Complex & Intelligent Systems (2023) 9:825–849
https://doi.org/10.1007/s40747-022-00830-6

ORIGINAL ARTICLE

Improved NSGA-II for energy-efficient distributed no-wait flow-shop with sequence-dependent setup time

Qing-qing Zeng[1] · Jun-qing Li[1,2] · Rong-hao Li[1] · Ti-hao Huang[1] · Yu-yan Han[1] · Hong-yan Sang[1]

Received: 24 July 2021 / Accepted: 7 July 2022 / Published online: 4 August 2022

Abstract

This paper addresses a multi-objective energy-efficient scheduling problem of the distributed permutation flowshop with sequence-dependent setup time and no-wait constraints (EEDNWFSP), which have important practical applications. Two objectives minimization of both makespan and total energy consumption (TEC) are considered simultaneously. To address this problem, a new mixed-integer linear programming (MILP) model is formulated. Considering the issues faced in solving large-scale instances, an improved non-dominated sorting genetic algorithm (INSGA-II) is further proposed that uses two variants of the Nawaz-Enscore-Ham heuristic (NEH) to generate high-quality initial population. Moreover, two problem-specific speed adjustment heuristics are presented, which can enhance the qualities of the obtained non-dominated solutions. In addition, four local and two global search operators are designed to improve the exploration and exploitation abilities of the proposed algorithm. The effectiveness of the proposed algorithm was verified using extensive computational tests and comparisons. The experimental results show that the proposed INSGA-II is more effective compared to other efficient multi-objective algorithms.

Keywords Distributed permutation flowshop scheduling · Energy-efficient · No-wait · Multi-objective · Improved NSGA-II

Introduction

With rapid economic globalization, transnational cooperation among enterprises has become common practice. Distributed manufacturing improves product quality, reduces production costs, and minimizes management risk [1]. Incidentally, development of the manufacturing industry causes environmental problems, and green manufacturing has been gaining attention. Production scheduling is vital in realizing intelligent green manufacturing [2]. Therefore, it is important to pay more attention to energy consumption in actual production, especially to combine energy-saving with traditional criterions.

The no-wait flow-shop scheduling problem (NWFSP) is a typical production shop scheduling problem, and is widely used in the plastic, chemical, and pharmaceutical industries [3]. When the number of machines exceeds three, the problem is proved to be NP-hard [4]. Specifically, in a no-wait flow-shop, each job needs to be processed continuously to complete a set of processes without interruption. Consequently, the concept of distributed scheduling is added to make the problem more complex. The distributed permutation flowshop scheduling problem (DPFSP) [5] is decomposed into two sub-problems: assign jobs to factories and get the sequence of each job in each factory. In the actual production process, the machine needs to perform additional operations between the processing of consecutive jobs, such as machine cleaning, tool replacement, and operation transportation [6]. Therefore, a sequence-dependent setup time (SDST) is also considered. A literature review of the above-mentioned problems is provided below.

Recently, the distributed flow shop scheduling problem has interested industry and academia. An increasing number of studies have focused on the distributed scheduling of permutation flow shop [6–11], hybrid flow shop [12–15], assembly flow shop [16–18] and so on. Lin et al. [19] developed an iterated cocktail greedy (ICG) algorithm to solve the distributed no-wait flow-shop scheduling problem

✉ Jun-qing Li
lijunqing@lcu-cs.com

1 School of Computer, Liaocheng University, Liaocheng 252059, China

2 School of Information Engineering, HengXing University, Qingdao 266100, China

(DNWFSP). Shao et al. [20] investigated the multiobjective DNWFSP with sequence-dependent setup time and solved it through a Pareto-based estimation of distribution algorithm (PEDA). Komaki and Malakooti [21] addressed the DNWFSP using a General Variable Neighborhood Search (GVNS). Shao et al. [22] proposed iterated greedy (IG) algorithms. Li et al. [23] solved the distributed heterogeneous DNWFSP using a discrete artificial bee colony algorithm (DABC). Other literature considered the sequence-dependent setup time in different actual production system [24–26].

To solve the multi-objective optimization problem of green shop scheduling, Wang et al. [7] proposed a knowledge-based cooperative algorithm (KCA) to solve an energy-efficient scheduling of the distributed permutation flow-shop (EEDPFSP). Wang et al. [8] used a multi-objective whale swarm algorithm (MOWSA) to solve EEDPFSP. Wu et al. [27] proposed a green scheduling algorithm NSGA-II that considered the energy consumption of equipment on–off and different rotating speeds to solve the flexible job shop scheduling problem (FJSP). Du et al. [28] investigated an FJSP with the time-of-use electricity price constraint via a hybrid multi-objective optimization algorithm of estimation of distribution algorithm (EDA) and deep Q-network (DQN) to minimize the makespan and total electricity price simultaneously. Li et al. [29] addressed an FJSP with crane transportation processes using a hybrid of the iterated greedy and simulated annealing algorithms to optimize both the makespan and energy consumption during machine processing as well as crane transportation. Qi et al. [30] researched a multi-objective time-dependent green vehicle routing problems and proposed a Q-learning-based multiobjective evolutionary algorithm to solve it. Jiang et al. [31] developed an effective modified multi-objective evolutionary algorithm with decomposition (MMOEA/D) to solve the energy-efficient distributed job shop scheduling problem. Li et al. [32] designed an energy-aware multi-objective optimization algorithm (EA-MOA) to solve the hybrid flow shop (HFS) scheduling problem with setup energy consumptions. Li et al. [33] introduced a distributed HFS with variable speed constraints using a knowledge-based adaptive reference points multi-objective algorithm. A collaborative optimization algorithm (COA) based on specific properties of the problem was proposed by Chen et al. [34] to solve an energy-efficient distributed no-idle permutation flow-shop scheduling problem. In addition, the indicator based multi-objective evolutionary algorithm with reference point adaptation (AR-MOEA) and hyperplane assisted evolutionary algorithm (hpaEA) are the latest algorithms to solve multi-objective problems. AR-MOEA is versatile in solving problems with various types of Pareto fronts, and is superior to several existing evolutionary algorithms for multi-objective optimization [35]. While solving multi-objective optimization problems, the proportion of non-dominated solutions in the population increases sharply with an increase in the number of objectives. It becomes more challenging to strengthen the selection pressure of population toward the Pareto-optimal front. To address these issues, Chen et al. [36] proposed the hpaEA.

In this study, we investigate a distributed no-wait permutation flow shop scheduling problem with sequence-dependent setup time to minimize the total energy consumption and makespan simultaneously. The contributions of this research work can be summarized as follows: (1) The energy-efficient distributed permutation flow shop scheduling problem with no-wait and setup time constraints in the presence of dynamic speed-scaling technique is formulated and solved for the first time. (2) Two problem-specific effective speed adjustment heuristics are proposed, which perform local enhancement together with four mutation operators. (3) Two types of crossover operators are designed to improve the algorithm's global search abilities.

The remainder of this paper is organized as follows. In "Problem description and formulation", the problem is formally described and the MILP is established. The subsequent section presents the problem-specific properties. and "Improved NSGA-II algorithm" introduces all the components of INSGA-II. In "Experiments and results", numerical experiments and comparative analysis are carried out. "Conclusion" concludes this paper, and points out several research directions.

Problem description and formulation

Definition of multi-objective optimization problem

To explain the proposed EEDNWFSP that considers the optimization of two conflicting objectives, the basic concepts of multi-objective optimization problems are briefly introduced here:

$$\min f(x) = \min\{f_1(x),\ f_2(x),\ \ldots,\ f_m(x)\}, \quad x \in \Omega, \tag{1}$$

where $f_1(x),\ f_2(x),\ \ldots,\ f_m(x)$ denote m conflicting objective functions, and Ω is the search space of a solution x.

(1) Pareto Dominance: For two feasible solutions x and x', if $\forall l \in \{1,\ 2,\ \ldots,\ m\}$, $f_l(x) \le f_l(x')$ and $\exists l' \in \{1,\ 2,\ \ldots m\}$, $f_{l'}(x) < f_{l'}(x')$, then x is said to dominate x' (denoted as $x \succ x'$). If the solution x is not dominated by any other feasible solutions, it is called a non-dominated solution.

(2) Pareto front: All the non-dominated solutions constitute the Pareto-optimal set and their projection in objective space forms the optimal Pareto front.

Description of EEDNWFSP

This paper considers an energy-efficient no-wait permutation flow shop scheduling problem to optimize both makespan and total energy consumption in the presence of dynamic speed-scaling technique. Specifically, there are g parallel factories, each one contains m machines. A set of jobs n can be assigned to any factory $f \in \{1, 2, ..., g\}$ and should be processed according to the same processing order, i.e., from machine $j = 1$ to $j =$ m in the assigned factory. The job processing sequence is the same for all machines in a factory. Formally, for each factory, it is a regular PFSP [5]. Moreover, each job $i \in \{1, 2, ..., n\}$ should be processed between consecutive operations without interruptions; that is, as soon as a job finishes its processing on one machine, it must be processed by the next one immediately. Each machine has a set of variable processing speeds s, which can be selected for each operation. After assigning a processing speed $v \in \{1, 2, ..., s\}$, the operation should be processed at speed v until its completion. Each job needs a basic processing time to be processed at the slowest speed. The basic processing time of jobs is fixed, and the actual processing time is obtained through dividing the basic processing time by the processing speed. Setup time and energy consumption are also considered. As show in Fig. 1, the EEDNWFSP is to assign jobs to factories, determine the job processing sequence of the assigned factory, and select the appropriate processing speed to optimize certain scheduling objectives. In this study, the objective is to minimize the maximum completion time (makespan) and the total energy consumption (TEC) simultaneously.

The assumptions and constraints for this study are as follows:

- All factories have the same processing capacities, i.e., the number of machines and processing ability are the same.
- All machines are available at time zero, and all jobs can be scheduled at this time.
- Each job can be processed on exactly one machine at a time in the same factory.
- Each machine can process only one job at a time.
- Preemption is not permitted, that is, one job should be completed on the assigned machine without any interruption.
- The processing speed of each machine can be adjusted; therefore, the actual processing time and machine energy consumption should be varied with the speed.
- During the processing of a job, the speed of each machine cannot be changed.
- For two successive operations on the same machine, the start processing time of the subsequent operation should be greater than or equal to the completion time of the previous one.
- No-wait constraint should be guaranteed, that is, after completing the previous stage, the job should start its next stage immediately.
- The sequence dependent setup times cannot be ignored.

Notations and parameters

Indexes

i:	Index for jobs, $i = 1, 2, ..., n$
k:	Index for job positions in a sequence, $i = 1, 2, ..., n$
j:	Index for machines, $j = 1, 2, ..., m$
f:	Index for factories, $f = 1, 2, ..., g$
v:	Index for speeds, $v = 1, 2, ..., s$

Parameters

n:	The number of jobs
m:	The number of machines in each factory
g:	The number of factories
s:	The number of processing speeds
$p_{i,j}$:	The standard processing time of job i at machine j
V_v:	The processing speed at v th
$st_{j,i,i'}$:	The setup-time of job i' on machine j if job i is the immediately preceding job
$PE_{j,v}$:	The unit time energy consumption of machine j running at v th speed
SPE_j:	The unit time energy consumption of machine j in stand-by state
$SE_{j,i,i'}$:	The unit setup-time energy consumption of job i' on machine j if job i is the immediately preceding job
M:	A very large positive value

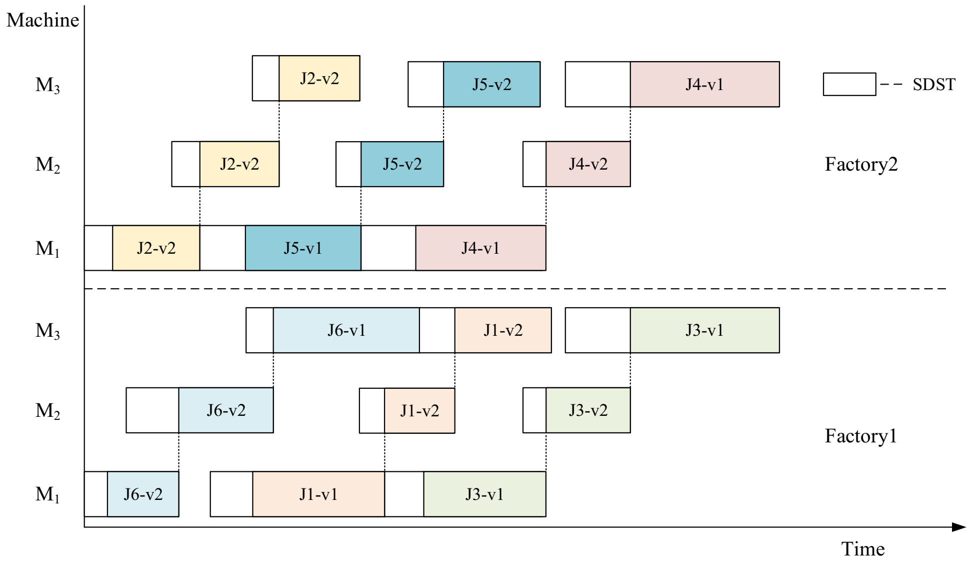


Fig. 1 Example of EEDNWFSP

Variables

$x_{i,f,k}$:	Binary variable, if job i occupies position k in factory f, then the value is 1, and 0 otherwise
$y_{i,j,v}$:	Binary variable, if job i is processed with speed v on machine j, then the value is 1, and 0 otherwise
$t_{i,j}$:	The actual processing time of job i at machine j
$pec_{i,j}$:	The processing energy consumption of job i at machine j
$S_{i,j,f}$:	The time that job i in factory f starts processing on machine j
$C_{i,j,f}$:	The completion time of job i on machine j in factory f
C_f:	The completion time of factory f
$C_{\max}$:	The total completion time, that is, the maximum completion time in g factories
PEC:	The energy consumption of the machine in the processing state
$SPEC$:	The energy consumption of the machine in stand-by state
SEC:	The energy consumption of the machine in the state of setup
TEC	The total energy consumption

