Swarm and Evolutionary Computation 62 (2021) 100861

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations

Yu Du [a], Jun-qing Li [a,b,*], Chao Luo [a], Lei-lei Meng [b]

[a] School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China
[b] School of Computer Science and Technology, Liaocheng University, Liaocheng 252059, China

ARTICLE INFO

Keywords:
Distributed flexible job shop scheduling
Crane transportation
Estimation of distribution algorithm
Variable neighborhood search
Multi-objective optimization

ABSTRACT

Distributed flexible job shop scheduling has attracted research interest due to the development of global manufacturing. However, constraints including crane transportation and energy consumption should be considered with the realistic requirements. To address this issue, first, we modeled the problem by utilizing an integer programming method, wherein the makespan and energy consumptions during the machine process and crane transportation are optimized simultaneously. Afterward, a hybrid algorithm consisting of estimation of distribution algorithm (EDA) and variable neighborhood search (VNS) was proposed to solve the problem, where an identification rule of four crane conditions was designed to make fitness calculation feasible. In EDA component, the parameters in probability matrices are set to be self-adaptive for stable convergence to obtain better output. Moreover, a probability mechanism was applied to control the activity of the EDA component. In VNS component, five problem-specific neighborhood structures including global and local strategies are employed to enhance exploitation ability. The simulation tests results confirmed that the proposed hybrid EDA-VNS algorithm can solve the considered problem with high efficiency compared with other competitive algorithms, and the proposed improving strategies are verified to have significance in better performance.

1. Introduction

With the development of global enterprises, single-factory manufacturing has been gradually replaced by a multi-factory manufacturing structure [1]. In the distributed flexible job shop problem (DFJSP), each factory has a flexible job shop scheduling system. In the manufacturing systems, maximum completion time (makespan) is an important objective to be considered, and the total energy consumption (TEC) is also important in green manufacturing. Various manufacturing systems, including steelmaking system, textile production process [2], equipment manufacturing [3], chemical materials processing, and automobile assembly, can be modeled as the DFJSP. In this study, crane transportation is included in the DFJSP, where each factory has a crane for transporting jobs between machines. Each crane has four different conditions based on the relationships between the job position, same-job predecessor position, and crane position. In this study, we propose a hybrid algorithm to solve the DFJSP with crane transportations (DFJSPC).

In the DFJSPC, three subproblems must be solved. First, all jobs should be distributed to different factories. Then, in each factory, the order of operations should be verified, and the assigned machine for each operation should be confirmed. The flexible job shop scheduling problem is NP-hard [4]; therefore, the DFJSPC is also NP-hard.

In this study, we propose a hybrid algorithm consisting of the estimation of distribution algorithm (EDA) and variable neighborhood search (VNS) to solve the DFJSPC. In the proposed hybrid algorithm, each solution of the DFJSPC is managed as an individual in a population. In the EDA component, superior individuals are selected according to their fitness in the population to generate new individuals and replace the same number of inferior individuals in the population. The new individuals are generated based on three probability matrices and inherit superior characteristics from superior individuals.

Premature convergence easily occurs in the EDA component during the iteration. Therefore, we set the parameters to be self-adaptive in the update probability matrices mechanism so that the model converges at a stable speed for improved output. In the proposed algorithm, the EDA component has strong exploration ability. To improve exploitation ability, VNS component is proposed, where five different neighborhood structures, including local and global heuristics, are proposed and organized. In each iteration of the proposed algorithm, the EDA component is used to explore larger space to identify optimal solutions, while the VNS component is used to obtain refined solutions in the neighborhood environments.

At each iteration of the proposed algorithm, the EDA and VNS components should operate simultaneously during the calculation process.

* Corresponding author at: School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China
E-mail address: lijunqing@lcu-cs.com (J.-q. Li).

https://doi.org/10.1016/j.swevo.2021.100861
Received 28 June 2020; Received in revised form 8 February 2021; Accepted 10 February 2021
Available online 16 February 2021
2210-6502/© 2021 Elsevier B.V. All rights reserved.

However, EDA is effective at the beginning of convergence and generally does not generate better solutions at later stages of convergence. In addition, the EDA component requires considerable calculation time during the iteration. Therefore, we propose a probability variable for controlling the activity of the EDA component at each iteration. If none of the probability matrices are changed in the previous iteration, the EDA component is deactivated.

The contributions of this study are as follows:

A hybrid algorithm of the EDA and VNS is proposed to solve the DFJSPC, where the makespan and TEC are the objectives.

To make the fitness calculation feasible, a well-designed identification rule is developed for four crane transportation conditions in each factory.

A constraint programming model of the DFJSPC is proposed.

In the EDA component, we set the parameters to be self-adaptive to have the solutions converge at a stable speed in the exploration function. In addition, a probability variable is considered to control the EDA component in the iteration to reduce computing resources.

In the VNS component, five neighborhood structures in the DFJSPC are organized to search for better solutions as the exploitation function.

The remainder of this paper is organized as follows. Section 2 presents a review of the research and applications of the distributed permutation flowshop scheduling problem (DPFSP), DFJSPC, and scheduling problems with transportation constraints. Section 3 provides the description and detailed decoding process of the DFJSPC. Section 4 describes the EDA-VNS algorithm, while Section 5 gives experimental analysis of the proposed algorithm and comparison experiments. Finally, conclusions are drawn in Section 6.

2. Literature review

2.1. Distributed scheduling problem

DPFSP is the most common type in distributed scheduling problems. Naderi and Ruiz proposed six different mixed integer linear programming (MILP) models, heuristic methods, and variable neighborhood descent methods [5] to solve the DPFSP. Wang et al. developed an EDA [6] to solve the DPFSP, and designed local search operators to enhance the local exploitation. Furthermore, Wang et al. proposed a fuzzy logic-based adaptive evolution strategy [7] to preserve the population diversity in the DPFSP. Deng and Wang designed a competitive memetic algorithm [8] for the multi-objective DPFSP, and knowledge-based local search operators were proposed to enhance its exploitation ability. Li et al. proposed an improved artificial bee colony (ABC) algorithm and a distributed iterated greedy (IG) heuristic [9] for the distributed flow shop scheduling problem. Meng et al. developed a variable neighborhood descent algorithm, an ABC algorithm, and an IG [10] for the DPFSP with customer order constraints. Ruiz et al. improved initialization, construction, and destruction procedures in IG [11] for the DPFSP. In addition, Wang and Wang proposed a knowledge-based cooperative algorithm [12] for the energy-efficient DPFSP. Li et al. designed a hybrid ABC algorithm [13] to improve the makespan for solving the DPFSP with deteriorating jobs. In the aforementioned studies, effective heuristics and strategies are proposed to improve performance; however, the main objective is the makespan, additional objectives should be considered as well. For other types of DPFSPs, Rifai et al. proposed an adaptive large neighborhood search [14] for the distributed reentrant permutation flow shop scheduling problem, while Shao et al. designed a Pareto-based EDA [15] for the distributed no-wait flow shop scheduling problem.

In addition, the distributed assembly permutation flowshop scheduling problem (DAPFSP) has been extensively researched. Wang and Wang designed an EDA-based memetic algorithm [16] to solve the DAPFSP, where EDA-based exploration and local-search-based exploitation were incorporated into the memetic algorithm framework. Pan et al. proposed a MILP model, three constructive heuristics, two VNS search methods, and an IG [17] for the DAPFSP, and in Ref [18], three novel heuristics, four effective metaheuristics, and problem-specific knowledge and accelerations were proposed to minimize the total flowtime of the DPFSP. Ferone et al. developed a biased-randomized iterated local search metaheuristic [19] for the DAPFSP that had fewer parameter. Sang et al. proposed invasive weed optimization algorithms [20] for the DAPFSP. The DAPFSP is closer to reality than the DPFSP model; however, the makespan is not the only factor that should be optimized in practice, and models should consider additional objectives.

Moreover, numerous recent published literatures have researched on distributed shop scheduling problems. Pan et al. proposed an effective cooperative co-evolutionary algorithm [21] for distributed flowshop group scheduling problems. Huang et al. established an effective IG [22] for DPFSP with sequence-dependent setup times. Jing et al. investigated an IG [23] for PDFSP with due windows. Lu et al. designed a knowledge-based multi-objective memetic optimization algorithm [24] for DPFSP with non-identical factory. Meng and Pan considered a heterogeneous DPFSP with lot-streaming and carryover sequence-dependent setup-time [25]. Huang et al. applied a discrete bee colony [26] and effective constructive heuristics for distributed flowshop with setup times. Nevertheless, most above researches should consider the transportation of jobs.

Other types of distributed scheduling problems have also been developed. Hsu et al. proposed an agent-based fuzzy constraint-directed negotiation mechanism [27] for the distributed job shop scheduling problem (DJSP). In addition, Zhang et al. developed a distributed ant colony system [28] to solve the flexible assembly job shop scheduling problem with multiple objectives. Zheng et al. proposed a cooperative co-evolution algorithm [29] for the multi-objective fuzzy distributed hybrid flow shop scheduling problem (DHFSP), where an EDA-mode search and IG-mode search were designed as a cooperation scheme for mode switching based on information entropy and the diversity of elite solutions. Shao et al. proposed a multi-neighborhood IG [30] for the DHFSP. However, energy factors should also be taken into account to satisfy the demands of green manufacturing.

2.2. Distributed flexible job shop scheduling problem

The DFJSP is another type of distributed scheduling problem, where each factory has a flexible job shop scheduling system. In studies on the DFJSP, the main objective is the makespan. De Giovanni and Pezzella proposed an improved genetic algorithm (GA) [31] for the DFJSP. Ziaee developed a fast heuristic algorithm [32] based on a constructive procedure to solve the DFJSP. Then, Lu et al. proposed a GA with a new and concise chromosome representation [33] for the DFJSP, modeling a three-dimensional scheduling solution using a one-dimensional scheme. Liu et al. proposed a refined GA [34] that integrates probability into the encoding method, reducing the length of chromosomes and saving computational space; and Liu et al. designed a GA [35] with an encoding operator to solve the DFJSP. Chang and Liu proposed a hybrid GA [36] to solve the DFJSP, where an encoding mechanism was used to handle invalid job assignment. Li et al. proposed a hybrid Pareto-based tabu search (TS) algorithm [2] to solve the DFJSP considering four objectives simultaneously. Wu et al. proposed a MILP model and an improved differential evolution algorithm [37] to solve the DFJSP with the assembly process. Additionally, Marzouki et al. proposed a chemical reaction optimization metaheuristic [38] in solving the DFJSP to minimize the makespan. Meng et al. proposed four MILP models and a constraint programming (CP) model [39] for the DFJSP. Recently, Luo et al. have proposed an efficient memetic algorithm (EMA) [40] to solve the DFJSP with transfers. Zhang et al. proposed a three-stage approach based on decomposition [41], providing the decomposition optimization theory of this study. Zou et al. established an effective IG [42] for solving a scheduling problem in a matrix manufacturing workshop. However, most of the researches did not consider the energy consumption and transportation process in the DFJSP. EDA is a newly developed statistics-

based optimization algorithm in evolutionary algorithms family. Compared with GA that generates new solutions with crossover and mutation operators, EDA sampled the superior area in search space to generate new superior solutions, making the population evolves based on the quality of solutions. Therefore, EDA has a stronger ability in global search and faster convergence. In this study, EDA is employed as the exploration component to solve DFJSP with crane transportation.

2.3. Scheduling problem with crane transportation

Crane and robot transportation have been considered constraints in the distributed scheduling problem. In Ref [39], the transportation time is included in the model and is added to the last operation of each job. In fact, additional transport constraints have been applied to other types of scheduling problems. Liu et al. [3] proposed a novel integrated green scheduling problem of flexible job shop and crane transportation, dividing crane transportation into four different conditions. In this study, we refer to these conditions as the crane transportation conditions in each factory. Dai et al. developed an improved GA [43] to solve an energy-efficient flexible job shop scheduling problem with transportation constraints. Ham proposed a CP model [44] for the flexible job shop scheduling problem with transfer robots, where the objective is the makespan. Li et al. proposed a hybrid IG algorithm [45] for flexible job shop problem with crane transportation, where lift operation is involved. And Li et al. designed an improved Jaya algorithm [46] for flexible job shop scheduling problem with transportation and set-up times constraints. An efficient multi-objective algorithm for the lot-streaming hybrid flowshop with variable sub-lots is proposed by Li et al. [47]. Moreover, Tao et al. designed a discrete imperialist competitive algorithm (ICA) [48] for hybrid flowshop problem with energy consumption. Li et al. designed efficient meta-heuristics for the FJSP with Type-2 fuzzy processing times [49], and vehicle routing problem with high performance [50].

Crane and robot transportation constraints are an active topic in the flexible job shop scheduling problem and job shop scheduling problem; however, the distributed scheduling problem with crane or robot transportation constraints has not attracted much attention thus far. From the above studies, it is evident that numerous algorithms have been applied in the scheduling problem for different environments. However, only a few studies considered crane transportation in DFJSP. Notably, considering the crane transportation, realistic constraints and factors need to be considered, such as energy consumptions of the machine processes and crane operations. Accordingly, in this study, an algorithm consisting of the EDA and VNS is investigated to solve the DFJSPC, where the EDA and VNS components are used for exploration and exploitation, respectively.

Details of the studies on the distributed scheduling problems are listed in Table 1.

3. Problem description

In the DFJSPC environment, I jobs must be distributed to F factories. For each job, one or more operations are performed one after another according to a given sequence, and all operations of a job must be performed in the selected factory. The execution of $O_{i,j}$ requires one machine out of a set of machines. Each factory has one crane to transport jobs in the operation process among various machines. In addition to the time and energy consumption of the machine process, the time and energy consumption of crane transportations are also considered. The positions of machines are provided as input of the DFJSPC.

3.1. Notations

Indices:

- f: index number of factories
- i: index number of jobs
- j: index number of operations
- k: index number of machines in all factories
- F: number of factories
- I: number of jobs
- J: number of operations
- K: number of machines in all factories

Parameters:

- $O_{i,j}$: jth operation of job i
- k': operating machine of predecessor operation
- $k(O_{i,j})$: assigned machine number of operation $O_{i,j}$, equaling 0 when k $\neq \delta_i$
- $Ts(O_{i,j})$: start time of $O_{i,j}$
- $Tc(O_{i,j})$: completion time of $O_{i,j}$
- $P(\pi_{1,f})$: initial position of the crane in factory f, equaling a machine number
- $\delta_{1,f}$: assigned machine position of the first performed operation in factory f
- x_{op}: x-coordinate position of operation position
- x_{cp}: x-coordinate position of crane position
- x_{pp}: x-coordinate position of same-job predecessor position
- y_{op}: y-coordinate position of operation position
- y_{cp}: y-coordinate position of crane position
- y_{pp}: y-coordinate position of same-job predecessor position
- a_{xs}: x-direction full load start acceleration of crane
- a_{xb}: x-direction full load braking acceleration of crane
- a_{ys}: y-direction full load start acceleration of crane
- a_{yb}: y-direction full load braking acceleration of crane
- Vx: operating speed of crane-gantry
- Vy: operating speed of crane-trolley
- P_{xs}: start power of crane-gantry
- P_{ys}: start power of crane-trolley
- P_{xd}: rated power of crane-gantry
- P_{yd}: rated power of crane-trolley
- Ps: standby power of the crane
- $T_{xnos,f}(O_{i,j})$: in factory f, x-direction start time of no-load operation in $O_{i,j}$
- $T_{xnob,f}(O_{i,j})$: in factory f, x-direction braking time of no-load operation in $O_{i,j}$
- $T_{xnod,f}(O_{i,j})$: in factory f, x-direction duration time of no-load operation in $O_{i,j}$
- $T_{xlos,f}(O_{i,j})$: in factory f, x-direction start time of load operation in $O_{i,j}$
- $T_{xlob,f}(O_{i,j})$: in factory f, x-direction braking time of load operation in $O_{i,j}$
- $T_{xlod,f}(O_{i,j})$: in factory f, x-direction duration time of load operation in $O_{i,j}$
- $T_{ynos,f}(O_{i,j})$: in factory f, y-direction start time of no-load operation in $O_{i,j}$
- $T_{ynob,f}(O_{i,j})$: in factory f, y-direction braking time of no-load operation in $O_{i,j}$
- $T_{ynod,f}(O_{i,j})$: in factory f, y-direction duration time of no-load operation in $O_{i,j}$
- $T_{ylos,f}(O_{i,j})$: in factory f, y-direction start time of load operation in $O_{i,j}$
- $T_{ylob,f}(O_{i,j})$: in factory f, y-direction braking time of load operation in $O_{i,j}$
- $T_{ylod,f}(O_{i,j})$: in factory f, y-direction duration time of load operation in $O_{i,j}$
- $T_{no,f}(O_{i,j})$: in factory f, time consumption of no-load operation in $O_{i,j}$
- $Ts_{no,f}(O_{i,j})$: in factory f, start time of no-load operation in $O_{i,j}$
- $Tc_{no,f}(O_{i,j})$: in factory f, completion time of no-load operation in $O_{i,j}$
- $T_{ns,f}(O_{i,j})$: in factory f, time consumption of no-load standbys in $O_{i,j}$
- $Ts_{ns,f}(O_{i,j})$: in factory f, start time of no-load operation in $O_{i,j}$

Table 1
Representative Studies on distributed scheduling problems.

