m*10}$, where *sumt* is the sum of job processing times including the remaining jobs of the old order and the new order, n_1' is the number of the remaining jobs from the old order, n_2 is the number of jobs from the new order, and T_i is a parameter to adjust the temperature which needs to be defined in the initialization. Otherwise, the current incumbent S is retained if the probability is not satisfied.

4.6 Calibration

In order to fully investigate algorithm potentials, parameter values should be decided carefully. In this paper, reference algorithms such as RS_NEH [12], RT_NEH [12], MR_WWD [25], RS_IG, RT_IG, and RVNS [13] are included. As few studies are involved with this topic, three existing heuristic algorithms, namely RS_NEH, RT_NEH, and MR_WWD are considered. No parameters need be tuned for them. IG is also adopted with RS and RT strategies, denoted by RS_IG and RT_IG for performance comparison. RVNS, as the existing study on this topic, is taken as the reference in this paper.

The parameters of IG algorithms in RS_IG and RT_IG algorithms should be carefully selected and kept same. Since the studied problem in this paper is similar to that of [13], $\delta = 4$ and $Ti = 0.4$ are selected as they have been confirmed robust in solving PFSPs and due date setting problems.

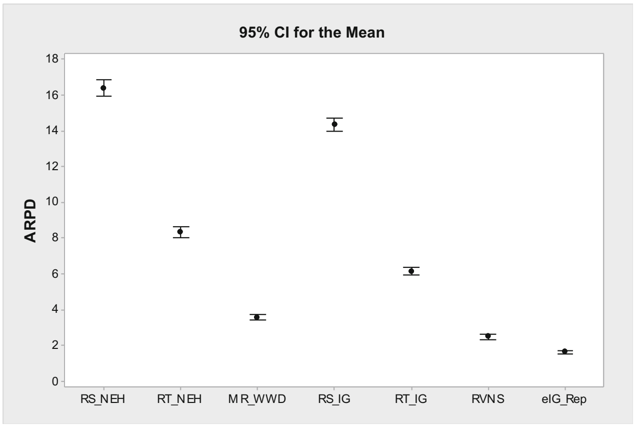

Fig. 7 Means and 95% HSD intervals for RS_NEH, RT_NEH, MR_WWD, RS_IG, RT_IG, RVNS, and eIG_Rep on VRF benchmark

Table 3 Paired samples *t* test results on Taillard benchmark

Pair	Paired differences				*t*	*p*
	Mean	SEM	Lower	Upper		
RS_NEH – eIG_Rep	14.94651	.45780	14.04001	15.85301	32.648	.000
RT_NEH – eIG_Rep	6.30151	.33727	5.63368	6.96934	18.684	.000
MR_WWD – eIG_Rep	3.64559	.23142	3.18736	4.10382	15.753	.000
RS_IG – eIG_Rep	13.02019	.47275	12.08409	13.95628	27.541	.000
RT_IG – eIG_Rep	4.26178	.30006	3.66764	4.85593	14.203	.000
RVNS – eIG_Rep	1.81990	.11024	1.60160	2.03819	16.508	.000

RVNS algorithm [13] focuses on setting tight due dates for new orders by minimizing makespan of the new order. It can be transformed into solving dynamic PFSPs with new order arrivals, and it is taken as a reference algorithm. In the RVNS algorithm, two parameters are required, i.e., $k_{\max}$ and q. $k_{\max}$ is the maximum number of jobs in shaking method and q is the percentage of jobs in escape procedure. According to [13], $k_{\max}$ is set as 40 and q is randomly obtained between 75 and 95% of the number of jobs in either the old or the new order depending on the feasibility of the solution.

In the new algorithm, eIG_Rep, four parameters including δ, z, δ', and Ti need to be defined. The test ranges of each parameter are described as follows.

$$\delta\in\{2,4,6,8\},$$
$$z\in\{1,2,4,6\},$$
$$\delta'\in\{4,6,8,10\},$$
$$Ti\in\{0.2,0.3,0.4,0.5,0.6\}.$$

Due to $\delta' > \delta$, $(4+3+2+1)\times4\times5=200$ combinations in total can be generated for the eIG_Rep algorithm. To choose the most robust parameters for the eIG_Rep algorithm, all 200 combinations are tested.