Formulation of EEDNWFSP

With the above notations, the following mathematical model of the problem is presented:

Objective functions:

$$\min C_{\max} = \max C_f, \quad f \in \{1, 2, ..., g\}, \tag{2}$$

$$\min TEC = PEC + SPEC + SEC, \tag{3}$$

$$PEC = \sum_{i=1}^{n}\sum_{j=1}^{m} \mathrm{pec}_{i,j}, \tag{4}$$

$$SPEC = \sum_{f=1}^{F}\sum_{j=1}^{m}\Bigg(C_f - \sum_{i=1}^{n}\sum_{k=1}^{n} t_{i,j}\cdot x_{i,f,k} - \sum_{i=1}^{n}\sum_{i'=1}^{n}\sum_{k=1}^{n-1} st_{j,i,i'}\cdot x_{i,f,k}\cdot x_{i',f,k+1} - \sum_{i=1}^{n} st_{j,i,i}\cdot x_{i,f,1}\Bigg)\cdot SPE_j, \tag{5}$$

$$SEC = \sum_{f=1}^{F}\sum_{j=1}^{m}\Bigg(\sum_{i=1}^{n}\sum_{i'=1}^{n}\sum_{k=1}^{n-1} x_{i,f,k}\cdot x_{i',f,k+1}\cdot st_{j,i,i'}\cdot SE_{j,i,i'} + \sum_{i=1}^{n} x_{i,f,1}\cdot st_{j,i,i}\cdot SE_{j,i,i}\Bigg). \tag{6}$$

Objective (2) is to minimize the maximum completion time, where C_f represents the completion time of factory f. Objective (3) is to minimize the total energy consumption, where *PEC* represents the energy consumption when the machines stay at the processing state. *SPEC* represents the energy consumption when the machines stay at the stand-by state. *SEC* represents the energy consumption when the machines stay at the setup state.

Constraints:

$$\sum_{f=1}^{F}\sum_{k=1}^{n} x_{i,f,k} = 1, \quad i \in \{1, 2, ..., n\}, \tag{7}$$

$$\sum_{i=1}^{n} x_{i,f,k} \le 1, \quad k \in \{1, 2, ..., n\}, \quad f \in \{1, 2, ..., g\}, \tag{8}$$

$$\sum_{i=1}^{n} x_{i,f,k} \ge \sum_{i=1}^{n} x_{i,f,k+1}, \quad k \in \{1, 2, ..., n-1\}, \quad f \in \{1, 2, ..., g\}, \tag{9}$$

$$\sum_{v=1}^{s} y_{i,j,v} = 1, \quad i \in \{1, 2, ...n\}, \quad j \in \{1, 2, ..., m\}, \tag{10}$$

$$t_{i,j} = p_{i,j} \cdot \sum_{v=1}^{s} \frac{y_{i,j,v}}{Vv}, \quad i \in \{1, 2, ...n\}, \quad j \in \{1, 2, ..., m\}, \tag{11}$$

$$S_{i,j,f} \ge st_{j,i,i} - M \cdot (1 - x_{i,f,1}), \quad i \in \{1, 2, ...n\}, \quad j \in \{1, 2, ..., m\}, \quad f \in \{1, 2, ..., g\}, \tag{12}$$

$$S_{i',j,f} \ge C_{i,j,f} + st_{j,i,i'} - M \cdot (2 - x_{i,f,k} - x_{i',f,k+1}), \quad i \in \{1, 2, ...n\}, \quad i' \in \{1, 2, ...n\}, \quad k \in \{1, ...n-1\}, \quad j \in \{1, 2, ..., m\}, \quad f \in \{1, 2, ..., g\}, \tag{13}$$

$$S_{i,j,f} = C_{i,j-1,f}, \quad i \in \{1, 2, ...n\}, \quad j \in \{2, ..., m\}, \quad f \in \{1, 2, ..., g\}, \tag{14}$$

$$C_{i,j,f} = S_{i,j,f} + t_{i,j}, \quad i = \{1, 2, ..., n\}, \quad j \in \{1, ..., m\}, \quad f \in \{1, 2, ..., g\}, \tag{15}$$

$$C_f \ge C_{i,m,f}, \quad i \in \{1, 2, ..., n\}, \quad f \in \{1, 2, ..., g\}, \tag{16}$$

$$C_f \ge 0, \quad i \in \{1, 2, ..., n\}, \quad f \in \{1, 2, ..., g\}, \tag{17}$$

$$\mathrm{pec}_{i,j} \ge t_{i,j} \cdot PE_{j,v} - M \cdot (2 - x_{i,f,k} - y_{i,j,v}), \quad i \in \{1, 2, ...n\}, \quad k \in \{1, 2, ..., n\}, \quad j \in \{1, 2, ..., m\}, \quad f \in \{1, 2, ..., F\}, \quad v \in \{1, 2, ..., s\}, \tag{18}$$

$$\mathrm{pec}_{i,j} \ge 0, \quad i \in \{1, 2, ...n\}, \quad j \in \{1, 2, ..., m\}, \tag{19}$$

$$x_{i,f,k} \in \{0, 1\}, \quad i \in \{1, 2, ..., n\}, \quad k \in \{1, 2, ..., n\}, \quad f \in \{1, 2, ..., g\}, \tag{20}$$

$$y_{i,j,v} \in \{0, 1\}, \quad i \in \{1, 2, ..., n\}, \quad j \in \{1, 2, ..., m\}, \quad v \in \{1, 2, ..., s\}. \tag{21}$$

Constraint (7) ensures that each job is assigned to only one factory and to one position in the assigned factory. Constraint (8) requires that a position in a factory is assigned to at most one job. Constraint (9) ensures that the jobs must be assigned at the preceding positions of a factory. Constraint (10) guarantees that each operation Oi, j has one and only one speed processing. Equation (11) calculates the actual processing time. Constraint (12) ensures that the first job assigned to a factory can begin only after the setup is finished, while Constraint (13) restricts that a job (except the job in the first position) can only start after the job at the preceding position as well as the setup have been finished. Constraint (14) ensures that the subsequent operation of each job is started immediately after the completion of the precursor. Constraint (15) specifies that the operation cannot be interrupted. Constraints (16) and (17) are defined to calculate the completion time of the factory f. Constraints (18) and (19) define the intermediate variable $pec_{i,j}$ for calculating the processing energy consumption. All the binary variables are defined in Constraints (20) and (21), where $x_{i,f,k}$ gives the factory assignment of each job and the processing sequence in the assigned factory and $y_{i,j,v}$ gives the speed of each operation.

In the problem, different machines can be set to different units of energy consumption at a given speed. Moreover, it is assumed that machines cannot be turned off until all jobs assigned to the factory are completed. It is obvious that the higher the processing speed, the shorter the processing time and the higher the energy consumption of the machine. Thus, the two optimization objectives are conflicting.

Problem-specific properties

In this section, two lemmas are proposed based on the problem-specific properties. Then, according to these properties and lemmas, two efficient heuristics are designed and utilized in the local search discussed in "DST-based speed adjustment heuristics".

Before the study of problem properties, an assumption is made: when a job $i \in n$ is processed at a higher speed on a machine $j \in m$, the processing time decreases, but the processing energy consumption is increased. This implies that

$$\forall v < v'(v, v' \in s), \ t_{i,j}(v) > t_{i,j}(v'), \quad PE_{j,v} \cdot t_{i,j}(v) < PE_{j,v'} \cdot t_{i,j}(v'), \tag{22}$$

with this assumption, the following property is observed.

Property 1 Consider two solutions $a = (\Pi, V)$ and $b = (\Pi, V')$, that is, the scheduling Π of both are fixed. The solution b is dominated by the solution a if the previous solution has a faster processing speed. Namely, we have $a \prec b$, if the following conditions are satisfied, then $TEC(a) < TEC(b)$:

(1) $C\max(a) \le C\max(b)$;
(2) $\forall i \in \{1, 2, ..., n\}$, $j \in \{1, 2, ..., m\}$, $v_{i,j}(a) \le v_{i,j}(b)$;
(3) $\exists i \in \{1, 2, ..., n\}$, $j \in \{1, 2, ..., m\}$, $v_{i,j}(a) < v_{i,j}(b)$.

This property can be proved by a method similar to Property 2 of Ding et al. [37]. To describe the lemmas in detail, the following definitions are given. For the sake of simplicity, the following $S_{i,j,f}$ represents the starting time of job i on machine j in factory f, that is, it is calculated from the time of preparation.

Definition 1 (*Right side idle time*): right side idle time $R_{i,j}$ of an operation $O_{i,j}$ is calculated as follows:

$$R_{i,j} = S_{i+1,j,f} - C_{i,j,f}. \tag{23}$$

Definition 2 (*Left side idle time*): left side idle time $L_{i,j}$ of an operation $O_{i,j}$ is defined by

$$L_{i,j} = S_{i,j,f} - C_{i-1,j,f}. \tag{24}$$

Definition 3 (*Critical machine*): machine $\underline{j}_{\pi(i)}$ or $\overline{j}_{\pi(i)}$ is called a critical machine for job i if $Ci,\ j,\ f = Si + 1,\ j,\ f$ (for right side idle time) or $Si,\ j,\ f = Ci - 1,\ j,\ f$ (for left side idle time). Note that there is only one $\underline{j}_{\pi(i)}$ and one $\overline{j}_{\pi(i)}$ for job i, that is, once found, the search will stop.

In particular, for the first processing job in the scheduling, there is only the right idle time. And for the last processing job, there is only the left idle time. A detailed explanation is provided in Fig. 2. It is obvious that $R_{i,j}$ of job i refers to the same area as $L_{i,j}$ of job $i + 1$.

Lemma 1 *For each job i, find its* $\underline{j}_{\pi(i)}$ *from* M_m *to* M_1*. If the critical machine is not used for the last operation of the job, there is* $R_{i,j}$ *of each subsequent operation. Slowing down the processing speed under the premise that the* $\Delta PT_{i,j}$ *will not exceed the minimum* $R_{i,j}$ *can reduce both the PEC and SPEC.*

Proof To prove this lemma, we first show that slowing down any operation that satisfies $\Delta PT_{i,j} \le \min R_{i,j}$ does not worsen the makespan. For each $O_{i,j}$, this condition guarantees that the starting time of job $i + 1$ is not be delayed. Both the start and completion times of all operations after the $O_{i,j}$ for job i are changed, as shown in Eqs. (26) and (27).

$$\Delta PT_{i,j} = \frac{p_{i,j}}{v'} - \frac{p_{i,j}}{v}, \tag{25}$$

$$C_{i,j,f} = C_{i,j,f} + \Delta PT_{i,j}, \tag{26}$$

$$S_{i,j+1,f} = S_{i,j+1,f} + \Delta PT_{i,j}. \tag{27}$$

In particular, for job i, all operations can be processed in advance if there is no $S_{i,j,f} = C_{i-1,j,f}$ due to the start time that is delayed. In this case, the completion time is be reduced, as shown in Fig. 2b and d.

Lemma 2 *For each job i, find its critical machine from* M_1 *to* M_m*. If the* $\overline{j}_{\pi(i)}$ *is not* M_1*, there is* $L_{i,j}$ *of each preceding operation. Slowing down the processing speed under the premise that the* $\Delta PT_{i,j}$ *will not exceed the minimum* $L_{i,j}$ *can reduce both the PEC and SPEC.*

Proof Similar to the proof of Lemma 1, for each $O_{i,j}$, condition $\Delta PT_{i,j} \le \min L_{i,j}$ ensures that the completion time of all jobs will not be delayed. Besides, the time of all operations before the $O_{i,j}$ for job i are changed, as shown in Eqs. (28) and (29).

$$S_{i,j,f} = S_{i,j,f} - \Delta PT_{i,j}, \tag{28}$$

$$C_{i,j-1,f} = C_{i,j-1,f} - \Delta PT_{i,j}. \tag{29}$$

Similarly, for job $i + 1$, all operations can be processed in advance if there is no $S_{i+1,j,f} = C_{i,j,f}$. As shown in Fig. 2b and f, the completion time will also reduce.

In sum, by slowing down the operations that satisfy $\Delta PT_{i,j} \le \min R_{i,j}$ or $\Delta PT_{i,j} \le \min L_{i,j}$, there is always $C_{\max}(a) \le C_{\max}(b)$. And combined with $v_{i,j}(a) < v_{i,j}(b)$, it follows from Property 1 that $TEC(a) < TEC(b)$.