Problem	Objective(s)	Methodology	Reference(s)
DPFSP	makespan	MILP, heuristics and variable neighborhood descent VND methods	[5]
	makespan	developed EDA	[6, 7]
	makespan	developed ABC	[9, 13, 25, 26]
	makespan	developed VNS, ABC, IG	[10]
	makespan	developed IG	[11, 22]
	makespan, negative social impact, total energy consumption	knowledge-based multi-objective memetic optimization algorithm	[24]
	makespan, total cost, average tardiness	multi-objective adaptive large neighborhood search	[14]
	makespan, total energy consumption	knowledge-based cooperative algorithm	[12]
	makespan, total tardiness	competitive memetic algorithm	[8]
	makespan, total weight tardiness	Pareto-based EDA	[15]
	total flowtime	three constructive heuristics and four metaheuristics	[18]
	total weighted earliness and tardiness	effective IG algorithm	[23]
DAPFSP	makespan	EDA-based memetic algorithm	[16]
	makespan	a MILP, three constructive heuristics, two VNS methods, and an IG algorithm	[17]
	makespan	biased-randomized iterated local search	[19]
	makespan	cooperative co-evolutionary algorithm	[21]
	total flowtime	effective invasive weed optimization algorithms	[20]
DFJSP	makespan	developed GA	[31, 33-36]
	makespan	fast heuristic algorithm	[32]
	makespan	chemical reaction optimization metaheuristics	[38]
	makespan	four MILP models and a CP model	[39]
	makespan, maximum workload, total workload, earliness/tardiness	hybrid Pareto-based TS	[2]
	makespan, maximum workload, total energy consumption of factories	EMA	[40]
	earliness/tardiness and total cost	DE-SA	[37]
DHFS	makespan	discrete ABC	[1]
	makespan	smallest-medium rule and multi-neighborhood IG	[30]
	fuzzy total tardiness, robustness	EDA and IG	[29]
DJSP	makespan	agent-based fuzzy constraint-directed negotiation mechanism	[27]

- $Tc_{ns,f}(O_{i,j})$: in factory f, completion time of no-load operation in $O_{i,j}$
- $T_{ls,f}(O_{i,j})$: in factory f, time consumption of load standbys in $O_{i,j}$
- $Ts_{ls,f}(O_{i,j})$: in factory f, start time of load standbys in $O_{i,j}$
- $Tc_{ls,f}(O_{i,j})$: in factory f, completion time of load standbys in $O_{i,j}$
- $T_{lo,f}(O_{i,j})$: in factory f, time consumption of load operations in $O_{i,j}$
- $Ts_{lo,f}(O_{i,j})$: in factory f, start time of load operations in $O_{i,j}$
- $Tc_{lo,f}(O_{i,j})$: in factory f, completion time of load operations in $O_{i,j}$
- T_{no}: total time consumption of all no-load operations in DFJSPC
- T_{ns}: total time consumption of all no-load standbys in DFJSPC
- T_{ls}: total time consumption of all load standbys in DFJSPC
- T_{lo}: total time consumption of all load operations in DFJSPC
- $E_{no,f}(O_{i,j})$: energy consumption of no-load operation in $O_{i,j}$
- $E_{ns,f}(O_{i,j})$: energy consumption of no-load standbys in $O_{i,j}$
- $E_{ls,f}(O_{i,j})$: energy consumption of load standbys in $O_{i,j}$
- $E_{lo,f}(O_{i,j})$: energy consumption of load operations in $O_{i,j}$
- E_{no}: total energy consumption of all no-load operations in DFJSPC
- E_{ns}: total energy consumption of all no-load standbys in DFJSPC
- E_{ls}: total energy consumption of all load standbys in DFJSPC
- E_{lo}: total energy consumption of all load operations in DFJSPC
- Ws: weight of lifting appliance
- Wx: weight of x-direction devices
- Wy: weight of y-direction devices
- Wj: weight of job i
- Qn: lifting weight of crane
- TEC_{mp}: energy consumption of all machine processes
- TEC_{ct}: energy consumption of all crane transportation processes
- TEC: total energy consumption of DFJSPC
- $\Psi(O_{i,j})$: set of available machines for operation $O_{i,j}$
- $\Theta< O_{i1,j1}, O_{i,j} >$: if $O_{i,j}$ is the immediate successor operation of $O_{i1,j1}$ being performed on the same machine

Decision variables:

- $u_{k,f}$: binary value set to 1 when machine k is in factory f; otherwise, $u_{k,f}$ is set to 0.
- $x_{i,j,k}$: binary value set to 1 when $O_{i,j}$ is assigned to machine k; otherwise, $x_{i,j,k}$ is set to 0.
- $y_{i,j,k,pr}$: binary value set to 1 when $O_{i,j}$ is assigned to machine k and $O_{i,j}$ is at the prth operation in scheduling; otherwise, $y_{i,j,k,pr}$ is set to 0.
- $z_{i,f}$: binary value set to 1 when job i is distributed in factory f; otherwise, $z_{i,f}$ is set to 0.

3.2. Assumptions

(1) All scheduled operations must be completed in the DFJSPC, and the operation order is strictly according to the operation schedule in each factory.
(2) At time zero, all jobs are released, and all machines are available.
(3) The crane and all machines work continuously and cannot be shut down during the working process.
(4) Preemption is not available; that is, each operation must be completed without interruption once it starts.
(5) An operation is performed by only one machine and cannot be performed by another machine at the same time.
(6) A machine performs jobs one by one and cannot perform other jobs when occupied.
(7) In each factory, a crane lifts only one job once.
(8) In each factory, the initial position of the crane is at the assigned machine position of the first operation in the factory.
(9) For safety, the load operation of a crane must wait until the target position machine is idle.

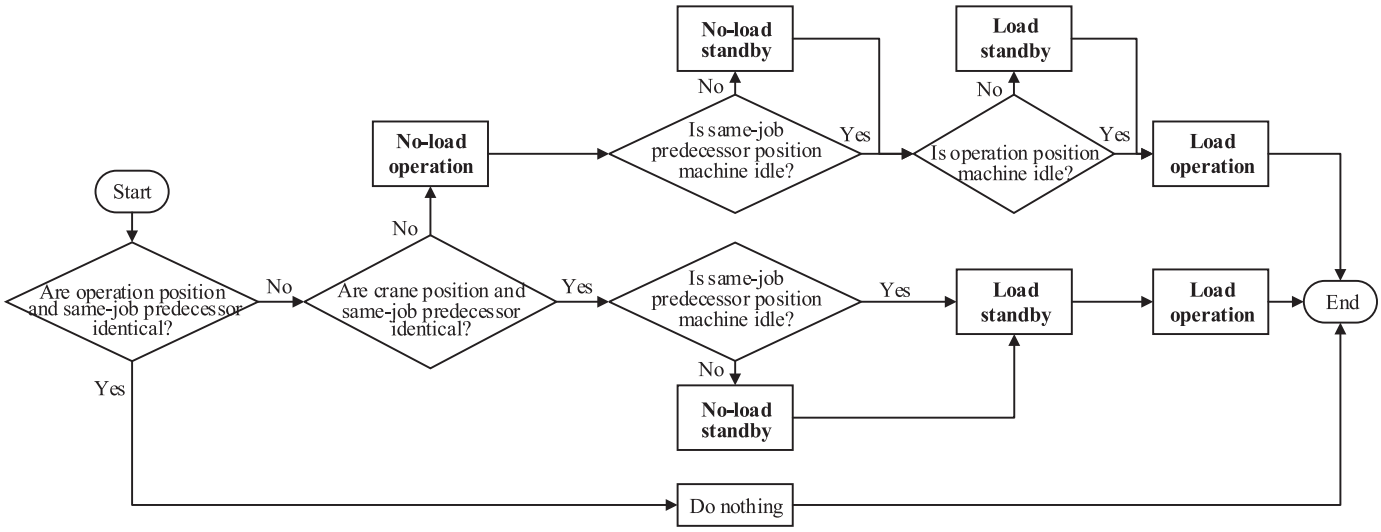


Fig. 1. Identification rules for crane conditions in each factory.

(10) The first operation of a job in each factory does not require crane transportation.
(11) In each factory, if the position and the predecessor position of an operation are identical, the operation does not require crane transportation.
(12) Once a job is assigned to a factory, it cannot be distributed to another factory.
(13) In DFJSPC, each factory can be seen as isolated, and production of each factory cannot be influenced by other factories.

3.3. Energy consumption of machine process

The energy consumption of $O_{i,j}$ is $E_{mp}(O_{i,j})$ and can be calculated as (1):

$$E_{mp}(O_{i,j}) = E_{i,j} \cdot T_{i,j} \tag{1}$$

where $E_{i,j}$ and $T_{i,j}$ are the unit energy consumption and process time of $O_{i,j}$, respectively.

In all factories, The TEC of the machine process TEC_{mp} is the sum of all $E_{mp}(O_{i,j})$, which is expressed in (2):

$$TEC_{mp} = \sum_{i=1}^{I} \sum_{j=1}^{J} E_{mp}(O_{i,j}) \tag{2}$$

3.4. Energy consumption of crane transportations

In every factory, all movement of jobs between different machines is realized by crane transportation. Based on the relationships between crane position, operation position, and same-job predecessor position, all crane conditions can be classified into the following four conditions: no-load operation, no-load standby, load standby, and load operation. In each factory, the order of the crane conditions should obey the identification rules illustrated in Fig. 1.

In every factory, the crane moves along the x-direction and y-direction and cannot move along the diagonal direction. As illustrated in Fig. 2, a crane includes a gantry (crane-gantry) and trolley (crane-trolley). Crane-gantry moves along the x-direction, while crane-trolley moves along the y-direction. Crane-trolley is on the track of the crane-gantry. For movement in the x-direction, the moving part of the crane lifts appliances, x-direction devices, and y-direction devices. For movement in the y-direction, the moving part of the crane lifts appliances and y-direction devices.

For both x- and y-direction movement, three moving stages are included, i.e. start time, duration time, and braking time. In start time, the crane accelerates uniformly from 0 m/s to V_x (in x-direction) or V_y (in y-direction). In duration time, the crane moves at a speed of V_x or V_y. In braking time, the crane decelerates uniformly from V_x or V_y to 0 m/s.

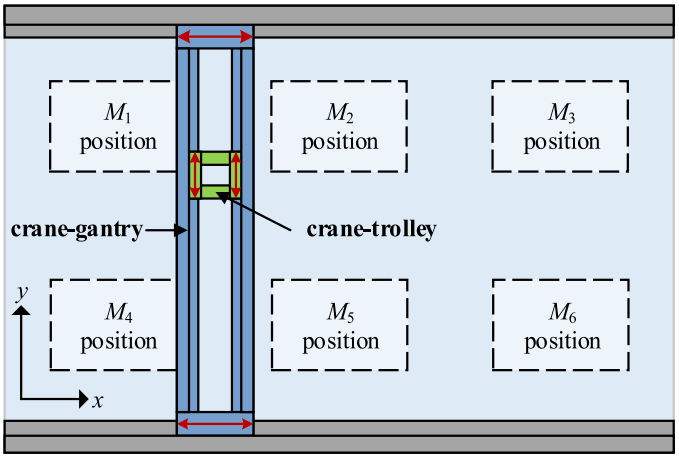


Fig. 2. Moving crane system in each factory.

3.4.1. Time and energy consumption of no-load operation

If job i is distributed in factory f, and $O_{i,j}$ has a no-load operation, the x-direction moving time is calculated by (3–6), and the y-direction moving time is calculated by (7–10). The total time consumption and TEC of the no-load operation in $O_{i,j}$ are calculated by (11) and (12), respectively. The TEC of all no-load operations in the DFJSPC (E_{no}) is the sum of $E_{no,f}(O_{i,j})$ for all no-load operations in corresponding factories. Specifically, the meaning of a in (3) can be explained in Fig. 3, where a is the rate of no-load sum weight to full-load sum weight of crane system in no-load operation.

$$a = \frac{Ws + Wx + Wy}{Qn + Wx + Wy} \tag{3}$$

$$T_{xnos,f}(O_{i,j}) = a \cdot \frac{Vx}{a_{xs}} \tag{4}$$

$$T_{xnob,f}(O_{i,j}) = a \cdot \frac{Vx}{a_{xb}} \tag{5}$$

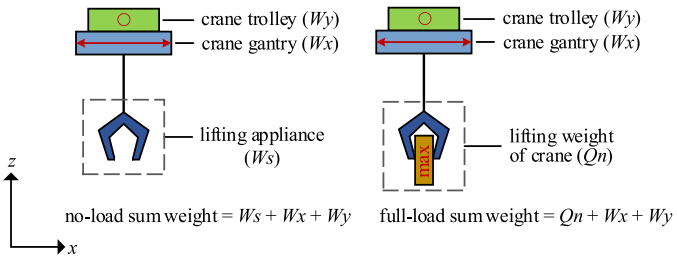


Fig. 3. Weight relationships of crane system in no-load operation.

$$T_{xnod,f}(O_{i,j}) = \max\left\{\frac{\left|x_{pp} - x_{cp}\right|}{Vx} - \frac{Vx}{2}\cdot a\cdot\left(\frac{1}{a_{xs}} + \frac{1}{a_{xb}}\right), 0\right\} \tag{6}$$

$$b = \frac{Ws + Wy}{Qn + Wy} \tag{7}$$

$$T_{ynos,f}(O_{i,j}) = b\cdot\frac{Vy}{a_{ys}} \tag{8}$$

$$T_{ynob,f}(O_{i,j}) = b\cdot\frac{Vy}{a_{yb}} \tag{9}$$

$$T_{ynod,f}(O_{i,j}) = \max\left\{\frac{\left|y_{pp} - y_{cp}\right|}{Vy} - \frac{Vy}{2}\cdot b\cdot\left(\frac{1}{a_{ys}} + \frac{1}{a_{yb}}\right), 0\right\} \tag{10}$$

$$\begin{aligned} T_{no,f}(O_{i,j}) = {} & T_{xnos,f}(O_{i,j}) + T_{xnob,f}(O_{i,j}) + T_{xnod,f}(O_{i,j}) \\ & + T_{ynos,f}(O_{i,j}) + T_{ynob,f}(O_{i,j}) + T_{ynod,f}(O_{i,j}) \end{aligned} \tag{11}$$

$$\begin{aligned} E_{no,f}(O_{i,j}) = {} & T_{xnos,f}(O_{i,j})\cdot P_{xs} + T_{ynos,f}(O_{i,j})\cdot P_{ys} \\ & + \frac{Ws}{Qn}\cdot\left(T_{xnod,f}(O_{i,j})\cdot P_{xd} + T_{ynod,f}(O_{i,j})\cdot P_{yd}\right) \end{aligned} \tag{12}$$

3.4.2. Time and energy consumption of no-load standby

If job i is distributed in factory f, and $O_{i,j}$ has no-load standby, the no-load standby time consumption of $O_{i,j}$ is calculated by (13), and the no-load standby energy consumption of $O_{i,j}$ is calculated by (14). The TEC of all no-load standbys in the DFJSPC (E_{ns}) is the sum of $E_{ns,f}(O_{i,j})$ for all no-load standbys in the corresponding factories.

$$T_{ns,f}(O_{i,j}) = \max\{Tc(O_{i,j-1}) - Ts_{no,f}(O_{i,j}), 0\} \tag{13}$$

$$E_{ns,f}(O_{i,j}) = T_{ns,f}(O_{i,j})\cdot Ps \tag{14}$$

3.4.3. Time and energy consumption of load standby

If job i is distributed in factory f, and $O_{i,j}$ has load standby, the load standby time consumption of $O_{i,j}$ is calculated by (15), while the load standby energy consumption of $O_{i,j}$ is calculated by (16). The TEC of all load standbys in the DFJSPC (E_{ls}) is the sum of $E_{ls,f}(O_{i,j})$ for all no-load standbys in the corresponding factories. $O_{i,j}$ is the immediate successor operation of $O_{i1,j1}$ that is performed on the same machine.

$$T_{ls,f}(O_{i,j}) = \max\{Tc(O_{i1,j1}) - Tc_{ns,f}(O_{i,j}), 0\}, \Theta < O_{i1,j1}, O_{i,j} > \tag{15}$$

$$E_{ls,f}(O_{i,j}) = T_{ls,f}(O_{i,j})\cdot Ps \tag{16}$$

3.4.4. Time and energy consumption of load operation

If job i is distributed in factory f, and $O_{i,j}$ has a load operation, the x-direction moving time is calculated by (17–20), while the y-direction moving time is calculated by (21–24). The total time consumption and TEC of a load operation in $O_{i,j}$ are calculated by (25) and (26), respectively. The TEC of all load operations in the DFJSPC (E_{lo}) is the sum of $E_{lo,f}(O_{i,j})$ for all load operations in the corresponding factories.

$$c = \frac{Ws + Wj + Wx + Wy}{Qn + Wx + Wy} \tag{17}$$

$$T_{xlos,f}(O_{i,j}) = c\cdot\frac{Vx}{a_{xs}} \tag{18}$$

$$T_{xlob,f}(O_{i,j}) = c\cdot\frac{Vx}{a_{xb}} \tag{19}$$

$$T_{xlod,f}(O_{i,j}) = \max\left\{\frac{\left|x_{op} - x_{pp}\right|}{Vx} - \frac{Vx}{2}\cdot c\cdot\left(\frac{1}{a_{xs}} + \frac{1}{a_{xb}}\right), 0\right\} \tag{20}$$

$$d = \frac{Ws + Wj + Wy}{Qn + Wy} \tag{21}$$

$$T_{ylos,f}(O_{i,j}) = d\cdot\frac{Vy}{a_{ys}} \tag{22}$$

$$T_{ylob,f}(O_{i,j}) = d\cdot\frac{Vy}{a_{yb}} \tag{23}$$

$$T_{ylod,f}(O_{i,j}) = \max\left\{\frac{\left|y_{op} - y_{pp}\right|}{Vy} - \frac{Vy}{2}\cdot d\cdot\left(\frac{1}{a_{ys}} + \frac{1}{a_{yb}}\right), 0\right\} \tag{24}$$

$$\begin{aligned} T_{lo,f}(O_{i,j}) = {} & T_{xlos,f}(O_{i,j}) + T_{xlob,f}(O_{i,j}) + T_{xlod,f}(O_{i,j}) + T_{ylos,f}(O_{i,j}) \\ & + T_{ylob,f}(O_{i,j}) + T_{ylod,f}(O_{i,j}) \end{aligned} \tag{25}$$

$$\begin{aligned} E_{lo,f}(O_{i,j}) = {} & T_{xlos,f}(O_{i,j})\cdot P_{xs} + T_{ylos,f}(O_{i,j})\cdot P_{ys} \\ & + \frac{Ws + Wj}{Qn}\cdot\left(T_{xlod,f}(O_{i,j})\cdot P_{xd} + T_{ylod,f}(O_{i,j})\cdot P_{yd}\right) \end{aligned} \tag{26}$$

3.4.5. TEC of crane transportations

The TEC of crane transportations is the sum of the energy consumption of the above four crane conditions, as expressed in (27). In addition, the TEC of the DFJSPC is expressed in (28).

$$TEC_{ct} = E_{no} + E_{ns} + E_{ls} + E_{lo} \tag{27}$$

$$TEC = TEC_{mp} + TEC_{ct} \tag{28}$$

3.5. Formulation of DFJSPC model

The DFJSPC model is formulated in (29–50) as follows:

$$\min f = w\cdot f_1 + (1 - w)\cdot f_2 \tag{29}$$

$$f_1 = \max\{Tc(O_{i,j})\} \tag{30}$$

$$f_2 = TEC \tag{31}$$

$$Ts(O_{i,j}) \geq Tc(O_{i,j-1}), \forall i, j \tag{32}$$

$$k(O_{i,j}) \in \Psi(O_{i,j}), \forall i, j \tag{33}$$

$$P(\pi_{1,f}) = \delta_{1,f}, \quad \forall f \tag{34}$$

$$\sum_{k=1}^{K} x_{i,j,k} = 1, \forall i, j \tag{35}$$

$$\sum_{k=1}^{K} \sum_{pr}^{PR} \left(y_{i,j+1,k,pr} \cdot pr\right) \geq \sum_{k=1}^{K} \sum_{pr}^{PR} \left(y_{i,j,k,pr} \cdot pr\right), \forall i, j \tag{36}$$

$$z_{i,f} = u_{k,f}, \forall f, x_{i,j,k} = 1 \tag{37}$$

$$Tc_{no,f}(O_{i,j}) = Ts_{ns,f}(O_{i,j}), \forall i, j, f \tag{38}$$

$$Tc_{ns,f}(O_{i,j}) = Ts_{ls,f}(O_{i,j}), \forall i, j, f \tag{39}$$

$$Tc_{ls,f}(O_{i,j}) = Ts_{lo,f}(O_{i,j}), \forall i, j, f \tag{40}$$

$$Tc_{lo,f}(O_{i,j}) = Ts(O_{i,j}), \forall i, j, f \tag{41}$$

$$Ts_{lo,f}(O_{i,j}) \geq Tc(O_{i1,j1}), \forall f, < O_{i1,j1}, O_{i,j} > \in \Theta \tag{42}$$

$$T_{no,f}(O_{i,1}) + T_{ns,f}(O_{i,1}) + T_{ls,f}(O_{i,1}) + T_{lo,f}(O_{i,1}) = 0, \forall i, f \tag{43}$$

$$T_{no,f}(O_{i,j}) + T_{ns,f}(O_{i,j}) + T_{ls,f}(O_{i,j}) + T_{lo,f}(O_{i,j}) = 0, \quad \forall f, k(O_{i,j}) = k' \tag{44}$$

$$u_{k,f} = \begin{cases} 1, & \text{if machine} k \text{ is in factory} f \\ 0, & \text{otherwise} \end{cases} \tag{45}$$

$$x_{i,j,k} = \begin{cases} 1, & \text{if} O_{i,j} \text{is processed on machine} k \\ 0, & \text{otherwise} \end{cases} \tag{46}$$

$$y_{i,j,k,pr} = \begin{cases} 1, & \text{if the} pr \text{th operation} O_{i,j} \text{is processed on machine} k \\ 0, & \text{otherwise} \end{cases} \tag{47}$$

$$z_{i,f} = \begin{cases} 1, & \text{if job} i \text{is distributed in factory} f \\ 0, & \text{otherwise} \end{cases} \tag{48}$$

The total objective of the proposed model is (29), where weight w indicates the preference between the two objectives (i.e., makespan in (30) and TEC in (31)). In this study, w is 0.8. Constraint (32) guarantees a sequential operation order for jobs, while constraint (33) guarantees that every operation is only performed from a set of available machines. Constraint (34) signifies that the initial position of the crane in each factory is at the assigned machine position of the first scheduled operation in the corresponding factory. Constraint (35) guarantees that each operation can only be performed on one machine. Constraint (36) guarantees that operations are performed according to their priority, while constraint (37) guarantees that a job is performed on machines in the selected factory. Constraints (38–41) describe the order of the four crane operations. Constraint (42) signifies that in each factory, the load operation must wait for the destination machine to be idle, while constraint (43) signifies that the first operation of every job does not require crane transportation. Constraint (44) signifies that if the position and the predecessor position of a job are identical, the operation does not require crane transportation. Constraints (45–48) define the range of the binary decision variables.

