An energy-efficient multi-objective permutation flow shop scheduling problem using an improved hybrid cuckoo search algorithm

Wenbin Gu¹, Zhuo Li², Min Dai³ and Minghai Yuan¹

Abstract
The flow shop scheduling problem has been widely studied in recent years, but the research on multi-objective flow shop scheduling with green indicators is still relatively limited. It is urgent to strengthen the research on effective methods to solve such interesting problems. To consider the economic and environmental factors simultaneously, the paper investigates the multi-objective permutation flow shop scheduling problems (MOPFSP) which minimizes the makespan and total carbon emissions. Since MOPFSP is proved to be a NP-hard problem for more than two machines. A hybrid cuckoo search algorithm (HCSA) is proposed to solve the problems. Firstly, a largest-order-value method is proposed to enhance the performance of HCS algorithm in the solution space of MOPFSP. Then, an adaptive factor of step size is designed to control the search scopes in the evolution phases. Finally, a multi-neighborhood local search rule is addressed in order to find the optimal sub-regions obtained by the HCSA. Numerical experiments show that HCSA can solve MOPFSP efficiently.

Keywords
Multi-objective, permutation flow shop, cuckoo search algorithm, carbon efficiency, green dispatch

Date received: 27 December 2020; accepted: 10 May 2021

Handling Editor: James Baldwin

Introduction
Recently, with the increasingly severe environmental problems in the manufacturing industry, the green manufacturing has received considerable attention.¹,² The problem model studied in this paper contains environmental indicators, which meets the current needs of ecological environment governance. According to a survey by Fang et al.,³ most of the world’s energy consumption comes from the industrial sector, and manufacturing companies have become one of the main factors for global warming. On the one hand, the control of greenhouse gas emissions by laws and regulations makes manufacturing companies have to limit carbon emissions; on the other hand, the high taxes brought by carbon emissions also make manufacturers seek practical and feasible ways to processed products.⁴

Scheduling time, processing cost, and product quality are the three most common optimization indicators

¹Department of Mechanical and Electrical Engineering, Hohai University, Changzhou, China
²Department of Mechanical Engineering and Science, Huazhong University of Science and Technology, Wuhan, China
³College of Mechanical Engineering, Yangzhou University, Yangzhou, China

Corresponding author:
Wenbin Gu, Department of Mechanical and Electrical Engineering, Hohai University, No. 200 Jinling Road, Changzhou 213022, China.
Email: guwenbing@hhuc.edu.cn

Creative Commons CC BY. This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
for traditional production scheduling problem.\textsuperscript{5} When the environmental and energy-saving problems are increasingly severe in the manufacturing industry, green indicators and economic indicators should be considered at the same time in order to meet the scheduling demands. The multi-objective permutation flow shop problem (MOPFSP) with green indicators studied in this paper has a strong industrial background.\textsuperscript{6} In terms of computational complexity, the permutation flow shop scheduling problem (PFSP) of more than two machines has been proved to be an NP-hard problem,\textsuperscript{7} so MOPFSP also belongs to the NP-hard problem. In summary, it is of great engineering and academic significance to solve the MOPFSP with green indicators.

In the past few decades, the MOPFSP and green manufacturing has been studied extensively. However, only few researchers have considered both economic and environmental indicators. Luo et al.\textsuperscript{8} designed an ant colony optimization method to address the multi-objective mixed FSP with power consumption. Ding et al.\textsuperscript{9} studied the multi-objective flow shop scheduling problem with total carbon emissions, and proposed an improved iterative greedy algorithm. Liu et al.\textsuperscript{10} proposed an adaptive multi-objective genetic algorithm that can effectively address a class of pipeline shop scheduling problems with two optimization indicators of carbon emissions and total weighted delay. Ding et al.\textsuperscript{11} addressed an iterative greedy algorithm based on non-dominated solution structure characteristics for solving the dual-objective flow shop scheduling problem with total carbon emissions. Tang et al.\textsuperscript{12} presented a modified particle swarm optimization algorithm, which can solve the flexible flow shop scheduling problem based on the two goals of energy consumption and maximum completion time. Lu\textsuperscript{13} proposed a hybrid multi-objective backtracking search algorithm based on machine setup time and workpiece transportation time to solve the MOPFSP with energy consumption. Wang and Tian\textsuperscript{14} established a dual-objective optimization model for the selection of milling parameters such that power consumption and process time are minimized, and proposed an improved artificial bee colony (ABC) intelligent algorithm to handle the dual-objective optimization mode. Fu and Tian\textsuperscript{15} proposed a scheduling model can make an interaction between the energy consumption and the production cost to realize an efficient and sustainable production process. In summary, the research on MOPFSP with green indicators is still relatively limited, and it is urgent to develop effective methods for solving such important problems.

