A quantum-inspired cuckoo co-evolutionary algorithm for no-wait flow shop scheduling

Haihong Zhu1 | Ning Luo2 | Xiaoping Li3

1School of Cyber Science and Engineering, Southeast University, Nanjing, 211189, China
2Institute of Software, Chinese Academy of Sciences, Beijing, 100190, China
3School of Computer Science and Engineering, Southeast University, Nanjing, 211189, China

Correspondence
Xiaoping Li, School of Computer Science and Engineering, Southeast University, Nanjing, 211189, China.
Email: xpli@seu.edu.cn

Funding information
National Key Research and Development Program of China, Grant/Award Number: 2017YFB1400801; The National Natural Science Foundation of China, Grant/Award Numbers: 61572127, 61872077, 61832004; the Collaborative Innovation Center of Wireless Communications Technology

Abstract
No-wait flow shop scheduling problems (NWFSFs) are widespread in practical applications. The authors propose a quantum-inspired cuckoo co-evolutionary algorithm for the NWFSF to minimize the makespan. There are three algorithm components: quantum solution construction, quantum population evolution, and an improved neighbourhood local search. They generate initial solutions, search solutions, and improve solution qualities, respectively. Parameters of the proposed algorithm are calibrated statistically. The proposal with calibrated parameters is compared with three existing algorithms on Reeves and Taillard benchmark instances with middle scales. Experimental results show that the proposal outperforms the compared algorithms.

1 | INTRODUCTION

The no-wait flow shop scheduling problem (NWFSF) is a significant combinatorial optimization problem, which has been studied widely because of its applications in the industrial field, for example, steel, food processing, and chemical engineering [1–3]. The NWFSF has attracted significant attention from researchers since the 1950s [4–7]. A comprehensive survey on the research and application of the NWFSF can be found in Hall and Sriskandarayah’s review paper [8]. In the NWFSF, waiting is not permitted, that is, once the processing of a job begins, subsequent processing must be continuously carried out on all machines with no interruption until its completion because of the constraints on processing jobs by equipment and technique. The common optimal objectives of the NWFSF include minimizing the makespan [9], total tardiness [10], and total flow time [11]. Among them, minimizing the makespan is an essential objective of the actual application.

NWFSF has been proved to be an NP-hard problem when the number of machines is more than two [12]. The heuristic and meta-heuristic algorithms are the most usual methods to solve the NWFSF. For decades, heuristics and meta-heuristics have been mainly developed for solving the NWFSF. Regarding heuristics, Allahverdi and Aldowaisan [13, 14] researched the NWFSF with a separate setup of three machines and sequence-dependent additive changeover time. Aldowaisan and Allahverdi [15] presented several heuristics for the NWFSF with setup times and m-machine constraints. Li et al. [16] introduced an objective increment method to judge whether a new schedule is better than its parent. The time complexity of the proposed heuristic algorithm can be decreased by one order. Gao et al. [17] presented two algorithms: ISDH (Improved Standard Deviation Heuristic) and IBH (Improved Bertolissi Heuristic). Computational results show that the composite heuristics perform significantly better than the existing ones. Sapkal and Laha [18] presented an efficient heuristic method in which the initial sequence of jobs is generated based on bottleneck machines.

Various noteworthy meta-heuristics have been developed for solving the NWFSF. A hybrid differential evolution (HDE) algorithm is proposed by Qian et al. [19]. Simulation results demonstrated its effectiveness. Pan et al. [20] presented a
discrete particle swarm optimization (DPSO) algorithm with variable neighbourhood descent (VND) local search for minimizing both makespan and total flow time. It was hybridized with the VND algorithm to improve the quality of solutions further. Tseng and Lin [21] proposed a hybrid genetic algorithm (HGA), which integrated the genetic algorithm (GA) into a novel local search scheme, and the results showed the superiority of the HGA. Jarboui et al. [22] designed a block order crossover operator for GA, regarded as GA-VNS. The experiments showed that GA-VNS provides competitive results and better upper bounds. Samarghandi and ElMekkawy [23] presented a hybrid tabu search (TS) and particle swarm optimization (PSO) algorithm, in which a new encoding scheme based on the natural relation between permutations and factoradics was developed. Akroud et al. [24] presented a hybrid greedy randomized adaptive search procedure (GRASP) of which the parameters are adjusted by differential evolution. Ding et al. [25] presented a tabu-mechanism improved iterated greedy algorithm (TMIIG) which adopts a tabu-based reconstruction strategy in the iterated greedy (IG) algorithm to improve the searching ability and the neighbourhood methods to obtain better solutions. Engin and Guelu [26] designed a hybrid ant colony optimization algorithm for solving the NWFSP with the makespan and the flowtime, which hybridized ant colony optimization (ACO) and simulated annealing (SA) algorithm. Zhao et al. [27] proposed a factorial based particle swarm optimization with a population adaptation (PA) mechanism to minimize the makespan in which the PA mechanism was designed to control the diversity of the population. Lin and Ying [28] proposed two efficient meta-heuristics to minimize the makespan for an asymmetric travelling salesman problem. Li et al. [29] researched the NWFSP with sequence-dependent setup times, learning and forgetting effects, and proposed an iterated greedy heuristic algorithm to minimize total flow time. Wang et al. [30] proposed an iterated greedy heuristic algorithm for solving mixed NWFSP with both wait and no-wait constraints. Shao et al. [31] considered a multiobjective distributed NWFSP with sequence-dependent setup time (MDNWFSP-SDST). In MDNWFSP-SDST, a Pareto-based estimation of distribution algorithm is presented for the makespan and the total weight tardiness objectives.

In addition, quantum computing has been drawn much attention and has widespread applications in combinatorial optimization. Since the late 1990s, a variety of quantum-inspired evolutionary algorithms such as the genetic-quantum algorithm [32], quantum-inspired gravitational search algorithm [33], and quantum differential evolution algorithm [34] have been successfully utilized in solving different engineering and combinatorial optimization problems. It is found that there is a growing interest in emerging quantum computing with evolutionary algorithms. One of the developed evolutionary algorithms is the cuckoo search (CS) algorithm [35] with global convergence, convenient to be coupled with other algorithms. Furthermore, the CS algorithm has been proved to be a useful optimization tool in practice. It is desirable to combine them for global optimization problems. As NWFSPs have unique characteristics, it is challenging to develop quantum-inspired evolutionary algorithms for them [36]. To the best of our knowledge, there are quite a few co-evolutionary algorithms for NWFSPs.

The authors propose an effective Quantum-inspired Cuckoo Co-evolutionary Algorithm (QCCA) with an improved local neighbourhood search method for the considered NWFSP to minimize makespan. The proposed algorithm combines quantum evolution and the cuckoo search mechanism. Furthermore, the QCCA with improved local neighbourhood search, which is called QCCA-INLS, enhances the quality of the solution. The main contributions of this paper are summarized as follows:

- A novel quantum solution construction (QSC) method is developed for the NWFSP to encode the population and constructs diversiform initial solutions.
- A quantum population evolutionary (QPE) method is designed with the cuckoo search mechanism and quantum NOT, which can effectively find the global solution.
- To efficiently improve the solution’s quality, an improved local neighbourhood search (INLS) is presented for minimizing the makespan.