Herein, the objective of setting a tight due date for the new order is taken for algorithm calibration since it can be validated on a large number of instances. The small instances of VRF benchmark [61] are used as samples. Each instance is evenly divided into two sets: the first half including the front part of the instance is taken as the old order while the second half is defined as the new order. Herein, the old order due date is set as a constant, $(1+\alpha)\times C_{\max}^{J_O}$, where α is the relaxation factor and $C_{\max}^{J_O}$ is the maximum completion time of the old order obtained by the NEH heuristic. α is defined as 0.4. The computation time limit $n\times\ (m/2)\ \times t$ milliseconds where $t=60$ [21, 62] is defined as the stopping criterion. To avoid stagnation, all algorithms are run for 10 independent times and the best solution for each instance is retained. Note that the stopping criterion is also used in the following sections. All algorithms are coded in Matlab R2014b and run on a CPU E5520 computer with 6G memory.

To evaluate each algorithm, the performance measure of relative percentage deviation (RPD) is employed, and it is computed as follows:

$$RPD=\frac{C_{\max}^{J_N}-UB}{UB}\times100\%, \qquad (4)$$

where $C_{\max}^{J_N}$ is the maximum completion time of the new order obtained on every instance, UB is the upper bound for each instance. The UB values provided by Vallada et al. [61] are used directly because the problem can be deemed as a static permutation flowshop scheduling problem if the slack between the completion time and due date of the old order is very loose and all jobs from the new order can be completed in the slack.

Figure 5 shows the results of each combination of the parameters on small instances of VRF benchmark. As shown in Fig. 5, the best three combinations $\{\delta, z, \delta', Ti\}$ associated with

Table 4 Paired samples *t* test results on VRF benchmark

Pair	Paired differences				*t*	*p*
	Mean	SEM	Lower	Upper		
RS_NEH – eIG_Rep	14.76971	.23650	14.30500	15.23442	62.451	.000
RT_NEH – eIG_Rep	6.69939	.16645	6.37233	7.02645	40.249	.000
MR_WWD – eIG_Rep	1.94909	.08031	1.79130	2.10689	24.271	.000
RS_IG – eIG_Rep	12.74025	.19755	12.35208	13.12841	64.492	.000
RT_IG – eIG_Rep	4.51068	.11758	4.27964	4.74172	38.362	.000
RVNS – eIG_Rep	.86038	.04103	.77976	.94101	20.969	.000

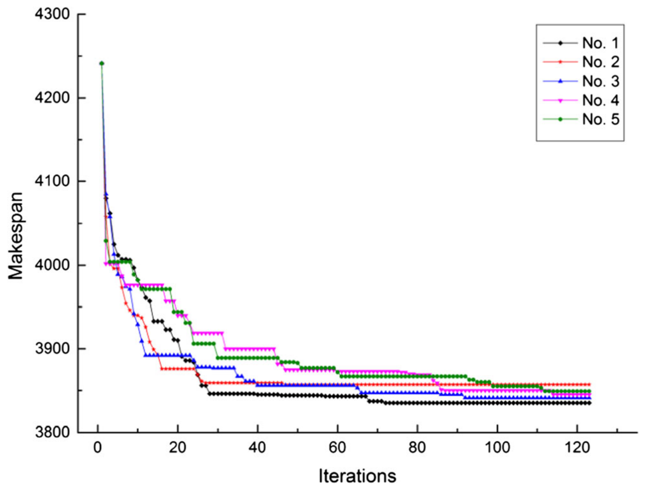

Fig. 8 Convergence curve of eIG_Rep on TA052 Taillard instance

the lowest average RPD (ARPD) values on every instance are {2, 1, 6, 0.5}, {4, 2, 6, 0.6}, and {6, 4, 6, 0.6}. To further check the performance of these parameter combinations, they are tested on VRF large instances and results show that the combination {4, 2, 6, 0.6} provides robust solutions on both small and large instances. So, the parameters of the new algorithm eIG_Rep are set as {4, 2, 6, 0.6}.