The CS (cuckoo search) algorithm is an algorithm that effectively solves the optimization problem by simulating the brood parasitism of some species of cuckoos that proposed by Yang and Deb.\textsuperscript{16} This algorithm searches according to the flight mechanism of cuckoos at the time of spawning, which can quickly and efficiently solve continuous optimization problems.\textsuperscript{17} Compared with other meta-heuristic algorithms, Cuckoo’s algorithm has better performance and efficiency in solving multi-objective optimization problems. Sheli and Chaudhuri\textsuperscript{18} verified the CS algorithm and compared with some classical Algorithms: GAs and PSO. From simulation and comparison, it is evident that CS is much better and efficient than the GAs and PSO for multi-objective functions. This is because in the case of CS, there are fewer parameters that require precise control. It is clear that once the number of nests is fixed, there is only one parameter discovery probability that needs to be adjusted. It should be noted that in the absence of discovery probability interference, the CS convergence speed of all the above test functions is very fast and the efficiency is high. Therefore, compared with other meta-heuristic algorithms, CS is more versatile and robust in many optimization problems. The robust optimization algorithm can be easily applied to multi-objective optimization problems. In recent years, the CS algorithm has also been extended to solve single-objective production scheduling problems with economic indicators. Li and Yin\textsuperscript{19} used NEH heuristics to generate part of the initial population and designed a hybrid CS algorithm with local search to solve the single-objective PFSP problem. Marichel et al.\textsuperscript{20} designed NEH rules to initialize part of the population at the initial stage of the CS algorithm, and then used it to solve a single-objective multi-stage mixed-line shop floor scheduling problem. Alaa and Alobaidi\textsuperscript{21} improved the Lévy flight equation in the CS algorithm, and enhanced the search for the optimal individual neighborhood of the population. The proposed algorithm was used to solve the single-objective flexible job shop scheduling problem. Wang et al.\textsuperscript{22} used the NEH rule to generate part of the initial population, and then proposed a Cuckoo algorithm with local search for solving a single-target pipeline shop scheduling problem. Zhang et al.\textsuperscript{23} proposed a dynamic adaptive cuckoo search algorithm (DACS). They introduced feedback into the algorithm framework, established a closed-loop control system for the parameters of the CS algorithm, and used Rechenberg’s 1/5 rule as an evolutionary evaluation index. However, the above literatures set the step size control factor to a certain constant and did not adjust it dynamically. In fact, as the number of iterations of algorithm increases, if the step size control factor is set too large, the cuckoo algorithm’s search tends to deviate high quality solution region, resulting poor performance; if the step size control factor is set too small, the algorithm is easy to fall into a local optimum and premature. Therefore, it is of great significance to design cuckoo algorithm that can dynamically adjust step size control factor to solve the green flow shop scheduling problem.
A hybrid cuckoo search algorithm is proposed to solve MOPFSP with the objective of minimizing the completion time and total carbon emissions in this paper. On one hand, an adaptive adjust strategy for step size control factor is presented to make the cuckoo algorithm have faster convergence speed and better global search performance; on the other hand, a multi-neighbor local search mechanism is also designed to improve the performance of HCS algorithm. To this end, a better balance between global and local search of the algorithm is achieved. Finally, the effectiveness of HCS is verified through simulation experiments and algorithm comparison.

Problem statement

Low carbon MOPFSP descriptions

Production scheduling is to allocate a set of machining tasks in time to an available set of machining machines to meet a performance index.\(^24\) It mainly consists of two factors: performance indicator and scheduling schemes.\(^{25,26}\) The performance indicators of production scheduling are classified into three categories: maximum capacity indicators, cost indicators, and customer satisfaction indicators.\(^{27}\) Among them, the maximum capacity indicators include minimizing the maximum completion time and maximum productivity. The cost indicators include minimizing operating costs and maximizing revenue.\(^{28}\) Customers satisfaction indicators include minimizing early or late penalties. Additionally, green indicators like energy consumption, carbon emissions have been considered in recent years.

The scheduling scheme generally represents a solution to a production scheduling problem, which is the arrangement of the processing order of processing workpieces on various machines.\(^{29}\) It can be usually expressed by a Gantt chart, as shown in Figure 1.

![Gantt chart](image)

Figure 1. Gantt chart.