4. Methodology

In this section, the proposed EDA-VNS algorithm for the DFJSPC is described. The algorithm framework is presented in Section 4.1, while the solution representation and encoding are discussed in Section 4.2. The decoding of the solution is provided in Section 4.3. The initialization and updating mechanism are explained in Section 4.4, and the new solution generation of the EDA component is presented in Section 4.5. The EDA probability mechanism is explained in Section 4.6. The detailed VNS neighborhood structures are described in Section 4.7, the complete VNS component structure is demonstrated in Section 4.8, and the computational complexity is analyzed in Section 4.9.

4.1. Algorithm framework

The framework of the proposed algorithm is illustrated in Fig. 4.

4.2. Representation and encoding

In this study, the solution of the DFJSPC is represented by three vectors: factory vector Λ, scheduling vector Π, and machine assignment vector Ξ. The length of Λ equals the number of jobs. The lengths of Π and Ξ are identical and equal to the total number of operations of all jobs. For factory vector $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, each element λ_i is represented by a distributed factory number. For scheduling vector $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$, each element π_i is represented by a job number, and the sequence of each job number is the processing order of the corresponding job. For machine assignment vector $\Xi = \{\delta_1, \delta_2, \ldots, \delta_n\}$, each element δ_i represents the available machine assignment for the corresponding operation.

The encoding solution of a 4-job, 2-factory, and 2-machine (4J2F2M) example is shown in Fig. 5, where the solution is composed of a factory vector, scheduling vector, and machine assignment vector. In Fig. 5, jobs 2 and 3 are distributed to factory 1, while jobs 1 and 4 are distributed to factory 2. Machines 1 and 2 are in factory 1, while machines 3 and 4 are in factory 2.

Fig. 6 presents the Gantt chart of the solution, the white rectangles represent the machine operation time, the four colored rectangles represent the time of the four different crane states, and the number in the colored rectangles represents the job number of the corresponding crane transportation. The according factory vector, scheduling vector, and machine assignment vector in Fig. 6 are illustrated in Fig. 5.

In factory 1, $O_{2,1}$ is the first operation, the job I_2 is at machine M_2, and the crane position is at the first operation position M_2. Then, operation $O_{2,2}$ needs to be performed at M_1, which requires the load operation of crane transportation from M_2 to M_1 with I_2. Next, operation $O_{3,1}$ is the first operation of job I_3, the operation does not require crane transportation. Afterwards, operation $O_{3,2}$ needs to be performed at M_2, the crane firstly waits the completion of $O_{3,1}$, transports I_3 from M_1 to M_2. Finally, operation $O_{3,3}$ is scheduled at M_1, and the crane waits the completion of $O_{3,2}$, and transports I_3 back to M_1.

In factory 2, operations $O_{4,1}$ and $O_{1,1}$ are the first operations of jobs I_4 and I_1, respectively, so the operations do not need crane transportation. The initial crane position in factory 2 is at machine M_3. Then, operation $O_{4,2}$ requires the load operation of crane transportation from M_3 to M_4. Next, for $O_{1,2}$, the crane should move to M_3 and wait the completion of the same-job predecessor operation $O_{1,1}$, which requires no-load operation and no-load standby; afterwards, the crane should take I_1 until the completion of $O_{4,2}$ on M_4 for safety and then transport the I_1 from M_3 to M_4, which requires load-standby and load operation. $O_{4,3}$ and its same-job predecessor operation $O_{4,2}$ share the same machine, so it does not need crane transportation. Finally, for $O_{1,3}$, the crane waits until the completion of $O_{1,2}$ and take I_1 from M_4 to M_3.

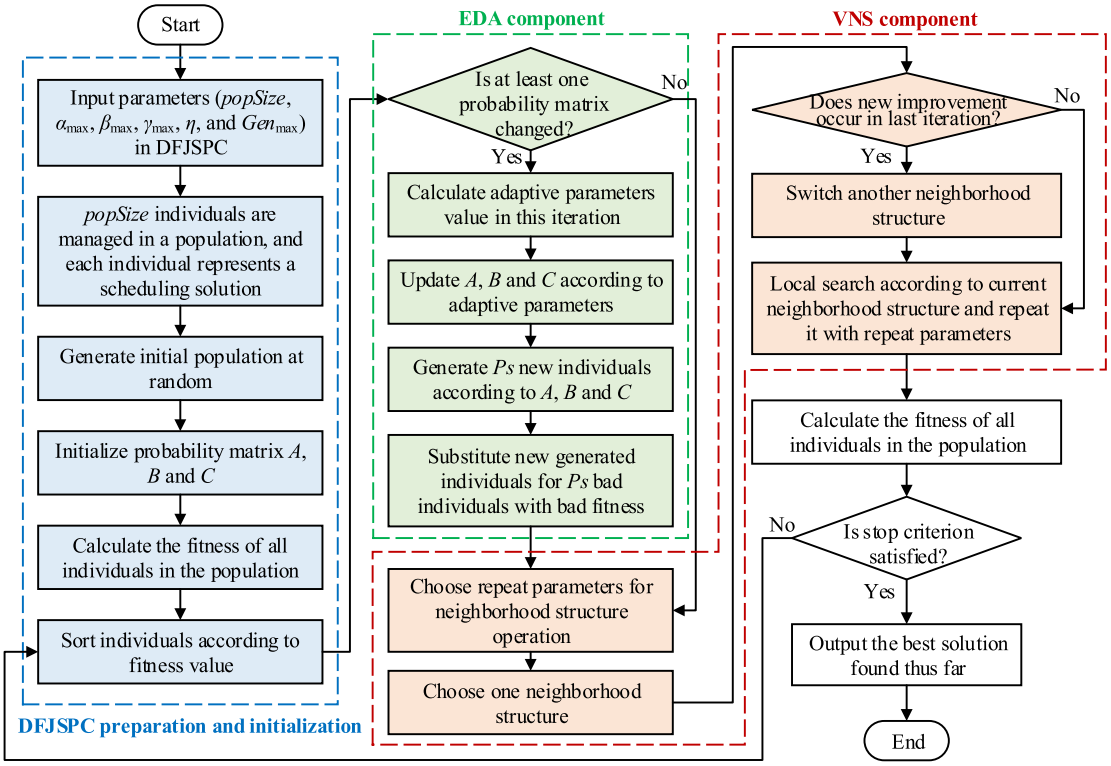


Fig. 4. Framework of proposed algorithm (EDA-VNS) for the distributed flexible job shop scheduling problem with crane transportations (DFJSPC).

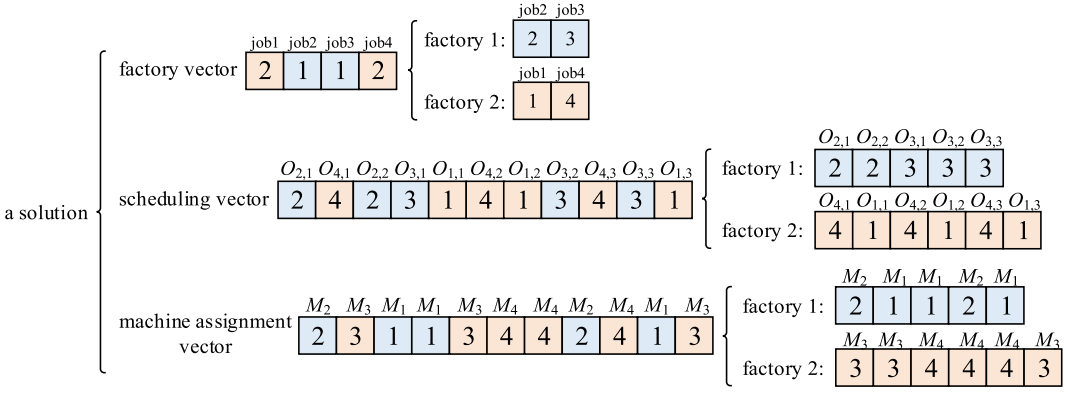


Fig. 5. Solution sturcture and encoding.

4.3. Decoding

When the solution is determined, its fitness can be calculated according to the given factory vector Λ, scheduling vector Π and machine assignment vector Ξ. The decoding heuristic is formulated in Algorithm 1.

4.4. Initialization and updating mechanism

The initialization of the proposed algorithm includes the initialization of the population solution and the initialization of probability matrices *A*, *B*, and *C* in the EDA component. The population solution is initialized at random. Probability matrices *A*, *B*, and *C* control the generation of the factory vector, scheduling vector, and machine assignment vector, respectively.

A_{if} is an element of probability matrix *A*, and $A_{if}(Gen)$ represents the probability that job *i* is assigned to factory *f* at the *Gen*th iteration ($Gen = 0$ represents the initial generation). The initialization of probability matrix *A* is expressed as (49), where *F* represents the number of factories. The update of *A* is based on (50), where α is the learning rate of probability matrix *A*, and *SI* is the number of superior individuals in the population. The indicator I_{if}^{s} is expressed as (51), where *s* is the *s*th superior individual in the population.

$$A_{if}(Gen = 0) = 1/F, \quad \forall i, f \tag{49}$$

$$A_{if}(Gen + 1) = (1 - \alpha) \cdot A_{if}(Gen) + \frac{\alpha}{SI} \sum_{s=1}^{SI} I_{if}^{s}, \quad \forall i, f \tag{50}$$

$$I_{if}^{s} = \begin{cases} 1, & \text{if job } i \text{ is distributed to factory } f \\ 0, & \text{else} \end{cases}, \forall i, f \tag{51}$$

B_{mi} is an element of probability matrix *B*, and $B_{mi}(Gen)$ represents the probability that job *i* is assigned to the *m*th position in the scheduling vector at the *Gen*th iteration ($Gen = 0$ represents the initial generation). The initialization of probability matrix *B* is expressed as (52), where *I* denotes the number of jobs. The update of *B* is based on (53), where β is

Algorithm 1
Decoding.

Algorithm 1: Decoding
input: factory vector, scheduling vector, and machine assignment vector of the solution
output: fitness (makespan and TEC) of the solution
For each factory
  For each operation in the factory
    Calculate the time and energy consumption of the machine process
    If it is not the first operation of the corresponding job
      If the operation machine is at the same-job predecessor machine
        the crane is not required to do anything for this operation
      else
        If the crane is at the same-job predecessor machine
          Calculate the time and energy consumptions of no-load standby, load standby, and load operation
        else
          Calculate the time and energy consumptions of no-load operation, no-load standby, load standby, and load operation
        end
      end
      Calculate the start time and completion time of the operation
    else
      Calculate the start time and completion time of the operation
    end
  end
end
Calculate the makespan according to the start time and completion time of every operation
Calculate the TEC according to the energy consumption of every operation
Output the fitness of the solution

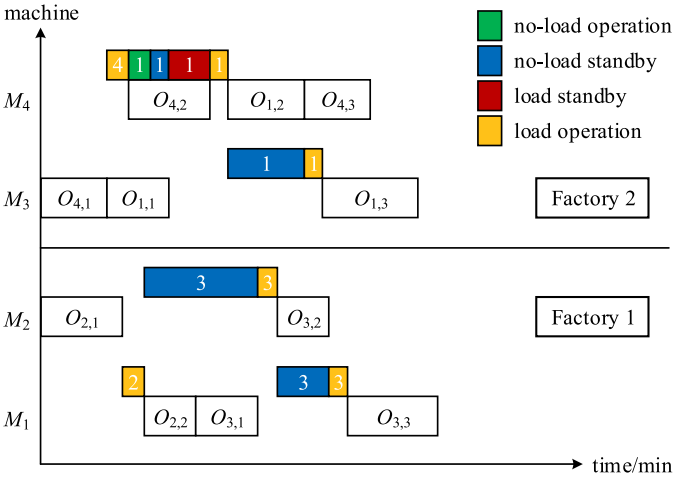


Fig. 6. Gantt chart of a 4J2F2M scale solution of distributed flexible job shop scheduling problem with crane transportations.

the learning rate of probability matrix B. The indicator I^s_{mi} is expressed as (54), where s is the sth superior individual in the population.

$$B_{mi}(Gen=0) = 1/I, \quad \forall m,i \tag{52}$$

$$B_{mi}(Gen+1) = (1-\beta)\cdot B_{mi}(Gen) + \frac{\beta}{SI}\sum_{s=1}^{SI} I^s_{mi}, \quad \forall m,i \tag{53}$$

$$I^s_{mi} = \begin{cases} 1, & \text{if job } i \text{ appears at or before position } m \\ 0, & \text{else} \end{cases}, \forall m,i \tag{54}$$

C_{fijk} is an element of probability matrix C, and $C_{fijk}(Gen)$ represents the probability that the jth operation of job i in factory f is assigned to machine k at the Genth iteration ($Gen = 0$ represents the initial generation). The initialization of probability matrix C is expressed as (55), where $K_{available}$ donotes the number of available machines for the jth operation of job i. The update of C is based on (56), where γ is the learning rate of probability matrix C. The indicator I^s_{fijk} is expressed as (57), where s is the sth superior individual in the population.

$$C_{fijk}(Gen=0) = \begin{cases} 1/K_{available}, & \text{if machine } k \text{ is available of } O_{i,j} \text{ in factory } f \\ 0, & \text{else} \end{cases}, \forall f,i,j,k \tag{55}$$

$$C_{fijk}(Gen+1) = (1-\gamma)\cdot C_{fijk}(Gen) + \frac{\gamma}{SI}\sum_{s=1}^{SI} I^s_{fijk}, \quad \forall f,i,j,k \tag{56}$$

$$I^s_{fijk}(Gen=0) = \begin{cases} 1, & \text{if } O_{i,j} \text{ is assigned to machine } k \text{ in factory } f \\ 0, & \text{else} \end{cases}, \forall f,i,j,k \tag{57}$$

The design of update Eqs. (50), (53), and (57) ensures that: (1) in the factory vector, the sum of the generation probability for all factories in job i is 1; (2) in the scheduling vector, the sum of the generation probability for all jobs in all position m is 1; and (3) in the machine assignment vector, the sum of the generation probability for all machines in operation $O_{i,j}$ is 1. Lemmas 1, 2, and 3 prove the conclusions by mathematical induction as follows.

Lemma 1. *For job i, the equation* $\sum_{f=1}^{F} A_{if}(Gen) = 1, \forall Gen \geq 0$ *will be valid.*

Proof. . For $Gen = 0$, according to Eq. (49), $\sum_{f=1}^{F} A_{if}(Gen)$ can be expressed as (58). Assume that for $Gen = k$ ($k > 0$), $\sum_{f=1}^{F} A_{if}(Gen=k) = 1$ is valid. For $Gen = k + 1$ ($k > 0$), according to update Eq. (50), the calculation process of $\sum_{f=1}^{F} A_{if}(Gen=k+1)$ is demonstrated in (59) . Note that if job i is assigned to factory f, then no other factories can perform job i; therefore, $\sum_{f=1}^{F} I^s_{if} = 1$. In (59), $\sum_{f=1}^{F} A_{if}(Gen=k) = \sum_{f=1}^{F} A_{if}(Gen=k+1) = 1$. Therefore, **Lemma 1** is proved.

$$\sum_{f=1}^{F} A_{if}(Gen=0) = \sum_{f=1}^{F}(1/F) = F\cdot(1/F) = 1, \quad \forall i \tag{58}$$

$$\begin{aligned}
&\sum_{f=1}^{F} A_{if}(Gen=k+1) = \sum_{f=1}^{F}\left[(1-\alpha)\cdot A_{if}(Gen=k)\right] + \sum_{f=1}^{F}\left(\frac{\alpha}{SI}\sum_{s=1}^{SI} I_{if}^{s}\right), \quad k>0, \forall i \\
&= (1-\alpha)\cdot \sum_{f=1}^{F} A_{if}(Gen=k) + \frac{\alpha}{SI}\sum_{f=1}^{F}\sum_{s=1}^{SI} I_{if}^{s}, \quad k>0, \forall i \\
&= (1-\alpha) + \frac{\alpha}{SI}\sum_{f=1}^{F}\sum_{s=1}^{SI} I_{if}^{s}, \quad k>0, \forall i \\
&= (1-\alpha) + \frac{\alpha}{SI}\sum_{s=1}^{SI} 1 = (1-\alpha)+\alpha = 1, \quad k>0, \forall i
\end{aligned} \tag{59}$$

Lemma 2. *For position m, the equation* $\sum_{i=1}^{I} B_{mi}(Gen) = 1, \forall Gen \geq 0$ *will be valid.*

Proof. . For $Gen = 0$, according to Eq. (52), $\sum_{i=1}^{I} B_{mi}(Gen)$can be expressed as (60). Assume that for $Gen = k$ $(k > 0)$, $\sum_{i=1}^{I} B_{mi}(Gen = k) = 1$is valid. For $Gen = k + 1$ $(k > 0)$, according to update Eq. (53), the calculation process of $\sum_{i=1}^{I} B_{mi}(Gen = k+1)$ is demonstrated in (61). Note that if job i appears in position m, no other jobs can appear in position m; therefore, $\sum_{i=1}^{I} I_{mi}^{s} = 1$. In (61), $\sum_{i=1}^{I} B_{mi}(Gen = k) = \sum_{i=1}^{I} B_{mi}(Gen = k+1) = 1$. Therefore, **Lemma 2** is proved.

$$\sum_{i=1}^{I} B_{mi}(Gen=0) = \sum_{i=1}^{I}(1/I) = I\cdot(1/I) = 1, \quad \forall m \tag{60}$$

$$\begin{aligned}
&\sum_{i=1}^{I} B_{mi}(Gen=k+1) = \sum_{i=1}^{I}\left[(1-\beta)\cdot B_{mi}(Gen=k)\right] + \sum_{i=1}^{I}\left(\frac{\beta}{SI}\sum_{s=1}^{SI} I_{mi}^{s}\right), \quad k>0, \forall m \\
&= (1-\beta)\cdot \sum_{i=1}^{I} B_{mi}(Gen=k) + \frac{\beta}{SI}\sum_{i=1}^{I}\sum_{s=1}^{SI} I_{mi}^{s}, \quad k>0, \forall m \\
&= (1-\beta) + \frac{\beta}{SI}\sum_{i=1}^{I}\sum_{s=1}^{SI} I_{mi}^{s}, \quad k>0, \forall m \\
&= (1-\beta) + \frac{\beta}{SI}\sum_{s=1}^{SI} 1 = (1-\beta)+\beta = 1, \quad k>0, \forall m
\end{aligned} \tag{61}$$

