Applied Intelligence (2020) 50:87–100
https://doi.org/10.1007/s10489-019-01497-2

A jigsaw puzzle inspired algorithm for solving large-scale no-wait flow shop scheduling problems

Fuqing Zhao[1] · Xuan He[1] · Yi Zhang[2] · Wenchang Lei[1] · Weimin Ma[3] · Chuck Zhang[4] · Houbin Song[1]

Published online: 2 July 2019

Abstract
The no-wait flow shop scheduling problem (NWFSP), as a typical NP-hard problem, has important ramifications in the modern industry. In this paper, a jigsaw puzzle inspired heuristic (JPA) is proposed for solving NWFSP with the objective of minimizing makespan. The core idea behind JPA is to find the best match for each job until all the jobs are scheduled in the set of process. In JPA, a waiting time matrix is constructed to measure the gap between two jobs. Then, a matching matrix based on the waiting time matrix is obtained. Finally, the optimal scheduling sequence is built by using the matching matrix. Experimental results on large-scale benchmark instances show that JPA is superior to the state-of-the-art heuristics.

Keywords No-wait flow shop scheduling · Makespan · Heuristic algorithm · Jigsaw puzzle

1 Introduction

Scheduling problems play an important role in the manufacturing system. As a kind of flow shop, the no-wait flow shop problems (NWFSP) are widely used in various industries. Such as iron production, mining, logistics, medical industry and food industry [1]. In this study, we consider the no-wait flow shop scheduling problems with the makespan criterion. The makespan is the completion time of the last operation of the last job finishing on the machine. The criterion is immensely important in the actual production and it has been widely studied. In the NWFSP, n jobs are processed on m machines. Each of n jobs has a predetermined processing time. The processing of each job must be successive. In other words, the waiting time between the operations of a job on all machines is zero. Each job is processed at most on one machine, and each machine processes at most one job. The starting time of jobs and machines are regarded as zero. The objective of the NWFSP is to schedule the job sequence to achieve an optimal makespan.

Due to the computational complexity property, NWFSP has been confirmed as a NP-hard problem [2]. Numerous approaches, which are mainly exact algorithms and meta-heuristic algorithms, have been proposed to address the NWFSP [3–6]. The exact algorithms are only suitable for solving small-sized problems. The meta-heuristic algorithm is an improvement of the heuristic algorithm, a combination of the random algorithm and the local search algorithm. Meta-heuristic is an iterative process, through the intelligent combination of different concepts. In the

✉ Fuqing Zhao
Fzhao2000@hotmail.com

Xuan He
2369084655@qq.com

Yi Zhang
1049440546@qq.com

Wenchang Lei
494391740@qq.com

Weimin Ma
mawm@tongji.edu.cn

Chuck Zhang
chuck.zhang@isye.gatech.edu

Houbin Song
523712418@qq.com

[1] School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, China

[2] School of Mechanical Engineering, Xijin University, Xi'an 710123, China

[3] School of Economics and Management, Tongji University, Shanghai 200092, China

[4] H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

process, learning strategies are employed to find approximate optimal solutions effectively. In general, the meta-heuristic algorithm obtains the approximate optimal solution to the problem at the expense of the computation time and storage space. Currently, meta-heuristic algorithms have attracted a wide variety of attention and become a key issue in the swarm intelligence.

A current and future idle time (CFI) constructive heuristic was proposed by Ye et al. [7] for the no-wait flow shop production to minimize the total completion time. The experimental results showed that the CFI heuristic more effective and efficient than other heuristics in literature at the time. An effective heuristics for the no-wait flow shop scheduling problem with the total flow time minimization was proposed by Gao et al. [8]. Two constructive heuristics, namely the improved standard deviation heuristic (ISDH) and improved Bertolissi heuristic (IBH), were presented. Four composite heuristics were obtained by combining the standard deviation heuristic and the Bertolissi heuristic with the procedure of the constructive heuristic of Laha. A heuristic for the no-wait flow shop scheduling was presented by Sapkal, Laha [9]. This method was based on the assumption that the priority of a job in the initial sequence was given by the sum of its processing times on the bottleneck machines. An enumeration algorithm was proposed by Samarghandi [10] to minimize the makespan in the no-wait flow shop scheduling problem with due date constraints. The new modeling technique and the resulting observations were incorporated into a new exact algorithm to solve the problem to optimality. A penalty-based construction algorithm was designed by Laha, Gupta [11] for the no-wait flow shop scheduling problem with the objective of minimizing the makespan and the total flow time of jobs. The proposed method was employed to generate an initial schedule of jobs. An average idle time (AIT) heuristic was proposed by Ye et al. [12] to minimize the makespan in the no-wait flow shop production. Compared with three existing best-known heuristics at the time, the smallest deviations from optimum were achieved by the AIT heuristic. A new exact algorithm was introduced by Samarghandi, Behroozi [13] to minimize the makespan in the no-wait flow shop scheduling problem with due date constraints. Computational results demonstrated that the developed algorithm was significantly faster than the mathematical models. A hybrid discrete optimization algorithm, which was based on teaching-probabilistic learning mechanism (HDTPL), was proposed by Shao et al. [14] to solve the no-wait flow shop scheduling with minimization of makespan. The HDTPL consists of four components, discrete teaching phase, discrete probabilistic learning phase, population reconstruction, neighborhood search. An average departure time (ADT) heuristic was introduced by Ye et al. [15] for minimizing the makespan in the no-wait flow shop production. Based on the computational experiment with instances of various sizes, the ADT heuristic performed better than the three existing best-known heuristics at the time in the same computational complexity environment.

A hybrid ant colony algorithm (HACO), which was based on the crossover and mutation mechanism, was combined by Engin, Guclu [16] for the no-wait flow shop scheduling with the makespan criterion. Since the operating parameters of ACO play an important role in the quality of the solution, the full factorial experimental design is considered for the NWFSP benchmark problems in the HACO. The computational experiments showed that the proposed HACO provided desirable results relative to certain state-of-the-art algorithms in the literature. An iterated greedy heuristic was proposed by Li et al. [4] to address a dependent setup times no-wait flowshop with learning and forgetting effects to minimize the total flowtime. In the proposed algorithm, a position-based learning and forgetting effects model was constructed. A novel quantum-inspired cuckoo co-search (QCCS) algorithm was introduced by Zhu et al. [17] to solve the no-wait flow shop scheduling problem. The numerical results verified that the QCCS is effective to the NWFSP with small and medium scales. Allahverdi et al. [3] developed dominance relations and proposed the *AA* Algorithm, which was a combination of the simulated annealing and the insertion algorithm. The total tardiness and the makespan were considered as criteria in the proposed algorithm. The computational analysis indicated that the proposed Algorithm *AA* performed significantly better than the existing algorithms in the literature. An enhanced migrating birds optimization algorithm (EMBO) was developed by Gao et al. [18] for the NWFSP with the total flow time criterion. In the EMBO, the population was divided into multiple migrating birds in an attempt to prevent the local optima. A hybrid NSGA-II and VNS was proposed by Asefi et al. [19] for solving a bi-objective no-wait flexible flow shop scheduling problems. A tabu mechanism improved iterated greedy (TMIIG) algorithm was introduced by Ding et al. [20] to solve the no-wait flow-shop scheduling problems with the makespan criterion. In the TMIIG, a powerful neighborhood search method which involves inserting, swapping, and double-inserting moves is applied to obtain desirable solutions. An evolutionary clustering search for the no-wait flow shop problem with sequence dependent setup times was introduced by Nagano et al. [21]. A hybrid heuristic algorithm was proposed by Riahi, Kazemi [22] for the no-wait flow shop scheduling problems (ACO-SA). ACO was employed in global search, and SA was utilized to avoid trapping into local optima. A particle swarm optimization was developed by Samarghandi [23] for solving the no-wait flow shop problems with due date constraints. An optimization of the makespan was introduced by Lin, Ying [24], in which efficient matheuristics was used to address the no-wait flowshop scheduling problems. The results obtained by the proposed algorithm were the best solutions and highest-precision at the present. In the proposed algorithm, the effects of certain dispatching rules for generating initial solutions were studied. In brief, different kinds of methods were proposed for solving the no-wait flow shop problems [25–31].