According to the description of the mentioned above, the MOPFSP with low carbon can be introduced as follows. There are \(m\) machines in the shop floor. The workpieces set is \(J = \{1, 2, \ldots, n\}\). The machine set is \(M = \{1, 2, \ldots, m\}\). The speed set is \(S = \{v_1, v_2, \ldots, v_s\}\). \(T_{i,k}\) represents the standard machining time of workpiece \(i\) on machine \(k\); \(V_{i,k}\) represents the machining speed of workpiece \(i\) on machine \(k\); \(T_{i,k}/V_{i,k}\) represents the real machining time of workpiece \(i\) when machining on machine \(k\) at speed \(V_{i,k}\). \(Q_{k,v}\) represents the unit energy consumption when machine \(k\) works at speed \(v\); \(Q_{k}\) represents the unit energy consumption when machine \(k\) is in standby state. When machine \(k\) works at speed \(v\) at time \(t\), \(x_{k}(t) = 1\), and otherwise, \(x_{k}(t) = 0\); at time \(t\), the machine is in standby state \(y_{k}(t) = 1\), and otherwise, \(y_{k}(t) = 0\); \(\pi = (j_1,j_2,\ldots,j_n)\) is some sort of all workpieces; \(\prod\) is different sort the set of \(\pi\); \(C(j,k) = (1, 2, \ldots, n)\) is the completion time of workpiece \(j\) on machine \(k\).

Two processing rules of PFSP are given: (1) After the workpiece \(i \in J\) is processed on the machine \((k - 1) \in M\), the workpiece \(j \in J\) can be processed on the machine \(k \in M\). (2) Each machine cannot process multiple workpieces at the same time. Assume that the workpieces \(j_1\) to \(j_m\) are sequentially processed on machine \(1\) to machine \(m\) in order. Therefore, the multi-objective models of PFSP can be described based on economic and environmental indicators.

For the economic indicator, the longest completion time is considered. Its mathematical model is written as

\[
C_{\text{max}} = C(j_n, m) \tag{1}
\]

For the green indicator, assuming \(e\) is the amount of CO\(_2\) emissions per unit energy consumption, the total CO\(_2\) emissions calculation equation is considered as:

\[
C_{\text{CO}_2} = eC_T = e \int_{0}^{C_{\text{max}}} \left( \sum_{v=1}^{s} \sum_{k=1}^{m} Q_{k,v}x_{kv}(t) dt \right) + \sum_{k=1}^{m} Q_{k}y_{k}(t) \tag{2}
\]

Among them, \(C_T\) represents total energy consumption; \(C_{\text{CO}_2}\) represents total CO\(_2\) emissions, hereinafter referred to as total carbon emissions (TCE).

In order to calculate the total CO\(_2\) emission, a rate matrix \(A_{n \times m}\) for the machine needs to be constructed, where \(A_{i,k} \in \{1, 2, \ldots, s\}\). Assuming \(s = 3\), \(A_{i,k} \in \{1, 2, 3\}\), \(n = 3\), \(m = 2\), then one of the cases of matrix \(A\) is expresses as follows:

\[
A_{3 \times 2} = \begin{bmatrix}
1 & 2 \\
2 & 2 \\
3 & 1
\end{bmatrix} \tag{3}
\]
Where $A_{31}$ represents the machining speed of third workpieces when machining on first machine is 3.

Furthermore, the constraints of the low carbon MPFSP are given in the following.

$$C(j_1, 1) = T_{j_1, 1}/V_{j_1, 1}$$  
(4)

$$C(j_i, 1) = C(j_{i-1}, 1) + T_{j_i, 1}/V_{j_i-1}$$  
(5)

$$C(j_1, k) = C(j_1, k - 1) + T_{j_i, k}/V_{j_i,k}$$  
(6)

$$C(j_i, k) = \text{max}\{C(j_{i-1}, k), C(j_i, k - 1)\} + T_{j_i, k}/V_{j_i,k}$$  
(7)

For the low carbon MPFSP, it is impossible to obtain the global optimal solution of makespan and TCE simultaneously. But a set of solutions can be obtained: the continued optimization of any one target must sacrifice the effective solution of other objective function values.\(^{27,30}\) The set of such effective solutions is called a non-inferior solution, which is a Pareto solution set.

**Related concepts of multi-objective problem**

In this article, MOPFSP is described as follows:

$$\min F(\pi) = [f_1(\pi), f_2(\pi)], \pi \in \psi$$  
(8)

$$f_1 = C_{\max}$$  
(9)

$$f_2 = C_{\text{co}2}$$  
(10)

Where $\psi$ is feasible region.

(1) Dominate.

For the target vectors $F(\pi_1) = [f_1(\pi_1), f_2(\pi_1)]$ and $F(\pi_2) = [f_1(\pi_2), f_2(\pi_2)]$, if and only if $\forall i \in \{1, 2\} : f_i(\pi_1) \leq f_i(\pi_2)$ and $\exists i \in \{1, 2\} : f_i(\pi_1) < f_i(\pi_2)$, we call $F(\pi_1)$ dominates $F(\pi_2)$ and denote it as $F(\pi_1) < F(\pi_2)$.

(2) Pareto optimal solution.