The remainder of the paper is organized as follows. Section 2 describes and formalizes the problem of minimizing the makespan in NWFSP. The proposed algorithm is presented in Section 3. Experimental evaluations of parameter calibration and algorithm comparison are shown in Section 4, followed by the conclusions and future research in Section 5.

2 | PROBLEM FORMULATION

In this section, we present a formal description of the problem. The major symbols and definitions used for the description are given in Table 1.

The NWFSP with makespan minimization is denoted as $E(m, n, w, \tau)$ with the makespan $C_{\text{max}}$ described as follows. There are $n$ jobs $(J_1, J_2, \ldots, J_n)$ to be processed on a set of $m$ machines $(M_1, M_2, \ldots, M_m)$. All jobs are processed in the same order without any pre-emption and interruption, that is, once a job is started on the first machine, it has to be continuously processed through $m$ machines without interruption. At a time, job $J_i$ ($1 \leq i \leq n$) is being processed at most by one machine, and machine $M_k$ ($1 \leq k \leq m$) can execute no more than one job. The objective of the problem is to find a feasible schedule $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ for the $n$ jobs such that the makespan denoted as $C_{\text{max}}(\pi)$ is minimized, where $C_{\text{max}}(\pi)$ is also equivalent to the finishing time of the last job on the last machine, which can be denoted as $C_{\text{max}}(\pi_n, m)$ and is obtained using the following equation

$$C_{\text{max}}(\pi) = C_{\text{max}}(\pi_n, m) = \sum_{k=2}^{n} D_{\pi_k-1, \pi_k} + \sum_{j=1}^{m} P_{\pi_n, j}$$ (1)
TABLE 1  Major symbols and definitions

| Symbol | Definition |
|--------|------------|
| $m$    | Number of machines in the NWFSP |
| $n$    | Number of jobs to be scheduled |
| $M_j$  | Machine $j$ |
| $J_i$  | Job $i$ |
| $\pi$  | A schedule involving a permutation of the given jobs |
| $\pi_0$ | An initial schedule involving a permutation of the given jobs |
| $\pi_{\text{global}}$ | Global schedule involving a permutation of the given jobs |
| $\pi_{\text{best}}$ | Best schedule involving a permutation of the given jobs |
| $\pi_k$ | $\pi_k \in \{J_1, J_2, \ldots, J_n\}$ is the job in position $k$ of $\pi$ |
| $P_{\pi,j}$ | Processing time of job $i(i=1, 2, \ldots, n)$ processed on machine $M_j(j=1, 2, \ldots, m)$ |
| $D_{\pi_k-1, \pi_k}$ | Completion time distance between adjacent jobs in position $k-1$ and $k$ of a schedule $\pi$ |
| $C_{\text{max}}(\pi)$ | Makespan of a given schedule $\pi$ |

![Figure 1](image)

**FIGURE 1** A Gantt chart of $3 \times 3$ NWFSP

The no-wait constraints of the problem ensure that the completion time distance between two adjacent jobs is determined by the processing times of two jobs, regardless of the other jobs in the permutation. Therefore, to satisfy the no-wait constraint, the start of some jobs on the first machine needs to be delayed. A completion time distance is defined between each pair of jobs. The completion time distance between two adjacent jobs, $\pi_{k-1}$ and $\pi_k$, is calculated by

$$D_{\pi_{k-1}, \pi_k} = \max_{1 \leq j \leq m} \left\{ \sum_{j=1}^{k-1} P_{\pi_{k-1}, j} - \sum_{j=2}^{k} P_{\pi_{k-1}, j-1} \right\}, k=2, 3, \ldots, n. \quad (2)$$

As an example shown in Figure 1, three jobs are processed on three machines. It assumes that the processing sequence of three jobs is $\pi = (\pi_1, \pi_2, \pi_3)$. As illustrated in Figure 1, it is clear that the $C_{\text{max}}(\pi)$ for processing three jobs can be easily obtained by summing up the completion time distances $(D_{\pi_1, \pi_2}, D_{\pi_2, \pi_3})$ and the total processing time $\sum_{j=1}^{3} P_{\pi,j}$ of these three jobs, which can be formulated by

$$C_{\text{max}}(\pi) = C_{\text{max}}(\pi_1, 3) = \sum_{k=2}^{3} D_{\pi_{k-1}, \pi_k} + \sum_{j=1}^{3} P_{\pi,j} = D_{\pi_1, \pi_2} + D_{\pi_2, \pi_3} + P_{\pi_1, 1} + P_{\pi_2, 2} + P_{\pi_3, 3} \quad (3)$$

## 3 | QUANTUM-INSPIRED CUCKOO CO-EVOLUTIONARY ALGORITHM

In this section, we introduce an efficient quantum-inspired cuckoo co-evolutionary algorithm for $F_m | \text{nwt} | C_{\text{max}}$. QCCA is cooperative with quantum-inspired evolution and cuckoo search mechanism. Firstly, we adopt a QSC to initialize the job sequence, a better characteristic of population diversity. Then, QPE hybridized CS is used to iterate the population to drive them towards better solutions. However, in the generation,
there are some individuals eliminated with a certain probability. Thus, a variation operator is applied to the quantum encoding of these individuals. Finally, an INLS is presented to intensify the quality of the global solution of QCCA. Therefore, the QCCA with INLS is also called QCCA-INLS. The QCCA-INLS framework is detailed in Algorithm 1.

Algorithm 1 QCCA-INLS Framework

1. Initialize job sequence $\pi_0$ by $QSC(ps)$;
2. Evaluate $C_{max}$;
3. while terminal conditions not met do
4. Population evolution by $QPE(Q(0))$;
5. Evaluate $C_{max}$ and reserve the current best solution;
6. Perform $INLS(\pi_{global})$ on the global best solution;
7. return.