5 Experimental validation

To validate the new algorithm, two evaluation schemes are conducted. One is to set the earliest due dates for new orders, and the second is to maximize the accepted number of orders whose due dates are given by a case study. The criterion of accepting an order is based on the due date. If the new order can be completed by its due date, it is accepted. Otherwise, it is rejected.

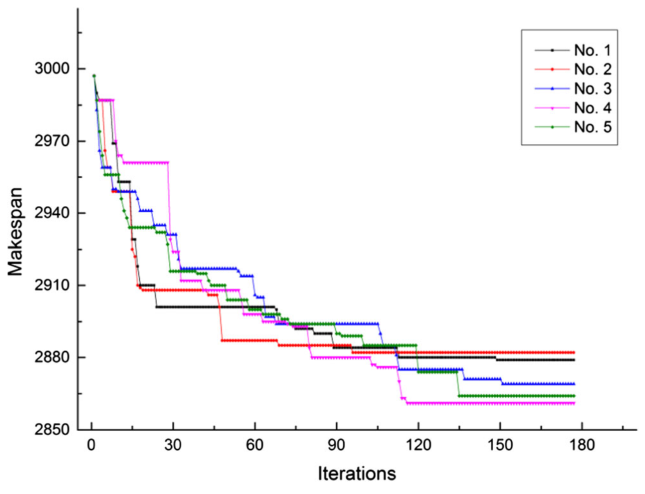

Fig. 9 Convergence curve of eIG_Rep on VRF 30_20_2 instance

Table 5 The acceptance status of each new order

Order no.		1	2	3	4	5	6	7	8	9	10
Arrival time		0	360	846	1038	2882	3991	5312	6264	7603	8213
Due date		1800	2964	4103	5050	6214	6621	7791	9022	10,604	11,976
RS_IG	$C_{\max}^N$	1586	3321	4404	5470	6506	7435	8824	9915	10,971	12,027
	$T_{\max}^N$	−214	357	301	420	292	814	1033	893	367	51
	Status	A	R	R	R	R	R	R	R	R	R
RT_IG	$C_{\max}^N$	1582	2868	3990	5060	6216	7139	8342	9269	10,515	11,573
	$T_{\max}^N$	−218	− 96	− 113	10	2	518	551	247	− 89	− 403
	Status	A	A	A	R	R	R	R	R	A	A
RVNS	$C_{\max}^N$	1800	2919	4058	4987	6181	7125	8313	9307	10,557	11,599
	$T_{\max}^N$	0	− 45	− 45	− 63	− 33	504	522	285	− 47	− 377
	Status	A	A	A	A	A	R	R	R	A	A
eIG_Rep	$C_{\max}^N$	1743	2376	4051	4502	5589	6638	7758	8836	9974	10,925
	$T_{\max}^N$	− 57	− 588	− 52	− 548	− 625	17	− 33	− 186	− 630	− 1051
	Status	A	A	A	A	A	R	A	A	A	A

A accepted, *R* rejected

5.1 Setting a tight due date

To fully evaluate the effectiveness of the new eIG_Rep algorithm, both Taillard [63] and VRF benchmarks are used, and the performance measure ARPD is adopted. As described in section 4, each instance is evenly divided into two sets representing old and new orders. The old order due date is set as $(1+\alpha)\times C_{\max}^{J_o}$, where α set as 0.4. The *UB* values for Taillard benchmark are taken from [64]. Each meta-heuristic algorithm is run for ten times and the best one on each instance is retained.