For $\pi \in \psi$, if and only if $\pi' \in \psi$ makes that $F(\pi') < F(\pi)$ does not exist, $\pi$ is the Pareto optimal solution.

(3) Pareto frontier.

For Pareto frontier $P_F$: $P_F = \{F(\pi) = (f_1(\pi_2), f_2(\pi_2)) | \pi \in P\}$.

(4) Pareto solution set.

For Pareto solution set $P$: $P = \{\pi \in \psi | \exists \pi' \in \psi : F(\pi') < F(\pi)\}$.

**Hybrid cuckoo algorithm**

**Standard Cuckoo algorithm**

The CS (cuckoo search) algorithm can effectively solve the optimization problem by simulating the brood parasitism of some species of cuckoos.\(^{31}\) Lévy-flights is a kind of action mode, the step size of random walk satisfies a heavy-tailed stable distribution. In this form of walking, the exploratory jumps of short distance and occasionally the longer distances working are interspersed. The trajectory of Lévy-flights on a two-dimensional plane is shown in Figure 2.\(^{32}\)

![Figure 2. Lévy-flights motion diagram on a two-dimensional plane.](image)

The flight of many animals in nature conforms to the Lévy-flights model, and the flight of cuckoo is one of them.\(^{33}\) When the cuckoo is searching, its flight path is the combination of various long and short distances. Each distance differs from the previous distances by a small angle.\(^{34}\) Short distances occur most frequently, while the longer distances are rare. Lévy-flights may seem disorganized, but it is not. The distances and the angle of deviations follow a very definite statistical distribution. Through long-term observation of the cuckoo’s living habits, it is found that cuckoos only lay eggs and do not build nests.\(^{35}\) And their eggs often lay in the nests of different birds. Inspired by this nature phenomenon, a new search algorithm—cuckoo search algorithm (CS) in cuckoo’s nest-seeking behavior is proposed.\(^{36}\) Each nest represents a potential solution. The host bird may find that the egg is not its own egg. It may abandon the egg and or the entire nest. Cuckoo birds lay eggs in this way, so that the nest position is continuously optimized.
Table 1. The standard cuckoo search algorithm.

| Begin |
|---|
| Objective function \( f(x), x = (x_1, x_2, \ldots, x_d) \) |
| Generate initial population of \( n \) host nests \( x_i = (i = 1, 2, \ldots, n) \) |
| While (\( t < \text{Max Generation} \) or (stop criterion)) |
| Get a cuckoo randomly by Levy flights |
| Evaluate its quality/fitness \( F_i \) |
| Choose a nest among \( n \) randomly |
| If \( (F_i > F_j) \) replace \( j \) by the new solution; |
| End |
| A fraction \( (p_a) \) of worse nests are abandoned and new ones are built; |
| Keep the best solutions (or nests with quality solutions); |
| Rank the solutions and find the current best |
| End while |
| Postprocess results and visualization |
| End |

The pseudocode of the standard cuckoo search algorithm is given in Table 1. The flowchart of the standard CS algorithm is given in Figure 3.

There are two ways to update individuals in the cuckoo search algorithm, the first is to use Levi’s flight equation:

\[
\text{Cuckoo finds new nest location with a certain probability} \\
\]  

\[
x_{i,t+1} = x_{i,t} + \alpha \odot \text{Levy}(\lambda) \quad (11)
\]

Among them, \( x_{i,t} \) and \( x_{i,t+1} \) are the position vectors of the \( i \)th individual at \( t \)th and \((k + 1)\)th generations; \( \odot \) represents a point multiplication; \( \alpha \) is a step factor controlling the step size, and in most cases, \( \alpha = O(1) \); \( \text{Levy}(\lambda) \) is levy flight search path.

As can be seen from equation (11), this distribution makes a continuous distribution of cuckoo birds form a probability distribution with heavy tails, which can expend the search range, increase population diversity, and easily converge to global optimum.

The second update method is to determine whether to generate new individuals according to the relationship between a fixed discovery probability \( p_a \) and a random number \( \beta \), the update equation is as follows:

\[
x_{i,t+1} = x_{i,t} + \gamma H(P_a - \beta) \odot |x_{o,t} - x_{k,t}| \quad (12)
\]

Among them, \( \gamma \) and \( \beta \) both obey uniform distribution and \( \gamma, \beta \in [0, 1] \); \( x_{i,t}, x_{o,t}, x_{k,t} \) are three different random in the individuals of \( t \)th generation; \( H \) is the Heaviside function, and its calculation equation is:

\[
H(P_a - B) = \begin{cases} 
0 & (P_a > B) \\
0.5 & (P_a = B) \\
1 & (P_a < B)
\end{cases}
\]