Lemma 3. *For* $O_{i,j}$ *in factory f, the equation* $\sum_{k=1}^{K} C_{fijk}(Gen) = 1, \forall Gen \geq 0$ *will be valid, where K is the number of the available machines in factory f, after updating the probability matrices.*

Proof. . For $Gen = 0$, according to Eq. (55), $\sum_{k=1}^{K} C_{fijk}(Gen)$ can be expressed as (62). Assume that for $Gen = p$ $(p > 0)$, $\sum_{k=1}^{K} C_{fijk}(Gen = p) = 1$ is valid. For $Gen = p + 1$ $(p > 0)$, according to update Eq. (56), the calculation process of $\sum_{k=1}^{K} C_{fijk}(Gen = p+1)$ is demonstrated in (63). Note that if operation $O_{i,j}$ appears on machine k, no other machines can perform $O_{i,j}$; therefore, $\sum_{k=1}^{K} I_{fijk}^{s} = 1$. In (63), $\sum_{k=1}^{K} C_{fijk}(Gen = p) = \sum_{k=1}^{K} C_{fijk}(Gen = p+1) = 1$. Therefore, **Lemma 3** is proved.

$$\sum_{k=1}^{K} C_{fijk}(Gen=0) = \sum_{k=1}^{K}(1/K_{available}) = K\cdot(1/K_{available}) = 1, \quad \forall f,i,j \tag{62}$$

$$\begin{aligned}
&\sum_{k=1}^{K} C_{fijk}(Gen=p+1) = \sum_{k=1}^{K}\left[(1-\gamma)\cdot C_{fijk}(Gen=p)\right] + \sum_{k=1}^{K}\left(\frac{\gamma}{SI}\sum_{s=1}^{SI} I_{fijk}^{s}\right), \quad p>0, \forall f,i,j \\
&= (1-\gamma)\cdot \sum_{k=1}^{K} C_{fijk}(Gen=p) + \frac{\gamma}{SI}\sum_{k=1}^{K}\sum_{s=1}^{SI} I_{fijk}^{s}, \quad p>0, \forall f,i,j \\
&= (1-\gamma) + \frac{\gamma}{SI}\sum_{k=1}^{K}\sum_{s=1}^{SI} I_{fijk}^{s}, \quad p>0, \forall f,i,j \\
&= (1-\gamma) + \frac{\gamma}{SI}\sum_{s=1}^{SI} 1 = (1-\gamma)+\gamma = 1, \quad p>0, \forall f,i,j
\end{aligned} \tag{63}$$

For improved output, *SI*, α, β, and γ are self-adaptive during the convergence process. *SI* represents the degree to which the population in this iteration learns the information from the superior individuals from the previous generation. The smaller *SI* is, the better the characteristics the next generation will learn. Therefore, *SI* should be reduced uniformly during the iteration, as expressed in (64). For learning rates α, β, and γ, the larger α, β, and γ are, the larger the number of superior characteristics the next generation will learn. Therefore, α, β, and γ should be increased uniformly during the iteration, as expressed in (65–67).

$$Si = \begin{cases} 20 + \mathrm{int}\left[\frac{Gen}{Gen_{\max}}(popSize\cdot\eta - 20)\right], & \text{if } Gen \leq Gen_{\max} \\ popSize\cdot\eta, & \text{else} \end{cases} \tag{64}$$

$$\alpha = \begin{cases} \alpha_{\max} - \frac{Gen}{Gen_{\max}}(\alpha_{\max} - 0.01), & \text{if } Gen \leq Gen_{\max} \\ 0.01, & \text{else} \end{cases} \tag{65}$$

$$\beta = \begin{cases} \beta_{\max} - \frac{Gen}{Gen_{\max}}(\beta_{\max} - 0.01), & \text{if } Gen \leq Gen_{\max} \\ 0.01, & \text{else} \end{cases} \tag{66}$$

$$\gamma = \begin{cases} \gamma_{\max} - \frac{Gen}{Gen_{\max}}(\gamma_{\max} - 0.01), & \text{if } Gen \leq Gen_{\max} \\ 0.01, & \text{else} \end{cases} \tag{67}$$

In (64), int[x] is the maximum integer number less than or equal to x. $\alpha_{\max}$, $\beta_{\max}$, $\gamma_{\max}$, and $Gen_{\max}$ are constant, and $\alpha_{\max}$, $\beta_{\max}$, and $\gamma_{\max}$ are the maximum values of α, β, and γ during the self-adaptive iteration process. It can be concluded from (64–67) that with the iteration process of the proposed algorithm, the value intervals of *SI*, α, β, and γ uniformly change from 20 to $popSize \times \eta$, $\alpha_{\max}$ to 0.01, $\beta_{\max}$ to 0.01, and $\gamma_{\max}$ to 0.01, respectively, where *popSize* denotes the number of individuals in the population (in this study, $popSize = 100$), η is the proportion of superior individuals selected from the population, and $\alpha_{\max}$, $\beta_{\max}$ and $\gamma_{\max}$ are larger than 0.01. From Eqs. (64–67), $Gen_{\max}$ is the iteration range of self-adaptive process for parameters, it controls the degree of self-adaptive parameters mode in optimization.

4.5. New solution generation of EDA component

The factory vector, scheduling vector, and machine assignment vector of the solution are generated by probability matrices *A*, *B*, and *C*, respectively.

For the generation of the factory vector, the assigned factory of every job is generated based on the probability distribution of the factory vector. Each position i in factory vector is generated based on the probability distribution of $\{A_{i1}, A_{i2}, \ldots, A_{iF}\}$, where F is the number of factories.

For the generation of the scheduling vector, each position m in scheduling vector is generated based on the probability distribution of $\{B_{m1}, B_{m2}, \ldots, B_{mI}\}$, where I is the number of jobs. Note that if all operations for a job are distributed in the scheduling vector, the distribution probability of the job should be set to 0. This signifies that the probability sum of each following position is not 1. To generate a feasible solution, when the distribution probability is changed at position c during the generation process, a repair process is required, which is expressed as (68):

$$B_{mi} \leftarrow B_{mi} / \sum_{i=1}^{I} B_{mi}, \ m \geq c \tag{68}$$

For the generation of the machine assignment vector, each machine assignment of operation $O_{i,j}$ is generated based on the probability distribution of $\{C_{fij1}, C_{fij2}, \ldots, C_{fijKavailable}\}$ that all machines in the available machine set can perform $O_{i,j}$.

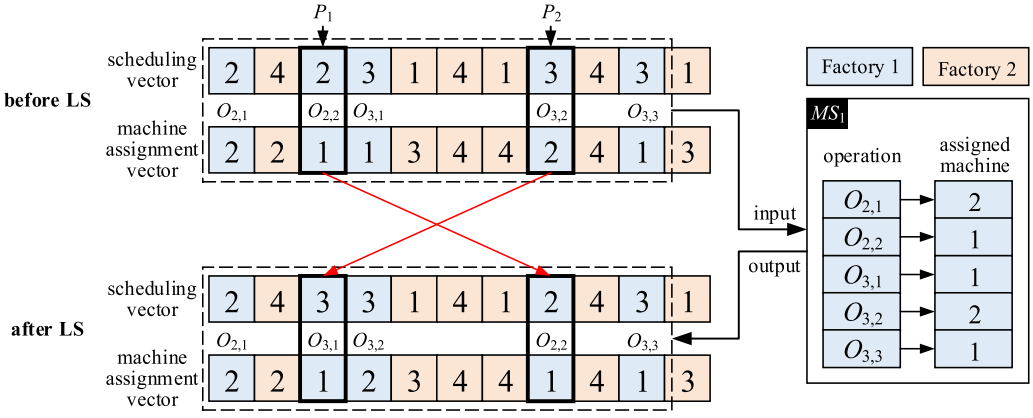


Fig. 7. Local swap neighborhood structure.

4.6. EDA probability mechanism

Compared with the VNS component, the EDA component requires more computational time. In addition, the EDA component is effective at the initial stage of the convergence process, but not at the middle or final stage. Therefore, if the EDA is called at every iteration, time will be wasted.

In this study, we use probability P_{EDA} to adjust the activity of the EDA component. The initial value of P_{EDA} is 1, signifying that the EDA component must be called at this iteration. If at least one probability matrix is changed in this iteration, P_{EDA} retains its value. However, if none of the three probability matrices are changed in this iteration, P_{EDA} is set to 0, signifying that the EDA component will not be called in this iteration or later iterations.

It should be noted that the VNS component can contribute to the change of probability matrices; therefore, the EDA component can still be active even it is not effective at the beginning of convergence process.

4.7. VNS neighborhood structures

In the VNS component, we propose five different neighborhood structures including local and global strategies. These neighborhood structures preserve the feasibility of the new individuals. It is difficult for the individuals with bad fitness to generate new best solution with neighborhood structures. In each neighborhood structure, to save the calculation resources, only top 5% of individuals in population according to fitness are applied VNS neighborhood structure for better output.

4.7.1. Local swap neighborhood structure

The local swap (LS) neighborhood structure is employed to generate a neighbor solution by swapping operation positions in the same factory. Fig. 7 illustrates the procedures in the LS neighborhood structure, and the detailed steps of LS are as follows.

Step 1. Select a factory at random, where the number of jobs in the factory must be at least 2.

Step 2. In the scheduling vector, randomly select two positions called P_1 and P_2, which must be in the assigned factory.

Step 3. Record the machine assigned for each operation from the selected factory in a set called MS_1.

Step 4. Swap the elements in P_1 and P_2 in the scheduling vector.

Step 5. In the machine assignment vector, for each position in the selected factory, replace the element with the assigned machine in MS_1 by each corresponding operation in the scheduling vector.

4.7.2. Local insert neighborhood structure

The local insert (LI) neighborhood structure is utilized to generate a neighbor solution by deleting an operation and inserting it into another position in a factory. Fig. 8 illustrates the LI procedure, and the detailed steps of LI are as follows.

Step 1. Select a factory at random, where the number of jobs in the factory must be at least 2.

Step 2. In the scheduling vector, randomly select two positions called P_1 and P_2, which must be in the selected factory.

Step 3. Record the machine assigned for each operation from the selected factory in a set called MS_2.

Step 4. In the selected factory, move the element in P_1 to P_2 in the scheduling vector, and move all elements between P_1 and P_2 (only including P_2) forward one position.

Step 5. In the machine assignment vector, for each position of the selected factory, replace the element with the assigned machine in MS_2 by each corresponding operation in the scheduling vector.

4.7.3. Local move neighborhood structure

The local move (LM) neighborhood structure is employed to generate a neighbor solution by changing the assigned machine of an operation that has the maximum machine makespan. Fig. 9 illustrates the LM procedure, and the detailed steps of LM are as follows.

Step 1. Identify the machine that has the maximum machine makespan.

Step 2. Select an operation at random that is assigned to the machine.

Step 3. Change the assigned machine for the operation to another available machine that belongs to the same factory.

4.7.4. Global move neighborhood structure

The global move (GM) neighborhood structure is utilized to generate a neighbor solution by moving all operations of one job from one factory to another factory. Fig. 10 illustrates the LM procedure, and the detailed steps of LM are as follows.

Step 1. In the factory vector, select a job and change its factory assignment.

Step 2. In the scheduling vector, select all operations of the job, and change their assigned factory to another factory.

Step 3. In the machine assignment vector, change the assigned machines of the moved operations. The assigned machines are in the new factory, and the machines are assigned randomly from a set of available machines.

4.7.5. Global exchange neighborhood structures

The global exchange (GE) neighborhood structure is employed to generate a neighbor solution by exchanging the operations of two jobs from different factories. Fig. 11 illustrates the GE procedure, and the detailed steps of GE are as follows.

Step 1. Randomly select two jobs from different factories.

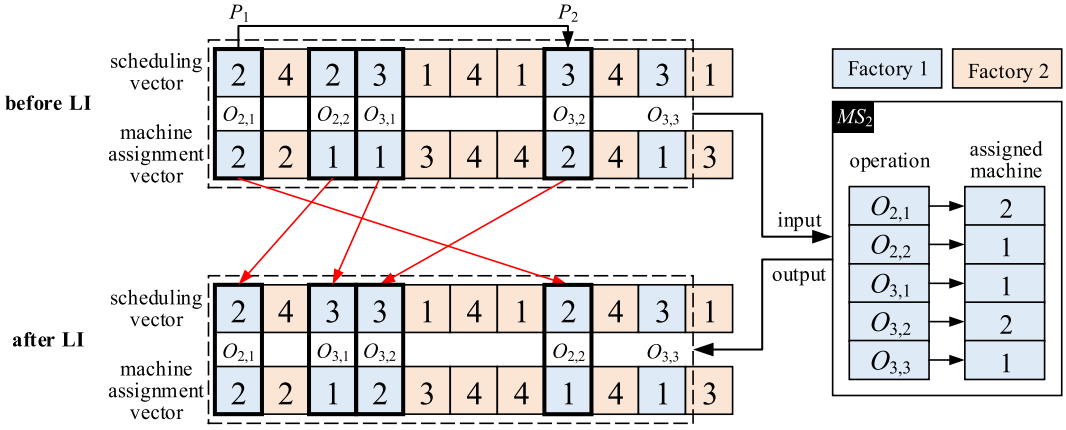


Fig. 8. Local insert neighborhood structure.

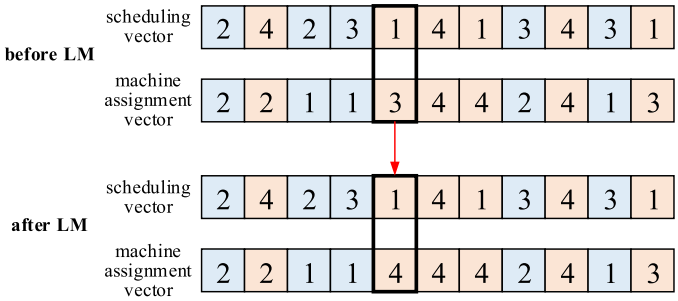


Fig. 9. Local move neighborhood structure.

Step 2. In the factory vector, exchange the factory assignment of the two jobs.

Step 3. In the scheduling vector, select all operations of the two jobs and exchange their assigned factories with each other.

Step 4. In the machine assignment vector, change the assigned machines of the moved operations. The assigned machines are in the new factory, and the machines are assigned randomly from a set of available machines.

4.8. VNS component structure

In every iteration of the VNS component, only one neighborhood structure is used to generate new individuals. The repetition time R for LS, LI, LM, GM, and GE is R_{LS}, R_{LI}, R_{LM}, R_{GM}, and R_{GE}, respectively. In each repetition, the original solution and generated solution are compared, and the better solution is retained. The repetition time R for neighborhood structures differs based on the current iteration number *Gen* (for the first iteration, *Gen* is set to 1). In the initial stage of iteration process, the job distribution of factories has great influence on the solution outcome; therefore, we set global strategies and local strategies are equally important when $1 \le Gen \le 100$. Later, the job distribution of most individuals are reasonable, so adjusting scheduling in factories can improve the quality more effectively. As a consequence, when $101 \le Gen \le 200$, we decrease the repetition times of global strategies and increase that of local strategies. For the later stage, the temporary best individual has a satisfying fitness, making global strategies hard to obtain better new best individuals; so, we enhance the repetition of local strategies for exploiting better individuals when $Gen > 200$.

To improve the proposed algorithm, the five neighborhood structures are organized into eight groups. The detailed process of the VNS component is illustrated in Algorithm 2.

4.9. Computational complexity analysis

For each generation of the proposed EDA-VNS, its computational complexity can be roughly analyzed as follows.

In the EDA component, firstly, the proposed algorithm sorts individuals with the computational complexity $O(popSize \log popSize)$; then, it updates *A*, *B*, and *C* of the *SI* superior individuals with the computational complexity $O[SI \times (FI + LI + LM)]$, where *L* is the length of scheduling vector; next, it generates new $(popSize-SI)$ individuals with the computational complexity $O[(popSize-SI) \times (I + L + L)]$. Thus, the computational complexity of the EDA component is $O\{SI \times [I \times (F-1) + L \times (I + M-2)] + popSize \times (I + 2L)\}$. In the VNS component, each generation is performed by only one of the five neighborhood structures. Each neighborhood structure performs *SI* individuals; thus, the computational complexity of the VNS component is $O(SI \times L)$.

Therefore, if the EDA component is active in the iteration, the computational complexity of the proposed algorithm is

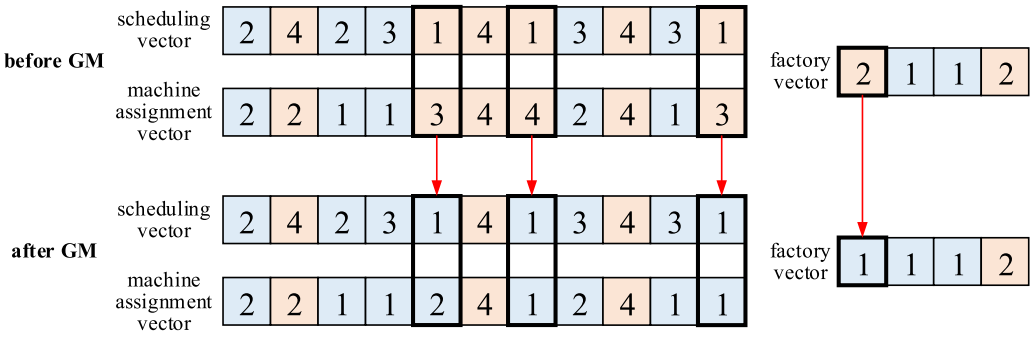


Fig. 10. Global move neighborhood structure.

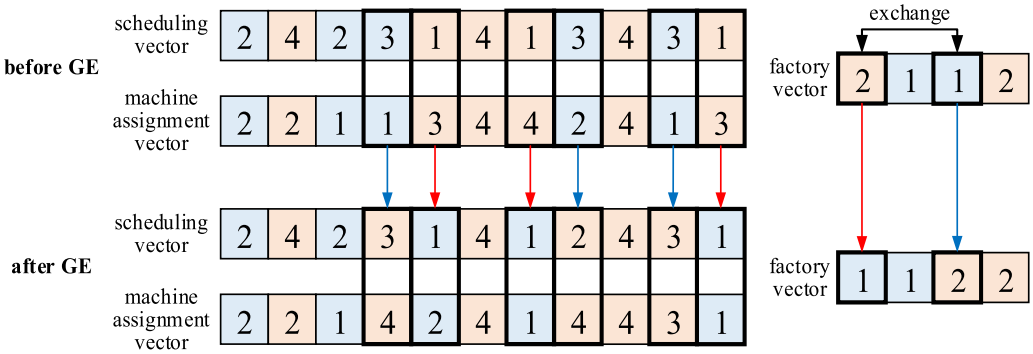


Fig. 11. Global exchange neighborhood structure.

Algorithm 2
VNS component framework.