Table 1 shows the test results of each algorithm on Taillard benchmark. The new eIG_Rep algorithm has an ARPD value of 2.10 which is much better than that of all the other references including RS_NEH, RT_NEH, MR_WWD, RT_IG, RS_IG, and RVNS algorithms. It performs best under most situations. Note that these algorithms, such as MR_WWD, RVNS, and eIG_Rep, allowing order mixing, show better performance than RS_NEH, RT_NEH, RS_IG, and RT_IG algorithms in which no order mixing is allowed.

Test results on VRF benchmark are shown in Table 2. The same conclusion can be obtained. The eIG_Rep algorithm provides the best result in terms of ARPD, followed by RVNS, MR_WWD, RT_IG, RT_NEH, RS_IG, and RS_NEH algorithms.

Figures 6 and 7 show the means plots and 95 confidence intervals of each algorithm on both benchmarks. No overlaps of ARPD means are observed between eIG_Rep and other algorithms, demonstrating that eIG_Rep shows significantly better performance.

In order to check if the differences of these ARPD values are statistically significant, the paired samples *t* test is conducted with a confidence level of 95%. Results are shown in Table 3 and Table 4 for both Taillard and VRF benchmarks. The *p* values for the comparison of the new algorithm are 0.000 on both test beds. Thus, it can be concluded that there are significant differences between the performances of the new algorithm eIG_Rep comparing to these reference algorithms.

Figures 8 and 9 show the convergence curve of the new algorithm on two instances, TA052 from Taillard and VRF 30_20_2 instance. The algorithm, eIG_Rep, is run for five times independently on each instance. It can be observed that the new algorithm generates stable solutions each time. The solution quality improves after several iterations as the size of the new destruction and construction method varies, escaping from a local optimum.

5.2 Case study on maximizing the number of accepted orders

To check the capability of each algorithm for the objective of maximizing the acceptance of new orders, a case study is conducted. It is assumed that ten orders with assigned individual due dates arrive continuously and randomly. Both the arrival times and due dates of each order are generated randomly. The ten orders are selected from Taillard benchmark [63], i.e., TA011-20 instances.

Table 5 shows the test results of order acceptance. No more than half of the new orders are completed by due dates if RS_IG and RT_IG algorithms are used. RVNS algorithm can accept seven new orders while the new eIG_Rep algorithm can complete nine new orders before their due dates. It can be seen that the algorithms with order mixing show a better chance of obtaining high-quality schedules against makespan. More orders can be absorbed and completed by their due dates. And eIG_Rep is more effective than RS_IG, RT_IG, and RVNS algorithms for this case.

6 Conclusions

This paper proposes a new algorithm for solving dynamic scheduling problems in permutation flowshops with new order arrivals. A new algorithm, named as eIG_Rep, is introduced based on a new destruction and construction method together with a novel effective repair method. The new destruction and construction method is developed by changing the strength of perturbation if the population performance remains unchanged for some iterations, in order to escape from a local optimum while maintaining high computational efficiency. The new repair method is developed for the first time by swapping tardy and early jobs. By using the new repair method, the infeasible solutions are converted into feasible ones while maintaining their solution quality as much as possible.

Statistical test results show that the new algorithm eIG_Rep is effective and outperforms the existing and reference algorithms including RS_NEH, RT_NEH, MR_WWD, RS_IG, RT_IG, and RVNS. This new eIG_Rep algorithm can be well applied to three types of problems, i.e., due date setting problem, order acceptance problem, and maximizing the number of accepted new orders. For the problem of setting tight due dates of orders, both the Taillard and VRF test beds are used for algorithm validation and the effectiveness of the new algorithm is demonstrated on both benchmarks. For the problems of order acceptance and revenue maximization, ten orders are assumed by assigning random arrival times and due dates when evaluating the new algorithm. Test results show that more orders can be completed by their due dates than existing algorithms by using the new algorithm eIG_Rep.

Funding information The Ph.D. scholarship supported by the Queen's University Belfast, UK and China Scholarship Council (201306050018) is acknowledged.