Hybrid cuckoo algorithm

Adaptive step size factor. Although the CS algorithm is more effective than other swarm optimization algorithms, it has its own inherent shortcomings: Levi’s flight is a Markov chain, which is only related to the current situation and has large randomness. Therefore, convergence accuracy and search depth are two shortcomings of the CS algorithm. The CS algorithm defines a step size control factor \( \alpha \) in equation (11), which is generally set to a fixed constant in the standard algorithm (e.g. the value is usually 0.01). If \( \alpha \) is set too large, it will easily cause the search of algorithm deviate from high-quality solution, which will slow down the convergence speed; on the contrary, if \( \alpha \) is set too small, the CS algorithm will easily get into the local optimal solution prematurely, resulting in weak algorithm performance. Therefore, the improvement of the \( \alpha \) can enhance the performance of algorithm effectively. If a large \( \alpha \) is used in the early stage of algorithm search, it is beneficial to easily locate the area of high-quality solution in the global scope; at the same time, as the algorithm search advances, the \( \alpha \) should be gradually reduced to strengthen the fine search of the local optimal solution region in order to improve the algorithm performance and convergence speed. This paper modified the standard CS algorithm from the step size control factor: replace the original fixed step size control factor with a dynamic step size control factor.

In this paper, the standard cuckoo search algorithm is improved from step size control factors: replacing fixed step size control factors with dynamic step size control factors. In the optimization process, when the
Individual quality gradually improves, the search scope is appropriately reduced to strengthen the search depth, which is conducive to searching for better solutions. The reasonable step size control factor should gradually decrease with the increase of the evolutionary algebra, so that the algorithm can easily find high-quality individuals in the later stage of evolution.

This paper adaptively adjusts the step size control factor $\alpha$ according to the following aspects: the cosine function is introduced to make a decrease as the evolution generation increases and an improved equation for $\alpha$ is proposed:

$$
\alpha^{(k+1)} = \begin{cases} 
\alpha_{\text{max}} \cos \left( \frac{\pi}{2} \frac{k}{T_{\text{max}}} \right) & R \leq 0.2 \\
\alpha_k & 0.2 < R < 0.5 \\
\alpha_{\text{min}} & 0.5 \leq R 
\end{cases}
$$

Where, $R$ is the ratio of the current evolution generation to the total evolution generation; $\alpha_{\text{max}}$ is the lower limit of $\alpha$; $\alpha_{\text{max}}$ is the upper limit of $\alpha$; $T_{\text{max}}$ is the maximum number of iterations; $k$ is evolution generation. At the beginning of HCS algorithm, $R \approx 0.2$. At this time, there should be a large step to find the high-quality solution area. Therefore, the step control factor $\alpha$ should gradually decreases as the evolutionary algebra increases; when $0.2 < R < 5$, the further search will be applied in the high-quality solution area, and should strengthen the local fine search ($\alpha$ remains unchanged); at the later stage of the algorithm, the $R > 5$. At this time, the individual gradually approaches the Pareto front, without the need for large steps, so keep the $\alpha$ in lower bound.

**Multi-neighborhood local search.** In order to further improve local search accuracy of the cuckoo algorithm, a multi-neighborhood local search strategy is proposed in this paper to perform fine search based on different neighborhoods. Specifically, it performs a local search based on three neighborhoods on individuals in the current non-inferior solution set of the algorithm. The three neighborhood searches are: interchange local search, insert local search, 2-opt local search. The specific definitions are as follows:

Interchange local search: Sort each workpiece, randomly select two different locations, and exchange the workpiece at the location. For example, 10 workpieces are sorted as [4, 2, 7, 1, 3, 5, 9, 8, 10, 6], and two positions $p_1 = 3$ and $p_2 = 9$ are generated randomly. Nine workpieces 10 swap positions and get a new sort [4, 2, 10, 1, 3, 5, 9, 8, 7, 6]. Then, the workpiece 7 at position 3 and the workpiece 10 at position 9 are exchanged to obtain a new order [4, 2, 10, 1, 3, 5, 9, 8, 7, 6].

Insert local search: This step can be divided into front insert and post insert. Sort the workpieces of each individual and randomly select two different positions $p_1$ and $p_2$, assuming $p_1 > p_2$. Post-insertion means inserting the workpiece at position $p_1$ into position $p_2$, and the workpieces at positions $p_1 + 1 \sim p_2$. Move forward one position; Insert forward refers to insert the workpiece of position $p_2$ into position $p_1$, and the workpieces of positions $p_1 \sim p_2 - 1$ are moved backward one position. For example, the 10 workpieces are sorted as [4, 2, 7, 1, 3, 5, 9, 8, 10, 6], and two positions $p_1 = 3$ and $p_2 = 9$ are randomly generated. According to the above, after inserting, the new order is [4, 2, 1, 3, 5, 9, 8, 10, 7, 6], and the new order obtained by the previous insertion is [4, 2, 10, 7, 1, 3, 5, 9, 8, 6].