3.1 Quantum solution construction

To encode the population, we use a qubit as the probabilistic representation of an individual, represented by the superposition of “1” state, “0” state, or any superposition of the two. The state of a qubit is given by

$$|\Phi\rangle = \alpha|0\rangle + \beta|1\rangle$$

(4)

where $\alpha$ and $\beta$ are complex numbers that specify the probability amplitudes of the corresponding states. Normalization of the state to unity guarantees $|\alpha|^2 + |\beta|^2 = 1$. Therefore, the probability amplitudes of a qubit can be expressed by a quantum angle, which is given by

$$|\Phi\rangle = \cos\theta|0\rangle + \sin\theta|1\rangle$$

(5)

where $\theta$ is the quantum angle. Thus, a qubit can be expressed by $|\Phi\rangle = \begin{bmatrix} \cos\theta & \sin\theta \end{bmatrix}$. A qubit description is shown in Figure 2. The quantum encoding of the qubit is directly used to generate cuckoo populations initially. The cuckoo population can be formulated by $Q(t) = \{q_i(t) | i = 1, 2, ..., ps\}$ at generation $t$. A quantum individual $q_i(t)$ can be given by

$$q_i(t) = \begin{bmatrix} \cos\theta^{(i)}_1 & \cos\theta^{(i)}_2 & \cdots & \cos\theta^{(i)}_n \\ \sin\theta^{(i)}_1 & \sin\theta^{(i)}_2 & \cdots & \sin\theta^{(i)}_n \end{bmatrix}$$

(6)

where $ps$ is the population size, $n$ is the length of a qubit as well as the number of jobs, and $\theta$ is randomly generated in $[0, 2\pi]$. For solving $F_{max}$, $\pi_{max}$, a quantum individual solution construction (QISC) is used to transform individuals from quantum encoding to job sequences so that the quantum representation of solutions can be applied for solving NWFSPs.

As Figure 3 illustrates with a simple instance ($n = 6$), an individual is ranked by the largest $\cos\theta$ or smallest $\sin\theta$, which is denoted as $\{j_1 \rightarrow 0.564, j_2 \rightarrow 0.281, j_3 \rightarrow 0.37, j_4 \rightarrow 0.009, j_5 \rightarrow 0.963, j_6 \rightarrow 0.55\}$. Jobs are put to the corresponding positions according to rank values. The sequence of rank values is $(2, 5, 4, 6, 1, 3)$. According to the rank values and related jobs, the quantum individual’s job sequence can be written as $(J_5, J_1, J_6, J_3, J_2, J_4)$.

The quantum individual solution construction is used for the QSC, formally described as Algorithm 2. We denote the maximum value $MAX$ as the initial $C_{max}(\pi_0)$ and the initial solution $\pi_0$ is set to $\emptyset$. $\pi_i(t)$ is the corresponding solution of the $i$th quantum individual. Each quantum individual is generated by the procedure of QISC and evaluated by $C_{max}$. The time complexity of QSC is $O(n)$.

Algorithm 2 QSC(ps)

Input: the population size ps
Output: initial solution $\pi_0$
1. $Q(0) \leftarrow \emptyset$;
2. $\pi_0 \leftarrow \emptyset$, $C_{max}(\pi_0) \leftarrow MAX$;
3. for $i = 1$ to $ps$ do
4. Generate a quantum individual $q_i(0)$;
5. $Q(0) \leftarrow Q(0) \cup \{q_i(0)\}$;
6. $\pi_i(0) \leftarrow QISC(q_i(0))$;
7. if $C_{max}(\pi_i(0)) < C_{max}(\pi_0)$ then
8. $\pi_0 \leftarrow \pi_i(0)$;
9. $C_{max}(\pi_0) \leftarrow C_{max}(\pi_i(0))$;
10. return.
3.2 | Quantum population evolution

In quantum-inspired evolutionary algorithm (QEA), the following quantum rotation gate is adopted to update the iterative population.

\[ U(\Delta \theta) = \begin{bmatrix} \cos \Delta \theta & -\sin \Delta \theta \\ \sin \Delta \theta & \cos \Delta \theta \end{bmatrix} \]  \hspace{1cm} \text{(7)}

Therefore, a qubit can be updated by

\[
|\Phi'\rangle = U(\Delta \theta) |\Phi\rangle = \begin{bmatrix} \cos \Delta \theta & -\sin \Delta \theta \\ \sin \Delta \theta & \cos \Delta \theta \end{bmatrix} \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix} = \begin{bmatrix} \cos(\theta + \Delta \theta) \\ \sin(\theta + \Delta \theta) \end{bmatrix}
\]  \hspace{1cm} \text{(8)}

where \( \Delta \theta \) is the rotation angle as Figure 4 illustrated. The quantum individual \( q_i(t + 1) \) is updated by

\[
q_i(t + 1) = [\cos(\theta_i + \Delta \theta)^{(t+1)} \ldots \cos(\theta_n + \Delta \theta)^{(t+1)}] \ldots [\sin(\theta_i + \Delta \theta)^{(t+1)} \ldots \sin(\theta_n + \Delta \theta)^{(t+1)}]
\]  \hspace{1cm} \text{(9)}

The cuckoo search simulates the process that cuckoos lay their eggs in the nests of other host birds. The cuckoo population searches new nests by Lévy flight at the generation. We consider generating quantum angle increment of the population by cuckoo search behaviour, which is performed by

\[
\theta_j^{(t+1)} = \theta_j^{(t)} + \alpha \otimes L(\mu, \nu) , j = 1, 2, \ldots, n
\]  \hspace{1cm} \text{(10)}

where \( \alpha(\alpha > 0) \) is the step factor which should be related to the scale of \( F_m |\text{new} - \text{old}|C_{max} \).

In most cases, we can use \( \alpha = \mathcal{O}(1) \) in Equation (10). The symbol \( \otimes \) presents entry-wise multiplications. Lévy flights essentially provide a random walk while their random steps are drawn from a Lévy distribution for large steps

\[
L(\mu, \nu) = \frac{\mu}{|\nu|^{1/\beta}}
\]  \hspace{1cm} \text{(11)}

where \( \beta \in [1, 2] \) is Lévy exponent, \( \mu \) and \( \nu \) are drawn from normal distributions

\[
\nu \sim N(0, \sigma_\nu^2), \mu \sim N(0, \sigma_\mu^2)
\]  \hspace{1cm} \text{(12)}

with

\[
\sigma_\mu = \left\{ \frac{\Gamma(1 + \beta)\sin(\pi \beta / 2)}{\Gamma(1 + \beta) / 2 \beta^{2 \beta / 2 - 1}} \right\}^{1/\beta} , \sigma_\nu = 1
\]  \hspace{1cm} \text{(13)}

respectively, and \( \Gamma \) being the standard Gamma function.

During the procedure of quantum population evolution, there are some worse nests abandoned. Therefore, new nests can be discovered with a \( P_m \) probability based on the cuckoo search mechanism. Moreover, a quantum mutation operation can modify the job sequence and the best solution is reserved for the next generation. Quantum NOT gate is used to generate some mutated individuals to avoid the premature convergence in quantum population evolution. The qubit of a mutated quantum individual is operated by

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure4.png}
\caption{A qubit update}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure5.png}
\caption{Procedure of quantum mutation operation}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure6.png}
\caption{Neighbourhood structure and insert points}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure7.png}
\caption{Quantum encoding}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure8.png}
\caption{Mutation operation}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure9.png}
\caption{Rank value}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure10.png}
\caption{Sequence of jobs}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure11.png}
\caption{Neighbourhood job and insert points}
\end{figure}
\[ |\varphi^0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix} = \begin{bmatrix} \sin \theta \\ \cos \theta \end{bmatrix} \]  

(14)

The procedure of QPE is described as Algorithm 3. The QPE contains two layers of the loop. The inner loop shows the procedure of quantum individual updates. The time complexity of this loop is \( O(n) \). The outer loop presents the number of iterations \( t \) and \( t \) is a constant. The time complexity of QPE is \( O(cn) \).