References

1. Xu J, Zhou X (2009) A class of multi-objective expected value decision-making model with birandom coefficients and its application to flow shop scheduling problem. Inf Sci (Ny) 179:2997–3017
2. Liu CY, Chang SC (2000) Scheduling flexible flow shops with sequence-dependent setup effects. IEEE Trans Robot Autom 16: 408–419
3. Aghezzaf E, Van Landeghem H (2002) An integrated model for inventory and production planning in a two-stage hybrid production system. Int J Prod Res 40:4323–4339
4. Boukef H, Benrejeb M, Borne P (2007) A proposed genetic algorithm coding for flow-shop scheduling problems. Int J Comput Commun Control 2:229–240
5. Liao L, Huang C (2010) Tabu search for non-permutation flowshop scheduling problem with minimizing total tardiness. Appl Math Comput 217:557–567
6. Liu W, Jin Y, Price M (2017) A new improved NEH heuristic for permutation flowshop scheduling problems. Int J Prod Econ 193: 21–30. https://doi.org/10.1016/j.ijpe.2017.06.026
7. Liu W, Jin Y, Price M (2016) A new Nawaz–Enscore–Ham-based heuristic for permutation flow-shop problems with bicriteria of makespan and machine idle time. Eng Optim 48:1808–1822
8. Ravindran D, Haq AN, Selvakuar SJ, Sivaraman · R (2005) Flow shop scheduling with multiple objective of minimizing makespan and total flow time. Int J Adv Manuf Technol 25:1007–1012. https://doi.org/10.1007/s00170-003-1926-1
9. Chakravorty A, Laha D (2017) A heuristically directed immune algorithm to minimize makespan and total flow time in permutation flow shops. Int J Adv Manuf Technol 93:3759–3776. https://doi.org/10.1007/s00170-017-0679-1
10. Garey MR, Johnson DS, Sethi R (1976) The complexity of flowshop and jobshop scheduling. Math Oper Res 1:117–129
11. Vasiljevic D, Danilovic M (2015) Handling ties in heuristics for the permutation flow shop scheduling problem. J Manuf Syst 35:1–9
12. Rahman HF, Sarker R, Essam D (2015) A real-time order acceptance and scheduling approach for permutation flow shop problems. Eur J Oper Res 247:488–503
13. Perez-Gonzalez P, Framinan JM (2010) Setting a common due date in a constrained flowshop: a variable neighbourhood search approach. Comput Oper Res 37:1740–1748
14. Safari E, Sadjadi SJ (2011) A hybrid method for flowshops scheduling with condition-based maintenance constraint and machines breakdown. Expert Syst Appl 38:2020–2029
15. Nandi A, Rogers P (2004) Using simulation to make order acceptance/rejection decisions. SIMULATION 80:131–142
16. Perez-Gonzalez P, Framinan JM (2015) Assessing scheduling policies in a permutation flowshop with common due dates. Int J Prod Res 53:5742–5754
17. Koch S, Wäscher G (2016) A grouping genetic algorithm for the order batching problem in distribution warehouses. J Bus Econ 86: 131–153. https://doi.org/10.1007/s11573-015-0789-x
18. Kulak O, Sahin Y, Taner ME (2012) Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms. Flex Serv Manuf J 24:52–80. https://doi.org/10.1007/s10696-011-9101-8