Although heuristics and meta-heuristics achieve optimal or near-optimal results, the evolutionary processing is always time-consuming owing to the local neighborhood search. In addition, the performance of meta-heuristic is sensitive to the value of the parameter and the meta-heuristics must establish a mapping rule between the continuous vector and job sequence. Finally, the result of each run of the meta-heuristic is not unique. Therefore, it is necessary to design a simple and efficient heuristic algorithm to address these problems. In this paper, a jigsaw puzzle inspired algorithm (JPA) for solving the NWFSP with the makespan criterion is introduced. The way to search the suboptimal solution of the problem for JPA is completely different from meta-heuristics. Heuristic refers to the method of solving the problem through inductive reasoning and experimental analysis of the past experience, namely, means a certain intuitive judgment or heuristic method is used to obtain the suboptimal solution of the problem. Versatility, stability, and faster convergence are the main criteria for measuring the performance of heuristics.

The principle of JPA is simple and easily implemented. The key of JPA is to find the best match for each job until all the jobs are scheduled. In the JPA, the waiting time matrix is adopted to measure the gap between two jobs. Then, the matching matrix based on the waiting time matrix is obtained. Finally, the optimal scheduling sequence is constructed by using the matching matrix. Experimental results on 360 benchmark instances show that JPA is superior to the state-of-the-art algorithms in terms of the solution quality.

The rest of the paper is structured as follows. In Section 2, the definition of NWFSP is described. The framework of JPA is presented in Section 3. Computational results with analysis are provided in Section 4. The conclusions and future work are summarized in Section 5.

2 The no-wait flow shop scheduling problems

The notation that is used throughout this paper is listed as follows.

n	The number of jobs
m	The number of machines
π	A sequence of n jobs, $\pi = [\pi(1), \pi(2), \dots, \pi(n)]$
$p_{j,i}$	The processing time of job j on machine i, where $j = 1, 2, \dots, n$ and $i = 1, 2, \dots, m$
$C_{j,i}$	The completion time of job j on machine i
$d_{j-1,j}$	The distance between the completion times of two adjacent jobs on the last machine
P	The processing time matrix
D	The distance matrix
$C_{\max}(\pi)$	The makespan of π
WTJa, Jb	The waiting time between J_a and J_b
π_{min}	The best scheduling sequence with minimal makespan

The following assumptions are used for the no-wait flow shop scheduling [32]. Let $\pi = [\pi(1), \pi(2), \dots, \pi(n)]$ be a feasible scheduling solution. The processing time of job j on machine i, $p_{\pi(j),\ i}$ is deterministic. The setup times are included in the processing times. The transportation times among machines are negligible. In addition, all jobs are available to be processed at time zero on the first machine, each job is only processed once and only once on each machine, each machine can process only one job at a time, and there is no machine breakdown. Once a job starts to be processed, it is not interrupted before completion. Namely, preemption is not allowed. Based on these assumptions, the objective is to find a sequence of jobs that minimizes the makespan. The calculation of $C_{\max}(\pi)$ for the no-wait flow shop production is illustrated as follows [33].

Suppose that the start time of job J_j on the first machine is the same with the completion time of job J_{j-1} on the last machine, as shown in the green dotted box in Fig. 1. Given initial conditions that $CT_{\pi(0),\ m} = 0$, $P_{\pi(j),\ 0} = 0$, $P_{\pi(0),\ i} = 0$, $\sum_{k=1}^{0} p_{\pi(j),k}$, and $\sum_{k=m+1}^{m} p_{\pi(j),k} = 0$.

$$ST_{\pi(j),1} = CT_{\pi(j-1),m} \tag{1}$$

$$ST_{\pi(j),i} = ST_{\pi(j),1} + \sum_{k=1}^{i-1} p_{\pi(j),k} \tag{2}$$

$$CT_{\pi(j),i} = CT_{\pi(j),m} - \sum_{k=i+1}^{m} p_{\pi(j),k} \tag{3}$$

Where, $j = 1, 2, \cdots n$ and $i = 1, 2, \cdots m$, $ST_{\pi(j),\ i}$ denotes the time when job j starts processing on machine i. $CT_{\pi(j),\ i}$ denotes processing completion time of job j on machine i. The potential distances between the start time of job j and the completion time of job $j-1$ on machine i are calculated by Equation 4.

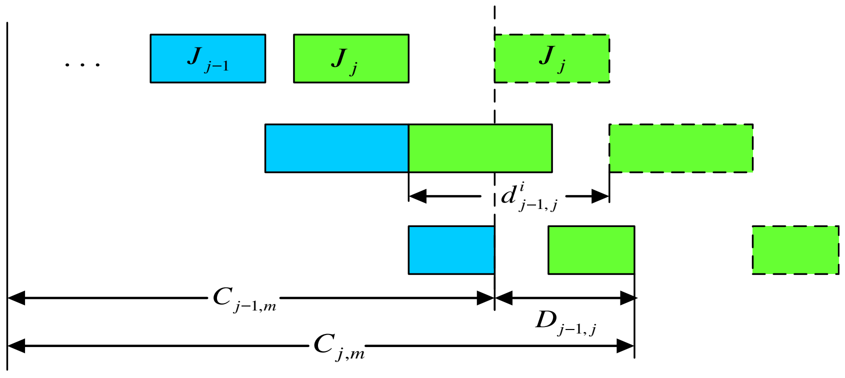

Fig. 1 The distance between two adjacent jobs

$$d^i_{\pi(j-1),\pi(j)} = ST_{\pi(j),i} - CT_{\pi(j-1),i} = ST_{\pi(j),1} - CT_{\pi(j-1),m} + \sum_{k=1}^{i-1} p_{\pi(j),k} + \sum_{k=i+1}^{m} p_{\pi(j-1),k} = \sum_{k=1}^{i-1} p_{\pi(j),k} + \sum_{k=i+1}^{m} p_{\pi(j-1),k} \quad (4)$$

As shown in Figure 1, there exists at least one machine i that $ST_{\pi(j),\ i}$ equals to $CT_{\pi(j-1),\ i}$ $(1 \le i \le m)$. Therefore, the completion time of job j on machine i $(C_{\pi(j),\ i})$ is calculated by Equation 5.

$$C_{\pi(j),i} = C_{\pi(j-1),m} + \sum_{k=1}^{i} p_{\pi(j),k} - \min_{1\le i\le m} d^i_{\pi(j-1),\pi(j)} \quad (5)$$

Namely,

$$C_{\pi(j),i} = C_{\pi(j-1),m} + \sum_{k=1}^{i} p_{\pi(j),k} - \min_{1\le i\le m}\left(\sum_{k=1}^{i-1} p_{\pi(j),k} + \sum_{k=i+1}^{m} p_{\pi(j-1),k}\right) \quad (6)$$

The distance between the completion times of two adjacent jobs on the last machine $d_{\pi(j-1),\ \pi(j)}$ is calculated by Equation 7.

$$d_{\pi(j-1),\pi(j)} = C_{\pi(j),m} - C_{\pi(j-1),m} = \sum_{k=1}^{m} p_{\pi(j),k} - \min_{1\le i\le m}\left(\sum_{k=1}^{i-1} p_{\pi(j),k} + \sum_{k=i+1}^{m} p_{\pi(j-1),k}\right) = \max_{1\le i\le m}\left(\sum_{k=1}^{m} p_{\pi(j),k} - \sum_{k=1}^{i-1} p_{\pi(j),k} - \sum_{k=i+1}^{m} p_{\pi(j-1),k}\right) = \max_{1\le i\le m}\left(\sum_{i=1}^{m} p_{\pi(j),k} - \sum_{k=i+1}^{m} p_{\pi(j-1),k}\right). \quad (7)$$

$$\mathrm{C}_{\max}(\pi) = \sum_{i=1}^{m} p_{\pi(1),i} + \sum_{j=2}^{n} D_{\pi(j-1),\pi(j)}. \quad (8)$$