For illustrating the mutation operation procedure distinctly, an example is shown in Figure 5, which is associated with Figure 3. From Figure 5, the job sequence is a solution of the NWFSP, obtained in the procedure of population evolution. It assumes that the quantum individuals of jobs \( J_2 \) and \( J_6 \) are mutational with a \( P_a \) probability, which mutates according to quantum NOT gate. Then, the mutated quantum individuals are encoded to a new job sequence by the QISC method.

Algorithm 3 QPE(\( Q(0) \))

**Input:** the step factor \( \alpha \), Lévy exponent \( \beta \\
**Output:** Global best solution \( \pi_{\text{best}} \)

1. \( t \leftarrow 0 \);
2. **while** terminal conditions not met **do**
   3. **for** \( i = 1 \) to \( ps \) **do**
      4. Update \( q_i(t + 1) \) using Eq. (9) and Eq. (10);
      5. Randomly generate a fraction \( R \);
      6. if \( R < P_a \) then
         7. Mutate \( q_i(t + 1) \) by quantum NOT gate;
         8. \( \pi_i(t + 1) \leftarrow QISC(q_i(t + 1)) \);
      9. if \( C_{\text{max}}(\pi_i(t + 1)) > C_{\text{max}}(\pi_i(t)) \) then
         10. \( \pi_i(t + 1) \leftarrow \pi_i(t) \);
         11. \( Q(t + 1) \leftarrow Q(t + 1) \cup \{q_i(t + 1)\} \);
      12. \( t \leftarrow t + 1 \);
   13. \( \pi_{\text{best}} \leftarrow \min_{i=1,\ldots,ps} \{C_{\text{max}}(\pi_i(t))\} \);
14. **return**.

**FIGURE 7** Swap operation

**FIGURE 8** Insert operation

**FIGURE 9** Means plot and 95.0% confidence level Tukey HSD intervals for \( ps \)
3.3 Improved neighbourhood local search

Because of $F_m | \text{net} | C_{\text{max}}$ is an NP-hard problem in the which solution space is very complicated, so a fast and efficient local search algorithm is essential [37].

The proposed INLS maintains the diversity of the population and avoids falling into a local optimum. The INLS is based on insert operation and swap operation. We give some job set definitions for searching optimized solutions. There are two random jobs $\pi_i, \pi_j (1 \leq i \leq n, 1 \leq j \leq n)$ and their neighbourhood jobs $\pi_{i-1}, \pi_{i+1}, \pi_{j-1}, \pi_{j+1}$. $\pi_{\text{local}} = \{\pi_{i-1}, \pi_i, \pi_{i+1}, \pi_{j-1}, \pi_j, \pi_{j+1}\}$ is denoted as a local neighbourhood job set, which is illustrated in Figure 6. The remaining jobs are denoted as $\pi_{\text{rem}} = \pi_{\text{global}} - \pi_{\text{local}}$. If $i$ and $j$ is the insert position, we denote the insert point set $I = \{I_{\text{point}}\} = \{(\pi_{i-1}, \pi_i), (\pi_i, \pi_{i+1}), (\pi_{j-1}, \pi_j), (\pi_j, \pi_{j+1})\}$. Any job from $\pi_{\text{rem}}$ can execute swap operation with jobs from $\pi_{\text{local}}$. Two jobs $\pi_y$ from $\pi_{\text{rem}}$ and $\pi_x$ from $\pi_{\text{local}}$ can be swapped as shown in Figure 7. In Figure 7, $\pi_x$ can be selected from $\pi_{j-1}, \pi_j, \pi_{j+1}$ and $\pi_{j-1}, \pi_j, \pi_{j+1}$ which are also $\pi_{\text{local}}$. $\pi_y$ can be exchanged with $\pi_{\alpha}$ which is anyone of $\pi_{\text{local}}$. In the same way, $\pi_y$ can also be inserted to any point from $I$, as shown in Figure 8. After swap operation and insert operation, the new $\pi_{\text{global}}$ is re-evaluated according to the increment of $C_{\text{max}}$ and the job schedule with the smallest increment is used for new $\pi_{\text{global}}$.

INLS is described in Algorithm 4. INLS contains two layers of the loop. The inner loop shows that the increment

![Figure 10](image1.png)

**Figure 10** Means plot and 95.0% confidence level Tukey HSD intervals for max_iter

![Figure 11](image2.png)

**Figure 11** Means plot and 95.0% confidence level Tukey HSD intervals for $\alpha$
evaluation process corresponds to swap operation and insert operation. The time complexity of inner loop is $O(n)$. The outer loop improves the time for a constant count. Therefore, the time complexity of INLS is $O(mn)$.

**Algorithm 4 INLS($\pi_{global}$)**

| Input: Global best solution $\pi_{global}$ |
|--------------------------------------------|
| Output: Optimized solution $\pi_{best}$ |
| count $\leftarrow 0$; $\Delta_{min} \leftarrow \text{MAX}$; |
| Randomly select two jobs $\pi_i$ and $\pi_j$ from $\pi_{global}$; |
| Construct $\pi_{local}$ and $I$; |
| while $\pi_{local}$ and $I$ not null do |
| Select $\pi_x$ from $\pi_{local}$ and $\pi_y$ from $\pi_{rem}$; |
| $\pi_{sw} \leftarrow \text{Swap}(\pi_x, \pi_y)$; |
| $\Delta_{sw} = C_{max}(\pi_{sw}) - C_{max}(\pi_{global})$; |
| Select $\pi_y$ insert to $I_{point}$ from $I$; |
| $\pi_{ins} \leftarrow \text{Insert}(\pi_x, I_{point})$; |
| $\Delta_{ins} = C_{max}(\pi_{ins}) - C_{max}(\pi_{global})$; |
| $\Delta_{min} \leftarrow \min(\Delta_{sw}, \Delta_{ins}, 0)$; |
| if $\Delta_{min} = \Delta_{sw}$ then |
| $\pi_{local} \leftarrow \pi_{local} - \{\pi_x\}$; |
| $\pi_{best} \leftarrow \pi_{sw}$; |
| else if $\Delta_{min} = \Delta_{ins}$ then |
| $I \leftarrow I - \{I_{point}\}$; |
| $\pi_{best} \leftarrow \pi_{ins}$; |
| else |
| break; |
| count $\leftarrow$ count + 1; |
| return. |

4 | EXPERIMENTAL EVALUATIONS

To evaluate the performance of QCCA-INLS for solving the Fm | nwt | Cmax, the parameter calibration and performance evaluation are described in this section. All tested algorithms are coded in C++ and run on an Intel i7-3770 CPU and 3.10 GHz with 8 GB of RAM. The termination criterion is set to a maximum iteration.