Algorithm 2: VNS component framework

input: system parameters; data from EDA component
output: best solution found thus far in this iteration

Define $L \leftarrow 1$ as the neighborhood structure switch variety, *Gen* as the iteration number
For every iteration
  Switch *Gen*
    case $1 \le Gen \le 100$,
      $R_{LS} = 2$, $R_{LI} = 2$, $R_{LM} = 2$, $R_{GM} = 2$, $R_{GE} = 2$
    case $101 \le Gen \le 200$,
      $R_{LS} = 5$, $R_{LI} = 5$, $R_{LM} = 5$, $R_{GM} = 1$, $R_{GE} = 1$
    case $Gen > 200$,
      $R_{LS} = 8$, $R_{LI} = 8$, $R_{LM} = 8$, $R_{GM} = 0$, $R_{GE} = 0$
  end
  Switch *L*
    case $L == 1$, perform LS(R_{LS}) neighborhood structure
    case $L == 2$, perform LI(R_{LI}) neighborhood structure
    case $L == 3$, perform LM(R_{LM}) neighborhood structure
    case $L == 4$, perform GM(R_{GM}) neighborhood structure
    case $L == 5$, perform GE(R_{GE}) neighborhood structure
    case $L == 6$, perform LS(R_{LS}) neighborhood structure
    case $L == 7$, perform LI(R_{LI}) neighborhood structure
    case $L == 8$, perform LM(R_{LM}) neighborhood structure
    case $L == 9$, $L \leftarrow 1$
  end
    Compare the new mutation solution with the best solution of the population gained thus far, and retain the better solution as the best solution of the population in the next generation.
  If current neighborhood structure cannot find a better solution in this iteration
    $L \leftarrow L + 1$
  end
end
Output the best solution found thus far

$O\{SI \times [I \times (F-1) + L \times (I + M-1)] + popSize \times (\log popSize + I + 2L)\}$; if the EDA component is not active in the iteration, the computational complexity is $O(popSize \log popSize + SI \times L)$. It can be seen that the computational complexity of the proposed algorithm is acceptable, and the EDA probability mechanism can effectively improve the calculation efficiency.

5. Experimental results

In this section, we present the experimental results to evaluate the performance of the proposed algorithm. Our algorithm and other compared algorithms were implemented in C++ on an Intel Core *i*7 2.6-GHz PC with 8 GB memory. To compare the performance of the proposed algorithm to that of other efficient algorithms, the relative percentage increase (RPI) is used as a performance measure and is calculated as follows:

$$\mathrm{RPI} = \frac{f_c - f_b}{f_b} \times 100, \tag{69}$$

where f_c is the average fitness value determined by the given algorithm, and f_b is the best fitness value collected by all f_c of the compared algorithms.

5.1. Experimental instances

This study involved three types of instances. The first type had 36 instances that we generated ourselves. In the first type of instances, the number of factories was $F = \{2, 3, 5\}$, the number of jobs was $I = \{20, 40, 60, 80, 100, 200\}$, and the number of machines in each factory was $M = \{3, 5\}$. The first type of instances was used for comparing the performance of the proposed algorithm with that of other algorithms (see Sections 5.3, 5.4, 5.5, 5.6, and 5.9). The second type had 10 instances of different sizes that were automatically-generated by code. This type was used for calibration of the experimental parameters (see Sections 5.2 and 5.7). In the second type of instances, the number of factories was $F = 2$, the number of jobs was $I = \{20, 40, 60, 80, 100\}$, and the number of machines in each factory was $M = \{3, 5\}$. The third type had 12 in-

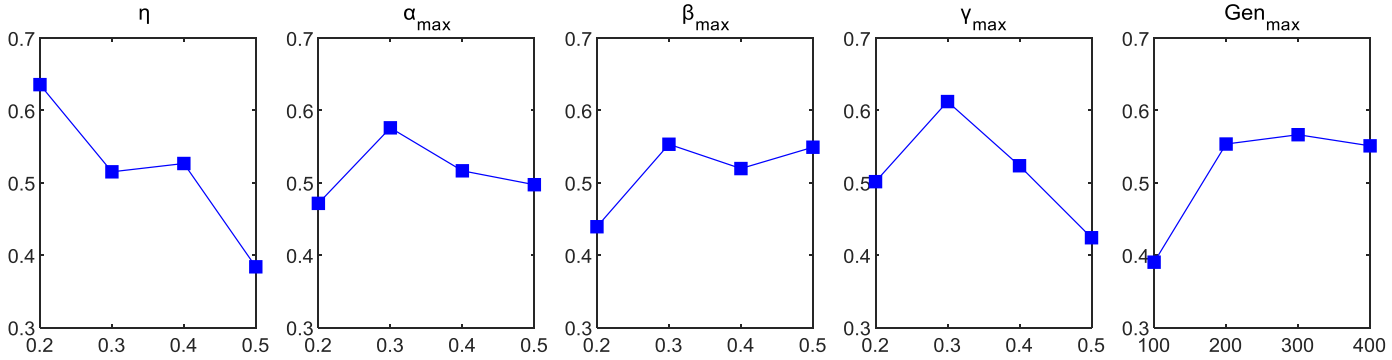


Fig. 12. Factor level trend of parameters in proposed algorithm (EDA-VNS).

Table 2
Experimental parameters and factor level values.

Parameters	Factor level			
	1	2	3	4
η	0.2	0.3	0.4	0.5
α_{max}	0.2	0.3	0.4	0.5
β_{max}	0.2	0.3	0.4	0.5
γ_{max}	0.2	0.3	0.4	0.5
Gen_{max}	100	200	300	400

stances that were used for comparison with the exact solver CPLEX (see Section 5.8). For the third type of instances, the number of factories was $F = \{2, 3\}$, the number of jobs was $I = \{2, 3, 4, 5, 6\}$, and the number of machines in each factory was $M = \{2, 3\}$. For all types of instances, the processing times were distributed uniformly in the interval [10,20].

5.2. Experimental parameters

The EDA-VNS algorithm contains five key parameters: η, α_{max}, β_{max}, γ_{max}, and Gen_{max}. To investigate the influence of these parameters on the performance of the proposed algorithm, the Taguchi method of design of experiment (DOE) [51] was implemented for the calibrated instances. Each parameter was set with four factor levels, as illustrated in Table 2. The orthogonal array $L_{16}(5^4)$ [52] is presented in Table 3, where each experimental combination is for one pair of five parameters.

Based on the orthogonal array, we ran the proposed algorithm on the second type of instances with each experimental combination of parameters for 15 independent runs. The fitness value of the parameter setting was the average value of the 15 runs. To remove the influence of different scale results, the average fitness was normalized between 0 and 1. For each combination, the normalized average fitness of instance i (NAF_i) can be calculated by (70):

$$NAF_i = \frac{AF_i - \min AF_i}{\max AF_i - \min AF_i} \tag{70}$$

where AF_i is the average fitness value of instance i, and $\min AF_i$ and $\max AF_i$ are the minimum and maximum values of all combinations of instance i, respectively.

NAF of an entire combination is the average of all normalized NAF_i values calculated by (71):

$$NAF = \frac{1}{CI}\sum_{i=1}^{CI} NAF_i, \tag{71}$$

where CI is the total number of calibration instances for the parameter setting.

Using the results in Table 3, we present the trend of each factor level in Fig. 12. It can be seen that η is the most significant parameter among the five parameters. According to the above analysis, we set $\eta = 0.5$, $\alpha_{max} = 0.2$, $\beta_{max} = 0.2$, $\gamma_{max} = 0.5$, and $Gen_{max} = 100$ for EDA-VNS in the following numerical test.

5.3. Experiment to evaluate self-adaptive parameter heuristic

To investigate the effectiveness of the self-adaptive parameter heuristic, we coded two algorithm variants: EDA-VNS without the self-adaptive parameter process (EDA-VNS-NA), and EDA-VNS with all components (EDA-VNS). In EDA-VNS-NA, $SI = popSize * \eta$ ($\eta = 0.5$), $\alpha = \alpha_{max} = 0.2$, $\beta = \beta_{max} = 0.2$, and $\gamma = \gamma_{max} = 0.5$. All components of the two compared algorithms were identical except for the self-adaptive parameter heuristic.

To evaluate whether the difference between the two methods was significant, we performed a multifactor analysis of variance (ANOVA) considering the compared methods as factors. Fig. 15(a) presents the means and the 95% least significant difference (LSD) intervals for the fitness values of the two compared methods. The p-value was smaller than 0.01; therefore, there was a significant difference between the compared methods. It can thus be concluded from the experimental results that the self-adaptive parameter heuristic significantly improved the performance of the proposed algorithm.

5.4. Exploration comparison experiment

To evaluate the exploration ability of the proposed algorithm, we coded two types of algorithms: EDA-VNS without the EDA component (NEDA-VNS), and EDA-VNS with all components (EDA-VNS). All other components of the two compared algorithms were identical. Fig. 15(b) presents the means and 95% LSD intervals for the fitness values of the two compared methods. The p-value was smaller than 0.01; therefore, there was a significant difference between the compared methods. It can thus be concluded from the experimental results that the EDA component significantly improved the exploration ability of the proposed algorithm.

5.5. Exploitation comparison experiment

To investigate the exploitation ability of the proposed algorithm, we coded two types of algorithms: EDA-VNS without the VNS component (EDA-NVNS), and EDA-VNS with all components (EDA-VNS). All other components of the two compared algorithms were identical. Fig. 15(c) presents the means and 95% LSD intervals for the fitness values of the two compared methods. The p-value was smaller than 0.01; therefore, there was a significant difference between the compared methods. The experimental results thus demonstrate that the VNS component significantly improved the exploitation ability of the proposed algorithm.

5.6. EDA probability mechanism comparison experiment

To investigate the effect of the EDA probability mechanism of the proposed algorithm, we coded two types of algorithms: EDA-VNS with-

Table 3
Orthogonal array and normalized average fitness values.

Experimental Combinations	Factor level					*NAF*	Experimental Combinations	Factor level					*NAF*
	η	α_{max}	β_{max}	γ_{max}	Gen_{max}			η	α_{max}	β_{max}	γ_{max}	Gen_{max}	
1	1	1	1	1	1	0.3769	9	3	1	3	4	2	0.4341
2	1	2	2	2	2	0.8695	10	3	2	4	3	1	0.5043
3	1	3	3	3	3	0.7002	11	3	3	1	2	3	0.5848
4	1	4	4	4	4	0.5963	12	3	4	2	1	4	0.5836
5	2	1	2	3	4	0.5536	13	4	1	4	2	3	0.5216
6	2	2	1	4	3	0.4592	14	4	2	3	1	4	0.4711
7	2	3	4	1	2	0.5746	15	4	3	2	4	1	0.2067
8	2	4	3	2	1	0.4734	16	4	4	1	3	2	0.3363

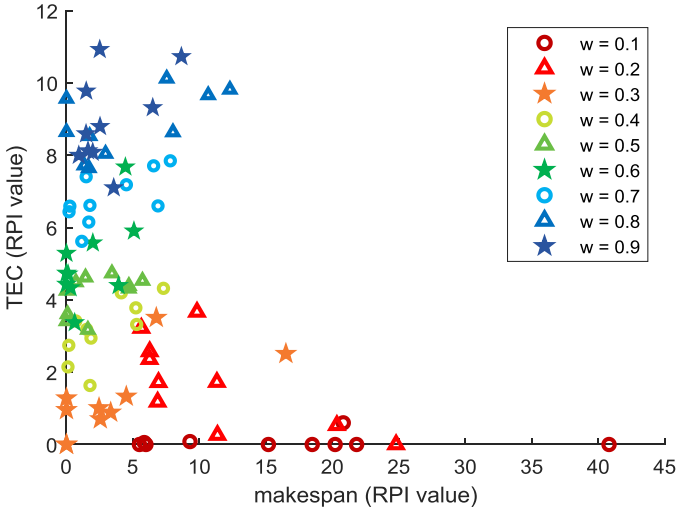


Fig. 13. Relationship between makespan and total energy consumption (TEC) for different weights.

out the EDA probability mechanism (EDA-VNS-NPM), and EDA-VNS with all components (EDA-VNS). All other components of the two compared algorithms were identical. Fig. 15(d) presents the means and 95% LSO intervals for the fitness values of the two compared methods. The *p*-value was smaller than 0.01; therefore, there was a significant different between the compared methods. The experimental results thus demonstrate that the EDA probability mechanism improved the performance of the proposed algorithm.

5.7. Discussion of weight in optimization

For the fitness calculation, we selected weight w ($w = 0.8$) to correlate two objectives (i.e., makespan and TEC) with fitness. However, the weight is determined by the decision-maker for balance between the objectives. In this section, we set nine weight value from 0.1 to 0.9 to test the effect on the makespan and TEC. The second type of instance was used in this experiment, and each instance was run independently 15 times.

Fig. 13 presents the RPI results of the experiment, where each point represents the average RPI value of the corresponding instance under the given weight. Each RPI value of the makespan and TEC was calculated by all the makespan values and TEC values in the same instance at different weights with Eq. (71), respectively. Fig. 13 reveals the following: (1) The weight can influence the balance of the optimization between the makespan and TEC. When w is small, the optimization prefers TEC, whereas when w is large, the optimization prefers makespan; and (2) The effect of weight for fitness is more allergic toward makespan than toward TEC.

5.8. Performance comparison with exact solver CPLEX

To evaluate the feasibility of the proposed MILP model, we employed an exact solver, IBM ILOG CPLEX 12.7, to calculate the model for 12 small-scale instances that were generated at random. In the exact solver, the maximum number of threads was 3, and the CPU runtime limit was set to 3 h for each run. The proposed algorithm was run 10 times for each instance, and the CPU runtime limit was set to 30 s for each run.

Table 4 presents the results of comparison between the proposed EDA-VNS algorithm (best fitness solutions) and the CPLEX solver. The second column provides the problem scale (where 3J2F2M signifies that the instance includes three jobs, two factories, and two machines in each factory). The best results collected by the two methods are provided in the third column. The weighted fitness values and RPI values for each instance are provided in the Fitness and RPI columns for EDA-VNS and CPLEX, respectively.

The following can be observed from Table 4: (1) EDA-VNS obtained higher quality solutions than the CPLEX solver; and (2) with more complex instances, the CPLEX solver demonstrated poorer performance.

5.9. Comparison with efficient algorithms

The compared algorithms include the EMA [40] proposed by Luo et al., the TS [2] proposed by Li et al., and the improved differential evolution simulated annealing (DE-SA) algorithm [37] proposed by Wu et al. The parameters of the three compared algorithms are set according to the corresponding studies, which are shown in Table 5, where L is the length of scheduling vector. For fairness, the stop criterions of all compared algorithms are the same as the proposed EDA-VNS.

The results of the proposed algorithm and compared algorithms are presented in Table 6, where the first column presents the number of instances, the second column presents the scale of the instance (where 20J2F3M signifies that the instance includes 20 jobs, two factories, and three machines in each factory), and the third column presents the best average fitness of the four algorithms. The fourth to seventh columns list the average fitness values over 30 independent runs of the compared algorithms and the proposed algorithm. Lastly, the 8th to 11th columns present the average RPI values over 30 independent runs of the four algorithms.

Table 6 reveals that EDA-VNS obtained 34 better values out of the given 36 instances, while the three compared algorithms demonstrated poor performance. The results of the average of all instances indicate that EDA-VNS had better performance than that of the compared algorithms. Fig. 14 (a–h) illustrate the convergence process for Instances 1, 7, 10, 16, 20, 26, 30 and 36. For every instance in Fig. 14, the convergence data were collected by an average of 30 runs according to the runtime for every algorithm. Fig. 15(e) presents the means and the 95% LSD intervals for the four algorithms. It can be seen that the proposed EDA-VNS algorithm demonstrated significantly better performance than the three compared algorithms.

For further comparison, the makespan and TEC of every run were statistically collected in an x-y-coordinate graph and analyzed as single-optimization parameters. The relationship between the makespan and

Table 4
Comparison between proposed algorithm (EDA-VNS) and CPLEX solver.

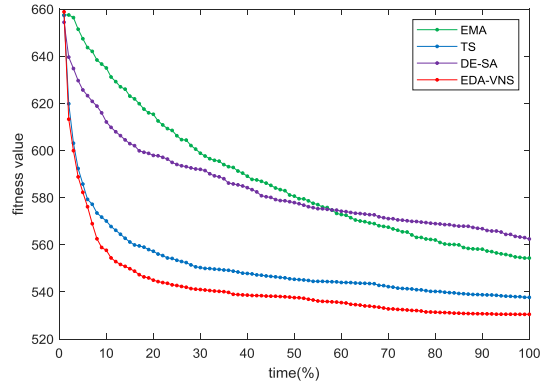
Instance	Scale	Best	Fitness		RPI	
			EDA-VNS (30 s)	CPLEX (3 h)	EDA-VNS (30 s)	CPLEX (3 h)
1	3J2F2M	63.44	63.44	63.44	**0.00**	**0.00**
2	3J2F3M	59.86	59.86	61.77	**0.00**	3.18
3	4J2F2M	77.91	77.91	78.78	**0.00**	1.12
4	4J2F3M	74.45	74.45	75.36	**0.00**	1.21
5	4J3F2M	78.00	78.00	78.27	**0.00**	0.34
6	4J3F3M	61.33	61.33	64.98	**0.00**	5.95
7	5J2F2M	94.85	94.85	96.71	**0.00**	1.97
8	5J2F3M	78.17	78.17	90.73	**0.00**	16.07
9	5J3F2M	81.91	81.91	94.95	**0.00**	15.92
10	5J3F3M	72.03	72.03	81.78	**0.00**	13.53
11	6J2F2M	111.21	111.21	129.49	**0.00**	16.44
12	6J2F3M	96.30	96.30	124.72	**0.00**	29.51
average	79.12	79.12	86.75	**0.00**	8.77	

Table 5
Parameters of compared algorithms.

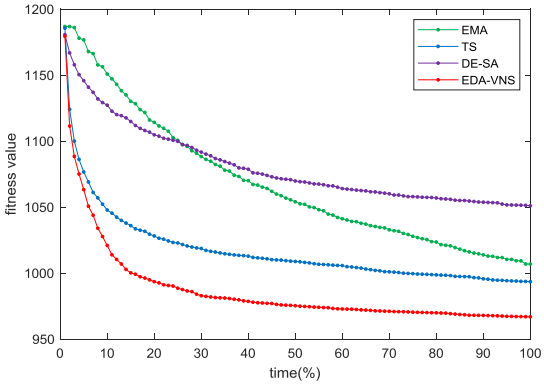
Parameters	EMA [40]	DE-SA [37]	TS [2]
size of population	150	100	100
crossover probability	0.7	randomly between 0.5 to 0.1	\
mutation probability	0.4	randomly between 0.5 to 0.1	\
population proportion of local search	0.15	0.25	\
initial temperature of SA	\	0.5	\
tabu tenure	\	\	$L/2$
tabu neighborhood size	\	\	from $L/5$ to L

Table 6
Performance comparison between EDA-VNS, EMA [40], DE-SA [37] and TS [2] to solve DFJSPC (average value over 30 runs).