From Equation 7, the $d_{\pi(j-1),\ \pi(j)}$ depends only on two adjacent jobs rather than on the positions of other jobs in the sequence, thus a distance matrix provides values of $d_{\pi(j-1),\ \pi(j)}$ for any two adjacent jobs. Although the calculation of distances between two adjacent jobs is not sequence dependent, the calculation of $\mathrm{C}_{\max}$ is sequence dependent. Therefore, the makespan for a given sequence is calculated by Equation 8. The waiting time of two adjacent job is calculated as follows.

$$WT_{J_a,J_b} = \left(\sum_{i=1}^{m}\left(\sum_{k=1}^{i-1} p_{J_b,k} + \sum_{k=i+1}^{m} p_{J_a,k}\right)\right) - m\cdot\left(\min_{1\le i\le m}\left(\sum_{k=1}^{i-1} p_{J_b,k} + \sum_{k=i+1}^{m} p_{J_a,k}\right)\right) \quad (9)$$

To-wait flow shop with three machines and three jobs is presented in Figure 2 to provide a better understanding of the problem formulation. The makespan is easily obtained by summing up the three completion time distances ($d_{0,\ 1} = 6$, $d_{1,\ 2} = 4$ and $d_{2,\ 3} = 3$) marked in Figure 2. The scheduling sequence in Figure 2 is $\pi = \{1, 2, 3\}$, and the corresponding makespan equals to 13. For more details about applications of no-wait flow shop production, readers are referred to [32].

The Gantt chart of a no-wait flow shop with three machines and three jobs is presented in Figure 2 to provide an understanding of the problem formulation. The makespan is easily obtained by summing up the three completion time distances ($\mathrm{d}_{0,\ 1} = 6$, $\mathrm{d}_{1,\ 2} = 4$ and $\mathrm{d}_{2,\ 3} = 3$) marked in Figure 2. The scheduling sequence in Figure 2 is $\pi = \{1, 2, 3\}$, and the corresponding makespan equals to 13. For more details about applications of the no-wait flow shop production, readers are referred to [32].

3 Jigsaw puzzle inspired algorithm

3.1 Description of jigsaw puzzle

In this section, a simple example is taken to explain the working principle of the Jigsaw Puzzle, which is the foundation of the proposed algorithm. The example is designated in detail. The steps are as follows.

Step 1: Cutting a paper into five parts by four random curves as Figure 3.

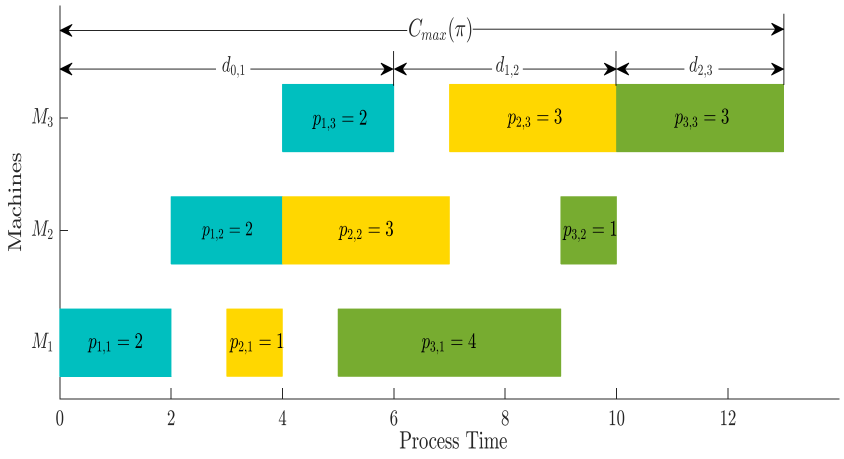

Fig. 2 Gantt chart with $n = 3$ and $m = 3$

Step 2: Disrupt the order of the five parts as Figure 4.

Step 3: Determine the head part. The head part is chosen randomly. In Figure 5, suppose that the yellow part is selected as the current part. All the other parts are matched to the current part independently. As the result in Figure 5, there is no gap between the yellow part and green part. Therefore, the green part is the best matching part of the yellow part. In the same way, the rest of the parts are matched in the best way and eventually the best results are determined as Figure 6. Namely, red part is selected as the head part.

3.2 The proposed algorithm (JPA)

According to the above descriptions of the Jigsaw Puzzle, a jigsaw puzzle inspired algorithm (JPA) is proposed for solving the NWFSP. The pseudo-code of JPA is presented in Algorithm 1 which consists of the following main steps.

Step 1: Construct the matching matrix. Calculate the waiting time of each job with the other jobs by the Equation 9. According to the waiting time, the best match is determined for each job independently. For instance, in the following matrix, J_1 denotes the job one, and $J_{1,\,min\,(WT)}$ represents the best match for job one, $J_{1,\ secmin(WT)}$ means the second-best match for job one, in this way, it constitutes a matching queue for job one. J_2 is the job two, $J_{2,\,min\,(WT)}$ denotes the best match for job two. $J_{2,\ secmin(WT)}$ represents the second-best match for job two. Similarly, it also constitutes a matching queue for job two. Finally, a matching matrix is got, which is shown as follows.

$$\begin{bmatrix} J_1 & J_{1,min(WT)} & \dots & J_{1,max(WT)} \\ J_2 & J_{2,min(WT)} & \dots & J_{2,max(WT)} \\ \vdots & \vdots & \vdots & \vdots \\ J_n & J_{n,min(WT)} & \dots & J_{n,max(WT)} \end{bmatrix} \tag{10}$$

In the matrix, J_i is the i th job. $J_{i,\,min\,(WT)}$ means the job with the minimum waiting time after J_i. $J_{i,\,max\,(WT)}$ denotes the job with the maximum waiting time after J_i.

Step 2: Generate a scheduling solution. Pick each job J_i ($i = 1, 2\cdots n$) as the head job (starting job) in the above matching matrix. And push the job into a stack. Pop the job to J_{new} from the stack. J_{new} is a temporary

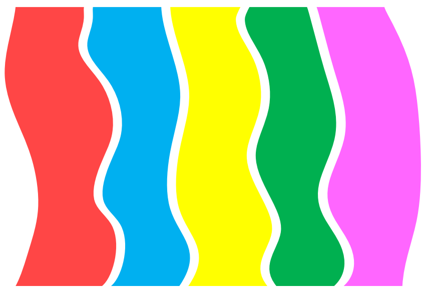

Fig. 3 Cutting paper

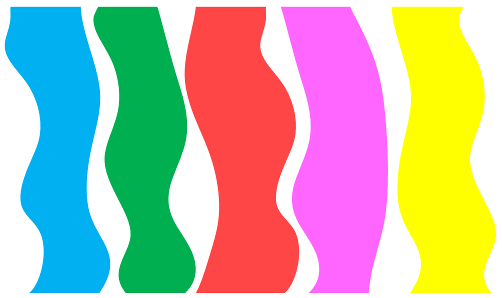

Fig. 4 Disorganizing parts

variable which is used to save the jobs. Choose the J_{next} which has minimum $WT_{new,\ next}$ and $J_{next} \notin$ *stack* from the matching matrix. J_{next} is the best match for the current job. The process of the generating solutions in detail is showed in Figure 7. The first job J_1 is picked. Pop the job J_1 and save it into J_{new}. Then choose best match (J_{next}) for the job according to the above matching matrix, push the J_{next} into the stack, and pop the J_{next} and save it into the J_{new}. In this way, scheduling solutions are achieved.

Step 3: Calculate the makespan. Calculate the makespan of the scheduling sequence with different jobs as the head job. Then return the minimum one.