2-opt local search: Sort each workpiece, randomly select two different positions $p_1$ and $p_2$, and sort the workpieces of $p_1 \sim p_2$ in reverse order. For example, 10 workpieces are sorted as [4, 2, 7, 1, 3, 5, 9, 8, 10, 6], and two positions $p_1 = 3$ and $p_2 = 9$ are randomly generated. According to the above, the new order is [4, 2, 10, 9, 5, 3, 1, 7, 6].

Let $\pi(X)$ be the workpiece ordering of the individual $X$ based on the LOV rule, $\pi(X')$ be the workpiece ordering of the individual $X'$ based on the LOV rule, and $k$ is the number of perturbations or explorations. The specific steps of performing a multi-neighborhood local search on individual $X$ are as follows:

1. The disturbance phase. ① Let $k = 0$; ② randomly select 2 different positions $p_1$ and $p_2$, $\pi(X) = \text{Insert}(\pi(X), p_1, p_2)$, $k = k + 1$; ③ If $k < 2$, return to step ②.

2. Exploration phase. ① Set $k = 0, t = 0$; ② randomly choose two different positions $p_1$ and $p_2$; if $t = 0$, $\pi(X') = \text{Insert}(\pi(X), p_1, p_2)$, otherwise if $t = 1$, $\pi(X') = \text{Interchange}(\pi(X), p_1, p_2)$, otherwise if $t = 2$, $\pi(X') = 2 - \text{opt}(\pi(X), p_1, p_2)$; ③ if $\pi(X') < \pi(X)$, then $\pi(X) = \pi(X')$, otherwise $t = t + 1$; ④ if $k < 30$, skip to step ③. Otherwise, stop exploring and output $\pi(X)$ and $X$; ⑤ If $t < 3$, return to step ②, otherwise $t = 0$, return to step ②.

**Steps of HCS algorithm to solve MOPFSP.** The main steps to solve MOPFSP based on the improved HCS algorithm are as follows:

1. Parameter initialization. Set the population size $N$; the upper and lower bounds of individuals, and initialize the population $W$ within the bounds; set the maximum number of iterations $g_{\text{ence}}$ or the algorithm running time $T_r$.

2. Individual discretization. The LOV rule is used to transform continuous individuals into discrete sorting; two objective function values for each individual are calculated.
Table 2. Test function.

| Function | Dimension | Range         | Theoretical value |
|----------|-----------|---------------|-------------------|
| $f_1(x) = \sum_{i=1}^{d-1} [(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2]$ | 10        | $[-100, 100]$ | 0                 |
| $f_2(x) = \sum_{i=1}^{d} x_i^2$ | 50        | $[-5.12, 5.12]$ | 0                 |
| $f_3(x) = 10d + \sum_{i=1}^{d} \left|x_i^2 - 10 \cos(2\pi x_i)\right|$ | 20        | $[-5.12, 5.12]$ | 0                 |
| $f_4(x) = \sum_{i=1}^{n} \left|x_i^2 - 10 \cos(2\pi x_i) + 10\right|$ | 10        | $[-5.12, 5.12]$ | 0                 |
| $f_5(x) = -\cos(x) \cos(y) \exp[-(x - \pi)^2 - (y - \pi)^2]$ | 2         | $[-100, 100]$ | $-1$              |
| $f_6(x) = 1 + \frac{1}{4000} \sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right)$ | 100       | $[-600, 600]$ | 0                 |

(3) Individual update. An individual $x_i$ is randomly selected, and the individual is updated using Levi’s flight according to equation (11) to generate a new individual $x_{i1}$; the new and old individuals are optimized using non-dominated principle. If they do not control each other, then randomly retained one and stored in $x_{i1}$.

(4) Application of abandonment probability. Determine whether to operate on the individual $x_j$ retained in step (3) according to the discard probability. If you want to operate it, use equation (12) to update and obtain a new individual $x_{j2}$. Finally, use the non-dominated principle to optimize the new and old individuals. If they do not control each other, then randomly retained one and stored in $x_{j1}$.

(5) Keep the Pareto front. The non-dominated principle is used to find the Pareto front of the current generation, and the individuals of the Pareto front of the current generation are stored in the Pareto solution set $P$.

(6) Multi-neighborhood search. The multi-neighborhood search is performed on the sort order corresponding to each individual $X_{p_t}$ in the current generation set $P_{t-1}$ to obtain the updated Pareto solution set $P_{t}$ of this generation.