19. Menéndez B, Pardo EG, Alonso-Ayuso A, Molina E, Duarte A (2017) Variable neighborhood search strategies for the order batching problem. Comput Oper Res 78:500–512. https://doi.org/10.1016/j.cor.2016.01.020
20. Menéndez B, Pardo EG, Sánchez-Oro J, Duarte A (2017) Parallel variable neighborhood search for the min–max order batching problem. Int Trans Oper Res 24:635–662. https://doi.org/10.1111/itor.12309
21. Ruiz R, Stützle T (2007) A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur J Oper Res 177:2033–2049. https://doi.org/10.1016/j.ejor.2005.12.009
22. Aytug H, Lawley MA, McKay K, Mohan S, Uzsoy R (2005) Executing production schedules in the face of uncertainties: a review and some future directions. Eur J Oper Res 161:86–110
23. Herroelen W, Leus R (2004) Robust and reactive project scheduling: a review and classification of procedures. Int J Prod Res 42: 1599–1620
24. Perez-Gonzalez P, Framinan JM (2009) Scheduling permutation flowshops with initial availability constraint: analysis of solutions and constructive heuristics. Comput Oper Res 36:2866–2876
25. Liu W, Jin Y, Price M (2017) New scheduling algorithms and digital tool for dynamic permutation flowshop with newly arrived order. Int J Prod Res 55:3234–3248. https://doi.org/10.1080/00207543.2017.1285077
26. O'Donovan R, Uzsoy R, McKay KN (1999) Predictable scheduling of a single machine with breakdowns and sensitive jobs. Int J Prod Res 37:4217–4233
27. Zhu H, Zhou H (2014) Single machine predictive scheduling using inserted idle times. J Appl Math 2014:1–5
28. Liu Q, Ullah S, Zhang C (2011) An improved genetic algorithm for robust permutation flowshop scheduling. Int J Adv Manuf Technol 56:345–354. https://doi.org/10.1007/s00170-010-3149-6
29. Wang K, Choi SH, Qin H (2014) An estimation of distribution algorithm for hybrid flow shop scheduling under stochastic processing times. Int J Prod Res 52:7360–7376. https://doi.org/10.1080/00207543.2014.930535
30. Moratori P, Petrovic S, Vázquez A (2008) Match-up strategies for job shop rescheduling. In: New frontiers in applied artificial intelligence, Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, pp 119–128
31. Moratori P, Petrovic S, Vázquez-Rodríguez JA (2012) Match-up approaches to a dynamic rescheduling problem. Int J Prod Res 50:261–276
32. Zakaria Z, Petrovic S (2012) Genetic algorithms for match-up rescheduling of the flexible manufacturing systems. Comput Ind Eng 62:670–686. https://doi.org/10.1016/j.cie.2011.12.001
33. Zhang L, Gao L, Li X (2012) A hybrid intelligent algorithm and rescheduling technique for job shop scheduling problems with disruptions. Int J Adv Manuf Technol 65:1141–1156. https://doi.org/10.1007/s00170-012-4245-6
34. Wang DJ, Liu F, Wang JJ, Wang YZ (2016) Integrated rescheduling and preventive maintenance for arrival of new jobs through evolutionary multi-objective optimization. Soft Comput 20:1635–1652. https://doi.org/10.1007/s00500-015-1615-7
35. Pan QK, Ruiz R (2014) An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem. Omega 44:41–50. https://doi.org/10.1016/j.omega.2013.10.002