Algorithm 1. The proposed JPA.

```
1  **Inputs:** The processing time matrix $P$, the distance matrix $D$, the job set $\pi = \{J_1, J_2, \cdots, J_n\}$.
2  **Initialize:** The best block $\pi_{\min} = \{\}$.
3  **For** $i = 1{:}n$ **do**
4      **For** $j = 1{:}n$ **do**
5          **If** $i == j$ **then**
6              $WT_{i,j} = \infty$
7          **Else**
8              Calculate the $WT_{i,j}$ by using Equation (9).
10         **End if**
11     **End for**
12 Jobs is sorted according to the value of $WT_{i,j}$ from small to large
13 **End for**
14 **For** $i = 1{:}n$ **do**
15     Push $J_i$ to stack
16     **For** every job except $J_i$ **do**
17         Pop the job in the stack to $J_{new}$
18         $J_j = J_{new,min(WT)}$
19         **While** $J_j \in stack$ **do**
20             $J_j = J_{new,secmin(WT)}$
21         **End while**
22         Push $J_{new}$ to stack
23         Push $J_j$ to stack
24     **End for**
25 Take the scheduling sequence $\pi'$ from the stack and calculate $C_{\max}(\pi')$ according to Equation (8
26 **End for**
27 **Output:** the best solution $\pi_{\min}$.
```

In order to illustrate the mechanism of the JPA, a simple example is showed as follow. Suppose that the processing time matrix P and the distance matrix D are as follows.

$$P = \begin{bmatrix} 2 & 3 & 7 \\ 1 & 5 & 1 \\ 4 & 2 & 3 \end{bmatrix} \quad (11)$$

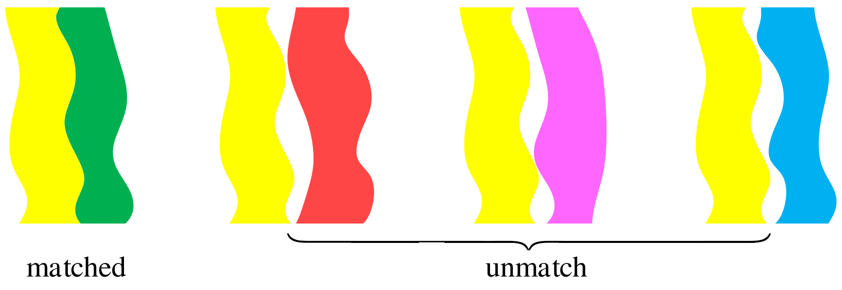

Fig. 5 Matching parts independently

$$D = \begin{bmatrix} \infty & 1 & 3 \\ 9 & \infty & 4 \\ 7 & 3 & \infty \end{bmatrix} \tag{12}$$

Step 1: Construct a 3 × 3 matching matrix.

$$\begin{bmatrix} J_1 & WT_{1,2}=6 & WT_{1,3}=9 \\ J_2 & WT_{2,3}=2 & WT_{2,1}=5 \\ J_3 & WT_{3,1}=0 & WT_{3,2}=3 \end{bmatrix}$$

Step 2: Generate a scheduling solution. In the above matching matrix, we take the first line to show how to produce a scheduling solution. Owing to $WT_{1,2}=6<WT_{1,3}=9$, the next job is J_2 when J_1 is the head job. And then it is the J_3 after J_2. Thus, the scheduling solution is [1,2,3]. In a similar way, the scheduling solution is [2,3,1] when J_2 is the head job. The scheduling solution is [3,1,2] when J_3 is the head job.

Step 3: According to the Equation 7 and Equation 8, the makespans of the scheduling solutions produced in Step 2 are 17, 18 and 21 respectively. Thus, the [1,2,3] is the final solution produced by JPA.

3.3 The time complexity of JPA

Definition: if J_j is the best match for J_i, and the J_j has already existed in the stack, during the process of searching the best match for J_i, it meets a J_j again. Thus, it forms a loop, which is named $l_{i,j}$. And $l(l\geq 0)$ is the number of loops. $n(n>2)$ is the number of jobs and $m(m>1)$ is the number of machines.

According to the algorithm 1, the frequency of step-8 is n^2m. The frequency of step-12 is $n^2 \log n$. The frequency of step-15 is n. The frequencies of step-17 and step-18 are n^2. The frequency of step-20 is n^2l. l is the number of loop. The frequencies of step-22 and step-23 are n^2. The frequency of step-25 is n^2m. The frequency of step-27 is n. Therefore, the frequency of JPA-key-steps is described as follows.

$$T(n) = n^2m + n^2logn + n + 2n^2 + n^2l + n^2 + n^2m + n$$
$$T(n) = 2n^2m + n^2logn + n^2l + 3n^2 + 2n$$
$$T(n) = n^2\left(2m + logn + l + 3 + \frac{2}{n}\right)$$

Due to $n>2$, $m>1$ and $l\geq 0$, $T(n)=O(n^2 \max(m, logn, l))$.

Therefore, the time complexity of JPA is $O(n^2 \max(m, logn, l))$. Compared with the meta-heuristic, there is no iteration behavior in JPA. And JPA obtains desired results solution in the fixed time.

4 The experiment and comparisons

In this section, 360 typical benchmark instances with different scales are employed to verify the effectiveness of the JPA in this paper. These benchmarks include two types: (1) 120 benchmarks were provided by Taillard [34]. The benchmark set includes 12 groups of different sizes problems ranging from 20 jobs and five machines to 500 jobs and 20 machines, and each group has 10 instances; (2) 240 benchmarks were provided by Vallada et al. [35]. These benchmarks are newly proposed for testing the performance of the algorithm. There are 480 instances in total. Here, we only take 240 large scale instances with the sizes from ranging from 100 jobs and 20

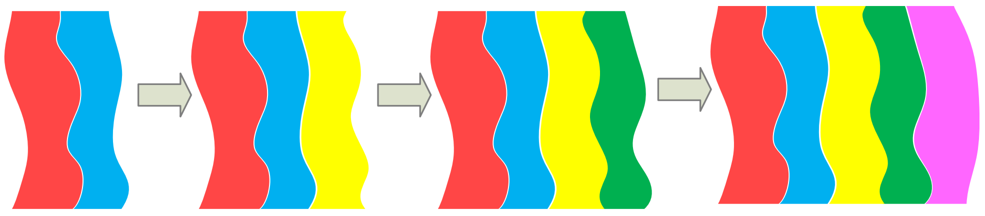

Fig. 6 The matching process of jigsaw puzzles

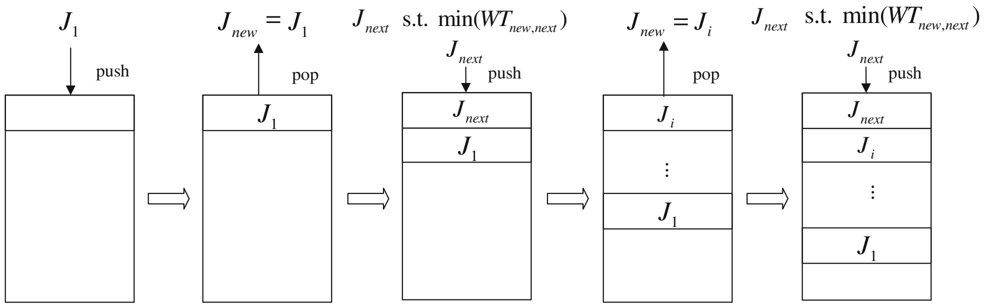

Fig. 7 The process of generating solutions

machines to 800 jobs and 60 machines. These well-known benchmarks well evaluate the algorithm performance and are widely used.

All the compared algorithms in this paper are coded in MATLAB and run on an Intel Core i7-2760QM @ 2.4GHZ with 4GB of RAM. JPA only run one time since it is not a random algorithm. For the compared algorithms, each of the instances is executed 30 times independently.

4.1 Results and comparison with other algorithms for makespan criterion

In this paper, Relative Percentage Increase (RPI) is employed to measure the solution quality of the algorithm, which is shown as follows.

$$RPI = \frac{C_r - C_r^*}{C_r^*} \times 100 \qquad (13)$$

where C_r is the rth solution produced by the corresponding algorithm. C_r^* is the best solution among all comparison algorithms. Obviously, the value of RPI represents the performance of the algorithm.

JPA is compared with the state-of-the-art algorithms including ADT [15], ACO-SA [22], AIT [36] and PAPSO [37] to evaluate the performance of the proposed algorithm. For each algorithm, the control parameter values, which were suggested, were used in the cited and original papers.