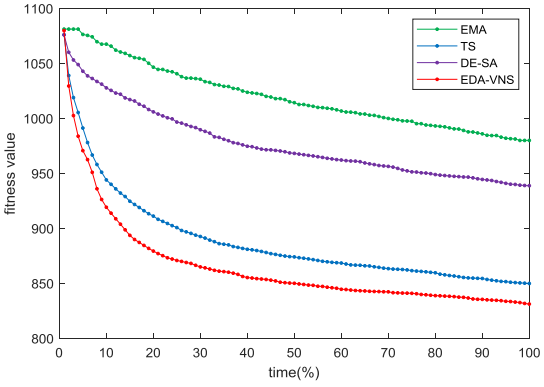
Instance	Scale	Best	Fitness				RPI			
			EMA [40]	DE-SA [37]	TS [2]	EDA-VNS	EMA [40]	DE-SA [37]	TS [2]	EDA-VNS
1	20J2F3M	530.47	554.34	562.29	537.49	530.47	4.50	6.00	1.32	**0.00**
2	20J2F5M	420.98	463.31	475.37	430.89	420.98	10.06	12.92	2.36	**0.00**
3	20J3F3M	417.56	431.69	438.33	418.91	417.56	3.38	4.98	0.32	**0.00**
4	20J3F5M	402.20	436.97	444.07	405.04	402.20	8.65	10.41	0.71	**0.00**
5	20J5F3M	414.05	454.79	454.73	414.05	423.98	9.84	9.82	**0.00**	2.40
6	20J5F5M	381.89	428.49	429.93	381.89	386.08	12.20	12.58	**0.00**	1.10
7	40J2F3M	966.75	1006.78	1050.59	992.98	966.75	4.14	8.67	2.71	**0.00**
8	40J2F5M	925.04	1052.70	1063.72	954.96	925.04	13.80	14.99	3.23	**0.00**
9	40J3F3M	833.51	912.02	895.39	845.10	833.51	9.42	7.42	1.39	**0.00**
10	40J3F5M	830.74	980.11	938.23	849.91	830.74	17.98	12.94	2.31	**0.00**
11	40J5F3M	807.05	923.42	885.41	819.51	807.05	14.42	9.71	1.54	**0.00**
12	40J5F5M	776.90	876.76	889.42	800.58	776.90	12.85	14.48	3.05	**0.00**
13	60J2F3M	1621.00	1674.53	1748.29	1661.44	1621.00	3.30	7.85	2.49	**0.00**
14	60J2F5M	1304.34	1477.16	1521.57	1381.38	1304.34	13.25	16.65	5.91	**0.00**
15	60J3F3M	1413.28	1512.38	1544.33	1465.46	1413.28	7.01	9.27	3.69	**0.00**
16	60J3F5M	1269.97	1434.09	1437.36	1331.80	1269.97	12.92	13.18	4.87	**0.00**
17	60J5F3M	1268.57	1390.08	1377.44	1289.36	1268.57	9.58	8.58	1.64	**0.00**
18	60J5F5M	1202.50	1407.40	1359.13	1227.73	1202.50	17.04	13.03	2.10	**0.00**
19	80J2F3M	2128.69	2352.94	2337.15	2195.04	2128.69	10.53	9.79	3.12	**0.00**
20	80J2F5M	1848.41	2158.49	2142.79	1922.70	1848.41	16.78	15.93	4.02	**0.00**
21	80J3F3M	1801.38	2019.36	1963.29	1831.26	1801.38	12.10	8.99	1.66	**0.00**
22	80J3F5M	1669.96	1903.56	1918.27	1723.63	1669.96	13.99	14.87	3.21	**0.00**
23	80J5F3M	1597.35	1751.13	1765.01	1620.98	1597.35	9.63	10.50	1.48	**0.00**
24	80J5F5M	1593.35	1834.21	1800.38	1597.02	1593.35	15.12	12.99	0.23	**0.00**
25	100J2F3M	2589.78	2874.43	2818.83	2649.58	2589.78	10.99	8.84	2.31	**0.00**
26	100J2F5M	2159.20	2605.58	2543.47	2287.09	2159.20	20.67	17.80	5.92	**0.00**
27	100J3F3M	2315.98	2560.05	2540.81	2363.24	2315.98	10.54	9.71	2.04	**0.00**
28	100J3F5M	2181.31	2545.85	2523.09	2249.97	2181.31	16.71	15.67	3.15	**0.00**
29	100J5F3M	2104.46	2339.59	2311.59	2134.09	2104.46	11.17	9.84	1.41	**0.00**
30	100J5F5M	1963.44	2278.51	2251.73	2046.33	1963.44	16.05	14.68	4.22	**0.00**
31	200J2F3M	5382.33	6031.15	6032.70	5516.23	5382.33	12.05	12.08	2.49	**0.00**
32	200J2F5M	4744.29	5588.01	5650.78	4955.44	4744.29	17.78	19.11	4.45	**0.00**
33	200J3F3M	4745.72	5267.80	5205.51	4771.90	4745.72	11.00	9.69	0.55	**0.00**
34	200J3F5M	4400.32	5245.83	5115.39	4556.62	4400.32	19.21	16.25	3.55	**0.00**
35	200J5F3M	4248.63	4749.86	4727.04	4318.31	4248.63	11.80	11.26	1.64	**0.00**
36	200J5F5M	4046.98	4791.70	4666.89	4300.56	4046.98	18.40	15.32	6.27	**0.00**
average	1869.68	2119.86	2106.40	1923.57	1870.07	12.19	11.86	2.54	**0.10**	



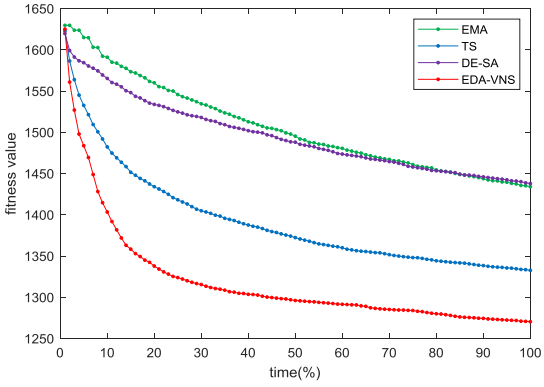
(a) Convergence curve for Instance 1



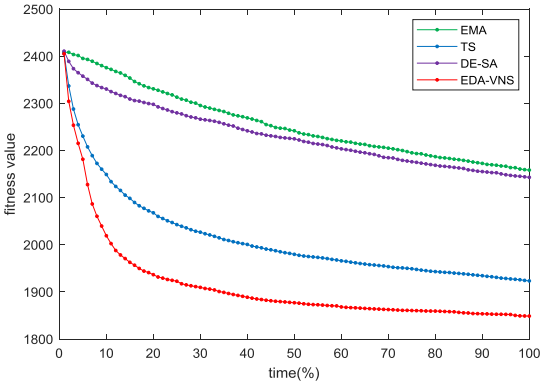
(b) Convergence curve for Instance 7



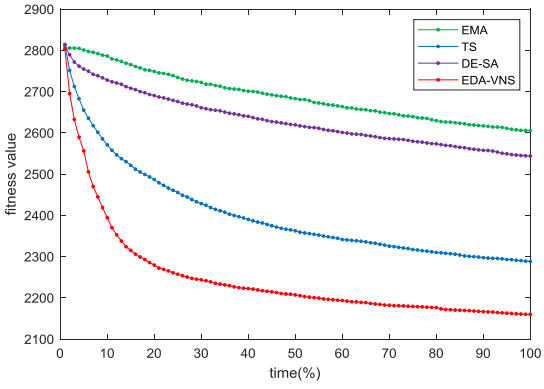
(c) Convergence curve for Instance 10



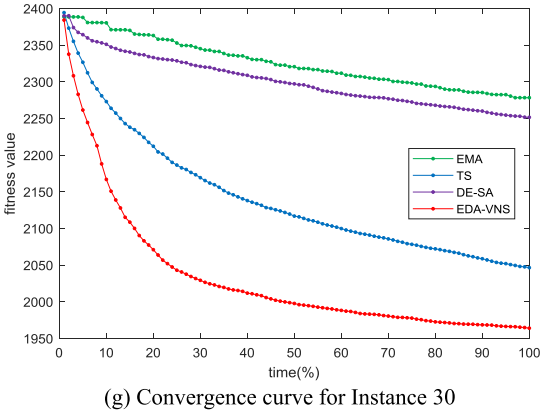
(d) Convergence curve for Instance 16



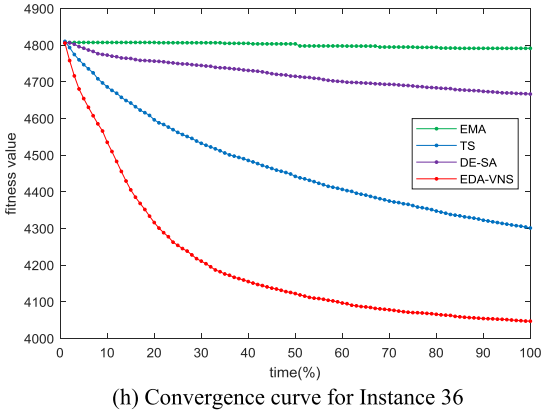
(e) Convergence curve for Instance 20



(f) Convergence curve for Instance 26

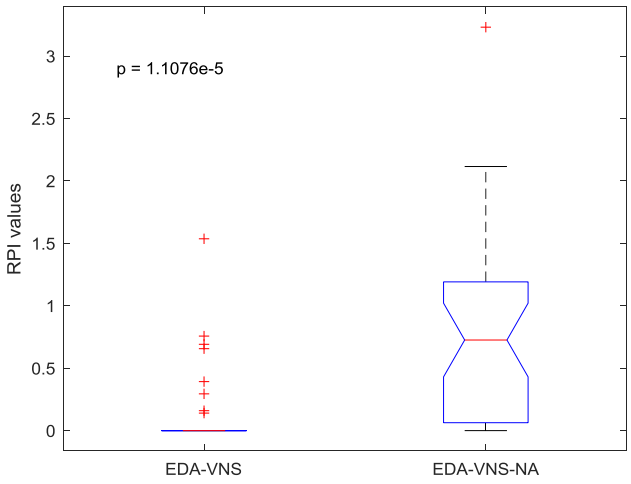


(g) Convergence curve for Instance 30

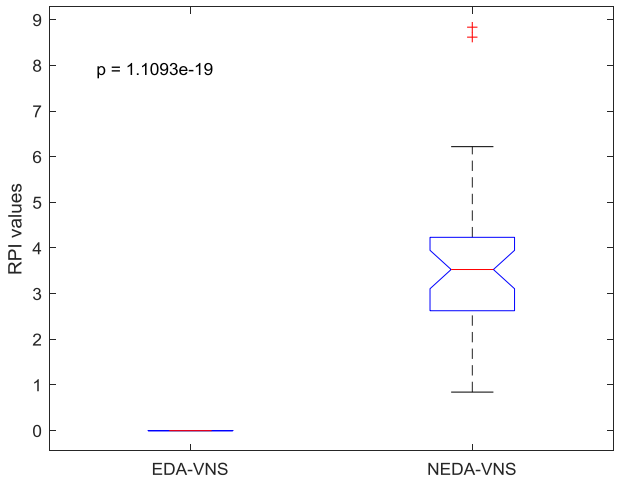


(h) Convergence curve for Instance 36

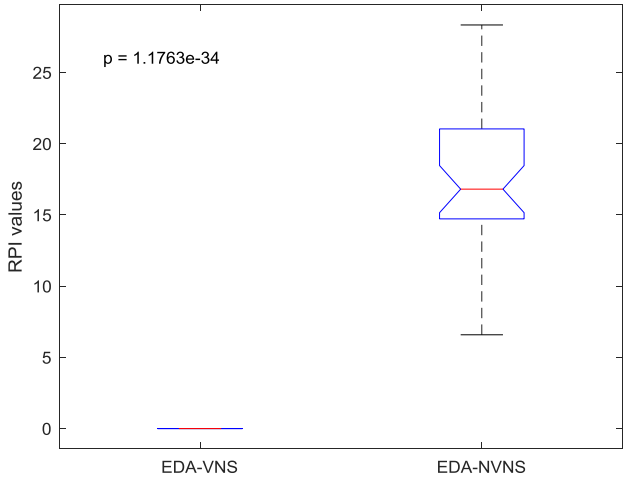
Fig. 14. Convergence curve comparison between EDA-VNS, EMA, DE-SA and TS to solve DFJSPC.



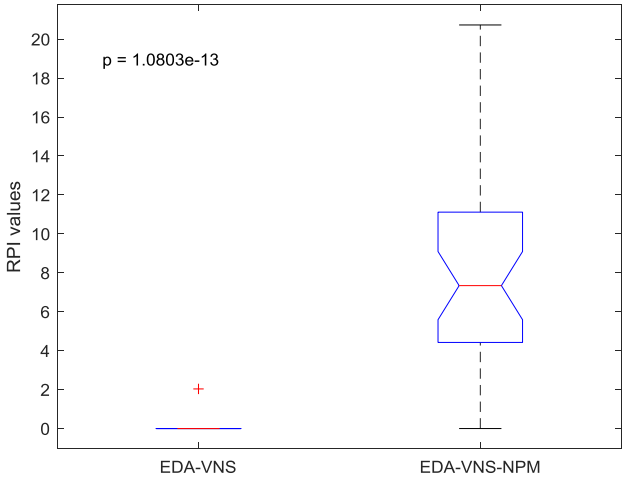
(a) Comparison of self-adaptive parameter process.



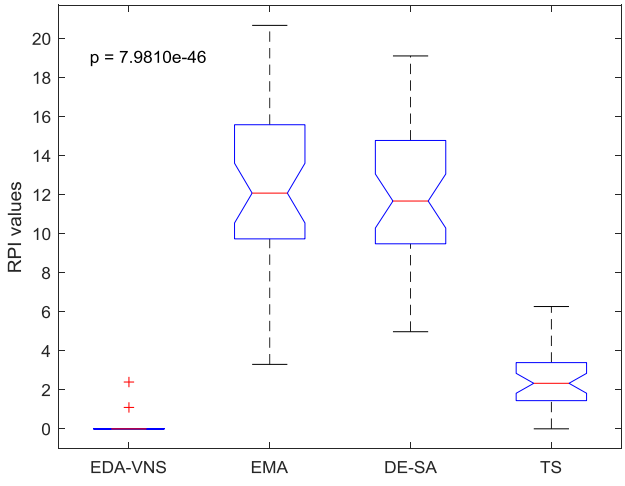
(b) Comparison of exploration ability



(c) Comparison of exploitation ability



(d) Comparison of EDA probability mechanism



(e) Analysis of variance (ANOVA) results for compared algorithms

Fig. 15. Comparison results for the proposed algorithm.

TEC of all 30 runs of Instance 1, 7, 10, 16, 20, 26, 30 and 36 of the first type of instances are presented in Fig. 16, which indicates that the results of the proposed algorithm are superior to those of other algorithms for both the makespan and TEC. It can be observed from Fig. 16 that, with the increase of instance scale, the optimization range of different algorithms has become more and more obvious; and the proposed algorithm (EDA-VNS) performs better than the compared algorithms.

Fig. 17 illustrates a feasible solution of a 20J3F5M scale instance optimized by the proposed algorithm. The *X*-*Y* in white rectangles represents the process time of operation $O_{X,\,Y}$.

5.10. Experiments analysis

According to the experimental results from aforementioned sections, the proposed EDA-VNS performed better than other compared algorithms. The main reasons can be concluded that: (1) the computational complexity analysis verified that the proposed EDA-VNS algorithm has acceptable iteration speed, and the EDA probability mechanism has lower computational complexity, which is calculated in Section 4.9; (2) the self-adaptive parameters heuristic prevented premature convergence, and convergence with stable speed can obtain better optimization, which is proved in Section 5.3; (3) the solutions generated by the EDA component is based on probability mechanism, making the solutions more diversified than other evolutionary algorithm that directly inherit the information from parent individuals, and the exploration ability of the EDA component in the proposed algorithm is proved in Section 5.4; (4) five neighborhood structures is organized by the VNS component, reducing more computing resources for more effective neighborhood structures, and the exploitation ability of the VNS component in the proposed algorithm is proved in Section 5.5; (5) the EDA probability mechanism improved the calculation efficiency of the proposed algorithm, which is proved in Section 5.6; (6) the weight discussion in a spectrum of 0.1∼0.9 indicates the relationships between the makespan and TEC under different weight influence, which is shown in Section 5.7; and (7) the comparison experiment with the global-optimization solver CPLEX and other effective algorithms verified the performance of the proposed algorithm in Sections 5.8 and 5.9.

6. Conclusion

In this study, we propose the hybrid EDA-VNS algorithm by modifying the EDA and VNS components and introducing new operations to be applicable to the DFJSPC. The EDA component is controlled by a probability variable for reducing the calculation time of an iteration, and parameters are set to be self-adaptive to prevent premature convergence. The population is evolved by the EDA component owing to its exploration ability, and VNS is applied owing to its exploitation ability. In the VNS component, five neighborhood structures, including global and local strategies, are organized to obtain optimized solutions. We compared the performance of the proposed algorithm with instances with three other algorithms. Detailed experimental analysis confirmed the reliability and robustness of the proposed EDA-VNS.

In future research, we will investigate other scheduling methods for crane transportations in various situations. We plan to achieve this by: (1) considering multi-crane scheduling in a factory under the objective of minimizing energy consumption; (2) considering Pareto-based multi-optimization methods to obtain improved solutions; (3) considering distributed scheduling problems with assembly process; (4) considering other realistic applications of green scheduling to improve economic benefits; and (5) where considering adaptive neighborhood structures switch strategy to enhance the exploitation ability, parameters and indices including machine makespan, factory makespan, and machine idle time will be concerned.

Author contributions

1. Yu Du's contributions are as follows: (1) code the compared algorithms and test the simulation; (2) design several heuristics for the proposed algorithm; (3) test all the compared algorithms; and (4) analyzed the algorithm's performance.

2. Jun-qing Li's contributions are as follows: (1) propose estimation of distribution algorithm and variable neighborhood search for the considered problem; (2) design several heuristics for the problem; (3) design the simulation tests; and (4) make a detailed analysis for the results.

3. Chao Luo's contributions are as follows: (1) analyze the problem features and proposed problem-specific heuristic; and (2) design the algorithm parameter and test them.

4. Lei-lei Meng's contributions are as follows: (1) verify the mathematical model; and (2) give a carefully check for the grammar in the paper.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is partially supported by National Science Foundation of China under Grant 61773192, 61803192, and 61773246, and major Program of Shandong Province Natural Science Foundation (ZR2018ZB0419).

Appendix

This section gives an example to calculate the fitness of a solution. The data of the example are designed for illustration. The considered solution includes a factory vector {2, 1, 2}, a scheduling vector {2, 3, 2, 1, 3, 1}, and a machine assignment vector {2, 3, 2, 3, 4, 4}. The job weight and machine process time of this example are shown in Table 7, while the parameters of machines are shown in Table 8.