Improved NSGA-II algorithm

Framework of INSGA-II

The nondominated sorting genetic algorithm II (NSGA-II) presented by Deb et al. [38] is a classical algorithm tailored to solve multi-objective problems. It has three outstanding contributions to address the shortcomings of the NSGA: fast nondominated sorting, crowded-comparison approach, and a novel elite selection strategy. Although the existing NSGA-II has made creative improvements in the above three aspects and showed the superiority to generate good individuals, two classical genetic operators of crossover and mutation have not been further researched. The traditional crossover and mutation operators are random and aimless, which cannot guarantee the generation of high-quality offspring and affects the efficacy of the algorithm. Therefore, an improved nondominated sorting genetic algorithm (INSGA-II) is proposed

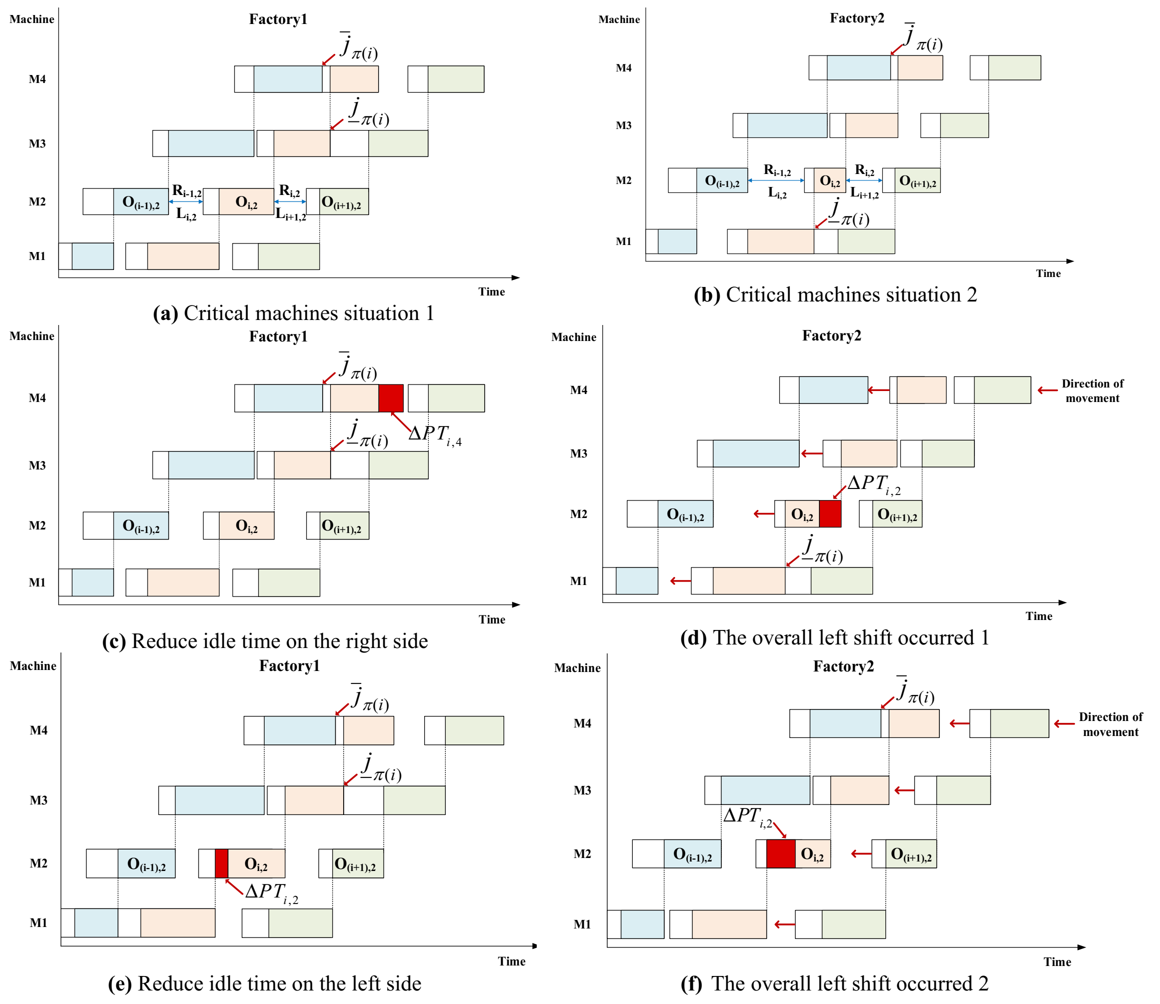

Factory1
Factory2
Machine
M1
M2
M3
M4
Time
$O_{(i-1),2}$
$O_{i,2}$
$O_{(i+1),2}$
$R_{i-1,2}$
$L_{i,2}$
$R_{i,2}$
$L_{i+1,2}$
$\bar{j}_{\pi(i)}$
$\underline{j}_{\pi(i)}$
$\Delta PT_{i,4}$
$\Delta PT_{i,2}$
Direction of movement
(a) Critical machines situation 1
(b) Critical machines situation 2
(c) Reduce idle time on the right side
(d) The overall left shift occurred 1
(e) Reduce idle time on the left side
(f) The overall left shift occurred 2

Fig. 2 Illustration of deceleration and left shift

here for arriving at the multi-objective solutions of the problem.

The proposed improved NSGA-II algorithm is presented, which includes the main framework, an encoding and decoding method, two efficient initialization heuristics, two problem-specific speed adjustment heuristics combined with four mutation operators to form a local search strategy, and two crossover operators to conduct global search. The framework of the INSGA-II for solving the EEDNWFSP is described in Fig. 3.

Encoding and illustration

For the EEDNWFSP, a solution is represented by two vectors, that is, the scheduling vector, which contains the factory assignment, and the speed vector for each job processed on each machine. The scheduling vector contains $n + g - 1$ elements [39], i.e., $\Pi = (\pi_1, \pi_2, ..., \pi_i, ..., \pi_{n+g-1})$, where $\pi_i \in \{0, 1, 2, ..., n\}$. There are n indexes of jobs and $g - 1$ indexes with value '0' as the separators. The separators divide Π into g sections, each of which contains a scheduling sequence of partial jobs. The speed of each operation is listed in $m * n$ elements. Thus, a solution is encoded as $(\Pi, V) = (\pi_1, \pi_2, ..., \pi_i, ..., \pi_{n+g-1}; v_{1,1}, ..., v_{1,m}, ..., v_{n,1}, ...v_{n,m})$. As show in Fig. 4.

A simple example is provided to understand the encoding and decoding method. There are two parallel factories, each of which has three machines. Each machine has two different speeds for processing. And there are six jobs to be processed. That is, $g = 2$, $m = 3$, and $n = 6$. The standard processing times, the energy consumption of each machine run at each speed and in stand-by mode are listed in Table 1. The sequence-dependent setup time and the setup energy consumption are given in Table 2.

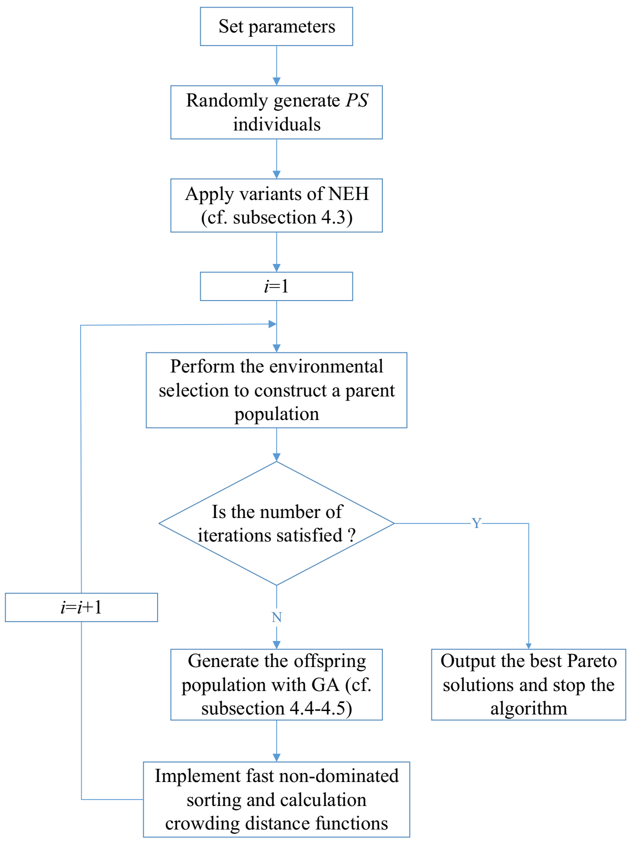

Set parameters
Randomly generate *PS* individuals
Apply variants of NEH (cf. subsection 4.3)
i=1
Perform the environmental selection to construct a parent population
Is the number of iterations satisfied ?
Y
N
i=*i*+1
Generate the offspring population with GA (cf. subsection 4.4-4.5)
Output the best Pareto solutions and stop the algorithm
Implement fast non-dominated sorting and calculation crowding distance functions

Fig. 3 Framework of the INSGA-II

Table 1 The standard processing time and the unit energy consumption

	M_1	M_2	M_3
Speed/PEC	1/2	1/4	1/2
	2/6	2/12	2/6
SPEC	1	2	1
J_1	32	21	24
J_2	21	20	31
J_3	11	29	18
J_4	29	12	10
J_5	28	18	24
J_6	14	13	33

Table 2 The SDST and the setup energy consumption

		J_1	J_2	J_3	J_4	J_5	J_6
M_1	J_1	3/1	2/1	1/2	3/1	4/1	2/2
	J_2	3/2	2/2	8/2	6/2	3/1	6/2
	J_3	2/1	4/2	6/1	2/1	8/2	5/1
	J_4	3/1	5/1	5/1	4/2	6/1	9/2
	J_5	9/2	8/2	3/2	5/1	10/2	3/2
	J_6	3/1	9/2	5/2	2/2	3/2	1/1
M_2	J_1	5/1	3/1	5/2	10/2	9/2	6/1
	J_2	8/2	9/2	4/1	9/2	9/2	3/1
	J_3	2/2	4/2	8/1	5/1	10/1	6/2
	J_4	3/1	5/1	5/2	8/2	6/1	4/2
	J_5	3/1	4/2	6/2	5/1	10/1	7/2
	J_6	6/2	9/1	5/1	7/2	8/1	10/1
M_3	J_1	3/1	6/1	2/1	9/1	5/2	8/2
	J_2	3/2	10/2	8/2	6/2	5/1	6/2
	J_3	2/2	4/1	6/1	5/1	8/1	3/1
	J_4	4/2	5/2	5/2	10/2	8/2	3/2
	J_5	9/1	8/1	5/1	5/1	10/1	6/2
	J_6	3/2	9/1	5/2	2/2	3/2	6/1

For ease of presentation, V is presented as the following matrix:

$$V = \begin{bmatrix} 2, & 2, & 1, & 2, & 2, & 2 \\ 2, & 2, & 2, & 1, & 1, & 2 \\ 2, & 2, & 2, & 1, & 1, & 2 \end{bmatrix}^T .$$

Considering a solution $\Pi = \{2, 5, 4, 0, 6, 3, 1\}$, a detailed description is given as follows: job 2 is processed first, then jobs 5 and 4 are processed in turns in factory 1. The real processing time of job 2 on machine 1 is $t_{2,1} = p_{2,1}/V_{2,1} = 21 \div 2 = 10.5$. The operation times related to the sequence are $st_{1,2,2} = 2$, $st_{2,2,2} = 9$, $st_{3,2,2} = 10$. Consequently, the completion time of factory 1 is the completion time of job 4 on machine 3, which is $C(1) = 88.5$ after calculation. Similarly, the completion time of factory 2 can be obtained $C(2) = 66.5$. Thus, $C_{\max}$ is 88.5. Next, the calculation of energy consumption is explained. For instance, in factory 1, machine 1–3 run at the speed of 2, 2, and 2 when processing job 2, respectively. Thus, the energy consumption of job 2 in the

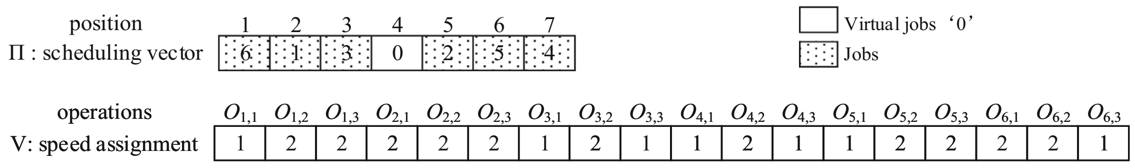

position
1 2 3 4 5 6 7
Π : scheduling vector
6 1 3 0 2 5 4
Virtual jobs '0'
Jobs
operations
$O_{1,1}$ $O_{1,2}$ $O_{1,3}$ $O_{2,1}$ $O_{2,2}$ $O_{2,3}$ $O_{3,1}$ $O_{3,2}$ $O_{3,3}$ $O_{4,1}$ $O_{4,2}$ $O_{4,3}$ $O_{5,1}$ $O_{5,2}$ $O_{5,3}$ $O_{6,1}$ $O_{6,2}$ $O_{6,3}$
V: speed assignment
1 2 2 2 2 2 1 2 1 1 2 1 1 2 2 2 2 1

Fig. 4 Encoding of EEDNWFSP

processing and setup states is calculated as follows: $PEC_2 = t_{2,1} \cdot PE_{1,2} + t_{2,2} \cdot PE_{2,2} + t_{2,3} \cdot PE_{3,2} = 10.5 \times 6 + 10 \times 12 + 15.5 \times 6 = 276$; $SEC_2 = st_{1,2,2} \cdot SE_{1,2,2} + st_{2,2,2} \cdot SE_{2,2,2} + st_{3,2,2} \cdot SE_{3,2,2} = 2 \times 2 + 9 \times 2 + 10 \times 2 = 42$. Therefore, the *TEC* can be calculated by accumulating the *PEC*, *SPEC*, and *SEC* of each factory.