- ADT is an effective heuristic method for the no-wait flowshop scheduling optimization. In the ADT, average departure time of each job was considered to construct an initial solution, then, the insertion technique similar to the NEH heuristic is used to further improve the quality of the solution.
- ACO-SA is a new hybrid algorithm combined the ant colony optimization with simulated annealing algorithm for solving the NWFSP, which ant colony optimization is used as a global search method and the simulated annealing algorithm is used as a local search method.
- An average idle time (AIT) heuristic was proposed by Ye et al. [36] to minimize the makespan in the no-wait flow shop production. In the AIT, the current idle times and future idle times are considered to generate an initial sequence. The insertion and neighborhood exchanging methods were applied to improve candidate quality in the solutions set. The computational results showed that

Table 1 Comparison of makespans based on Ta's benchmark set

Instance	ADT	ACO-SA	AIT	PAPSO	NN + NEH	JPA
20 × 5	1.54E+03	1.94E+03	1.51E+03	1.89E+03	1.54E+03	1.59E+03
20 × 10	2.05E+03	2.63E+03	2.02E+03	2.65E+03	2.06E+03	2.12E+03
20 × 20	3.03E+03	3.94E+03	3.01E+03	3.93E+03	3.08E+03	3.18E+03
50 × 5	3.51E+03	4.63E+03	3.40E+03	4.59E+03	3.48E+03	3.48E+03
50 × 10	4.49E+03	6.35E+03	4.42E+03	6.31E+03	4.50E+03	4.57E+03
50 × 20	6.14E+03	8.83E+03	6.12E+03	8.80E+03	6.16E+03	6.35E+03
100 × 5	6.78E+03	9.34E+03	6.55E+03	9.13E+03	6.68E+03	6.51E+03
100 × 10	8.51E+03	1.25E+04	8.43E+03	1.24E+04	8.51E+03	8.43E+03
100 × 20	1.12E+04	1.70E+04	1.12E+04	1.69E+04	1.13E+04	1.13E+04
200 × 10	1.63E+04	2.48E+04	1.61E+04	2.46E+04	1.63E+04	1.59E+04
200 × 20	2.10E+04	3.33E+04	2.10E+04	3.31E+04	2.11E+04	2.09E+04
500 × 20	4.96E+04	8.38E+04	4.86E+04	8.29E+04	4.97E+04	4.83E+04
Average	1.12E+04	1.74E+04	1.10E+04	1.73E+04	1.12E+04	1.11E+04

Table 2 Comparison of RPI based on Ta's benchmark set

Instance	ADT		ACO-SA		AIT		PAPSO		NN + NEH		JPA	
	RPI	SD	RPI	SD	RPI	SD	RPI	SD	RPI	SD	RPI	SD
20 × 5	1.95	1.62	28.49	4.36	0.05	0.14	25.36	4.06	2.25	1.46	5.14	3.39
20 × 10	2.22	1.93	30.69	4.25	0.36	0.68	31.85	2.45	2.65	1.72	5.76	2.18
20 × 20	0.97	1.24	31.42	6.15	0.09	0.16	30.79	3.59	2.59	1.91	5.78	1.93
50 × 5	3.22	1.12	36.11	3.58	0.00	0.01	35.03	3.66	2.30	1.10	2.41	1.82
50 × 10	1.66	0.68	43.75	4.70	0.00	0.00	42.70	3.97	1.96	1.13	3.52	1.51
50 × 20	0.59	0.57	44.66	3.71	0.24	0.50	44.19	4.45	0.95	0.45	3.98	1.86
100 × 5	4.20	0.77	43.56	2.22	0.68	0.65	40.44	3.32	2.76	0.93	0.07	0.12
100 × 10	1.34	0.98	48.56	2.64	0.38	0.48	47.62	4.13	1.42	0.67	0.38	0.39
100 × 20	0.67	0.61	52.46	3.03	0.19	0.28	51.32	3.07	0.95	0.65	0.90	0.57
200 × 10	2.43	0.38	55.72	1.22	1.17	0.47	54.62	2.26	2.29	0.53	0.00	0.00
200 × 20	0.73	0.47	59.34	2.97	0.45	0.38	58.52	2.39	0.81	0.58	0.08	0.17
500 × 20	2.53	0.36	73.36	1.88	0.58	0.28	71.49	1.90	2.76	0.42	0.00	0.00
Average	1.88	0.89	45.68	3.39	0.35	0.34	44.49	3.27	1.97	0.96	2.34	1.16

Table 3 Comparison of makespans based on VRF's benchmark set

Instance	ADT	ACO-SA	AIT	PAPSO	NN + NEH	JPA
100 × 20	1.11E+04	1.73E+04	1.60E+04	1.71E+04	1.13E+04	1.13E+04
100 × 40	1.54E+04	2.40E+04	2.25E+04	2.38E+04	1.55E+04	1.56E+04
100 × 60	1.88E+04	2.92E+04	2.76E+04	2.91E+04	1.90E+04	1.91E+04
200 × 20	2.09E+04	3.36E+04	3.15E+04	3.33E+04	2.09E+04	2.08E+04
200 × 40	2.81E+04	4.63E+04	4.34E+04	4.58E+04	2.82E+04	2.81E+04
200 × 60	3.38E+04	5.60E+04	5.38E+04	5.54E+04	3.41E+04	3.38E+04
300 × 20	3.04E+04	5.02E+04	4.58E+04	4.97E+04	3.06E+04	3.01E+04
300 × 40	4.06E+04	6.82E+04	6.23E+04	6.80E+04	4.09E+04	4.03E+04
300 × 60	4.87E+04	8.30E+04	7.85E+04	8.26E+04	4.89E+04	4.84E+04
400 × 20	3.95E+04	6.65E+04	6.14E+04	6.58E+04	4.02E+04	3.93E+04
400 × 40	5.27E+04	9.07E+04	8.55E+04	9.02E+04	5.31E+04	5.23E+04
400 × 60	6.31E+04	1.10E+05	1.05E+05	1.10E+05	6.35E+04	6.25E+04
500 × 20	4.87E+04	8.37E+04	7.72E+04	8.31E+04	4.97E+04	4.83E+04
500 × 40	6.54E+04	1.13E+05	1.08E+05	1.13E+05	6.56E+04	6.43E+04
500 × 60	7.76E+04	1.36E+05	1.28E+05	1.36E+05	7.79E+04	7.64E+04
600 × 20	5.90E+04	1.00E+05	9.23E+04	9.93E+04	5.91E+04	5.73E+04
600 × 40	7.73E+04	1.36E+05	1.29E+05	1.36E+05	7.76E+04	7.59E+04
600 × 60	9.19E+04	1.64E+05	1.62E+05	1.65E+05	9.21E+04	9.03E+04
700 × 20	6.81E+04	1.17E+05	1.08E+05	1.16E+05	6.83E+04	6.62E+04
700 × 40	8.94E+04	1.59E+05	1.50E+05	1.59E+05	8.95E+04	8.76E+04
700 × 60	1.06E+05	1.92E+05	1.79E+05	1.92E+05	1.06E+05	1.04E+05
800 × 20	7.75E+04	1.34E+05	1.18E+05	1.33E+05	7.76E+04	7.51E+04
800 × 40	1.01E+05	1.81E+05	1.64E+05	1.80E+05	1.01E+05	9.90E+04
800 × 60	1.20E+05	2.18E+05	1.98E+05	2.18E+05	1.20E+05	1.17E+05
Average	5.77E+04	1.00E+05	9.36E+04	1.00E+05	5.80E+04	5.68E+04

the AIT algorithm outperformed state-of-the-art algorithms in the literature.

- PAPSO is a particle swarm optimization algorithm with the population adaptation method. The proposed population adaptation method identifies the moment when the diversity of population is poor or the population stagnates by measuring the Euclidean distance between the particle position and the particles average position of a population. The results show that the population adaptation method significantly improves the performance of the PSO algorithm.
- The idea of using the NN heuristic and the NEH heuristic (NN + NEH) to construct the initial population is presented by [38] in a discrete water wave optimization (DWWO) algorithm for the no-wait flow shop scheduling problem.