### 4.1 Parameter calibration

As previously discussed, the proposed algorithm has several parameters: (1) the population size ($ps$), (2) maximum iteration ($max\_iter$), (3) the step factor ($a$), (4) the Lévy exponent ($\beta$), and (5) the mutation probability ($P_m$). The tested values of the five parameters to be calibrated are set as $ps \in \{50, 100, 150\}$, $max\_iter \in \{30, 50, 70\}$, $a \in \{0.01, 0.05, 0.1\}$, $\beta \in \{1, 1.5, 2\}$, and $P_m \in \{0.25, 0.5, 0.75\}$. There are $3^5 = 243$ combinations of calibration instances with five parameters. Each combination runs 15 times on each NWFSP instance. Solutions are evaluated by the RPD (Relative Percentage Deviation) defined as follows:

$$RDP(\%) = \frac{S - S_{best}}{S_{best}} \times 100$$

where $S$ is the solution obtained by the corresponding algorithm on a given instance and $S_{best}$ is the best solution obtained on the same instance.

Experimental results are analysed by the ANOVA (analysis of variance) technique. First, the three main hypotheses (normality, homoscedasticity, and independence of the residuals) are checked from the experiments’ residuals. All three hypotheses are acceptable from this analysis. When the $p$-value is infinitesimal and close to zero, they are significant statistically. Figures 9–13 show the means plot and 95.0% confidence level Tukey HSD (honest significant difference) intervals for $ps$, $max\_iter$, $a$, $\beta$, and $P_m$, respectively.

Since the $p$-value in Figure 9 is close to 0, Figure 9 illustrates that the observed differences are statistically significant for $ps$. It can be observed that $ps$ with 150 is significantly better than that with 100 and 50. It implies that a relatively big population size can get a good solution. Similar to

![Figure 12](image-url) Means plot and 95.0% confidence level Tukey HSD intervals for $\beta$
Figure 13: Means plot and 95.0% confidence level Tukey HSD intervals for $P_a$.

Table 2: RPDs (%) for HGA, TS-PSO, TMIIG, QCCA, and QCCA-INLS in Reeves benchmark instances.

| Instance | $n \times m$ | HGA RPD | TS-PSO RPD | TMIIG RPD | QCCA INLS RPD | QCCA-INLS RPD |
|----------|--------------|---------|------------|----------|---------------|---------------|
| Rec01    | 20 $\times$ 5 | 0.00    | 0.00       | 0.00     | 0.00           | 0.00           |
| Rec03    | 20 $\times$ 5 | 0.00    | 0.00       | 0.00     | 0.00           | 0.00           |
| Rec05    | 20 $\times$ 5 | 0.00    | 0.42       | 0.00     | 0.00           | 0.00           |
| Average  |              | 0.00    | 0.14       | 0.00     | 0.00           | 0.00           |
| Rec07    | 20 $\times$ 10| 0.00    | 0.20       | 0.00     | 0.00           | 0.00           |
| Rec09    | 20 $\times$ 10| 0.00    | 0.00       | 0.00     | 0.00           | 0.00           |
| Rec11    | 20 $\times$ 10| 0.00    | 0.21       | 0.00     | 0.00           | 0.00           |
| Average  |              | 0.00    | 0.14       | 0.00     | 0.00           | 0.00           |
| Rec13    | 20 $\times$ 15| 0.00    | 0.11       | 0.00     | 0.00           | 0.00           |
| Rec15    | 20 $\times$ 15| 0.00    | 0.00       | 0.00     | 0.00           | 0.00           |
| Rec17    | 20 $\times$ 15| 0.00    | 0.04       | 0.00     | 0.00           | 0.00           |
| Average  |              | 0.00    | 0.05       | 0.00     | 0.00           | 0.00           |
| Rec19    | 30 $\times$ 10| 0.00    | 0.72       | 0.00     | 0.00           | 0.00           |
| Rec21    | 30 $\times$ 10| 0.28    | 0.17       | 0.07     | 0.00           | 0.00           |
| Rec23    | 30 $\times$ 10| 0.00    | 0.07       | 0.00     | 0.00           | 0.00           |
| Average  |              | 0.09    | 0.32       | 0.02     | 0.00           | 0.00           |
| Rec25    | 30 $\times$ 15| 0.00    | 0.52       | 0.00     | 0.00           | 0.00           |
| Rec27    | 30 $\times$ 15| 0.00    | 0.18       | 0.01     | 0.00           | 0.00           |
| Rec29    | 30 $\times$ 15| 0.00    | 0.29       | 0.03     | 0.00           | 0.00           |
| Average  |              | 0.00    | 0.33       | 0.01     | 0.00           | 0.00           |
| Rec31    | 50 $\times$ 10| 0.54    | 1.01       | 0.13     | 0.49           | 0.00           |
| Rec33    | 50 $\times$ 10| 0.59    | 0.98       | 0.30     | 0.16           | 0.00           |
| Rec35    | 50 $\times$ 10| 0.62    | 0.63       | 0.19     | 0.27           | 0.00           |
| Average  |              | 0.58    | 0.87       | 0.21     | 0.31           | 0.00           |
| Rec37    | 75 $\times$ 20| 1.33    | 1.21       | 0.33     | 0.07           | 0.00           |
| Rec39    | 75 $\times$ 20| 0.72    | 1.32       | 0.27     | 0.59           | 0.00           |
| Rec41    | 75 $\times$ 20| 0.44    | 1.52       | 0.13     | 0.30           | 0.00           |
| Average  |              | 0.83    | 1.35       | 0.24     | 0.32           | 0.00           |
| Global average |            | 0.22    | 0.46       | 0.07     | 0.09           | 0.00           |
Figure 9, Figure 10 with infinitesimal p-value illustrates that the differences are statistically significant for max_iter. It follows that the proposed algorithm has minimal RPD when max_iter = 70. Thus, relatively numerous iterations can also get a good solution.

Figures 11–13 indicate that the differences are statistically significant for different values of α, β, and P_a. It implies that different values of α, β, and P_a exert a great influence on the proposed algorithm’s performance. Similarly, it can be observed that there is an increasing tendency of RPD with a decrease in α, β, and P_a. As Figures 11–13 show, the proposed algorithm has the minimal RPD when α = 0.01, β = 1.0, and P_a = 0.25. Therefore, we set α = 0.01, β = 1.0, and P_a = 0.25 in the following experiments.