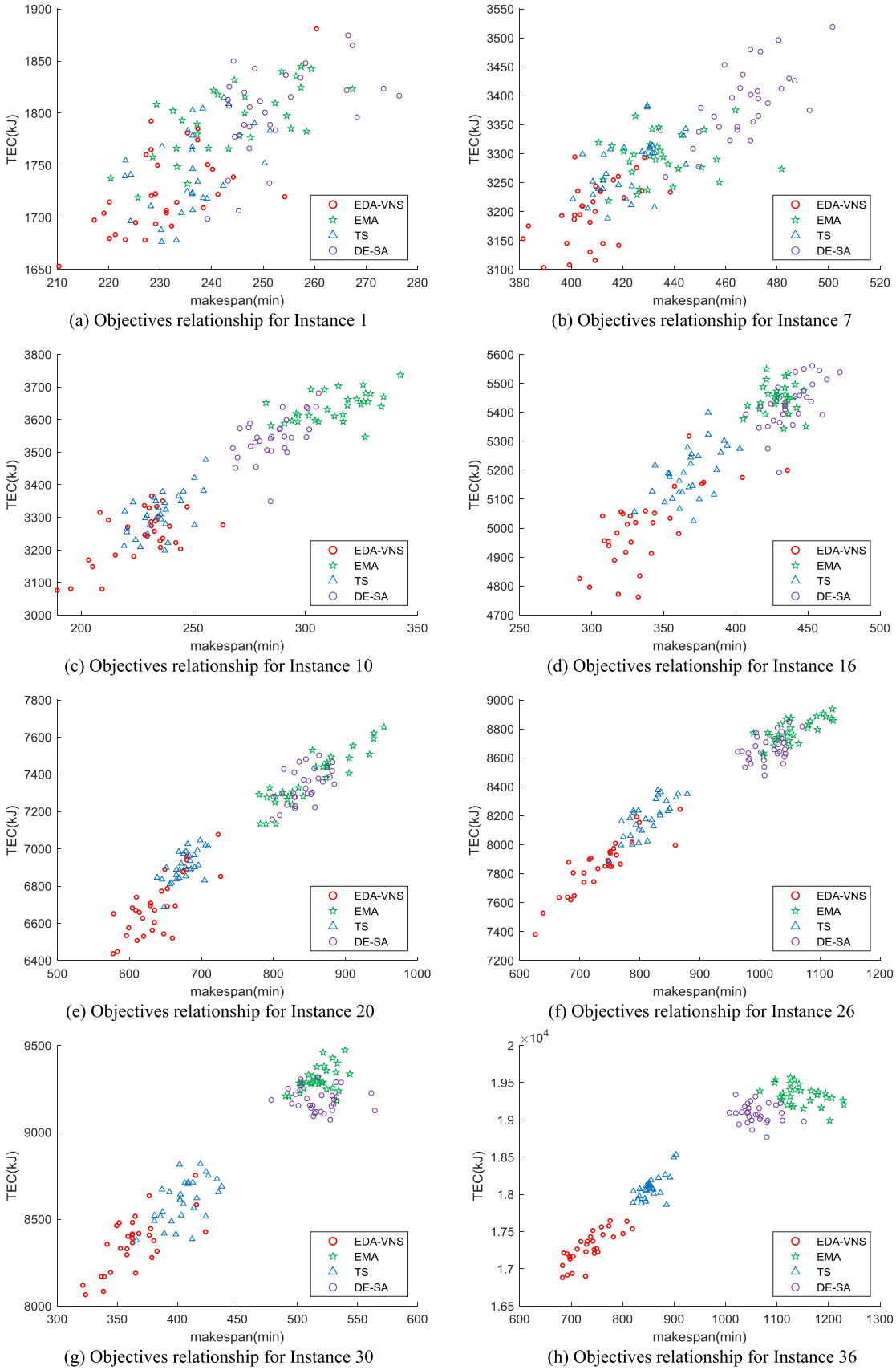
The calculation of fitness is based on Algorithm 1. According to the factory vector, job 2 is distributed to factory 1, jobs 1 and 3 are distributed to factory 2. Therefore, in factory 1, the operation sequence and the corresponding machine assignment are $\{O_{2,1}, O_{2,2}\}$ and $\{M_2, M_2\}$, respectively; in factory 2, the operation sequence and the corresponding machine assignment are $\{O_{3,1}, O_{1,1}, O_{3,2}, O_{1,2}\}$ and $\{M_3, M_3,$

Table 7
Machine process time of 4J2F2M.

Operation	Job weight W_j (kg)	Factory 1 $T_{i,j}$ (min)		Factory 2 $T_{i,j}$ (min)	
		M_1	M_2	M_3	M_4
$O_{1,1}$	2100	8	6	**5**	10
$O_{1,2}$	2100	5	12	8	**6**
$O_{2,1}$	1510	–	**7**	8	10
$O_{2,2}$	1510	6	**4**	15	–
$O_{3,1}$	1100	–	10	**5**	8
$O_{3,2}$	1100	8	7	8	**6**

Table 8
Parameters of machines.

Machine	Operating power (W)	Location (m)	
		x position	*y* position
M_1	1100	20	20
M_2	1350	20	60
M_3	1200	160	20
M_4	1050	160	60



(a) Objectives relationship for Instance 1

(b) Objectives relationship for Instance 7

(c) Objectives relationship for Instance 10

(d) Objectives relationship for Instance 16

(e) Objectives relationship for Instance 20

(f) Objectives relationship for Instance 26

(g) Objectives relationship for Instance 30

(h) Objectives relationship for Instance 36

Fig. 16. Relationship between makespan and TEC in proposed algorithm and compared algorithms.

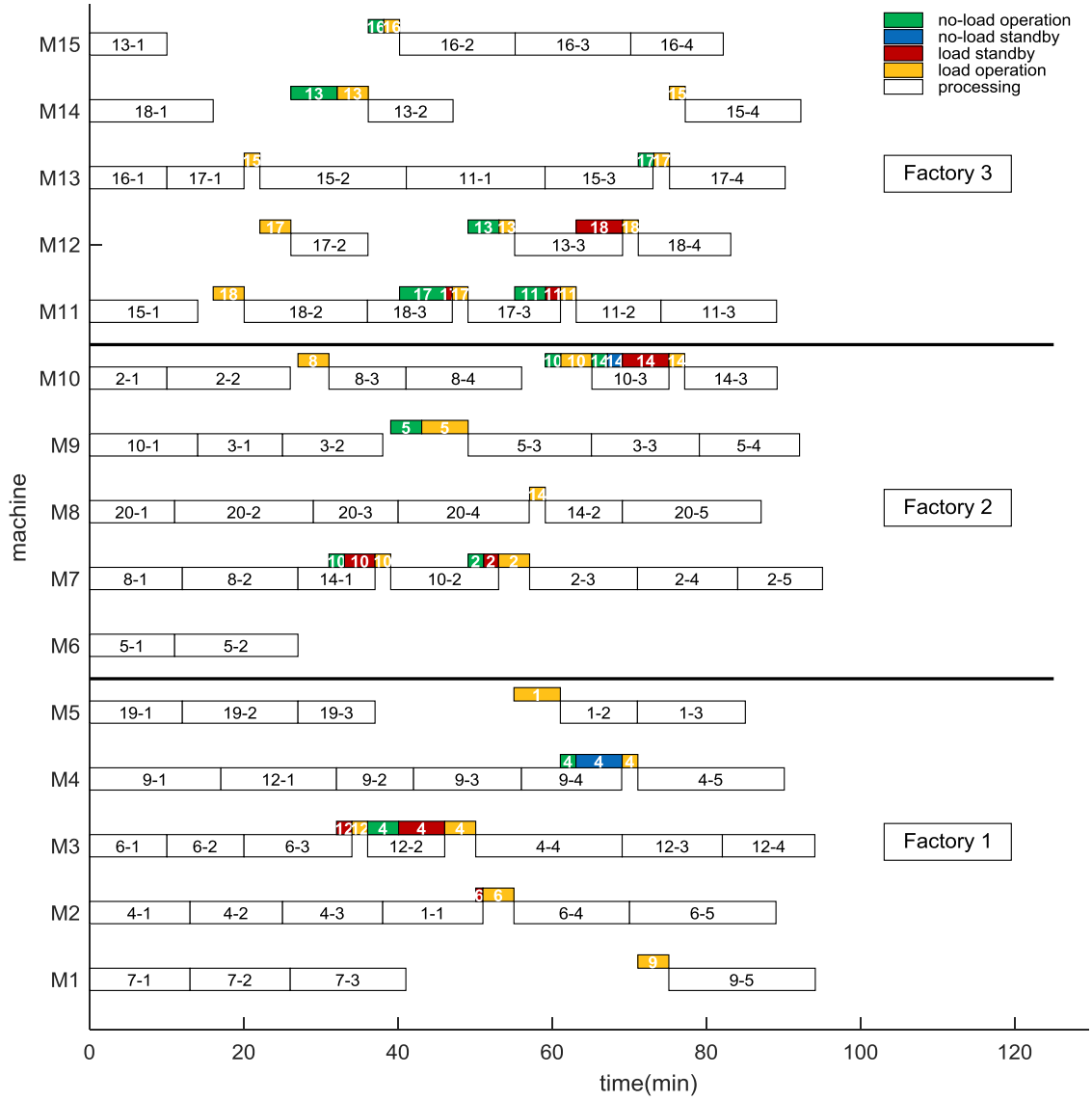


Fig. 17. Gantt chart for 20J3F5M scale instance optimized by proposed algorithm (EDA-VNS) (22nd run result).

Table 9
Calculation of machine process energy consumption.

Operation	Assigned machine	$E_{i,j}$ (W)	$T_{i,j}$(min)	$E_{mp}(O_{i,j}) = E_{i,j} * T_{i,j}$
$O_{1,1}$	M_3	1200	5	1200•5•60 / 1000 = 360 (kJ)
$O_{1,2}$	M_4	1050	6	1050•6•60 / 1000 = 378 (kJ)
$O_{2,1}$	M_2	1350	7	1350•7•60 / 1000 = 567 (kJ)
$O_{2,2}$	M_1	1100	4	1100•4•60 / 1000 = 264 (kJ)
$O_{3,1}$	M_3	1200	5	1200•5•60 / 1000 = 360 (kJ)
$O_{3,2}$	M_4	1050	6	1050•6•60 / 1000 = 378 (kJ)

M_4, M_4}, respectively. According to Eq. (1), the $E_{mp}(O_{i,j})$ of all operations are calculated in Table 9.

In factory 1, $O_{2,1}$ is the first operation of the factory, according to assumption (10), $O_{2,1}$ need not any crane transportation; therefore, $O_{2,1}$ only has machine process. As for $O_{2,2}$, according to Fig. 1, the operation position and same-job predecessor are identical, the crane does not do anything, so $O_{2,2}$ only has machine process.

In factory 2, *Ws, Wx, Wy*, and *Qn* are 900 kg, 9700 kg, 5500 kg, and 10,000 kg, respectively; P_{xs}, P_{xd}, P_{ys}, P_{yd}, and *Ps* are 20,000 W, 8000 W, 12,500 W, 5000 W, and 750 W respectively; V_x and V_y are 50 m/min and 30 m/min, respectively; a_{xs}, a_{xb}, a_{ys}, and a_{yb} are 0.5 m/s^2, 0.8 m/s^2, 0.4 m/s^2, and 0.6 m/s^2, respectively.

$O_{3,1}$ and $O_{1,1}$ are the first operation of jobs 3 and 1, respectively, according to assumption (10), the two operations do not need crane transportation. According to assumption (8), the initial position of crane is at M_3. Next, for $O_{3,2}$, the crane should take job 3 from M_3 to M_4, which only requires load operation by crane transportation. According to Eqs. (17) – (26), the time and energy consumption of load operation of $O_{3,2}$ is calculated in (72) – (81).

$$c = (Ws + Wj + Wx + Wy)/(Qn + Wx + Wy)$$
$$= (900 + 1100 + 9700 + 5500)/(10000 + 9700 + 5500) = 0.6825 \quad (72)$$

$$T_{xlos,2}(O_{3,2}) = c \cdot Vx/a_{xs} = 0.6825 \cdot 50/(0.5 \cdot 60^2) = 0.0190(\text{min}) \quad (73)$$

$$T_{xlob,2}(O_{3,2}) = c \cdot Vx/a_{xb} = 0.6825 \cdot 50/(0.8 \cdot 60^2) = 0.0118(\text{min}) \quad (74)$$

$$\begin{aligned} T_{xlod,2}(O_{3,2}) &= \max\left\{\frac{\left|x_{op}-x_{pp}\right|}{Vx} - \frac{Vx}{2}\cdot c\cdot\left(\frac{1}{a_{xs}}+\frac{1}{a_{xb}}\right),0\right\} \\ &= \max\left\{\frac{|160-160|}{50} - \frac{50}{2}\cdot 0.6825\cdot\left(\frac{1}{0.5\cdot 60^2}+\frac{1}{0.8\cdot 60^2}\right),0\right\} = 0(\text{min}) \end{aligned} \quad (75)$$

$$\begin{aligned} d &= (Ws + Wj + Wy)/(Qn + Wy) \\ &= (900 + 1100 + 5500)/(10000 + 5500) = 0.4839 \end{aligned} \quad (76)$$

$$T_{ylos,2}(O_{3,2}) = d \cdot Vy/a_{ys} = 0.4839 \cdot 30/(0.4 \cdot 60^2) = 0.0101(\text{min}) \quad (77)$$

$$T_{ylob,2}(O_{3,2}) = d \cdot Vy/a_{yb} = 0.4839 \cdot 30/(0.6 \cdot 60^2) = 0.0067(\text{min}) \quad (78)$$

$$\begin{aligned} T_{ylod,2}(O_{3,2}) &= \max\left\{\frac{\left|y_{op}-y_{pp}\right|}{Vy} - \frac{Vy}{2}\cdot d\cdot\left(\frac{1}{a_{ys}}+\frac{1}{a_{yb}}\right),0\right\} \\ &= \max\left\{\frac{|60-20|}{30} - \frac{30}{2}\cdot 0.4839\cdot\left(\frac{1}{0.4\cdot 60^2}+\frac{1}{0.6\cdot 60^2}\right),0\right\} = 1.3249(\text{min}) \end{aligned} \quad (79)$$

$$\begin{aligned} T_{lo,2}(O_{3,2}) &= T_{xlos,2}(O_{3,2}) + T_{xlob,2}(O_{3,2}) + T_{xlod,2}(O_{3,2}) + T_{ylos,2}(O_{3,2}) \\ &\quad + T_{ylob,2}(O_{3,2}) + T_{ylod,2}(O_{3,2}) \\ &= 0.0190 + 0.0118 + 0 + 0.0101 + 0.0067 + 1.3249 = 1.3725(\text{min}) \end{aligned} \quad (80)$$

$$\begin{aligned} E_{lo,2}(O_{3,2}) &= T_{xlos,2}(O_{3,2})\cdot P_{xs} + T_{ylos,2}(O_{3,2})\cdot P_{ys} \\ &\quad + (Ws + Wj)/Qn\cdot\left(T_{xlod,2}(O_{3,2})\cdot P_{xd} + T_{ylod,2}(O_{3,2})\cdot P_{yd}\right) \\ &= \left[0.0190\cdot 20000 + 0.0101\cdot 12500 + (900+1100)/10000\right. \\ &\quad \left.\cdot(0\cdot 8000 + 1.3249\cdot 5000)\right]\cdot 60/1000 = 109.8078(\text{kJ}) \end{aligned} \quad (81)$$

Then, for the last operation $O_{1,2}$ in factory 2, the crane is at M_4, according to Fig. 1, four crane transportation conditions, i.e., no-load operation, no-load standby, load standby, and load operation are needed. According to equations (3) – (12), the time and energy consumption of no-load operation of $O_{1,2}$ is calculated in (82) – (91).

$$\begin{aligned} a &= (Ws + Wx + Wy)/(Qn + Wx + Wy) \\ &= (900 + 9700 + 5500)/(10000 + 9700 + 5500) = 0.6389 \end{aligned} \quad (82)$$

$$T_{xnos,2}(O_{1,2}) = a \cdot Vx/a_{xs} = 0.6389 \cdot 50/(0.5 \cdot 60^2) = 0.0177 \quad (83)$$

$$T_{xnob,2}(O_{1,2}) = a \cdot Vx/a_{xb} = 0.6389 \cdot 50/(0.8 \cdot 60^2) = 0.0111 \quad (84)$$

$$\begin{aligned} T_{xnod,2}(O_{1,2}) &= \max\left\{\frac{\left|x_{pp}-x_{cp}\right|}{Vx} - \frac{Vx}{2}\cdot a\cdot\left(\frac{1}{a_{xs}}+\frac{1}{a_{xb}}\right),0\right\} \\ &= \max\left\{\frac{|160-160|}{50} - \frac{50}{2}\cdot 0.6389\cdot\left(\frac{1}{0.5\cdot 60^2}+\frac{1}{0.8\cdot 60^2}\right),0\right\} = 0(\text{min}) \end{aligned} \quad (85)$$

$$\begin{aligned} b &= (Ws + Wy)/(Qn + Wy) = (900 + 5500)/(10000 + 5500) = 0.4129 \end{aligned} \quad (86)$$

$$T_{ynos,2}(O_{1,2}) = b \cdot Vy/a_{ys} = 0.4129 \cdot 30/(0.4 \cdot 60^2) = 0.0086 \quad (87)$$

$$T_{ynob,2}(O_{1,2}) = b \cdot Vy/a_{yb} = 0.4129 \cdot 30/(0.6 \cdot 60^2) = 0.0057 \quad (88)$$

$$\begin{aligned} T_{ynod,2}(O_{1,2}) &= \max\left\{\frac{\left|y_{pp}-y_{cp}\right|}{Vy} - \frac{Vy}{2}\cdot b\cdot\left(\frac{1}{a_{ys}}+\frac{1}{a_{yb}}\right),0\right\} \\ &= \max\left\{\frac{|20-60|}{30} - \frac{30}{2}\cdot 0.4129\cdot\left(\frac{1}{0.4\cdot 60^2}+\frac{1}{0.6\cdot 60^2}\right),0\right\} = 1.3262(\text{min}) \end{aligned} \quad (89)$$

$$\begin{aligned} T_{no,2}(O_{1,2}) &= T_{xnos,2}(O_{1,2}) + T_{xnob,2}(O_{1,2}) + T_{xnod,2}(O_{1,2}) + T_{ynos,2}(O_{1,2}) \\ &\quad + T_{ynob,2}(O_{1,2}) + T_{ynod,2}(O_{1,2}) \\ &= 0.0177 + 0.0111 + 0 + 0.0086 + 0.0057 + 1.3262 = 1.3693(\text{min}) \end{aligned} \quad (90)$$

$$\begin{aligned} E_{no,2}(O_{1,2}) &= T_{xnos,2}(O_{1,2})\cdot P_{xs} + T_{ynos,2}(O_{1,2})\cdot P_{ys} \\ &\quad + Ws/Qn\cdot\left(T_{xnod,2}(O_{1,2})\cdot P_{xd} + T_{ynod,2}(O_{1,2})\cdot P_{yd}\right) \\ &= \left[0.0177\cdot 20000 + 0.0086\cdot 12500 + 900/10000\cdot(0\cdot 20000\right. \\ &\quad \left.+1.3262\cdot 5000)\right]\cdot 60/1000 = 63.5544(\text{kJ}) \end{aligned} \quad (91)$$

According to Eqs. (13) and (14), the time and energy consumption of no-load standby of $O_{1,2}$ is calculated in (92) and (93).

$$\begin{aligned} T_{ns,2}(O_{1,2}) &= \max\{Tc(O_{1,1}) - Tc_{no,2}(O_{1,2}), 0\} \\ &= \max\{10 - 7.7418, 0\} = 2.2582(\text{min}) \end{aligned} \quad (92)$$

$$E_{ns,2}(O_{1,2}) = T_{ns,2}(O_{1,2})\cdot Ps = 2.2582\cdot 750\cdot 60/1000 = 101.6190(\text{kJ}) \quad (93)$$

According to Eqs. (15) and (16), the time and energy consumption of load standby of $O_{1,2}$ is calculated in (94) and (95). Specifically, in (94), $O_{1,2}$ is the immediate successor operation of $O_{1,2}$ being performed on the same machine.

$$\begin{aligned} T_{ls,2}(O_{1,2}) &= \max\{Tc(O_{3,2}) - Tc_{ns,2}(O_{1,2}), 0\} \\ &= \max\{12.3725 - 10, 0\} = 2.3725(\text{min}) \end{aligned} \quad (94)$$

$$E_{ls,2}(O_{1,2}) = T_{ls,2}(O_{1,2})\cdot Ps = 1.0032\cdot 750\cdot 60/1000 = 45.1440(\text{kJ}) \quad (95)$$

According to Eqs. (17) – (26), the time and energy consumption of load operation of $O_{1,2}$ is calculated in (96) – (105).

$$\begin{aligned} c &= (Ws + Wj + Wx + Wy)/(Qn + Wx + Wy) \\ &= (900 + 2100 + 9700 + 5500)/(10000 + 9700 + 5500) = 0.7222 \end{aligned} \quad (96)$$

$$T_{xlos,2}(O_{1,2}) = c \cdot Vx/a_{xs} = 0.7222 \cdot 50/(0.5 \cdot 60^2) = 0.0201(\text{min}) \quad (97)$$

$$T_{xlob,2}(O_{1,2}) = c \cdot Vx/a_{xb} = 0.7222 \cdot 50/(0.8 \cdot 60^2) = 0.0125(\text{min}) \quad (98)$$

$$\begin{aligned} T_{xlod,2}(O_{1,2}) &= \max\left\{\frac{\left|x_{op}-x_{pp}\right|}{Vx} - \frac{Vx}{2}\cdot c\cdot\left(\frac{1}{a_{xs}}+\frac{1}{a_{xb}}\right),0\right\} \\ &= \max\left\{\frac{|160-160|}{50} - \frac{50}{2}\cdot 0.7222\cdot\left(\frac{1}{0.5\cdot 60^2}+\frac{1}{0.8\cdot 60^2}\right),0\right\} = 0(\text{min}) \end{aligned} \quad (99)$$

Table 10
start time and completion time of all operations in machine process and crane transportation (unit: min).