(7) Record the current Pareto solution set. Fuse the updated Pareto solution set $P_{t}$ and the Pareto solution set $P$ retained in the ($t-1$)th generation, use the non-dominated principle to find the current $t$-generation Pareto solution set $P$, and use this set $P$ instead of the current generation of inferior solution set $P_{t}$ for easy use in the next generation.

(8) The step size control factor $\alpha$ is updated according to equation (14). Each generation must judge and update the step size control factor.

(9) Termination conditions. If the current number of iterations is less than the maximum number of iterations $g_{\text{max}}$ or the evolution time is less than the algorithm running time $T_e$, repeat steps (2)–(8); otherwise output the current Pareto set $P$ and end the algorithm.

Based on the above steps, the algorithm flow chart of using HC algorithm to solve MOPFSP is shown in Figure 4.

Numerical experiments

In order to verify that the algorithm in this chapter is feasible and effective, this chapter first selects six classic benchmark test functions for verification as shown in Table 2.

In this experiment, a simulation program written in MATLAB 7.1 was used. The operating environment was a Windows 7 professional operating system. The hardware configuration was a Celeron (R) Dual-core CPU T3100, a 1.90 GHZ processor, and a 2G memory laptop.

Among the six benchmark test functions, there are both high-dimensional and low-dimensional functions. Perform 20 independent experiments on each of them to find the minimum, maximum, average, and variance, and compare them with the basic cuckoo algorithm. It can be seen from Table 3 that for the minimum, maximum, or average value of HCS, the calculation accuracy is significantly improved compared with the basic CS. Meanwhile, the optimal solution obtained by HCS is closer to the theory value. The optimal values of functions $f_1$ and $f_6$ are increased by $10^{-2}$, the optimal value of function $f_2$ is increased by $10^{-1}$, and the optimal value of function $f_3$ is increased by $10^{-3}$, especially the performance of function $f_5$ is more obvious, and the theoretical optimal value is reached each time. From
the experimental results, the HCS algorithm proposed in this paper performs well.

The following shows the convergence curve of the average fitness of the six test functions in 20 independent experiments. The ordinate of Figure 9 is the average fitness, while the other graphs are all the log of the average fitness.

Figures 5 to 10 are the convergence curves of the HCS and CS algorithms in this chapter. It can be intuitively seen that the HCS algorithm converges faster than the CS algorithm and has fewer iterations.

Table 3. Experimental comparison between CS and HCS.

| Function | Algorithm | Minimum      | Maximum      | Average   | Variance      |
|----------|-----------|--------------|--------------|-----------|---------------|
| $f_1$    | CS        | 462.9039     | 3.1197e+00   | 1.4446e+00| 6.7116e+00   |
|          | HCS       | 4.8562       | 158.8805     | 25.4291   | 1.4115e+00   |
| $f_2$    | CS        | 7.4681       | 18.0782      | 11.7679   | 8.1463       |
|          | HCS       | 0.8318       | 3.0577       | 1.8499    | 0.3465       |
| $f_3$    | CS        | 111.8739     | 131.4907     | 120.5582  | 32.4853      |
|          | HCS       | 106.2796     | 133.0725     | 114.5563  | 42.6203      |
| $f_4$    | CS        | 2.4898       | 5.1542       | 3.3210    | 0.5428       |
|          | HCS       | 1.5285e-005  | 1.1551       | 0.0580    | 0.0667       |
| $f_5$    | CS        | -1.0000      | -0.9952      | -0.9996   | 1.1418e-06   |
|          | HCS       | -1           | -1           | -1        | 0            |
| $f_6$    | CS        | 121.0992     | 199.6183     | 150.3869  | 511.1894     |
|          | HCS       | 82.7011      | 147.4886     | 112.0223  | 224.0648     |

To further verify the effectiveness and superiority of the HCS algorithm to solve MOPFSP, this paper selects 10 kinds of test instances with different sizes, and uses the standard CS algorithm and INSGA-II algorithm for comparison experiments.

In the test problem, the processing time of the workpiece on each machine is generated according to the size of the problem using random number ranging from [1, 100]; the machine speed gear was set to $A_{i,k} \in \{1, 2, 3\}$. In order to find the HCS algorithm parameters with the least number of iterations and the
Figure 5. $f_1$ Curves of the objective function value.

Figure 6. $f_2$ Curves of the objective function value.

Figure 7. $f_3$ Curves of the objective function value.

Figure 8. $f_4$ Curves of the objective function value.

Figure 9. $f_5$ Curves of the objective function value.

Figure 10. $f_6$ Curves of the objective function value.
best robustness, four parameters (the step size control factor $\alpha$, population size, discovery probability $P_a$, and random number $\beta$) that affect the HCS algorithm are selected as control factors. Considering the optimization method of experimental design, each parameter takes three levels: the step size control factor is 0.1, 0.2, and 0.3; the population size is 10, 30, and 50; the discovery probability is 0.25, 0.5, and 0.75; the random number is 0.25, 0.5, and 0.75. Since the experiment has four parameters, the orthogonal test table of $L_9(3^{4})$ is selected for experimental research. The specific scheme is shown in Table 4.