4.2 Performance evaluation

According to the calibrated parameters, the QCCA and QCCA-INLS are compared with some well-performing algorithms: HGA [21], TS-PSO [23], and TMIIG [25]. There are 21 instances with n ∈ {20, 30, 50, 75} and m ∈ {5, 10, 15, 20} in Reeves benchmark, which are grouped by 20 × 5, 20 × 10, 20 × 15, 30 × 10, 30 × 15, 50 × 10, and 75 × 20. Each group consists of three instances. In addition, the proposed QCCA-INLS are also compared with HGA and TMIIG in small and middle scales of the Taillard benchmark. There are 90 instances with n ∈ {20, 50, 100} and m ∈ {5, 10, 20} in Taillard benchmark for experiments, which are grouped by 20 × 5, 20 × 10, 20 × 20, 50 × 5, 50 × 10, 50 × 20, 100 × 5, 100 × 10, and 100 × 20. Each group consists of 10 instances.

Table 2 gives the RPDs of HGA, TS-PSO, TMIIG, QCCA, and QCCA-INLS and the average RPDs in each group. From Table 2, we can observe that the average RPDs of QCCA-INLS in all Reeves benchmark instances are all 0.00, which are best compared with HGA, TS-PSO, TMIIG, and QCCA. Besides, the number of upper bounds of QCCA is more than those of HGA, TS-PSO, and TMIIG. However, the average RPDs of QCCA is more than those of TMIIG when n > 30. Furthermore, the global average RPD of QCCA-INLS with 0.00 is less than that of the other algorithms with 0.22, 0.46, 0.07, 0.09, respectively. It implies that the efficiency of QCCA-INLS is better than that of compared algorithms, but the efficiency of QCCA is worse than that of TMIIG when n > 30 and demonstrates that the QCCA-INLS is effective for solving NWFSP in Reeves benchmark instances.

To make comparisons more intuitively, Figure 14 shows the interactions between the compared algorithms and Reeves benchmark instances with 95.0% confidence level Tukey HSD intervals. The p-value is close to 0, indicating that the RPD of QCCA-INLS is significantly smaller than those of the other four algorithms from Figure 14. Therefore, we can see that the QCCA-INLS is statistically better than HGA, TS-PSO, TMIIG, and QCCA in all the Reeves benchmark instances.

Table 3 shows the RPDs and the average RPDs of HGA, TMIIG, and QCCA-INLS in Taillard benchmark instances. From Table 3, we can observe that the RPDs and the average RPDs of QCCA-INLS are all 0.00 in groups of 20 × 5, 20 × 10, 20 × 20, 50 × 5, 50 × 10, and 50 × 20, which means that the QCCA-INLS obtained the upper bounds when n ≤ 50. In groups of 100 × 5, 100 × 10, and 100 × 20, the average RPDs of QCCA-INLS are the smallest with 0.01, 0.02, 0.03, respectively. In addition, the average RPDs of TMIIG and QCCA-INLS are both 0.01, while that of TMIIG is 0.02. It implies that QCCA-INLS and TMIIG are equally effective.

Figure 15 shows the interactions between the compared algorithms and Taillard benchmark instances with 95.0%
| Instance | n × m | HGA RPD | TMIIG | QCCA-INLS | Instance | n × m | HGA RPD | TMIIG | QCCA-INLS | Instance | n × m | HGA RPD | TMIIG | QCCA-INLS |
|----------|-------|--------|-------|-----------|----------|-------|--------|-------|-----------|----------|-------|--------|-------|-----------|
| Ta01     | 20 × 5 | 0.00   | 0.00  | 0.00      | Ta11     | 20 × 10 | 0.00   | 0.00  | 0.00      | Ta21     | 20 × 20 | 0.00   | 0.00  | 0.00      |
| Ta02     | 20 × 5 | 0.00   | 0.00  | 0.00      | Ta12     | 20 × 10 | 0.00   | 0.00  | 0.00      | Ta22     | 20 × 20 | 0.00   | 0.00  | 0.00      |
| Ta03     | 20 × 5 | 0.00   | 0.00  | 0.00      | Ta13     | 20 × 10 | 0.00   | 0.00  | 0.00      | Ta23     | 20 × 20 | 0.00   | 0.00  | 0.00      |
| Ta04     | 20 × 5 | 0.00   | 0.00  | 0.00      | Ta14     | 20 × 10 | 0.00   | 0.00  | 0.00      | Ta24     | 20 × 20 | 0.00   | 0.00  | 0.00      |
| Ta05     | 20 × 5 | 0.00   | 0.00  | 0.00      | Ta15     | 20 × 10 | 0.00   | 0.00  | 0.00      | Ta25     | 20 × 20 | 0.00   | 0.00  | 0.00      |
| Ta06     | 20 × 5 | 0.00   | 0.00  | 0.00      | Ta16     | 20 × 10 | 0.00   | 0.00  | 0.00      | Ta26     | 20 × 20 | 0.00   | 0.00  | 0.00      |
| Ta07     | 20 × 5 | 0.00   | 0.00  | 0.00      | Ta17     | 20 × 10 | 0.00   | 0.00  | 0.00      | Ta27     | 20 × 20 | 0.00   | 0.00  | 0.00      |
| Ta08     | 20 × 5 | 0.00   | 0.00  | 0.00      | Ta18     | 20 × 10 | 0.00   | 0.00  | 0.00      | Ta28     | 20 × 20 | 0.00   | 0.00  | 0.00      |
| Ta09     | 20 × 5 | 0.00   | 0.00  | 0.00      | Ta19     | 20 × 10 | 0.00   | 0.00  | 0.00      | Ta29     | 20 × 20 | 0.00   | 0.00  | 0.00      |
| Ta10     | 20 × 5 | 0.00   | 0.00  | 0.00      | Ta20     | 20 × 10 | 0.00   | 0.00  | 0.00      | Ta30     | 20 × 20 | 0.00   | 0.00  | 0.00      |
| **Average** |       | 0.00   | 0.00  | 0.00      | **Average** |       | 0.00   | 0.00  | 0.00      | **Average** |       | 0.00   | 0.00  | 0.00      |