Initialization procedure

In multi-objective evolutionary algorithms, the quality of the initial population has an important impact on the algorithm performance. As we know, the two objectives considered in this paper conflict with each other, and finding the extreme value of each objective can effectively guide optimization. The NEH [40] heuristic has been identified as one of the most efficient constructive heuristics for the PFSP with the makespan criterion [41]. Combined with distributed characteristics, Ruiz et al. [5] proposed two factory assigned rules that extended the basic NEH. In addition, Wang et al. [7] proposed the extended NEHFF (NEHFF2) that considers energy efficiency. Inspired by these previous studies and based on the two current objectives, we propose two extended distributed NEH heuristics, named ENEH and ENEH2, respectively. In this paper, random initialization is performed, and then two individuals are replaced by two solutions generated by ENEH and ENEH2. The procedure of ENEH is shown in Algorithm 1. The difference between ENEH2 and ENEH is that the TEC is compared first in Step 14. Then, the makespan is compared so as to construct another solution that is more optimal in terms of energy consumption.

Local search

DST-based speed adjustment heuristics

Based on Lemmas 1 and 2 discussed in "Problem-specific properties", two effective speed adjustment heuristics are proposed. For either of the two algorithms, a new neighborhood solution is constructed by appropriately slowing down the speed matrix V. Therefore, a better solution is obtained, which can reduce the TEC without increasing the makespan values. The two heuristics, named dynamic speed-scaling technique 1 (DST1) and 2 (DST2), are described in Algorithm 2 and Algorithm 3 respectively.

Mutation operators

For the optimization of completion time, several search operators are designed. Specifically, this section proposes four local search methods to improve the solution using insert and swap operators between or within factories.

Definition 4 (*Critical factory*): critical factory f_c is the one with the maximum completion time.

(1) Operators to adjust factory assignment:

FA_i: a job is randomly removed from f_c and inserted into a random position in another factory. The factory to be inserted is selected from small to large according to completion times.

FA_s: a job i is randomly selected from a factory and job i' from another factory is randomly selected, and then the two jobs are swapped.

First, the factories are arranged in the descending order of completion times. Then while swapping, the first factory swaps with the last factory, the second factory swaps with the penultimate factory, and so on.

(2) Operators to adjust the scheduling of jobs in the same factory:

J_i**:** randomly select two jobs from the same factory and then insert the successor into the position before the previous one.

J_s**:** randomly select two jobs from the same factory and then swap them.

In order to find a better solution, based on the above four search operators and the two speed adjustment heuristics in "DST-based speed adjustment heuristics", a novel local search strategy is proposed which is described in Algorithm 4. To be specific, it combines four mutation operators and two speed adjustment heuristics and implements them according to probability. First, the completion time of each factory is normalized by Formula (28), where the $F_{\min}$ is 0. Then, find the factory ($C_{\min}$) with the smallest completion time, and select the corresponding mutation operator according to the completion time. For factories with $C_{\min} < 0.8$, FA_i is executed because it means that the completion times of this factory and critical factory fc are quite different. The insertion operator between the factories is used to balance the completion time. The remaining three mutation operators are executed with almost an equal probability. Finally, one of DST1 and DST2 is arbitrarily selected in accordance with the equal probability to adjust the speed.

$$C_t = (C_t - F_{\min})/(F_{\max} - F_{\min}). \tag{30}$$

Global search

Considerable research on intelligent scheduling exists, which focus on designing crossover operators for specific problems

Algorithm 1: Procedure of the ENEH heuristic

Input: the number of jobs n, number of factories g, number of machines m, process time of operations Ot

Output: a feasible solution

```
1:  Randomly generate the speed part, and its length is $n*m$.
2:  The sum of the standard processing time $Ot$ of each job on all machines is calculated and sorted in the non-
    ascending order. That is $\Pi_0 = (\pi_0(1), \pi_0(2), ..., \pi_0(n))$.
3:  For $f$ = 1 to $g$
4:      $\Pi_f(1) = \Pi_0(f)$      /* Assign a job to each factory. */
5:  End For
6:  For $k = g$+1 to $n$
7:      $i = \Pi_0(k)$
8:      For $f$ = 1 to $g$
9:          Insert the job $i$ into all possible positions to get all the candidate sequences $\Pi_{f'}$.
10:     End For
        /*There will be $g+k-1$ possible positions, so $g+k-1$ candidate sequences will be obtained.*/
11:     For $l$ = 1 to $g$+$k$-1
12:         The makespan and TEC of each candidate sequence are calculated.
13:     End For
14:     Find all the candidate sequences with the minimum makespan.
15:     If multiple sequences have the same makespan, the $TEC$ is compared.
16:     If there are multiple sequences with the same makespan and $TEC$, randomly select one from them.
17:     Replace the original sequence $\Pi_f$ in the corresponding factory with the selected candidate sequence $\Pi_{f'}$.
        And the scheduling sequence in other factories remains unchanged.
18: End For
```

to improve the performance of evolutionary algorithms. Specially, Han et al. [42] proposed two enhanced crossover operators based on similar block order crossover (SBOX) [43] and artificial chromosome (ACJOX) [44], which are named improved SBOX (ISBOX) and improved SJOX (ISJOX), respectively. Inspired by the idea of using the information of non-dominated solutions to preserve good gene blocks, the corresponding ISBOXII and ISJOXII are conceived according to the specific encoding.

The steps of ISBOXII are presented as follows:

Step 1: for each job, count the number of subsequent jobs in the current non-dominated solution set to find the one with the most occurrences. A temporary set is consisted of these gene pairs.
Step 2: two individuals were randomly selected as parents from the parent population.
Step 3: in a parent, for a job in each position, a gene pair is formed with its subsequent job, and the gene pair is searched in the temporary set.
Step 4: if the parent and the temporary set have the common gene pairs, the identical gene pairs are put into offspring at the same position.
Step 5: otherwise, compare two parent genes in the same position, and put the common gene into the same position of the corresponding offspring respectively.
Step 6: the genes of the offspring in the rest positions are filled using the one-point order crossover (OP) [45] based on the two parents.

An example is provided below to illustrate how ISBOXII works. Suppose that there are five non-dominated solutions in the current set, expressed as Π_i, $i = 1,2,3,4,5$, each containing seven jobs and two factories. The crossover operator only considers the operation of the scheduling part and ignores the speed part temporarily. That is, the speed part uses random crossover. Their expressions are given in Fig. 5. For all the non-dominated solutions, count the times job j ($j = 1,2,\ldots,7$) appears immediately after job i ($i = 1,2,\ldots,7$). Moreover, according to the occurrence of the highest number of gene pairs, a temporary set {(1, 4), (2, 1), (3, 5), (4, 5), (5, 2), (6, 3), (7,6)} I s obtained, as shown in Fig. 6.

Algorithm 2: Procedure of the DST1

Input: a feasible solution

Output: an improved solution

```
1: For $f$ = 1 to $g$
2:   If ($jobnum$ > 1) then    /* only optimize factory with more than one jobs */
3:     For $i$ = 1 to $jobnum$-1   /* the last job does not need to be considered */
4:       Calculate the start and finish times of each operation.
5:       For $j = m$ to 1   /* Find the critical machine $\underline{j}_{\pi(i)}$ for job i from $M_m$ to $M_1$ */
6:         If ($C_{i,j,f} = S_{i+1,j,f}$) then
7:           If ($j == m$) then   /* no opportunity to optimize */
8:             break
9:           Else
10:            For $k = j+1$ to $m$
11:              Calculate $R_{i,j}$ for $j \in [M_k, M_m]$ and find $\min R_{i,j}$.
12:              If the processing speed of the current operation $O_{i,k}$ can be reduced and the increased
                 processing time $\Delta PT_{i,j}$ is not greater than $\min R_{i,j}$, the most appropriate processing
                 speed is selected.
13:              Adjust the start and completion times of all subsequent operations for job $i$.
14:            End For
15:          End If
16:          break
17:        End If
18:      End For
19:    End For
20:  End If
21: End For
```

π_1	2	6	3	0	1	4	5	7
π_2	1	4	7	6	0	3	5	2
π_3	2	7	6	4	0	3	5	1
π_4	7	6	0	5	4	3	2	1
π_5	4	5	2	0	6	3	7	1

Fig. 5 Non-dominated solution set

	follower						
	1	2	3	4	5	6	7
job1	0	0	0	2	0	0	0
job2	1	0	0	0	0	1	1
job3	0	1	0	0	2	0	1
job4	0	0	1	0	2	0	1
job5	1	2	0	1	0	0	1
job6	0	0	2	1	0	0	0
job7	1	0	0	0	0	3	0

Fig. 6 Temporary set

Then, two parents from the population are randomly selected, e.g., *parent1* = (1, 6, 7, 4, 0, 3, 5, 2) and *parent2* = (5, 2, 7, 0, 1, 4, 3, 6), the common gene pairs between parent 1 and the temporary set, i.e., (3, 5) and (5, 2), are sought and put into offspring 1 at positions 6–8. The same gene 7 was also located in the same position in two parents, which was put into position 3 of offspring 1 and 2, respectively.

To obtain the genes for the unfilled position of offspring 1, a crossover point is randomly generated between positions

Algorithm 3: Procedure of the DST2

Input: a feasible solution

Output: an improved solution

```
1: For f = 1 to g
2:   If (jobnum > 1) then        /* only optimize factory with more than one jobs */
3:     For i = 2 to jobnum   /* the first job does not need to be considered */
4:       Calculate the start and finish times of each operation.
5:       For j = 1 to m    /* Find the critical machine $\bar{j}_{\pi(i)}$ for job i from $M_1$ to $M_m$ */
6:         If ( $S_{i,j,f} = C_{i-1,j,f}$ ) then
7:           If (j == 1) then    /* no opportunity to optimize */
8:             break
9:           Else
10:            For k = j-1 to 1
11:              Calculate $L_{i,j}$ for $j \in [M_1, M_k]$ and find $\min L_{i,j}$ .
12:              If the processing speed of the current operation $O_{i,k}$ can be reduced and the increased
                 processing time $\Delta PT_{i,j}$ is not greater than $\min L_{i,j}$ , the most appropriate processing speed
                 is selected.
13:              Adjust the start and completion times of all operations of job i processed before $O_{i,k}$ .
14:            End For
15:          End If
16:          break
17:        End If
18:      End For
19:    End For
20:  End If
21: End For
```

1 and 2, and the OP operator is performed on parents 1 and 2, leading *offspring1* to (1, 0, 7, 4, 6, 3, 5, 2). Similarly, *offspring2* = (5, 2, 7, 6, 1, 4, 0, 3) is generated. The whole process of generating offspring 1 and 2 is illustrated in Fig. 7.

The second new crossover operator, ISOJXII, can be described as follows. First, a temporary individual is generated based on the current non-dominated solution set. Then, two parents are randomly selected from the population, and the genes of each parent are compared with those of the temporary individual at the same position. Similarly, genes at the same position are compared between the two parents. If they are the same, the gene is put in the same position of its offspring. Furthermore, similar to ISBOXII, the genes in the rest positions of the offspring are generated by performing the OP operator between the two parents.

In the following, the same example as above is used to illustrate the main work of ISOJXII. Count the number of times job i appears in all the non-dominated solutions at position$k$ (k = 1,2,…,8). Then place the jobs with the most occurrences at each location, and obtain the temporary individual (2, 6, 2, 4, 1, 3, 5, 1). The generation process is shown in Fig. 8.

Then, randomly select two parents using the above-mentioned same example. To obtain the genes for the unfilled position of offspring 1, a crossover point is randomly generated between positions 2 and 3. Then the OP operator is performed on parents 1 and 2, leading *offspring1* to (1, 6, 7, 4, 2, 3, 5, 0). Similarly, *offspring2* = (5, 2, 7, 6, 1, 4, 0, 3) is generated. The whole process of generating offspring 1 and offspring 2 is illustrated in Fig. 9.