The ADT, AIT and NN + NEH belong to constructive heuristics. However, the way ADT, AIT, and NN + NEH construct solutions are different in the process of solving NWFSP. ACO-SA and PAPSO belong to meta-heuristics. The two algorithms are variants of ACO and PSO respectively. The computational results on Taillard instances are reported in Tables 1 and 2. The experiential results on VRF instances are shown in Tables 3 and 4. The bold values mean the optimal solutions among the compared algorithms. The makespan and RPI in the table are mean value according to different scale groups. SD denotes the standard deviation of RPI.

4.2 Analysis and discussion

From Tables 1 and 3, it is seen that the solutions generated by JPA are better than those obtained by ADT, ACO-SA, AIT, PAPSO, and NN + NEH on most of the test cases. The performance of JPA improved with the increasement of jobs. For example, VRF800_60 means that there are 800 jobs and 60 machines in the test case. The average RPI obtained by the compared algorithms in the instance are as follows: JPA/0.00, ADT/1.96, ACO-SA/85.34, AIT/68.80, PAPSO/85.50, and NN + NEH/2.24 respectively. The results produced by ACO-SA, and PAPSO are close to 85, which stated that there is no significant difference between the ACO-SA and PAPSO.

Table 4 Comparison of RPI based on VRF's benchmark set

Instance	ADT		ACO_SA		AIT		PAPSO		NN + NEH		JPA	
	RPI	SD	RPI	SD	RPI	SD	RPI	SD	RPI	SD	RPI	SD
100 × 20	0.07	0.20	55.15	3.43	43.75	6.37	53.13	4.34	1.22	0.84	1.51	1.40
100 × 40	0.18	0.23	55.77	2.00	46.24	4.73	54.65	2.78	0.76	0.79	1.15	0.91
100 × 60	0.10	0.29	55.83	2.71	47.34	3.41	55.11	2.78	1.13	0.78	1.77	1.22
200 × 20	0.61	0.37	61.59	3.32	51.55	5.19	60.31	3.78	0.81	0.46	0.10	0.27
200 × 40	0.45	0.48	65.43	2.11	55.04	8.24	63.52	2.21	0.81	0.58	0.45	0.54
200 × 60	0.46	0.42	66.14	2.77	59.65	7.10	64.62	1.84	1.27	0.74	0.29	0.38
300 × 20	1.08	0.51	66.87	1.11	52.05	2.87	65.10	1.82	1.80	0.53	0.00	0.00
300 × 40	0.73	0.58	69.15	1.56	54.43	2.10	68.55	1.77	1.35	0.49	0.02	0.06
300 × 60	0.55	0.53	71.57	1.52	62.24	8.79	70.61	1.80	1.12	0.61	0.07	0.22
400 × 20	0.63	0.41	69.40	1.48	56.49	7.17	67.53	1.92	2.34	0.36	0.01	0.03
400 × 40	0.84	0.37	73.50	1.48	63.69	4.35	72.64	1.35	1.63	0.39	0.00	0.00
400 × 60	0.91	0.41	75.62	1.02	67.70	6.77	75.41	1.70	1.56	0.26	0.00	0.00
500 × 20	0.90	0.43	73.32	1.80	59.91	6.50	72.04	2.41	2.88	0.39	0.02	0.05
500 × 40	1.80	0.26	76.61	1.97	67.41	4.33	76.21	1.42	2.08	0.33	0.00	0.00
500 × 60	1.52	0.12	78.40	1.92	68.10	3.72	77.81	1.76	1.96	0.22	0.00	0.00
600 × 20	2.92	0.38	75.10	1.43	61.07	6.06	73.36	2.02	3.12	0.42	0.00	0.00
600 × 40	1.84	0.29	79.56	1.34	70.38	6.30	78.90	1.49	2.24	0.39	0.00	0.00
600 × 60	1.74	0.19	81.98	1.12	79.58	7.59	82.16	1.46	1.97	0.20	0.00	0.00
700 × 20	2.95	0.19	76.56	1.35	62.97	3.00	75.63	1.19	3.20	0.24	0.00	0.00
700 × 40	2.01	0.13	81.86	1.18	71.33	5.01	81.71	0.91	2.21	0.25	0.00	0.00
700 × 60	1.99	0.33	84.70	1.35	72.10	9.22	84.69	1.18	2.35	0.29	0.00	0.00
800 × 20	3.27	0.22	78.08	0.96	57.40	3.27	76.70	1.33	3.38	0.20	0.00	0.00
800 × 40	2.28	0.17	82.33	1.27	65.76	3.73	82.00	1.58	2.47	0.18	0.00	0.00
800 × 60	1.96	0.30	85.34	1.48	68.80	5.79	85.50	1.32	2.24	0.31	0.00	0.00
Average	1.32	0.32	72.49	1.74	61.04	5.48	71.58	1.92	1.91	0.43	0.22	0.21

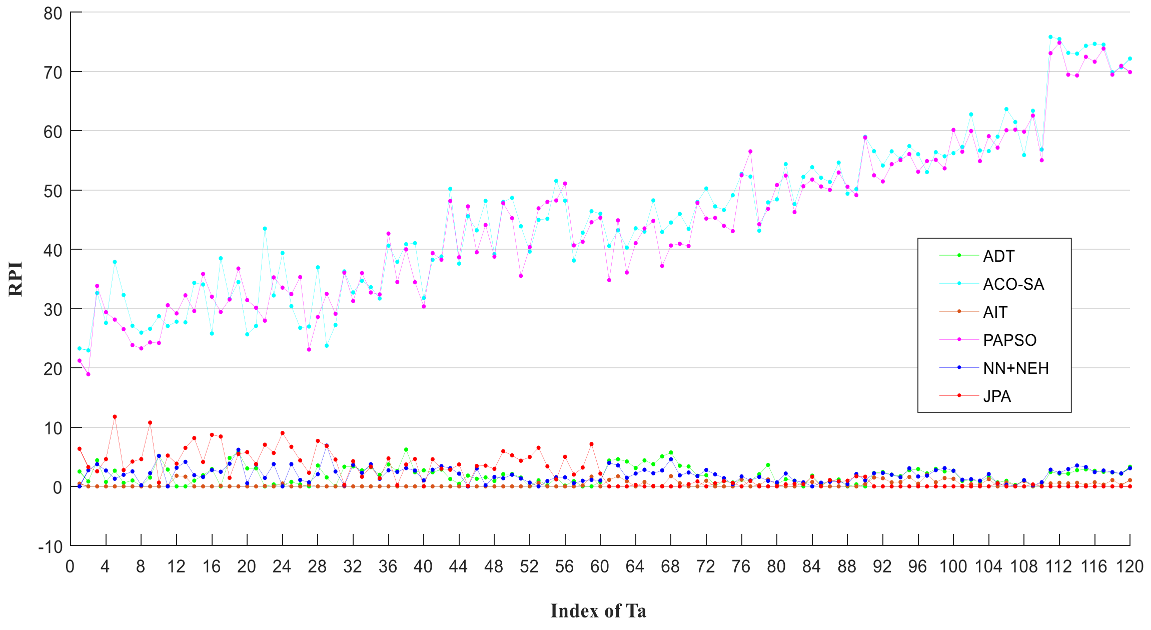

Fig. 8 The trend of compared algorithms on the Ta instances

However, the RPI obtained by JPA ranked the first place and significantly better than the other five algorithms. It is worth mentioning that JPA obtains the excellent results on large-scale problems, which indicates the JPA is suitable for solving the large-scale no-wait flow shop problems. Although NN + NEH, ADT, and AIT are better than JPA in certain instances, the time complexity of JPA ($O(n^2 max(m, logn, l))$) is lower than that of NN + NEH ($O(n^3)$). The computational complexity of the AIT heuristic is $O(n^3 + mn^2)$, which is the same as the computational complexity of the ADT.

The trend of compared algorithms on Ta and VRF instances are shown in Figures 8 and 9 to clearly show the performance of JPA. It is observed that the curve of JPA is lower than those of ACO-SA, and PAPSO. At the same time, with the increasing the jobs, the vertical distance of the compared algorithms increased correspondingly, which suggests that the performance of JPA is superior to the previous one.