| Instance | n × m | HGA RPD | TMIIG | QCCA-INLS | Instance | n × m | HGA RPD | TMIIG | QCCA-INLS | Instance | n × m | HGA RPD | TMIIG | QCCA-INLS |
|----------|-------|--------|-------|-----------|----------|-------|--------|-------|-----------|----------|-------|--------|-------|-----------|
| Ta31     | 50 × 5 | 0.00   | 0.00  | 0.00      | Ta41     | 50 × 10 | 0.00   | 0.00  | 0.00      | Ta51     | 50 × 20 | 0.00   | 0.00  | 0.00      |
| Ta32     | 50 × 5 | 0.00   | 0.00  | 0.00      | Ta42     | 50 × 10 | 0.00   | 0.00  | 0.00      | Ta52     | 50 × 20 | 0.00   | 0.00  | 0.00      |
| Ta33     | 50 × 5 | 0.00   | 0.00  | 0.00      | Ta43     | 50 × 10 | 0.00   | 0.00  | 0.00      | Ta53     | 50 × 20 | 0.00   | 0.00  | 0.00      |
| Ta34     | 50 × 5 | 0.03   | 0.03  | 0.00      | Ta44     | 50 × 10 | 0.00   | 0.00  | 0.00      | Ta54     | 50 × 20 | 0.00   | 0.00  | 0.00      |
| Ta35     | 50 × 5 | 0.00   | 0.00  | 0.00      | Ta45     | 50 × 10 | 0.00   | 0.00  | 0.00      | Ta55     | 50 × 20 | 0.00   | 0.00  | 0.00      |
| Ta36     | 50 × 5 | 0.00   | 0.00  | 0.00      | Ta46     | 50 × 10 | 0.00   | 0.00  | 0.00      | Ta56     | 50 × 20 | 0.00   | 0.00  | 0.00      |
| Ta37     | 50 × 5 | 0.00   | 0.00  | 0.00      | Ta47     | 50 × 10 | 0.00   | 0.00  | 0.00      | Ta57     | 50 × 20 | 0.00   | 0.00  | 0.00      |
| Ta38     | 50 × 5 | 0.00   | 0.00  | 0.00      | Ta48     | 50 × 10 | 0.00   | 0.00  | 0.00      | Ta58     | 50 × 20 | 0.00   | 0.00  | 0.00      |
| Ta39     | 50 × 5 | 0.07   | 0.07  | 0.00      | Ta49     | 50 × 10 | 0.00   | 0.00  | 0.00      | Ta59     | 50 × 20 | 0.00   | 0.00  | 0.00      |
| Ta40     | 50 × 5 | 0.00   | 0.00  | 0.00      | Ta50     | 50 × 10 | 0.00   | 0.00  | 0.00      | Ta60     | 50 × 20 | 0.00   | 0.00  | 0.00      |
| **Average** | 0.01  | 0.01  | 0.00  | **Average** | 0.00   | 0.00  | 0.00  | **Average** | 0.00   | 0.00  | 0.00  | **Average** | 0.00   | 0.00  | 0.00  |

| Instance | n × m | HGA RPD | TMIIG | QCCA-INLS | Instance | n × m | HGA RPD | TMIIG | QCCA-INLS | Instance | n × m | HGA RPD | TMIIG | QCCA-INLS |
|----------|-------|--------|-------|-----------|----------|-------|--------|-------|-----------|----------|-------|--------|-------|-----------|
| Ta61     | 100 × 5 | 0.08  | 0.00  | 0.00      | Ta71     | 100 × 10 | 0.02  | 0.00  | 0.02      | Ta81     | 100 × 20 | 0.00   | 0.00  | 0.00      |
| Ta62     | 100 × 5 | 0.10  | 0.00  | 0.00      | Ta72     | 100 × 10 | 0.08  | 0.00  | 0.00      | Ta82     | 100 × 20 | 0.00   | 0.00  | 0.14      |

(Continues)
| Instance | $n \times m$ | HGA RPD | TMIIG | QCCA-INLS | Instance | $n \times m$ | HGA RPD | TMIIG | QCCA-INLS | Instance | $n \times m$ | HGA RPD | TMIIG | QCCA-INLS |
|----------|-------------|---------|-------|-----------|----------|-------------|---------|-------|-----------|----------|-------------|---------|-------|-----------|
| Ta63     | 100 $\times$ 5 | 0.29    | 0.10  | 0.00      | Ta73     | 100 $\times$ 10 | 0.00    | 0.00  | 0.00      | Ta83     | 100 $\times$ 20 | 0.00    | 0.00  | 0.00      |
| Ta64     | 100 $\times$ 5 | 0.00    | 0.02  | 0.00      | Ta74     | 100 $\times$ 10 | 0.12    | 0.12  | 0.00      | Ta84     | 100 $\times$ 20 | 0.00    | 0.00  | 0.05      |
| Ta65     | 100 $\times$ 5 | 0.00    | 0.03  | 0.03      | Ta75     | 100 $\times$ 10 | 0.10    | 0.10  | 0.00      | Ta85     | 100 $\times$ 20 | 0.00    | 0.00  | 0.00      |
| Ta66     | 100 $\times$ 5 | 0.21    | 0.00  | 0.00      | Ta76     | 100 $\times$ 10 | 0.17    | 0.00  | 0.17      | Ta86     | 100 $\times$ 20 | 0.06    | 0.18  | 0.00      |
| Ta67     | 100 $\times$ 5 | 0.13    | 0.00  | 0.08      | Ta77     | 100 $\times$ 10 | 0.03    | 0.03  | 0.00      | Ta87     | 100 $\times$ 20 | 0.09    | 0.00  | 0.09      |
| Ta68     | 100 $\times$ 5 | 0.08    | 0.08  | 0.00      | Ta78     | 100 $\times$ 10 | 0.00    | 0.00  | 0.00      | Ta88     | 100 $\times$ 20 | 0.01    | 0.00  | 0.01      |
| Ta69     | 100 $\times$ 5 | 0.17    | 0.00  | 0.00      | Ta79     | 100 $\times$ 10 | 0.06    | 0.00  | 0.00      | Ta89     | 100 $\times$ 20 | 0.19    | 0.19  | 0.00      |
| Ta70     | 100 $\times$ 5 | 0.05    | 0.00  | 0.00      | Ta80     | 100 $\times$ 10 | 0.04    | 0.00  | 0.00      | Ta90     | 100 $\times$ 20 | 0.00    | 0.00  | 0.00      |
| Average  |             | 0.11    | 0.02  | **0.01**  | Average  |             | 0.06    | **0.02** | **0.02**  | Average  |             | **0.03** | 0.04  | **0.03**  |
| Global average |     | 0.02   | **0.01** |          |           |             |         |         |         |

**TABLE 3** (Continued)

ZHU et al.
5 | CONCLUSIONS AND FUTURE WORKS

The authors have studied NWFSP for minimizing the makespan. Based on the characteristics and objective of the NWFSP, we proposed the QCCA-INLS framework which consists of QSC, QPE, and INLS. QSC generates population by quantum encode to enhance the population diversity. In the QPE, we developed a co-evolutionary procedure to search global solutions iteratively. To improve the intensification and efficiency of QCCA, quantum NOT is used to enhance the search to avoid falling into local optimization. For improving the quality of the solution, the INLS method is presented for searching optimized solution. INLS is based on neighbourhood search and local search, which can speed up the search. By comparing with three existing algorithms, QCCA and QCCA-INLS show better performance for the problem under study.

It is worthwhile to investigate mixed FSPs that consider the wait and no-wait constraints. It may commonly exist in practical industries and are promising topics to investigate in the future. Further refinements of the algorithm performance are also critical future avenues of research.

ACKNOWLEDGEMENTS

This work is sponsored by the National Key Research and Development Program of China (No. 2017YFB1400801), the National Natural Science Foundation of China (Nos. 61572127, 61872077, 61832004), and the Collaborative Innovation Center of Wireless Communications Technology.