The framework of the proposed INSGAII is shown in Algorithm 5.

```
Algorithm 4: Procedure of Local Search
Input: a feasible solution, the completion time of each factory
Output: an improved solution
1:  Find the maximum completion time $Fmax$ from all the completion times.
2:  Normalize the completion time $Ct$ of all factories, that is $Normal(Ct) = Ct / F\max$.
3:  Find out the minimum one, as $Cmin$.
4:  If $Cmin < 0.8$ then      /* There is a large gap in the completion time between factories. */
5:      $FA_i$ is executed
6:  Else
7:      Generate a random number $k$. /* One of the following three operations is randomly selected according to probability.*/
8:      If $k < 0.3$ then
9:          $J_i$ is executed
10:     Else If $k < 0.6$ then
11:         $J_s$ is executed
12:     Else
13:         $FA_s$ is executed
14:     End If
15: End If
16: Generate a random number $k2$. /* One of the two heuristics is randomly selected according to probability. */
17: IF $k2 < 0.5$ then
18:     The individual is optimized by the DST1.
19: Else
20:     The individual is optimized by the DST2.
21: End If
```

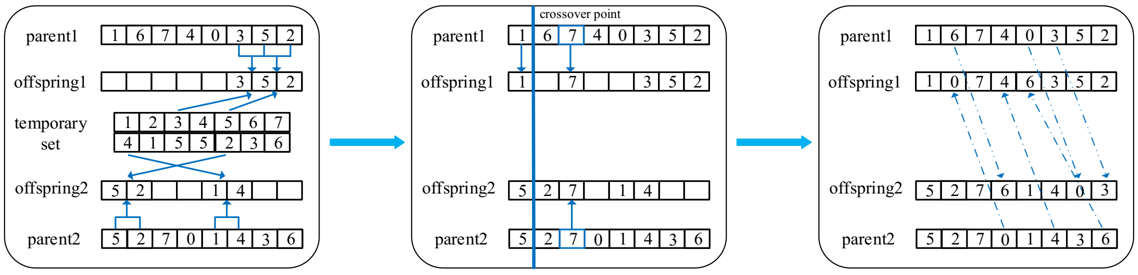


Fig. 7 Process of ISBOXII

Experiments and results

Experimental setup

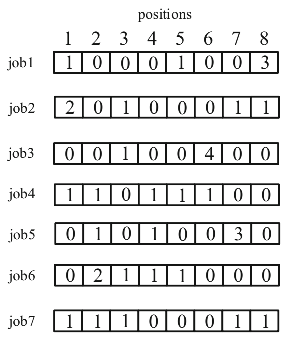
This section discusses computational experiments used to evaluate the performance of the proposed algorithm. All the algorithms are implemented in the PlatEMO v3.0 on a DELL with Intel Core i7-10,700 CPU operating at 2.90 GHz and 8 GB RAM, and the same library functions are employed to make peer comparisons. Furthermore, all the compared algorithms are recoded to adapt them to solve the considered problem, including the encoding and decoding method. The parameters are set in accordance with the literature. In this paper, the population size is 100. For each instance, the stopping criterion is set to 200 iterations. To verify the effectiveness and efficiency of the proposed algorithm, after 30 independent runs, the resulted non-dominated solutions found by all the compared algorithms were collected for performance comparisons. The relative percentage increase (RPI) is used for the ANOVA comparison, which is calculated as follows:

Algorithm 5: Framework of INSGAII

Input: the parent population P, crossover rate pc, mutation rate pm, population size PS

Output: a new population

```
1:  Generate PS random numbers as the crossover point of each individual's scheduling part.
2:  PS random numbers are generated as the crossover point of each individual's speed part.
3:  For i =1 to PS/2
4:  |   P1=P(i) /* Extract the two parents to generate offspring. */
5:  |   P2=P(i+PS/2)
6:  |   The crossover rate rc is generated randomly.
7:  |   The mutation rate rm is generated randomly.
8:  |   IF rc < pc then /* Perform crossover operations. */
9:  |   |   Generate a random number rc2. /* One of the following two operations is randomly selected according to the
    |   |   probability. */
10: |   |   If rc2<0.5 then
11: |   |   |   Implement the ISJOXII
12: |   |   Else
13: |   |   |   Implement the ISBOXII
14: |   |   End If
15: |   |   Carry out one-point order crossover on the speed part.
16: |   End If
17: |   If rm < pm then /* Perform mutation operations. */
18: |   |   Perform the local search of Algorithm 4. (c.f. Section 4.4.2) for P1 and P2 respectively.
19: |   End If
20: End For
```



	positions							
	1	2	3	4	5	6	7	8
job1	1	0	0	0	1	0	0	3
job2	2	0	1	0	0	0	1	1
job3	0	0	1	0	0	4	0	0
job4	1	1	0	1	1	1	0	0
job5	0	1	0	1	0	0	3	0
job6	0	2	1	1	1	0	0	0
job7	1	1	1	0	0	0	1	1

Fig. 8 Process of generating a temporary individual

The relative percentage increase (RPI) is used as a performance measure and is calculated as follows:

$$RPI(C) = \frac{C_c - C_b}{C_b} \times 100, \tag{31}$$

where C_b is the best fitness value of all the compared algorithms, and C_c denotes the fitness value of the current algorithm. The fitness values used in this paper are the performance indicators HV and IGD mentioned in "Performance indicators".

In order to test the performance of INSGA-II, we generate test instances based on the literature [46, 47]. To be specific, a set of instances includes several combinations of the number of jobs, machines, and factories, i.e., the combinations of $n = \{20, 40, 60, 80, 100\}$, $m = \{4, 8, 16\}$, $g = \{2, 3, 4, 5\}$. The processing time $P_{i,j}$ is uniformly distributed within the range of [5 h, 50 h] and the processing speed v is set as $\{1, 2, 3\}$. The EC is set as $PE_{j,v} = 4 \times vkW$, $SPE_j = 1kW$ and the $SE_{j,i,i'}$ is generated uniformly within the range [1 kW, 2 kW]. The setup times are 50% of the processing times, that is, the setup times are generated by a uniform [2 h, 25 h] distribution. The instances and experimental results can be obtained from the website: http://ischedulings.com/data/CAIS_EEDNWFSP.rar.

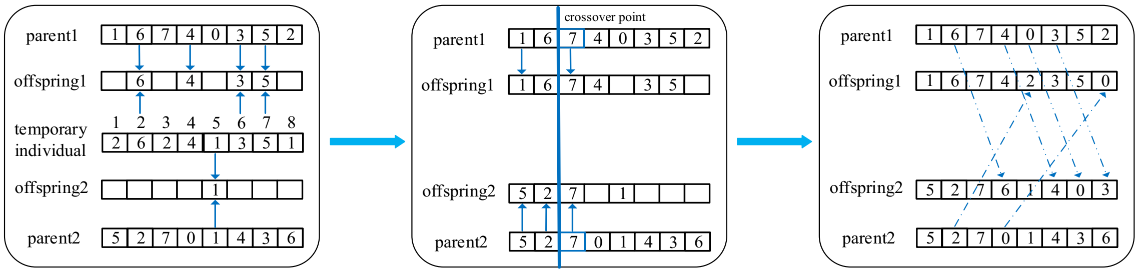


Fig. 9 Process of ISJOXII

Performance indicators

Since the exact Pareto front of the investigated problem is unknown, we use a so-called reference set to approximate it. The reference set is generated using a method similar to that of Wu et al. [3]. Specifically, all the non-dominated solutions obtained by the comparison algorithms are combined into a set, from which the dominated solutions are removed to obtain the reference set. In this study, each instance is solved by all the comparison algorithms iterating 30,000 times independently, and the non-dominated solutions obtained are considered the final reference set.

To assess the quality of the obtained solutions, the following two representative indicators are used in terms of convergence and diversity: the hypervolume (HV) [48] and the inverted generational distance (IGD) [49]. The detailed calculation process of the two indicators is as follows.

(1) hypervolume HV

$$HV = \delta\left(\bigcup_{i=1}^{|S|} v_i\right). \tag{32}$$

The Lebesgue measure is a metric used to measure the volume, denoted by δ. Where $|S|$ is the number of non-dominated solutions and v_i is the hypervolume of the ith solution in the reference solution set. The larger the volume of the region in the target space surrounded by the non-dominated solution set and reference points, the better the comprehensive performance of the algorithm. In this work, we select (1, 1) as the reference point.

(2) inverted generational distance IGD

$$IGD\left(P,\ P^*\right) = \frac{\sum_{x\in P^*} \min_{y\in P} \mathrm{dis}(x,\ y)}{|P^*|}. \tag{33}$$

IGD is used to compute the average distance from each reference solution to the nearest solution. Here P is the solution set obtained by the algorithm, and P^* represents a group of uniformly distributed reference solutions sampled from the reference solution set. The dis$(x,\ y)$ represents the Euclidian distance between the solution x in the reference solution set P^* and the solution y in the solution set P.

Table 3 Parameter values

Parameter	Factor level				
	1	2	3	4	5
pc	0.2	0.4	0.6	0.8	1.0
pm	0.2	0.4	0.6	0.8	1.0

Parameter setting

In the proposed algorithm, two parameters crossover probability (pc), and mutation probability (pm) have the main effects for the performance. In this section, we conduct a design of experiments (DOE) test for selecting the levels of the two parameters. More precisely, we apply a full factorial design using the two parameters as factors. Five levels are considered for each parameter, as listed in Table 3. Based on the results of detailed experiments, parameters of *pc* and *pm* are set to 0.8 and 0.4, respectively.

Efficiency of the proposed components

Effect of initialization

To investigate the effectiveness of the ENEH heuristics discussed in "Initialization procedure", we compare the INSGAII to the INSGAII with random initialization (denoted as G1). A multi-factor analysis of variance (ANOVA) is performed to test whether the performance differences between the algorithms are significant, and the two compared algorithms are considered as factors. Tables 4 and 5 report the comparison results of HV and IGD values for the given 60 different scale instances (each instance runs 10 times independently). Moreover, the 60 instances are further classified according to the number of factories. In the tables, the first column gives the instance number. Then,

Table 4 Comparisons of the HV values between INSGAII and G1

$n \times m$	HV($g = 2$)		HV($g = 3$)		HV($g = 4$)		HV($g = 5$)	
	INSGAII	G1	INSGAII	G1	INSGAII	G1	INSGAII	G1
20 × 4	**0.6958**	0.6877	**0.6692**	0.6594	**0.6706**	0.6603	**0.7495**	0.7474
20 × 8	**0.6612**	0.6560	**0.7135**	0.7100	**0.7305**	0.7239	**0.6907**	0.6872
20 × 16	**0.6194**	0.6167	0.6984	**0.7008**	**0.7199**	0.7135	**0.7324**	0.7192
40 × 4	**0.6851**	0.6639	**0.7157**	0.6860	**0.6695**	0.6434	**0.6794**	0.6533
40 × 8	**0.6483**	0.6183	**0.6476**	0.6252	**0.6331**	0.6098	**0.7386**	0.7274
40 × 16	**0.6115**	0.5900	**0.6092**	0.5789	**0.6750**	0.6641	**0.6964**	0.6923
60 × 4	**0.6549**	0.6282	**0.6905**	0.6712	**0.6732**	0.6396	**0.7021**	0.6750
60 × 8	**0.6333**	0.5981	**0.6360**	0.6096	**0.6978**	0.6797	**0.7028**	0.6940
60 × 16	**0.5986**	0.5740	**0.6021**	0.5748	**0.6085**	0.5755	**0.6790**	0.6691
80 × 4	**0.6499**	0.6136	**0.6692**	0.6293	**0.6472**	0.6171	**0.6490**	0.6171
80 × 8	**0.6386**	0.6090	**0.6556**	0.6202	**0.6366**	0.5970	**0.6342**	0.5945
80 × 16	**0.5989**	0.5708	**0.5959**	0.5671	**0.6051**	0.5703	**0.5936**	0.5504
100 × 4	**0.6586**	0.6168	**0.6460**	0.6022	**0.6975**	0.6659	**0.6528**	0.6027
100 × 8	**0.6264**	0.5920	**0.6229**	0.5807	**0.6251**	0.5826	**0.6343**	0.5824
100 × 16	**0.6253**	0.5883	**0.6037**	0.5810	**0.6083**	0.5836	**0.6732**	0.6561
Avg	**0.6404**	0.6149	**0.6517**	0.6264	**0.6599**	0.6351	**0.6805**	0.6579

Values in bold mean the best ones

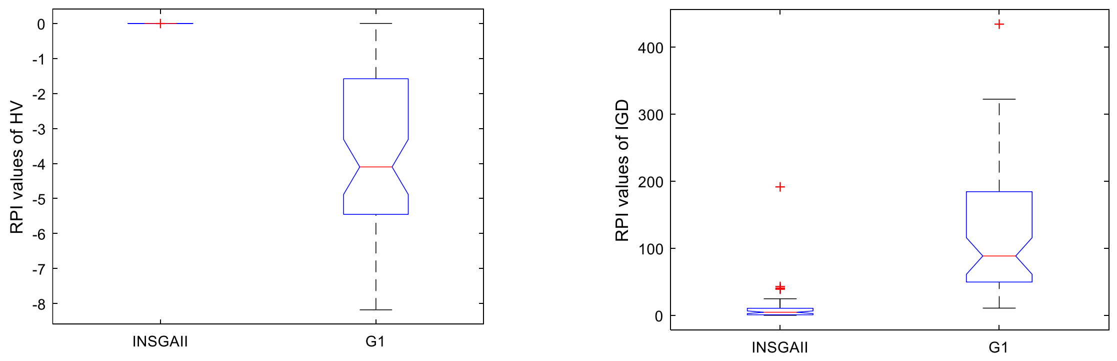
Table 5 Comparisons of the IGD values between INSGAII and G1