In order to further check the significant differences between JPA and the five competitors in the large-scale instances (VRF instances), the Friedman's test [39] is carried out to rank the algorithms, which we use in the experimental evaluations, in which Bonferroni–Dunn's procedure was used as a post-hoc procedure. The average ranking of the six algorithms obtained by the Friedman's

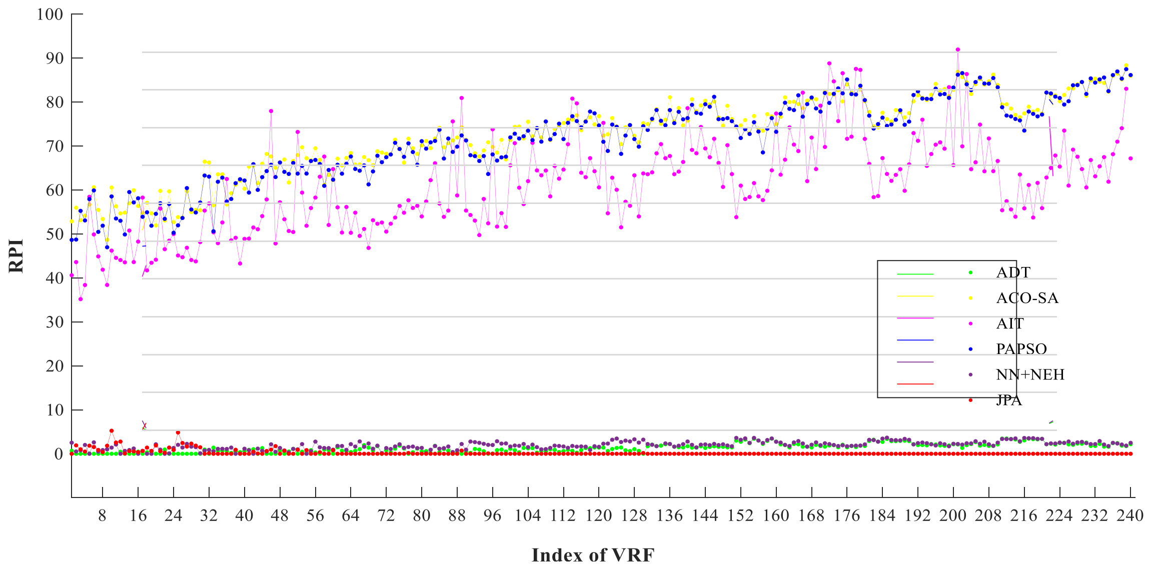

Fig. 9 The trend of compared algorithms on the VFR instances

Table 5 Ranking of ADT, ACO-SA, AIT, PAPSO, NN + NEH, and JPA obtained through Friedman's test and critical difference of Bonferroni-Dunn's procedure

Algorithms	Mean Rank
ADT	1.99
ACO-SA	5.59
AIT	4.21
PAPSO	5.20
NN + NEH	2.74
JPA	1.27
Crit.Diff. $\alpha = 0.05$	1.2443
Crit.Diff. $\alpha = 0.1$	1.1240

test is summarized in Table 5. The rank of the proposed JPA is among the smallest one. To evaluate the significance level of all the algorithms, an additional Bonferroni–Dunn's procedure was introduced as a post hoc procedure to calculate the critical difference (CD in the Equation 14) for comparing their differences with $\alpha = 0.05$ and $\alpha = 0.1$:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{14}$$

In the Equation 14, parameters k and N are the number of algorithms to be compared and the number of benchmarks, respectively. They are $k = 6$ and $N = 240$ in the experimental evaluations. When $\alpha = 0.05$, q_α is 2.576 and when $\alpha = 0.1$, q_α is 2.327 from Table B.16 (two-tailed$\alpha(2)$) of [40]. The results of Bonferroni–Dunn's test which considers JPA as control algorithm is sketched in Figure 10.

As shown in Figure 10, a significant difference among JPA, ACO-SA, AIT, PAPSO, and NN + NEH with $\alpha = 0.1$ and $\alpha = 0.05$ was observed. Although there is no significant difference between JPA and ADT, the rank of JPA is less than that of ADT. In summary, the statistical analysis of the results obtained by the algorithms in the comparative study showed that the performance of the proposed JPA is the best algorithm to solve the large-scale instances.

5 Conclusion and future research

This paper presents a jigsaw puzzle inspired algorithm (JPA) for addressing the no-wait flow shop scheduling problems (NWFSP) with the makespan criterion. The NWFSP has been a strong NP-hard problem. The key of JPA is to search for the best match for each job, and the principle of JPA is easy to understand. JPA is not a random algorithm so that it produces a certain solution for each case. There is no need to execute each case various times and achieve the average. Since there is no local search behavior in JPA, the speed of searching the global optimum is fast. Besides, it is not necessary to set initial values in the JPA. By analyzing the pseudo-code of JPA, the time complexity of JPA is $O(n^2\, max\,(m, logn, l))$. Finally, a large number of experiments on large scale instances show that JPA is efficient and effective in solving the NWFSP.

It is worth noting that although the performance of the constructive heuristic depends on the intrinsic properties of the specific problem, JPA is not specifically designed for the flow-shop scheduling problem. In fact, JPA is proposed to solve the type of combinatorial optimization problem like NWFSP. In the future work, JPA as an initialization algorithm is applied to reduce the time complexity of the

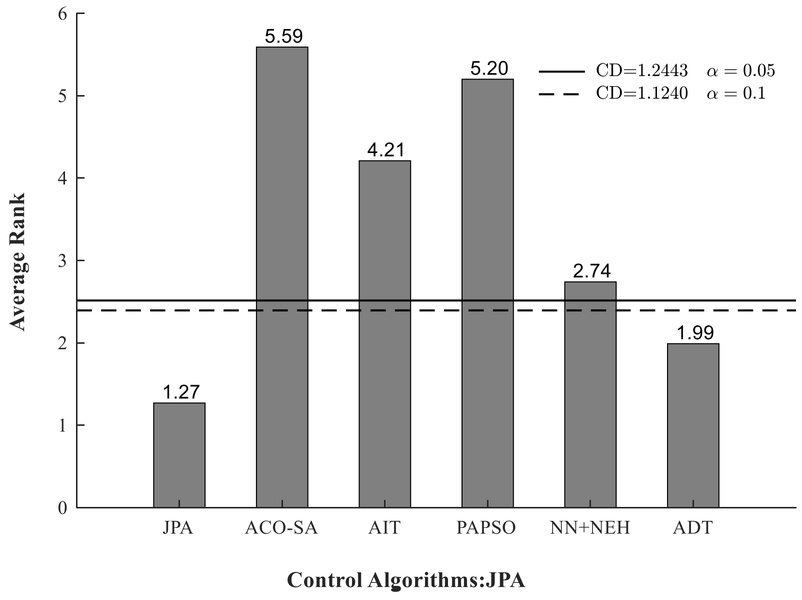

Fig. 10 Rankings obtained through the Friedman test and graphical representation of the Bonferroni-Dunn's test (Taking JPA as a control method). CD means critical difference

meta-heuristic algorithm and solve various practical problems. Such as job shop scheduling problems and the lot-streaming flow shop scheduling problems.

Acknowledgements This work was financially supported by the National Natural Science Foundation of China under grant numbers 61663023. It was also supported by the Key Research Programs of Science and Technology Commission Foundation of Gansu Province (2017GS10817), Lanzhou Science Bureau project (2018-rc-98), Zhejiang Provincial Natural Science Foundation (LGJ19E050001), Wenzhou Public Welfare Science and Technology project (G20170016), respectively.