ORCID

Haibong Zhu https://orcid.org/0000-0002-2740-4799

REFERENCES

1. Allahverdi, A.: A survey of scheduling problems with no-wait in process. Eur. J. Oper. Res. 255(3), 665–686 (2016)
2. Nagano, M.S., Miyata, H.H.: Review and classification of constructive heuristics mechanisms for no-wait flow shop problem. Int. J. Adv. Manuf. Technol. 86(5-8), 2161–2174 (2016)
3. Hoogeveen, J.A., Raaymakers, W.H.M.: Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. Eur. J. Oper. Res. 126(1), 131–151 (2000)
4. Gilmore, P.C., Gomory, R.E.: Sequencing a one-state-variable machine: a solvable case of the travelling salesman problem. Oper. Res. 12(5), 655–679 (1964)
5. Wismen, D.A.: Solution of the flowshop-scheduling problem with no intermediate queues. Oper. Res. 20(3), 689–697 (1972)
6. Ruiz, R., Allahverdi, A.: New heuristics for no-wait flow shops with a linear combination of makespan and maximum lateness. Int. J. Prod. Res. 47(20), 5717–5738 (2009)
7. Engin, O., Gündaydin, C.: An adaptive learning approach for no-wait flowshop scheduling problems to minimize makespan. Int. J. Comput. Intell. Syst. 4(4), 521–529 (2011)
8. Hall, N.G., Sriskandarajah, C.: A survey of machine scheduling problems with blocking and no-wait in process. Oper. Res. 44(3), 510–525 (1996)
9. Ye, H., Li, W., Abedini, A.: An improved heuristic for no-wait flow shop to minimize makespan. J. Manuf. Syst. 44, 273–279 (2017)
10. Ding, J., et al.: Accelerated methods for total tardiness minimisation in no-wait flowshops. Int. J. Prod. Res. 53(4), 1002–1018 (2015)
11. Qi, X., et al.: Fast local neighborhood search algorithm for the no-wait flow shop scheduling with total flow time minimization. Int. J. Prod. Res. 54(16), 4957–4972 (2016)
12. Röck, H.: The three-machine no-wait flow shop is np-complete. J. ACM. 31(2), 336–345 (1984)
13. Allahverdi, A., Aldowaisan, T.: No-wait and separate setup three-machine flowshop with total completion time criterion. Int. Trans. Oper. Res. 7(3), 245–264 (2000)

14. Allahverdi, A., Aldowaisan, T.: Minimizing total completion time in a no-wait flowshop with sequence-dependent additive changeover times. J. Oper. Res. Soc. 52(4), 449–462 (2001)

15. Aldowaisan, T., Allahverdi, A.: New heuristics for m-machine no-wait flowshop to minimize total completion time. Omega. 32(5), 345–352 (2004)

16. Li, X., Wang, Q., Wu, C.: Heuristic for no-wait flowshops with makespan minimization. Int. J. Prod. Res. 46(9), 2519–2530 (2008)

17. Gao, K., Pan, Q., Li, J., Suganthan, P.N.: 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 (2013)

18. Sapkal, S.U., Laha, D.: A heuristic for no-wait flow shop scheduling. Int. J. Adv. Manuf. Technol. 68(5-8), 1327–1338 (2013)

19. Qian, B., Wang, L., Hu, R., Huang, D., Wang, X.: A de-based approach to no-wait flow-shop scheduling. Comput. Ind. Eng. 57(3), 787–805 (2009)

20. Pan, Q.K., Tasgetiren, M.F., Liang, Y.C.: A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Comput. Oper. Res. 35(9), 2807–2839 (2008)

21. Tseng, L.Y., Lin, Y.T.: A hybrid genetic algorithm for no-wait flowshop scheduling problem. Int. J. Prod. Econ. 128(1), 144–152 (2010)

22. Jarboui, B., Eddaly, M., Siarry, P.: A hybrid genetic algorithm for solving no-wait flowshop scheduling problems. Int. J. Adv. Manuf. Technol. 54(9-12), 1129–1143 (2011)

23. Samarghandi, H., ElMekkawy, T.Y.: A meta-heuristic approach for solving the no-wait flow-shop problem. Int. J. Prod. Res. 50(24), 7313–7326 (2012)

24. Aktouf, H., et al.: New greedy randomized adaptive search procedure based on differential evolution algorithm for solving no-wait flow-shop scheduling problem. In: Proceedings of the International Conference on Advanced Logistics and Transport, pp. 327–334. IEEE (2013)

25. Ding, J.Y., et al.: An improved iterated greedy algorithm with a tabu-based reconstruction strategy for the no-wait flowshop scheduling problem. Appl. Soft. Comput. 30, 604–613 (2015)

26. Engin, O., Guclu, A.: A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems. Appl. Soft. Comput. 72, 166–176 (2018)

27. Zhao, F., et al.: A factorial based particle swarm optimization with a population adaptation mechanism for the no-wait flow shop scheduling problem with the makespan objective. Expert Syst. Appl. 126, 41–53 (2019)

28. Lin, S.W., Ying, K.C.: Optimization of makespan for no-wait flowshop scheduling problems using efficient heuristics. Omega. 64, 115–125 (2016)

29. Li, X., et al.: An iterated greedy heuristic for no-wait flow shops with sequence dependent setup times, learning and forgetting effects. Inf. Sci. 453, 408–425 (2018)

30. Wang, Y., et al.: An iterated greedy heuristic for mixed no-wait flowshop problems. IEEE Trans. Cyber. 48(5), 1553–1566 (2018)

31. Shao, W., Pi, D., Shao, Z.: 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), 1344–1360 (2019)

32. Han, K.H., Kim, J.H.: Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Trans. Evol. Comput. 6(6), 580–593 (2002)

33. Nezamabadi Pour, H.: A quantum-inspired gravitational search algorithm for binary encoded optimization problems. Eng. Appl. Artif. Intell. 40, 62–75 (2015)

34. Draa, A., et al.: A quantum-inspired differential evolution algorithm for solving the n-queens problem. Int. Arab. J. Inf. Technol. 7(1), 21–27 (2010)

35. Yang, X.S., Deb, S.: Cuckoo search via levy flights. In: Proceedings of 2009 World Congress on Nature & Biologically Inspired Computing, pp. 210–214 (2009)

36. Zhu, H., et al.: Quantum-inspired cuckoo co-search algorithm for no-wait flow shop scheduling. Appl. Intell. 49(2), 791–803 (2019)

37. Cao, Y., et al.: Comprehensive learning particle swarm optimization algorithm with local search for multimodal functions. IEEE Trans. Evol. Comput. 23(4), 718–731 (2018)

How to cite this article: Zhu H, Luo N, Li X. A quantum-inspired cuckoo co-evolutionary algorithm for no-wait flow shop scheduling. IET Collab. Intell. Manuf. 2021;3:105–118. https://doi.org/10.1049/cim2.12002