Operation	$Ts_{no,f}(O_{i,j})$	$Tc_{no,f}(O_{i,j})$	$Ts_{ns,f}(O_{i,j})$	$Tc_{ns,f}(O_{i,j})$	$Ts_{ls,f}(O_{i,j})$	$Tc_{ls,f}(O_{i,j})$	$Ts_{lo,f}(O_{i,j})$	$Tc_{lo,f}(O_{i,j})$	$Ts\ (O_{i,j})$	$Tc\ (O_{i,j})$
$O_{2,1}$	–	–	–	–	–	–	–	–	0	7
$O_{2,2}$	–	–	–	–	–	–	–	–	7	11
$O_{3,1}$	–	–	–	–	–	–	–	–	0	5
$O_{1,1}$	–	–	–	–	–	–	–	–	5	10
$O_{3,2}$	–	–	–	–	–	–	5	6.3725	6.3725	12.3725
$O_{1,2}$	6.3725	7.7418	7.7418	10	10	12.3725	12.3725	13.7480	13.7480	19.7480

$$d = (Ws + Wj + Wy)/(Qn + Wy) = (900 + 2100 + 5500)/(10000 + 5500) = 0.5484 \quad (100)$$

$$T_{ylos,2}(O_{1,2}) = d \cdot Vy/a_{ys} = 0.5484 \cdot 30/(0.4 \cdot 60^2) = 0.0114(\text{min}) \quad (101)$$

$$T_{ylob,2}(O_{1,2}) = d \cdot Vy/a_{yb} = 0.5484 \cdot 30/(0.6 \cdot 60^2) = 0.0076(\text{min}) \quad (102)$$

$$T_{ylod,2}(O_{1,2}) = \max\left\{\frac{|y_{op} - y_{pp}|}{Vy} - \frac{Vy}{2} \cdot d \cdot \left(\frac{1}{a_{ys}} + \frac{1}{a_{yb}}\right), 0\right\} = \max\left\{\frac{|60 - 20|}{30} - \frac{30}{2} \cdot 0.5484 \cdot \left(\frac{1}{0.4 \cdot 60^2} + \frac{1}{0.6 \cdot 60^2}\right), 0\right\} = 1.3238(\text{min}) \quad (103)$$

$$T_{lo,2}(O_{1,2}) = T_{xlos,2}(O_{1,2}) + T_{xlob,2}(O_{1,2}) + T_{xlod,2}(O_{1,2}) + T_{ylos,2}(O_{1,2}) + T_{ylob,2}(O_{1,2}) + T_{ylod,2}(O_{1,2}) = 0.0201 + 0.0125 + 0 + 0.0114 + 0.0076 + 1.3238 = 1.3755(\text{min}) \quad (104)$$

$$E_{lo,2}(O_{1,2}) = T_{xlos,2}(O_{1,2}) \cdot P_{xs} + T_{ylos,2}(O_{1,2}) \cdot P_{ys} + (Ws + Wj)/Qn \cdot \left(T_{xlod,2}(O_{1,2}) \cdot P_{xd} + T_{ylod,2}(O_{1,2}) \cdot P_{yd}\right) = [0.0201 \cdot 20000 + 0.0114 \cdot 12500 + (900 + 2100)/10000 \cdot (0 \cdot 8000 + 1.3238 \cdot 5000)] \cdot 60/1000 = 151.7858(\text{kJ}) \quad (105)$$

According to the time of machine process and crane transportation, the start and completion of these stages can be obtained. The start time and completion time of all operations in machine process and crane transportation are listed in Table 10, while the corresponding Gantt chart is demonstrated in Fig. 18.

From Table 10 and Fig. 18, the makespan of this example is 19.7480 min. According to Eqs. (2), (27) and (28) The TEC of this example is calculated as follow.

$$TEC_{mp} = \sum_{i=1}^{I}\sum_{j=1}^{J} E_{mp}(O_{i,j}) = 360 + 378 + 567 + 264 + 360 + 378 = 2307(\text{kJ}) \quad (106)$$

$$TEC_{ct} = E_{no} + E_{ns} + E_{ls} + E_{lo}) = E_{no,2}(O_{1,2}) + E_{ns,2}(O_{1,2}) + E_{ls,2}(O_{1,2}) + E_{lo,2}(O_{1,2}) + E_{lo,2}(O_{3,2}) = 63.5544 + 101.6190 + 45.1440 + 151.7858 + 109.8078) = 471.9110(\text{kJ}) \quad (107)$$

$$TEC = TEC_{mp} + TEC_{ct} = 2307 + 471.9110 = 2778.9110(\text{kJ}) \quad (108)$$

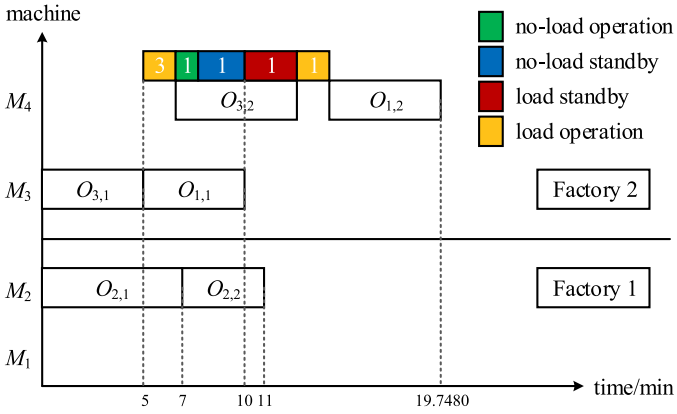


Fig. 18. Gantt chart for the example.

The fitness of the example can be calculated based on Eq. (29), which is shown as follow. In this study, w is 0.8.

$$f = w \cdot f_1 + (1 - w) \cdot f_2 = 0.8 \cdot 19.7480 + (1 - 0.8) \cdot 2778.9110 = 571.5806 \quad (109)$$

Therefore, the fitness of the example is 571.5806.

References

[1] Y. Li, X. Li, L. Gao, B. Zhang, Q.K. Pan, M.F. Tasgetiren, L. Meng, A discrete artificial bee colony algorithm for distributed hybrid flowshop scheduling problem with sequence-dependent setup times, Int. J. Prod. Res. (2020) 1–20, doi:10.1080/00207543.2020.1753897.

[2] J. Li, P. Duan, J. Cao, X. Lin, Y. Han, A hybrid pareto-based tabu search for the distributed flexible job shop scheduling problem with E/T criteria, IEEE Access 6 (2018) 58883–58897, doi:10.1109/ACCESS.2018.2873401.

[3] Z. Liu, S. Guo, L. Wang, Integrated green scheduling optimization of flexible job shop and crane transportation considering comprehensive energy consumption, J Clean. Prod. 211 (2019) 765–786, doi:10.1016/j.jclepro.2018.11.231.

[4] J.Q. Li, Q.K. Pan, M.F. Tasgetiren, A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities, Appl. Math. Model. 38 (3) (2014) 1111–1132, doi:10.1016/j.apm.2013.07.038.

[5] B. Naderi, R. Ruiz, The distributed permutation flowshop scheduling problem, Comput. Oper. Res. 37 (4) (2010) 754–768, doi:10.1016/j.cor.2009.06.019.

[6] S.Y. Wang, L. Wang, M. Liu, Y. Xu, An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem, Int. J. Prod. Econ. 145 (1) (2013) 387–396, doi:10.1016/j.ijpe.2013.05.004.

[7] K. Wang, Y. Huang, H. Qin, A fuzzy logic-based hybrid estimation of distribution algorithm for distributed permutation flowshop scheduling problems under machine breakdown, J. Oper. Res. Soc. 67 (1) (2016) 68–82, doi:10.1057/jors.2015.50.

[8] J. Deng, L. Wang, A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem, Swarm. Evol. Comput. 32 (2017) 121–131, doi:10.1016/j.swevo.2016.06.002.

[9] J.Q. Li, S.C. Bai, P.Y. Duan, H.Y. Sang, Y.Y. Han, Z.X. Zheng, An improved artificial bee colony algorithm for addressing distributed flow shop with distance coefficient in a prefabricated system, Int. J. Prod. Res. 57 (22) (2019) 6922–6942, doi:10.1080/00207543.2019.1571687.

[10] T. Meng, Q.K. Pan, L. Wang, A distributed permutation flowshop scheduling problem with the customer order constraint, Knowl. based Syst. 184 (2019) 104894, doi:10.1016/j.knosys.2019.104894.

[11] R. Ruiz, Q.K. Pan, B. Naderi, Iterated Greedy methods for the distributed permutation flowshop scheduling problem, Omega (Westport) 83 (2019) 213–222, doi:10.1016/j.omega.2018.03.004.

[12] J.J. Wang, L. Wang, A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop, IEEE Trans. Syst., Man, Cybern.: Syst. (99) (2018) 1–15, doi:10.1109/TSMC.2017.2788879.

[13] J.Q. Li, M.X. Song, L. Wang, P.Y. Duan, Y.Y. Han, H.Y. Sang, Q.K. Pan, Hybrid artificial bee colony algorithm for a parallel batching distributed flow-shop problem with deteriorating jobs, IEEE Trans. Cybern. 50 (6) (2020) 2425–2439, doi:10.1109/TCYB.2019.2943606.
[14] A.P. Rifai, H. Nguyen, S.Z. Dawal, Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling, Appl. Soft Comput. 40 (2016) 42–57, doi:10.1016/j.asoc.2015.11.034.
[15] W. Shao, D. Pi, Z. Shao, 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. 16 (3) (2019) 1344–1360, doi:10.1109/TASE.2018.2886303.
[16] S.Y. Wang, L. Wang, An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem, IEEE Trans. Syst., Man, Cybern.: Syst. 46 (1) (2016) 139–149, doi:10.1109/TSMC.2015.2416127.
[17] Q.K. Pan, L. Gao, L. Xinyu, F.M. Jose, Effective constructive heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem, Appl. Soft Comput. 81 (2019) 105492, doi:10.1016/j.asoc.2019.105492.
[18] Q.K. Pan, L. Gao, L. Wang, J. Liang, X.Y. Li, Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem, Expert Syst. Appl. 124 (2019) 309–324, doi:10.1016/j.eswa.2019.01.062.
[19] D. Ferone, S. Hatami, E.M. Gonzalezneira, A.A. Juan, P. Festa, A biased-randomized iterated local search for the distributed assembly permutation flow-shop problem, Int. Trans. Oper. Res. 27 (3) (2020) 1368–1391, doi:10.1111/itor.12719.
[20] H. Sang, Q. Pan, J. Li, P. Wang, Y. Han, K. Gao, P. Duan, Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion, Swarm Evol. Comput. 44 (2019) 64–73, doi:10.1016/j.swevo.2018.12.001.
[21] Q.K. Pan, L. Gao, L. Wang, An effective cooperative co-evolutionary algorithm for distributed flowshop group scheduling problems, IEEE Trans. Cybern. (2021), doi:10.1109/TCYB.2020.3041494.
[22] J.P. Huang, Q.K. Pan, L. Gao, An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times, Swarm Evol. Comput. 59 (2020) 100742, doi:10.1016/j.swevo.2020.100742.
[23] X.L. Jing, Q.K. Pan, L. Gao, Y.L. Wang, An effective Iterated Greedy algorithm for the distributed permutation flowshop scheduling with due windows, Appl. Soft Comput. 96 (2020) 106629, doi:10.1016/j.asoc.2020.106629.
[24] C. Lu, L. Gao, W.Y. Gong, C.Y. Hu, X.S. Yan, X.Y. Li, Sustainable scheduling of distributed permutation flow-shop with non-identical factory using a knowledge-based multi-objective memetic optimization algorithm, Swarm Evol. Comput. 60 (2021) 100803, doi:10.1016/j.swevo.2020.100803.
[25] T. Meng, Q.K. Pan, A distributed heterogeneous permutation flowshop scheduling problem with lot-streaming and carryover sequence-dependent setup time, Swarm Evol. Comput. 60 (2021) 100804, doi:10.1016/j.swevo.2020.100804.
[26] J.P. Huang, Q.K. Pan, Z.H. Miao, L. Gao, Effective constructive heuristics and discrete bee colony optimization for distributed flowshop with setup times, Eng. Appl. Artif. Intell. 97 (2021) 104016, doi:10.1016/j.engappai.2020.104016.
[27] C. Hsu, B. Kao, V.L. Ho, K.R. Lai, Agent-based fuzzy constraint-directed negotiation mechanism for distributed job shop scheduling, Eng. Appl. Artif. Intell. 53 (2016) 140–154, doi:10.1016/j.engappai.2016.04.005.
[28] S. Zhang, X. Li, B. Zhang, S. Wang, Multi-objective optimisation in flexible assembly job shop scheduling using a distributed ant colony system, Eur. J. Oper. Res. 283 (2) (2020) 441–460, doi:10.1016/j.ejor.2019.11.016.
[29] J. Zheng, L. Wang, J. Wang, A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid flow shop, Knowl. based Syst. 194 (2020) 105536, doi:10.1016/j.knosys.2020.105536.
[30] W. Shao, Z. Shao, D. Pi, Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem, Knowl. based Syst. 194 (2020) 105527, doi:10.1016/j.knosys.2020.105527.
[31] L. De Giovanni, F. Pezzella, An improved genetic algorithm for the distributed and flexible job-shop scheduling problem, Eur. J. Oper. Res. 200 (2) (2010) 395–408, doi:10.1016/j.ejor.2009.01.008.
[32] M. Ziaee, A heuristic algorithm for the distributed and flexible job-shop scheduling problem, J. Supercomput. 67 (1) (2014) 69–83.
[33] P. Lu, M. Wu, H. Tan, Y. Peng, C. Chen, A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems, J. Intell. Manuf. 29 (1) (2018) 19–34, doi:10.1007/s10845-015-1083-z.
[34] T. Liu, Y. Chen, J. Chou, Solving distributed and flexible job-shop scheduling problems for a real-world fastener manufacturer, IEEE Access 2 (2014) 1598–1606, doi:10.1109/ACCESS.2015.2388486.
[35] T. Liu, Y. Chen, J. Chou, Evolutionary scheduling system using a universal encoding operator in a distributed and flexible job-shop manufacturing environment, J. Chin. Soc. Mech. Eng. 36 (3) (2015) 221–232.
[36] H. Chang, T. Liu, Optimisation of distributed manufacturing flexible job shop scheduling by using hybrid genetic algorithms, J. Intell. Manuf. 28 (8) (2017) 1973–1986, doi:10.1007/s10845-015-1084-y.
[37] X. Wu, X.J. Liu, N. Zhao, An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem, Memetic Comput. 11 (4) (2019) 1865–9292, doi:10.1007/s12293-018-00278-7.
[38] B. Marzouki, O.B. Driss, K. Ghedira, Solving distributed and flexible job shop scheduling problem using a chemical reaction optimization metaheuristic, Proc. Comput. Sci. 126 (2018) 1424–1433, doi:10.1016/j.procs.2018.08.114.
[39] L. Meng, C. Zhang, Y. Ren, B. Zhang, C. Lv, Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem, Comput. Ind. Eng. 142 (2020) 106347, doi:10.1016/j.cie.2020.106347.
[40] Q. Luo, Q. Deng, G. Gong, L. Zhang, W. Han, K. Li, An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers, Expert Syst. Appl. 160 (2020) 113721, doi:10.1016/j.eswa.2020.113721.
[41] B. Zhang, Q.K. Pan, L. Gao, L.L. Meng, X.Y. Li, K.K. Peng, A three-stage multiobjective approach based on decomposition for an energy-efficient hybrid flow shop scheduling problem, IEEE Trans. Syst., Man, Cybern.: Syst. 50 (12) (2020) 4984–4999, doi:10.1109/TSMC.2019.2916088.
[42] W.Q. Zou, Q.K. Pan, M.F. Tasgetiren, An effective iterated greedy algorithm for solving a multi-compartment AGV scheduling problem in a matrix manufacturing workshop, Appl. Soft Comput. 99 (2021) 106945, doi:10.1016/j.asoc.2020.106945.
[43] M. Dai, D. Tang, A. Giret, M.A. Salido, Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints, Robot. Comput. Integr. Manuf. 59 (2019) 143–157, doi:10.1016/j.rcim.2019.04.006.
[44] A. Ham, Transfer-robot task scheduling in flexible job shop, J. Intell. Manuf. 1-11 (2020), doi:10.1007/s10845-020-01537-6.
[45] J.Q. Li, Y. Du, K.Z. Gao, P.Y. Duan, D.W. Gong, Q.K. Pan, P.N. Suganthan, A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem, IEEE Trans. Autom. Sci. Eng. (2021) In press.
[46] J.Q. Li, J.W. Deng, C.Y. Li, Y.Y. Han, J. Tian, B. Zhang, C.G. Wang, An improved jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times, Knowl. based Syst. 200 (2020) 106632, doi:10.1016/j.knosys.2020.106032.
[47] J. Li, X. Tao, B. Jia, Y. Han, C. Liu, P. Duan, ..., H. Sang, Efficient multi-objective algorithm for the lot-streaming hybrid flowshop with variable sub-lots, Swarm Evol. Comput. 52 (2020) 100600, doi:10.1016/j.swevo.2019.100600.
[48] X.R. Tao, J.Q. Li, T.H. Huang, P. Duan, Discrete imperialist competitive algorithm for the resource-constrained hybrid flowshop problem with energy consumption, Complex Intell. Syst. (2020), doi:10.1007/s40747-020-00193-w.
[49] J. Li, Z.M. Liu, C.D. Li, Z.X. Zheng, Improved artificial immune system algorithm for Type-2 fuzzy flexible job shop scheduling problem, IEEE Trans. Fuzzy Syst. 2020 (2020), doi:10.1109/TFUZZ.2020.3016225.
[50] J. Li, Y. Han, P. Duan, Y. Han, B. Niu, C. Li, Y. Liu, Meta-heuristic algorithm for solving vehicle routing problems with time windows and synchronized visit constraints in prefabricated systems, J. Clean. Prod. 250 (2020) 119464, doi:10.1016/j.jclepro.2019.119464.
[51] D.G. Montgomery, Design and Analysis of Experiments, 8th ed., John Wiley & Sons, Inc, 2012.
[52] Orthogonal arrays (Taguchi designs) from department of mathematics, the university of York. https://www.york.ac.uk/depts/maths/tables/orthogonal.htm (accessed 31 May 2020).