References

1. Hall NG, Sriskandarajah C (1996) A survey of machine scheduling problems with blocking and no-wait in process. Oper Res 44(3): 510–525
2. Garey MR, Johnson DS (1979) A Guide to the Theory of NP-Completeness. WH Freemann, New York
3. Allahverdi A, Aydilek H, Aydilek A (2018) No-wait flowshop scheduling problem with two criteria; total tardiness and makespan. Eur J Oper Res 269(2):590–601. https://doi.org/10.1016/j.ejor.2017.11.070
4. Li X, Yang Z, Ruiz R, Chen T, Sui S (2018) An iterated greedy heuristic for no-wait flow shops with sequence dependent setup times, learning and forgetting effects. Inf Sci 453:408–425. https://doi.org/10.1016/j.ins.2018.04.038
5. Zhao F, Zhang L, Liu H, Zhang Y, Ma W, Zhang C, Song H (2018) An improved water wave optimization algorithm with the single wave mechanism for the no-wait flow-shop scheduling problem. Eng Optim. https://doi.org/10.1080/0305215X.2018.1542693
6. Wang Y, Li X, Ruiz R, Sui S (2018) An Iterated Greedy Heuristic for Mixed No-Wait Flowshop Problems. IEEE T Cybern 48(5): 1553–1566. https://doi.org/10.1109/TCYB.2017.2707067
7. Ye H, Li W, Abedini A, Nault B (2017) An effective and efficient heuristic for no-wait flow shop production to minimize total completion time. Comput Ind Eng 108:57–69. https://doi.org/10.1016/j.cie.2017.04.002
8. Gao K, Pan Q, Suganthan PN, Li J (2012) Effective heuristics for the no-wait flow shop scheduling problem with total flow time minimization. Int J Adv Manuf Technol 66(9–12):1563–1572
9. Sapkal SU, Laha D (2013) A heuristic for no-wait flow shop scheduling. Int J Adv Manuf Technol 68(5–8):1327–1338
10. Samarghandi (2017) On the exact solution of the no-wait flow shop problem with due date constraints. Comput Oper Res 81:141–159. https://doi.org/10.1016/j.cor.2016.12.013
11. Laha D, Gupta JND (2016) A Hungarian penalty-based construction algorithm to minimize makespan and total flow time in no-wait flow shops. Comput Ind Eng 98:373–383. https://doi.org/10.1016/j.cie.2016.06.003
12. Ye H, Li W, Abedini A (2017) An improved heuristic for no-wait flow shop to minimize makespan. J Manuf Syst. https://doi.org/10.1016/j.jmsy.2017.04.007
13. Samarghandi H, Behroozi M (2016) An Enumeration Algorithm for the No-Wait Flow Shop Problem with Due Date Constraints. IFAC-PapersOnLine 49(12):1803–1808. https://doi.org/10.1016/j.ifacol.2016.07.844
14. Shao W, Pi D, Shao Z (2016) A hybrid discrete optimization algorithm based on teaching–probabilistic learning mechanism for no-wait flow shop scheduling. Knowl-Based Syst 107:219–234. https://doi.org/10.1016/j.knosys.2016.06.011
15. Ye H, Li W, Miao E (2016) An effective heuristic for no-wait flow shop production to minimize makespan. J Manuf Syst 40:2–7. https://doi.org/10.1016/j.jmsy.2016.05.001
16. Engin O, Guclu A (2018) A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems. Applied Soft Computing Journal 72:166–176. https://doi.org/10.1016/j.asoc.2018.08.002
17. Zhu H, Qi X, Chen F, He X, Chen L, Zhang Z (2018) Quantum-inspired cuckoo co-search algorithm for no-wait flow shop scheduling. Appl Intell. https://doi.org/10.1007/s10489-018-1285-0
18. Gao KZ, Suganthan PN, Chua TJ (2013) An enhanced migrating birds optimization algorithm for no-wait flow shop scheduling problem. In: Computational Intelligence in Scheduling (SCIS), 2013 IEEE Symposium on, 16–19 April 2013, pp 9–13. https://doi.org/10.1109/SCIS.2013.6613246
19. Asefi H, Jolai F, Rabiee M, Araghi MET (2014) A hybrid NSGA-II and VNS for solving a bi-objective no-wait flexible flowshop scheduling problem. Int J Adv Manuf Technol 75(5–8):1017–1033. https://doi.org/10.1007/s00170-014-6177-9
20. Ding J, Song S, Zhang R, Wu C, IEEE (2014) Minimizing makespan for a no-wait flowshop using Tabu Mechanism Improved Iterated Greedy Algorithm. 2014 IEEE Congress on Evolutionary Computation (Cec):1906–1911
21. Nagano MS, Silva AAD, Lorena LAN (2014) An evolutionary clustering search for the no-wait flow shop problem with sequence dependent setup times. Expert Syst Appl 41(8):3628–3633
22. Riahi V, Kazemi M (2015) A hybrid heuristic algorithm for the no-wait flowshop scheduling problem. In: Computer Science and Software Engineering (CSSE), 2015 International Symposium on, 18–19 Aug. 2015, pp 1–6. https://doi.org/10.1109/CSICSSE.2015.7369247
23. Samarghandi H (2015) A particle swarm optimisation for the no-wait flow shop problem with due date constraints. Int J Prod Res 53(9):1–18
24. Lin SW, Ying KC (2015) Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. Omega 64:115–125
25. Zhu X, Li X (2015) Iterative search method for total flowtime minimization no-wait flowshop problem. Int J Mach Learn Cybern 6(5):747–761. https://doi.org/10.1007/s13042-014-0312-7
26. Wang S, Liu M, Chu C (2015) A branch-and-bound algorithm for two-stage no-wait hybrid flow-shop scheduling. Int J Prod Res 53(4):1143–1167
27. Ramezani P, Rabiee M, Jolai F (2015) No-wait flexible flowshop with uniform parallel machines and sequence-dependent setup time: a hybrid meta-heuristic approach. J Intell Manuf 26(4):731–744
28. Sapkal SU, Laha D (2011) An improved scheduling heuristic algorithm for no-wait flow shops on total flow time criterion. In: Electronics Computer Technology (ICECT), 2011 3rd International Conference on, pp 159–163
29. Gao KZ, Li JQ, Liang JJ, Li H, Pan QK (2010) Hybrid heuristics based on harmony search to minimize total flow time in no-wait flow shop. In: Chinese Control and Decision Conference, pp 1184–1188
30. Gao K, Pan Q, Li J, He Y (2010) A novel grouping harmony search algorithm for the no-wait flow shop scheduling problems with total flow time criteria. In: 2010 International Symposium on Computer, Communication, Control and Automation Proceedings, vol 1, pp 77–80
31. Ding JY, Song S, Gupta JND, Zhang R, Chiong R, Wu C (2015) An improved iterated greedy algorithm with a Tabu-based reconstruction strategy for the no-wait flowshop scheduling problem. Appl Soft Comput 30:604–613
32. Pinedo ML (2016) Scheduling: theory, algorithms, and systems. Springer, Berlin

33. Ye HH, Li W, Miao EM (2016) An effective heuristic for no-wait flow shop production to minimize makespan. J Manuf Syst 40:2–7. https://doi.org/10.1016/j.jmsy.2016.05.001
34. Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64(2):278–285. https://doi.org/10.1016/0377-2217(93)90182-M
35. Vallada E, Ruiz R, Framinan JM (2015) New hard benchmark for flowshop scheduling problems minimising makespan. Eur J Oper Res 240(3):666–677
36. Ye H, Li W, Abedini A (2017) An improved heuristic for no-wait flow shop to minimize makespan. J Manuf Syst 44:273–279. https://doi.org/10.1016/j.jmsy.2017.04.007
37. Jana ND, Sil J (2014) Das S Particle Swarm Optimization with population adaptation. Evol Comput:1727–1732
38. Zhao FQ, Liu H, Zhang Y, Ma WM, Zhang C (2018) A discrete Water Wave Optimization algorithm for no-wait flow shop scheduling problem. Expert Syst Appl 91:347–363. https://doi.org/10.1016/j.eswa.2017.09.028
39. García S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. J Heuristics 15(6):617–644
40. Zar JH (1999) Biostatistical Analysis. 4th ed. NewDelhi: Pearson Education India

Fuqing Zhao received the B.Sc. and Ph.D. degrees from the Lanzhou University of Technology, Lanzhou, China, in 1994 and 2006, respectively. Since 1998, he has been with the School of Computer Science Department, Lanzhou University of Technolgy, Lanzhou, China, where he became a Full Professor in 2012. He has been as the post Doctor with the State Key Laboratory of Manufacturing System Engineering, Xi'an Jiaotong University, Xi'an, China in 2009. He has authored two academic book and over 50 refereed papers. His current research interests include intelligent optimization and scheduling.