$n \times m$	IGD($g = 2$)		IGD($g = 3$)		IGD($g = 4$)		IGD($g = 5$)	
	INSGAII	G1	INSGAII	G1	INSGAII	G1	INSGAII	G1
20 × 4	**5.8176**	28.1926	**5.1728**	18.5917	**0.8602**	10.7065	**10.7807**	21.6783
20 × 8	11.6968	**11.5433**	40.5108	**31.2564**	**5.5459**	16.3955	12.1780	**11.1968**
20 × 16	**4.6023**	35.3588	**11.6726**	64.3980	**24.6725**	26.9317	**10.2060**	44.2591
40 × 4	**5.7964**	60.1374	**0.0000**	54.7883	**0.2925**	33.6016	**0.0000**	41.7198
40 × 8	**0.7945**	64.4567	**4.5307**	74.2285	**0.9089**	37.0931	**11.9167**	65.8171
40 × 16	**2.3528**	76.8651	**0.7798**	113.3179	**18.9093**	54.3116	**42.9864**	44.8782
60 × 4	**0.8241**	96.7012	**7.4167**	63.6580	**1.5060**	68.8097	**5.4764**	59.7686
60 × 8	**0.0000**	118.5663	**5.6344**	94.1960	**23.8459**	115.1965	**14.0692**	82.6480
60 × 16	**2.3617**	124.9135	**5.3598**	135.3208	**1.1165**	139.8077	**9.6808**	63.2765
80 × 4	**0.0000**	141.6163	**0.0000**	155.6765	**3.8399**	69.7350	**8.5067**	68.3357
80 × 8	**2.2220**	183.5328	**2.3344**	184.9940	**3.4195**	263.4318	**0.0000**	186.2944
80 × 16	**8.8220**	254.4020	**16.3215**	194.3021	**7.3209**	238.1768	**0.0000**	232.2292
100 × 4	**0.0000**	211.4340	**1.4399**	153.5090	**9.6292**	176.1901	**0.0000**	114.2430
100 × 8	**0.0000**	289.5759	**0.7433**	241.9929	**0.0000**	206.9498	**0.3443**	195.0756
100 × 16	**1.1567**	322.1684	191.4744	**170.0620**	**39.3091**	251.1446	**38.9450**	434.0927
Avg	**3.0965**	134.6309	**19.5594**	116.6861	**9.4118**	113.8988	**11.0060**	111.0342

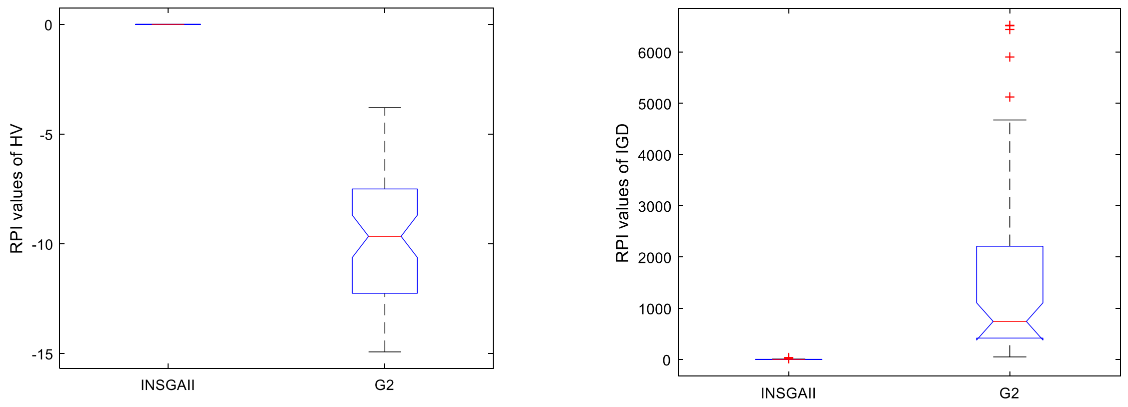
Values in bold mean the best ones

the results collected by INSGAII and G1 are shown in the following eight columns.

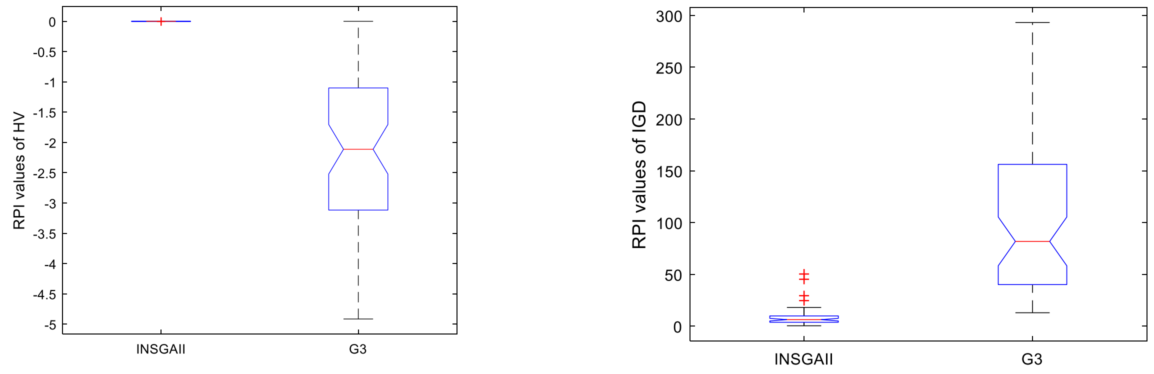
From the comparison results, the following can be observed: (1) considering the HV values, compared with G1, INSGAII obtains 59 better results, and the slightly worse results for only one instance; (2) for the IGD values, INSGAII obtains 56 better results, the rest one is slightly inferior in small-scale cases; and (3) from the average performance



(a) Comparison results of the initialization heuristics ENEH



(b) Comparison results of the local search strategy



(c) Comparison results of the crossover operators

Fig. 10 ANOVA comparison results

in HV and IGD given in the last line and the ANOVA results from Fig. 10a, it can be seen that INSGAII is significantly better than G1, which verify the efficiency of the proposed ENEH heuristics.

Effect of search strategies

To show the effectiveness of the local search strategy and crossover operators discussed in "Local Search" and "Global search", we conduct detailed comparisons of the algorithms with and without the proposed strategies. The algorithm without the local search strategy is represented by G2, the algorithm without the crossover operators is denoted as G3, and the INSGAII with all the components.

As illustrated by Tables 6 and 7, (1) IGD and HV values of INSGAII are significantly better than G2 in all instances; (2) the ANOVA results from Fig. 10b shows that the local search strategy is very effective in solving the EEDNWFSP; and (3) the results indicate that the proposed local search contributes to improve the diversity and convergence of solutions. This

Table 6 Comparisons of the HV values between INSGAII and G2

$n \times m$	HV($g = 2$)		HV($g = 3$)		HV($g = 4$)		HV($g = 5$)	
	INSGAII	G2	INSGAII	G2	INSGAII	G2	INSGAII	G2
20 × 4	**0.6777**	0.6451	**0.6637**	0.6311	**0.6713**	0.6458	**0.7699**	0.7281
20 × 8	**0.6451**	0.5966	**0.7213**	0.6757	**0.7023**	0.6667	**0.7271**	0.6894
20 × 16	**0.6079**	0.5677	**0.6686**	0.6257	**0.6831**	0.6471	**0.7245**	0.6855
40 × 4	**0.6635**	0.6104	**0.6688**	0.6183	**0.6510**	0.6055	**0.7205**	0.6637
40 × 8	**0.6343**	0.5641	**0.6392**	0.5743	**0.6539**	0.5997	**0.6510**	0.5846
40 × 16	**0.6016**	0.5221	**0.6095**	0.5377	**0.5904**	0.5342	**0.6936**	0.6383
60 × 4	**0.6808**	0.6278	**0.6527**	0.6043	**0.6617**	0.6040	**0.6690**	0.6132
60 × 8	**0.6308**	0.5540	**0.6463**	0.5808	**0.6258**	0.5522	**0.6353**	0.5712
60 × 16	**0.5865**	0.5097	**0.5924**	0.5131	**0.5975**	0.5215	**0.6683**	0.5964
80 × 4	**0.6529**	0.5889	**0.6645**	0.6013	**0.6674**	0.6184	**0.6590**	0.5998
80 × 8	**0.6177**	0.5427	**0.6336**	0.5613	**0.6416**	0.5674	**0.6405**	0.5747
80 × 16	**0.5873**	0.5073	**0.6140**	0.5270	**0.5969**	0.5153	**0.5878**	0.5158
100 × 4	**0.6475**	0.5991	**0.6497**	0.5904	**0.6619**	0.6087	**0.6651**	0.6061
100 × 8	**0.6314**	0.5524	**0.6491**	0.5651	**0.6332**	0.5523	**0.6308**	0.5535
100 × 16	**0.5848**	0.4975	**0.5825**	0.5008	**0.5793**	0.4972	**0.6653**	0.5730
Avg	**0.6300**	0.5657	**0.6437**	0.5805	**0.6411**	0.5824	**0.6738**	0.6129

Values in bold mean the best ones

Table 7 Comparisons of the IGD values between INSGAII and G2

$n \times m$	IGD($g = 2$)		IGD($g = 3$)		IGD($g = 4$)		IGD($g = 5$)	
	INSGAII	G2	INSGAII	G2	INSGAII	G2	INSGAII	G2
20 × 4	**13.9173**	562.0828	**5.9056**	679.5463	**3.8929**	48.6234	**7.2012**	692.0580
20 × 8	**1.8034**	171.5275	**3.0435**	994.7606	**1.3680**	155.1355	**2.9209**	171.1599
20 × 16	**1.1953**	579.4234	**0.0715**	426.1558	**0.4948**	286.2467	**4.2273**	368.9697
40 × 4	**4.8949**	173.7518	**1.8528**	164.6672	**0.6844**	217.5063	**3.3446**	265.9727
40 × 8	**0.0000**	748.2804	**0.0000**	625.5124	**0.6971**	474.5985	**0.0000**	735.3733
40 × 16	**0.0000**	2357.4690	**0.5243**	2063.0820	**0.0000**	1552.1410	**0.0000**	1289.8630
60 × 4	**2.0679**	284.6895	**0.1819**	235.5487	**1.0481**	284.0066	**0.1752**	406.9018
60 × 8	**0.0000**	1221.7430	**36.6981**	1253.6310	**0.1144**	1057.4070	**0.0000**	1164.8740
60 × 16	**0.0000**	3055.8730	**14.0706**	3275.6730	**0.0000**	3435.2940	**0.0000**	3025.9910
80 × 4	**0.0000**	489.2292	**0.8371**	426.4559	**2.8838**	292.5117	**0.0000**	572.1952
80 × 8	**0.0000**	1533.9490	**4.7919**	1502.6310	**0.0000**	1481.4350	**0.6971**	1868.7260
80 × 16	**0.0000**	4416.9660	**0.0000**	5125.1920	**0.0000**	4676.1170	**0.0000**	4396.7220
100 × 4	**3.8736**	457.7837	**0.0000**	455.5416	**14.9347**	664.3326	**6.7696**	542.6256
100 × 8	**0.0000**	1844.6790	**0.0000**	2471.5640	**0.0000**	2591.4540	**0.0000**	2001.6710
100 × 16	**0.9614**	6517.1050	**2.5021**	6442.5340	**0.0000**	5906.0340	**0.2764**	6523.7930
Avg	**1.9142**	1627.6368	**4.6986**	1742.8330	**1.7412**	1541.5229	**1.7075**	1601.7931

Values in bold mean the best ones

proves the effectiveness of the proposed local search, which helps in improving the performance of the algorithm. In the INSGAII, we take advantage of the knowledge specific to the problem, and then combine this knowledge with classical mutation operators to form a local search strategy. This local search based on knowledge can effectively guide the solution to Pareto optimal solution.

Tables 8 and 9 separately list the comparison results with G3 as follows: (1) considering the HV and IGD values, G3 also achieved several superior values, but the overall effect of INSGAII performs better; and (2) the ANOVA results from Fig. 10c shows that INSGAII obtained significantly better results, where p values of HV and IGD are both less than 0.05. This shows that the two crossover operators designed based on the knowledge of Pareto set are effective.

Comparisons of the efficient algorithms

To further verify the performance of the proposed algorithm compared with other efficient algorithms, the following three algorithms are selected, namely, NSGAII [31], ARMOEA [32], and hpaEA [33]. Each algorithm run 30 times independently on the same computer, and 15×4 instances are tested. According to the number of factories, the detailed results of the experimental comparison are shown in Tables 10, 11, 12 and 13.

Specifics of the table are as follows: the scales of the examples are presented in the first column, the second column presents the information of HV for all algorithms, and the last column is IGD value. It can be observed that: (1) for the given 60 instances, the proposed INSGAII algorithm obtains all the better indicators, which is significantly better than the other compared algorithms; and (2) the average values of the last line further verify the efficiency of the INSGAII, which prove that the solutions obtained by INSGAII have good convergence and distribution.

Figure 11 reports the Pareto results for a given scale instance (i.e., $M = 8$, $J = 20$) belonging to four factories. PF in the figure represents the near Pareto front obtained as described in "Performance indicators". Moreover, all the four compared algorithms are drawn with different marks. From the four sub-figures, the following can be concluded: (1) the results obtained by the proposed INSGAII algorithm have better dominance performance compared with other compared algorithms; (2) the population diversity of INSGAII is significantly better than the three compared algorithms; and (3) considering different scale instances, the proposed INSGAII is the best one to balance the abilities of diversity and convergence.

Comparative analysis

Through experimental comparison and analysis of the results, the following conclusions about the proposed algorithm are obtained:

(1) The use of ENEH heuristics in the initialization phase can produce high-quality individuals, thus effectively guiding the population to approach the optimal solutions.
(2) According to the distributed nature of the problem, the corresponding mutation operators are designed. Combined with the analysis of the characteristics of EEDNWFSP, the DST1 and DST2 heuristics are designed to greatly improve the local search ability of the algorithm.
(3) The Pareto-based crossover operators enhance the population diversity as well as quality.
(4) As the number of factories increases, the superiority of the INSGA-II becomes more prominent. Therefore, the proposed algorithm is efficient for solving EEDNWFSP.