36. Lin SW, Ying KC, Huang CY (2013) Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm. Int J Prod Res 51:5029–5038. https://doi.org/10.1080/00207543.2013.790571
37. Ribas I, Companys R, Tort-Martorell X (2011) An iterated greedy algorithm for the flowshop scheduling problem with blocking. Omega 39:293–301. https://doi.org/10.1016/j.omega.2010.07.007
38. Fatih Tasgetiren M, Pan QK, Suganthan PN, Buyukdagli O (2013) A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem. Comput Oper Res 40:1729–1743. https://doi.org/10.1016/j.cor.2013.01.005
39. Ding JY, Song S, Gupta JND, Zhang R, Chiong R, Wu C (2015) An improved iterated greedy algorithm with a Tabu-based reconstruction strategy for the no-wait flowshop scheduling problem. Appl Soft Comput J 30:604–613. https://doi.org/10.1016/j.asoc.2015.02.006
40. Pan QK, Wang L, Zhao BH (2008) An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion. Int J Adv Manuf Technol 38:778–786. https://doi.org/10.1007/s00170-007-1120-y
41. Ruiz R, Stützle T (2008) An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. Eur J Oper Res 187:1143–1159. https://doi.org/10.1016/j.ejor.2006.07.029
42. García-Martínez C, Rodriguez FJ, Lozano M (2014) Tabu-enhanced iterated greedy algorithm: a case study in the quadratic multiple knapsack problem. Eur J Oper Res 232:454–463. https://doi.org/10.1016/j.ejor.2013.07.035
43. Rodriguez FJ, Lozano M, Blum C, García-Martínez C (2013) An iterated greedy algorithm for the large-scale unrelated parallel machines scheduling problem. Comput Oper Res 40:1829–1841. https://doi.org/10.1016/j.cor.2013.01.018
44. Deng G, Gu X (2012) An iterated greedy algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times. Int J Syst Sci 45:351–362. https://doi.org/10.1080/00207721.2012.723054
45. Fanjul-Peyro L, Ruiz R (2010) Iterated greedy local search methods for unrelated parallel machine scheduling. Eur J Oper Res 207:55–69. https://doi.org/10.1016/j.ejor.2010.03.030
46. Congram RK, Potts CN, De VV, Steef L (2002) An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. INFORMS J Comput 14:52–67. https://doi.org/10.1287/ijoc.14.1.52.7712
47. Michalewicz Z, Nazhiyath G (1995) Genocop III: a co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. Evol Comput 1995IEEE Int Conf 2:647–651. https://doi.org/10.1109/ICEC.1995.487460
48. Xiao J, Michalewicz Z, Zhang L, Trojanowski K (1997) Adaptive evolutionary planner/navigator for mobile robots. IEEE Trans Evol Comput 1:18–28. https://doi.org/10.1109/4235.585889
49. Mühlenbein H (1992) Parallel genetic algorithms in combinatorial optimization. Comput Sci Oper Res:441–456
50. Liepins GE, Vose MD (1990) Representational issues in genetic optimization. J Exp Theor Artif Intell 2:101–115. https://doi.org/10.1080/09528139008953717
51. Davis L (1991) Handbook of genetic algorithms. Van Nostrand Reinhold Company
52. Orvosh D, Davis L (1994) Using a genetic algorithm to optimize problems with feasibility constraints. Proc First IEEE Conf Evol Comput:548–553
53. Coello Coello CA (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Methods Appl Mech Eng 191:1245–1287. https://doi.org/10.1016/S0045-7825(01)00323-1
54. Chootinan P, Chen A (2006) Constraint handling in genetic algorithms using a gradient-based repair method. Comput Oper Res 33:2263–2281
55. Ying KC, Cheng HM (2010) Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic. Expert Syst Appl 37:2848–2852. https://doi.org/10.1016/j.eswa.2009.09.006
56. Pranzo M, Pacciarelli D (2016) An iterated greedy metaheuristic for the blocking job shop scheduling problem. J Heuristics 22:587–611. https://doi.org/10.1007/s10732-014-9279-5
57. Appa G, Pitsoulis L, Williams HP (2006) Handbook on modelling for discrete optimization. Springer, US
58. Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann Discret Math 5:287–326
59. Nawaz M, Enscore Jr. EE, Ham I (1983) A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11:91–95
60. Taillard E (1990) Some efficient heuristic methods for the flow shop sequencing problem. Eur J Oper Res 47:65–74. https://doi.org/10.1016/0377-2217(90)90090-X
61. Vallada E, Ruiz R, Framinan JM (2015) New hard benchmark for flowshop scheduling problems minimising makespan. Eur J Oper Res 240:666–677
62. Ruiz R, Allahverdi A (2009) Minimizing the bicriteria of makespan and maximum tardiness with an upper bound on maximum tardiness. Comput Oper Res 36:1268–1283. https://doi.org/10.1016/j.cor.2007.12.013
63. Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64:278–285
64. Taillard E Summary of best known lower and upper bounds of Taillard's instances. http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/flowshop.dir/best_lb_up.txt. Accessed 22 May 2015