Conclusion

This study is the first one to consider the multi-objective energy-efficient scheduling of the distributed permutation flow-shop with sequence-dependent setup time and no-wait constraints. Two objectives, including minimizing of both makespan and the TEC, are adopted simultaneously. In order to solve the problem, based on the canonical multi-objective algorithm NSGA-II, some effective search strategies are designed according to the characteristics of the problem. First, the initial population is generated randomly, and then two individuals are replaced with the solutions generated by ENEH and ENEH2. Then, the characteristics of the problem are analyzed, two lemmas are proposed, and two speed adjustment heuristics are further developed. Combined with four mutation operators, a local enhancement process is designed, which effectively enhances the convergence of the algorithm and improves the quality of solutions. Finally, using the knowledge of non-dominated solution set, two effective crossover operators are designed to improve the diversity of solutions.

INSGA-II is tested with multiple scale instances and compared with two state-of-the-art multi-objective evolutionary algorithms. The experimental results demonstrate the superiority of the proposed algorithm.

As future work, it is worth investigating other types of distributed scheduling problems, such as resource constraints, heterogeneous factories, and high dimensional multi-objective. We will focus on the extraction and utilization of problem-specific knowledge as well as the design

Table 8 Comparisons of the HV values between INSGAII and G3

$n \times m$	HV($g = 2$)		HV($g = 3$)		HV($g = 4$)		HV($g = 5$)	
	INSGAII	G3	INSGAII	G3	INSGAII	G3	INSGAII	G3
20 × 4	**0.6620**	0.6607	0.6660	**0.6690**	**0.7501**	0.7476	**0.6609**	0.6525
20 × 8	**0.6607**	0.6509	**0.6728**	0.6615	**0.6988**	0.6931	**0.7064**	0.6896
20 × 16	**0.5955**	0.5921	**0.6630**	0.6454	**0.6394**	0.6147	**0.7367**	0.7005
40 × 4	**0.6664**	0.6583	**0.6797**	0.6638	**0.6577**	0.6442	**0.6804**	0.6611
40 × 8	**0.6525**	0.6384	**0.6236**	0.6170	**0.7167**	0.7128	**0.6217**	0.6131
40 × 16	**0.6082**	0.6063	**0.6672**	0.6616	**0.6065**	0.5995	**0.6840**	0.6773
60 × 4	**0.6734**	0.6496	**0.6398**	0.6169	**0.6503**	0.6300	**0.6630**	0.6411
60 × 8	**0.6205**	0.6093	**0.6490**	0.6300	**0.6905**	0.6778	**0.6349**	0.6233
60 × 16	0.5945	**0.5964**	**0.5975**	0.5910	**0.6034**	0.5932	**0.6844**	0.6804
80 × 4	**0.6633**	0.6365	**0.6637**	0.6329	**0.6478**	0.6263	**0.6593**	0.6394
80 × 8	**0.6403**	0.6239	**0.6327**	0.6138	**0.6186**	0.6043	**0.6124**	0.5934
80 × 16	**0.5866**	0.5800	**0.6030**	0.5861	**0.5967**	0.5949	**0.5928**	0.5853
100 × 4	**0.6377**	0.6168	**0.6535**	0.6277	**0.6549**	0.6327	**0.6426**	0.6154
100 × 8	**0.6433**	0.6217	**0.6270**	0.6044	**0.6365**	0.6182	**0.6245**	0.6074
100 × 16	**0.6136**	0.6047	**0.6122**	0.5982	**0.5895**	0.5843	**0.5924**	0.5824
Avg	**0.6346**	0.6230	**0.6434**	0.6280	**0.6505**	0.6382	**0.6531**	0.6375

Values in bold mean the best ones

Table 9 Comparisons of the IGD values between INSGAII and G3

$n \times m$	IGD($g = 2$)		IGD($g = 3$)		IGD($g = 4$)		IGD($g = 5$)	
	INSGAII	G3	INSGAII	G3	INSGAII	G3	INSGAII	G3
20 × 4	**6.4253**	28.1913	**17.3912**	36.3181	**2.6949**	20.9009	**5.1159**	87.1628
20 × 8	**6.6052**	27.3634	**6.5460**	108.4567	**5.3313**	14.3616	**4.5036**	45.4794
20 × 16	**10.5063**	12.9577	**3.7925**	79.7895	**11.3341**	57.7334	**6.9817**	240.6732
40 × 4	**6.1780**	32.6651	**1.9207**	43.3112	**2.7813**	43.2867	**2.2286**	43.7553
40 × 8	**5.1923**	36.9973	**4.7158**	33.0645	**2.4525**	31.9487	**5.6525**	47.2015
40 × 16	**6.6654**	32.6637	**7.3149**	34.1539	**4.1873**	32.8176	**29.4323**	158.7961
60 × 4	**0.7880**	93.7319	**2.2722**	55.3519	**2.3249**	64.4927	**1.6483**	99.5842
60 × 8	**7.7062**	56.7305	**6.1223**	148.4391	**9.2609**	63.0451	**6.6078**	65.3631
60 × 16	**14.8379**	28.2828	**16.5978**	83.9112	**6.8725**	69.4887	**9.2879**	60.2524
80 × 4	**5.5407**	135.7429	**3.0102**	170.8620	**4.9320**	103.9318	**3.1661**	88.8715
80 × 8	**7.2935**	154.8194	**45.2692**	220.5064	**12.6506**	96.5460	**4.0060**	140.0035
80 × 16	**9.0627**	71.4499	**4.8200**	263.2319	**10.6508**	23.1394	**18.0773**	149.5012
100 × 4	**5.0054**	157.6446	**3.0357**	213.9454	**7.2274**	127.1013	**2.6644**	233.0220
100 × 8	**3.7206**	173.4658	**16.5590**	246.1586	**12.7700**	175.4973	**0.3867**	208.5464
100 × 16	**9.6454**	159.3736	**10.1612**	293.1734	**50.3596**	96.7395	**24.6897**	231.9344
Avg	**7.0115**	80.1387	**9.9686**	135.3782	**9.7220**	68.0687	**8.2966**	126.6765

Values in bold mean the best ones

Table 10 Results of the algorithms ($g = 2$)

$n \times m$	HV				IGD			
	ARMOEA	INSGAII	NSGAII	hpaEA	ARMOEA	INSGAII	NSGAII	hpaEA
20×4	0.6350	**0.6689**	0.6363	0.6391	862.9526	**5.8334**	933.7459	961.2891
20×8	0.6152	**0.6592**	0.6222	0.6135	253.9407	**15.2828**	203.3305	240.2489
20×16	0.5488	**0.5976**	0.5518	0.5486	682.7621	**0.0000**	653.4532	711.6409
40×4	0.6015	**0.6433**	0.6007	0.5963	237.9978	**2.9417**	194.6513	200.4242
40×8	0.5552	**0.6238**	0.5566	0.5537	774.0414	**0.0000**	718.1670	767.6475
40×16	0.4859	**0.5699**	0.4901	0.4831	2505.5410	**0.0000**	2315.0910	2436.4890
60×4	0.6172	**0.6556**	0.6208	0.6091	473.9908	**10.8207**	406.4291	484.0508
60×8	0.5570	**0.6208**	0.5626	0.5571	1268.4460	**21.2009**	1226.8120	1303.0030
60×16	0.4762	**0.5687**	0.4755	0.4691	3524.6920	**0.0000**	3707.8340	3711.8760
80×4	0.5811	**0.6349**	0.5792	0.5767	616.9777	**6.3949**	607.9967	635.6730
80×8	0.4936	**0.5700**	0.4935	0.4879	2162.4690	**0.0000**	2115.2650	2118.2260
80×16	0.4562	**0.5594**	0.4647	0.4536	5984.4640	**0.0000**	5662.2860	5976.3330
100×4	0.5875	**0.6421**	0.5888	0.5817	961.6876	**57.5231**	878.0886	949.1375
100×8	0.5017	**0.5772**	0.5041	0.5014	3107.6710	**0.0000**	2905.0270	2912.3990
100×16	0.4301	**0.5488**	0.4323	0.4312	8563.7730	**0.0000**	8385.6190	8366.9910
Avg	0.5428	**0.6093**	0.5453	0.5401	2132.0938	**7.9998**	2060.9198	2118.3619

Values in bold mean the best ones

Table 11 Results of the algorithms ($g = 3$)

$n \times m$	HV				IGD			
	ARMOEA	INSGAII	NSGAII	hpaEA	ARMOEA	INSGAII	NSGAII	hpaEA
20×4	0.6349	**0.6753**	0.6350	0.6336	491.6849	**0.3865**	581.3584	580.5010
20×8	0.6482	**0.6984**	0.6547	0.6433	1379.3790	**0.0000**	1208.8590	1638.3280
20×16	0.6419	**0.7071**	0.6438	0.6387	692.1126	**0.0000**	636.8500	672.9804
40×4	0.6216	**0.6710**	0.6279	0.6193	334.1531	**5.0042**	273.1995	321.7073
40×8	0.5474	**0.6225**	0.5548	0.5448	935.7414	**0.0000**	776.2829	929.9833
40×16	0.5505	**0.6314**	0.5581	0.5503	2364.7160	**0.0000**	2121.2880	2402.1510
60×4	0.5979	**0.6471**	0.6037	0.5959	368.6063	**35.9517**	316.0852	365.4021
60×8	0.5290	**0.6078**	0.5302	0.5296	1569.6950	**0.0000**	1423.7610	1492.2550
60×16	0.4623	**0.5480**	0.4594	0.4583	4366.1770	**0.0000**	4169.7940	4292.8760
80×4	0.5863	**0.6365**	0.5849	0.5817	649.3023	**23.5702**	586.5835	606.5393
80×8	0.5241	**0.6088**	0.5230	0.5207	2309.5610	**0.0000**	2339.2460	2378.0090
80×16	0.4583	**0.5576**	0.4688	0.4535	6518.8710	**0.0000**	5962.8210	6432.5830
100×4	0.5666	**0.6119**	0.5629	0.5588	742.3209	**33.9751**	742.6668	716.7839
100×8	0.4860	**0.5749**	0.4872	0.4817	2988.5700	**0.0000**	2922.4550	2999.4420
100×16	0.4334	**0.5337**	0.4336	0.4308	7699.0730	**0.0000**	7717.3670	7835.0550
Avg	0.5525	**0.6221**	0.5552	0.5494	2227.3309	**6.5925**	2118.5745	2244.3064

Values in bold mean the best ones

Table 12 Results of the algorithms ($g = 4$)

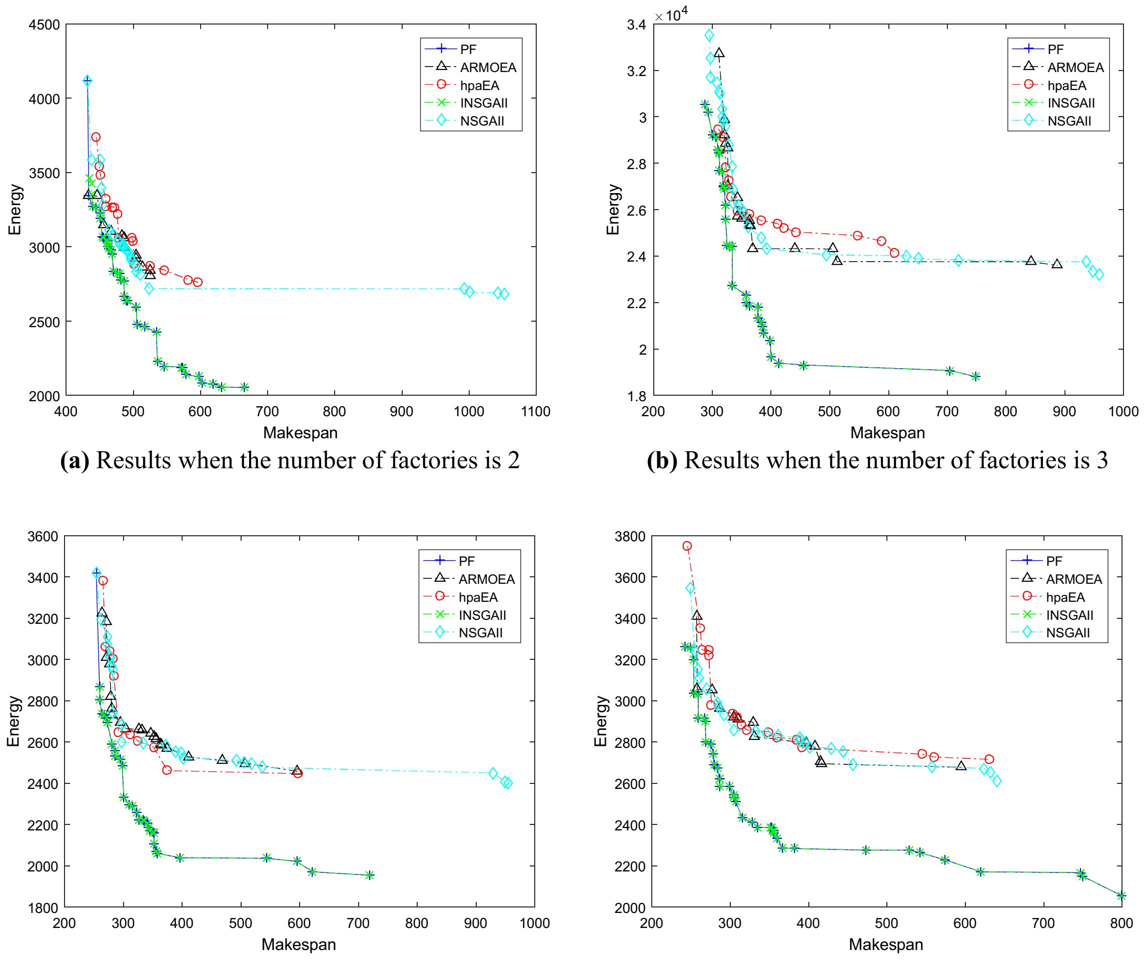
$n \times m$	HV				IGD			
	ARMOEA	INSGAII	NSGAII	hpaEA	ARMOEA	INSGAII	NSGAII	hpaEA
20 × 4	0.6213	**0.6698**	0.6303	0.6263	90.9816	**0.4818**	76.3124	86.5997
20 × 8	0.6753	**0.7178**	0.6812	0.6762	250.4894	**5.2796**	245.4039	215.3780
20 × 16	0.6643	**0.7001**	0.6634	0.6701	337.8108	**0.8978**	383.6756	321.1888
40 × 4	0.6171	**0.6624**	0.6251	0.6181	284.9572	**8.9542**	237.7833	253.4150
40 × 8	0.5050	**0.5778**	0.5111	0.5099	673.1945	**0.0000**	603.5707	713.5393
40 × 16	0.5576	**0.6253**	0.5648	0.5546	1982.4460	**0.0000**	1746.0310	2065.0820
60 × 4	0.6027	**0.6559**	0.6051	0.5992	384.5750	**7.7861**	353.4561	369.1107
60 × 8	0.5216	**0.6039**	0.5221	0.5196	1407.2100	**0.0000**	1385.7210	1431.3560
60 × 16	0.4875	**0.5772**	0.4894	0.4853	3828.7470	**0.0000**	3692.5760	3699.8410
80 × 4	0.5558	**0.6169**	0.5608	0.5557	627.7588	**0.0000**	590.8560	608.5870
80 × 8	0.5493	**0.6412**	0.5539	0.5451	2124.3480	**0.0000**	1900.1150	2127.7020
80 × 16	0.4634	**0.5620**	0.4699	0.4714	5865.7030	**0.0000**	5516.3790	5682.8610
100 × 4	0.5728	**0.6238**	0.5824	0.5714	793.5963	**56.5530**	615.9821	752.8644
100 × 8	0.4828	**0.5722**	0.4888	0.4815	3419.9320	**0.0000**	2894.0410	3130.0580
100 × 16	0.5399	**0.6453**	0.5436	0.5409	7578.3830	**0.0000**	7284.7820	7543.3520
Avg	0.5611	**0.6301**	0.5661	0.5617	1976.6755	**5.3302**	1835.1123	1933.3957

Values in bold mean the best ones

Table 13 Results of the algorithms ($g = 5$)

$n \times m$	HV				IGD			
	ARMOEA	INSGAII	NSGAII	hpaEA	ARMOEA	INSGAII	NSGAII	hpaEA
20 × 4	0.6164	**0.6491**	0.6084	0.5989	355.3706	**0.0000**	358.3120	548.0381
20 × 8	0.6574	**0.7129**	0.6645	0.6567	243.9721	**0.0000**	238.8503	281.5824
20 × 16	0.6683	**0.7296**	0.6732	0.6669	539.7077	**0.0000**	477.0620	550.6626
40 × 4	0.5941	**0.6512**	0.6011	0.5907	231.5507	**0.0000**	221.3578	233.3693
40 × 8	0.5350	**0.6109**	0.5385	0.5341	804.0522	**0.0000**	780.4658	729.1339
40 × 16	0.6112	**0.6704**	0.6154	0.6056	1798.5440	**0.0000**	1641.4210	1831.1760
60 × 4	0.6754	**0.7408**	0.6807	0.6798	617.5125	**10.5511**	548.1857	586.7113
60 × 8	0.6127	**0.6983**	0.6215	0.6140	1573.8320	**0.0000**	1321.1470	1593.4640
60 × 16	0.4920	**0.5676**	0.4939	0.4878	3084.0470	**0.0000**	3097.6250	3118.9630
80 × 4	0.6184	**0.6920**	0.6173	0.6113	692.2176	**0.0000**	755.0597	811.5215
80 × 8	0.5232	**0.6097**	0.5277	0.5237	2187.8330	**0.0000**	1889.8940	2120.5530
80 × 16	0.5511	**0.6568**	0.5570	0.5552	6236.6680	**0.0000**	5908.1990	5857.7450
100 × 4	0.5359	**0.6126**	0.5394	0.5388	955.2761	**0.0000**	814.5295	894.9756
100 × 8	0.4851	**0.5730**	0.4946	0.4841	2835.1090	**0.0000**	2469.9140	2599.6000
100 × 16	0.5475	**0.6521**	0.5537	0.5442	8317.0510	**0.0000**	7840.9230	8467.2710
Avg	0.5816	**0.6551**	0.5858	0.5794	2031.5162	**0.7034**	1890.8631	2014.9844

Values in bold mean the best ones



(a) Results when the number of factories is 2

(b) Results when the number of factories is 3

(c) Results when the number of factories is 4

(d) Results when the number of factories is 5

Fig. 11 Pareto front of the compared algorithms

of effective search operators. More and more realistic constraints will be considered, such as fuzzy processing times [50]. In addition, it is extremely important to apply these algorithms to real industrial problems.

Funding This work was supported in part by the National Science Foundation of China under Grant 62173216.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Wang B, Han K, Spoerre J et al (1997) Integrated product, process and enterprise design: why, what and how? Springer
2. Wang G, Li X, Gao L et al (2021) Energy-efficient distributed heterogeneous welding flow shop scheduling problem using a modified MOEA/D. Swarm Evol Comput 62(3):100858
3. Wu X, Che A, Lev B (2020) Energy-efficient no-wait permutation flow shop scheduling by adaptive multi-objective variable neighborhood search. Omega 94:102117
4. Grabowski J, Pempera J (2005) Some local search algorithms for no-wait flow-shop problem with makespan criterion. Comput Oper Res 32:2197–2212
5. Ruiz R, Naderi B (2010) The distributed permutation flowshop scheduling problem. Comput Oper Res 37:754–768

6. Huang JP, Pan QK, Gao L (2020) An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. Swarm Evol Comput 59:100742
7. Wang JJ, Wang L (2018) A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop. IEEE Trans Syst Man Cybern Syst 50(5):1805–1819
8. Wang GC, Gao L, Li XY et al (2020) Energy-efficient distributed permutation flow shop scheduling problem using a multi-objective whale swarm algorithm. Swarm Evol Comput 57:100716
9. Fernandez-Viagas V, Perez-Gonzalez P, Framinan JM (2018) The distributed permutation flow shop to minimise the total flowtime. Comput Ind Eng 118:464–477
10. Lu C, Gao L, Gong W et al (2021) Sustainable scheduling of distributed permutation flow-shop with non-identical factory using a knowledge-based multi-objective memetic optimization algorithm. Swarm Evol Comput 60:100803
11. Wang G, Li X, Gao L et al (2019) A multi-objective whale swarm algorithm for energy-efficient distributed permutation flow shop scheduling problem with sequence dependent setup times. IFAC-PapersOnLine 52(13):235–240
12. Li Y, Li X, Gao L et al (2020) An improved artificial bee colony algorithm for distributed heterogeneous hybrid flowshop scheduling problem with sequence-dependent setup times. Comput Ind Eng 2020:106638
13. Li JQ, Song MX, Wang L et al (2020) Hybrid artificial bee colony algorithm for a parallel batching distributed flow-shop problem with deteriorating jobs. IEEE Trans Cybern 50(6):2425–2439
14. Shao W, Shao Z, Pi D (2020) Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem. Knowl-Based Syst 194:105527
15. Cai J, Zhou R, Lei D (2020) Dynamic shuffled frog-leaping algorithm for distributed hybrid flow shop scheduling with multiprocessor tasks. Eng Appl Artif Intell 90:103540
16. Zhang ZQ, Qian B, Hu R et al (2021) A matrix-cube-based estimation of distribution algorithm for the distributed assembly permutation flow-shop scheduling problem. Swarm Evol Comput 60:100785
17. Huang YY, Pan QK, Huang JP et al (2020) An improved iterated greedy algorithm for the distributed assembly permutation flowshop scheduling problem. Comput Ind Eng 152(3):107021
18. Shao Z, Shao W, Pi D (2020) Effective constructive heuristic and metaheuristic for the distributed assembly blocking flow-shop scheduling problem. Appl Intell 50(1):4647–4669
19. Lin SW, Ying KC (2016) Minimizing makespan for solving the distributed no-wait flowshop scheduling problem. Comput Ind Eng 99:202–209
20. Shao W, Pi D, Shao Z (2019) A pareto-based estimation of distribution algorithm for solving multiobjective distributed no-wait flow-shop scheduling problem with sequence-dependent setup time. IEEE Trans Autom Sci Eng 2019:1–17
21. Komaki M, Malakooti B (2017) General variable neighborhood search algorithm to minimize makespan of the distributed no-wait flow shop scheduling problem. Prod Eng 11(3):315–329
22. Shao W, Pi D, Shao Z (2017) Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated greedy algorithms. Knowl-Based Syst 137:163–181
23. Li H, Li X, Gao L (2021) A discrete artificial bee colony algorithm for the distributed heterogeneous no-wait flowshop scheduling problem. Appl Soft Comput 100:106946
24. Behjat S, Salmasi N (2017) Total completion time minimisation of no-wait flowshop group scheduling problem with sequence dependent setup times. Eur J Indust Eng 11(1):22
25. 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(3):1143–1159
26. Ciavotta M, Minella G, Ruiz R (2013) Multi-objective sequence dependent setup times permutation flowshop: a new algorithm and a comprehensive study. Eur J Oper Res 227(2):301–313
27. Wu X, Sun Y (2018) A green scheduling algorithm for flexible job shop with energy-saving measures. J Clean Prod 172:3249–3264
28. Du Y, Li JQ, Chen XL, Duan PY, Pan QK (2022) A knowledge-based reinforcement learning and estimation of distribution algorithm for flexible job shop scheduling problem. IEEE Trans Emerg Topics Comput Intell. https://doi.org/10.1109/TETCI.2022.3145706
29. Li JQ, Du Y, Gao KZ, Duan PY et al (2022) A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem. IEEE Trans Autom Sci Eng 19(3):2153–2170
30. Qi R, Li JQ, Wang J, Jin H, Han YYQMOEA (2022) A Q-learning-based multiobjective evolutionary algorithm for solving time-dependent green vehicle routing problems with time windows. Inf Sci 608:178–201
31. Jiang ED, Wang L, Peng ZP (2020) Solving energy-efficient distributed job shop scheduling via multi-objective evolutionary algorithm with decomposition. Swarm Evol Comput 2020:100745
32. Li JQ, Sang HY, Han YY et al (2018) Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions. J Clean Prod 181:584–598
33. Li JQ, Chen XL, Duan PY, Mou JH (2021) KMOEA: a knowledge-based multi-objective algorithm for distributed hybrid flow shop in a prefabricated system. IEEE Trans Industr Inf. https://doi.org/10.1109/TII.2021.3128405
34. Chen JF, Wang L, Peng ZP (2019) A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. Swarm Evol Comput 50:100557
35. Tian Y, Cheng R, Zhang XY et al (2018) An indicator-based multiobjective evolutionary algorithm with reference point adaptation for better versatility. IEEE Trans Evol Comput 22(4):609–622
36. Chen H, Tian Y, Pedrycz W et al (2019) Hyperplane assisted evolutionary algorithm for many-objective optimization problems. IEEE Trans Cybern 2019:1–14
37. Ding JY, Song S, Wu C (2015) Carbon-efficient scheduling of flow shops by multi-objective optimization. Eur J Oper Res 000:1–14
38. Deb K, Pratap A, Agarwal S et al (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
39. Xiong F, Chu M, Li Z et al (2021) Just-in-time scheduling for a distributed concrete precast flow shop system. Comput Oper Res 129:105204
40. Muhammad N, Emory E, Enscore et al (1983) A heuristic algorithm for the m-machine, *n*-job flow-shop sequencing problem. Omega 11(1):91–95
41. Fernandez-Viagas V, Ruiz R, Framinan JM (2017) A new vision of approximate methods for the permutation flowshop to minimise makespan: state-of-the-art and computational evaluation. Eur J Oper Res 257(3):707–721
42. Han Y, Gong D, Jin Y et al (2019) Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns. IEEE Trans Cybern 49(1):184–197
43. Ruiz R, Maroto C, Alcaraz J (2006) Two new robust genetic algorithms for the flowshop scheduling problem. Omega 34(5):461–476
44. Yi Z, Li X, Qian W (2009) Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization. Eur J Oper Res 196(3):869–876
45. Michalewic Z (1996) Genetic algorithms + data structures = evolution programs. Springer, Berlin. https://doi.org/10.1007/978-3-662-03315-9
46. Deng J, Wang L, Wu C et al (2016) A competitive memetic algorithm for carbon-efficient scheduling of distributed flow-shop. Int

Conf Intell Comput 9771:476–488 (**Springer International Publishing**)

47. Ruiz R, Maroto C, Alcaraz J (2005) Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. Eur J Oper Res 165(1):34–54
48. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans Evol Comput 3(4):257–271
49. Coello CAC, Cortes NC (2005) Solving multiobjective optimization problems using an artificial immune system. Genet Program Evolvable Mach 6(2):163–190
50. Li JQ, Liu ZM, Li CD, Zheng ZX (2021) Improved artificial immune system algorithm for Type-2 fuzzy flexible job shop scheduling problem. IEEE Trans Fuzzy Syst 29(11):3234–3248