The Rastrigin benchmark function is used to test the HCS performance, $(x, y) \in [-5,5]$. As shown in Figure 11, this function has many local minimum points in the domain of definition, which has strong oscillation characteristics. During the test, the maximum number of iterations per run is 1000, and the error is less than $10^{-6}$. Each function is optimized 20 times, and then the average number of iterations is taken. The test results are shown in Table 5.

It can be seen from Table 5 that when the step size control factor $\alpha$, population size, discovery probability $P_a$, and random number $\beta$ are respectively $\alpha = 0.2$, population size = 30, $P_a = 0.75$, $\beta = 0.25$, the convergence rate of the HCS algorithm is the largest and the number of iterations is relatively small. Therefore, in the experiment, the population size of HCS algorithm is 30; the discovery probability of HCS algorithm is 0.25. On account of the fixed value of step size control factor $\alpha$ is 0.01 in standard CS algorithm, setting the value range of step size control factor to $[0.01, 0.2]$ in the HCS algorithm.

In the INSGA-II algorithm, the mutation probability is 0.3 and the crossover probability is 0.9. In the CS algorithm, the abandonment probability is 0.25; the step size control factor is 0.01. This paper uses $50n$ (unit: ms) as the termination condition for each algorithm, where $n$ is the number of work piece of each problem size. This allows all algorithms to run at the same time when testing the same problem size, ensuring fair comparisons. Each algorithm runs independently 20 times for test problem.

The analysis indicators used is this paper are multi-objective analysis indicators proposed in literature, which are $R_{NDS}(S_r)$ and $NDS\_NUM(S_r)$ respectively, and the calculation equation is as follows:

$$R\_{NDS}(S_r) = \frac{|S_r - \{x \in S_r | \exists y \in S : y < x\}|}{|S_r|}$$ (15)
expressed as $S = \{x \in S_r \mid y \in S : y < x\}$ (16)

Where $S_r$ refers to the Pareto solution set of the algorithm $r$; $S$ refers to the union of the Pareto solution set of the $K$ algorithms, which can be expressed as $S = S_1 \cup \cdots \cup S_r \cup \cdots S_K$; $y < x$ means that the individual $y$ completely dominates the individual $x$; $|S_r|$ refers to the individuals in the $S_r$ set; $NDS\_NUM(S_r)$ refers to the number of undominated individuals in algorithm $r$; $R\_NDS(S_r)$ refers to the ratio of the number of undominated individuals in algorithm $r$ to the total number of individuals in the Pareto solution set in algorithm $r$. $R\_NDS(S_r) = 1$ means all the Pareto individuals in the $S_r$ set are not dominated; $R\_NDS(S_r) = 0.9$ means that 90% of Pareto individuals in $S_r$ are not dominated.

The data results of each problem size in this paper can be found in Tables 6 and 7. The comparison data of HCS algorithm and CS algorithm are shown in Table 6. The comparison data of HCS algorithm and INSGA-II algorithm are shown in Table 7. According to the definition of $S$ above, $S$ in Table 6 can be expressed as $S = S_{HCS} \cup S_{CS}$, $S$ in Table 7 can be expressed as $S = S_{HCS} \cup S_{INSGA-II}$; $R\_NDS\_HCS$ is the average ratio of $20R\_NDS(S_{HCS})$ data, and $R\_NDS\_INSGA-II$ is 20. The average ratio of $R\_NDS(S_{INSGA-II})$ data, $NDS\_NUM\_CS$ represents the average number of $20NDS\_NUM(S_{CS})$ data, $NDS\_NUM\_INSGA-II$ represents the average number of $20NDS\_NUM(S_{INSGA-II})$ data.

It is obvious that $R\_NDS\_HCS$ is all 1 and $R\_NDS\_CS$ is almost 0 from Table 6, indicating that when solving the above-mentioned problem size MOPFSP, the HC algorithm completely dominates the standard CS algorithm, and the number of $NDS\_NUM$ is in the range of 3–6, in accordance with the population size of 30; From Table 7, we can see that $R\_NDS\_HCS$ is greater than $R\_NDS\_INSGA-II$, and there are eight problem sizes $R\_NDS\_HCS$ greater than $R\_NDS\_INSGA-II$, indicating that more than 80% of the Pareto individuals of the HC algorithm dominate the Pareto individuals of the INSGA-II algorithm, while in the other two problem sizes, The ratio is above 65%. The operation of size 10_5 and size 100_30 are shown in Figures 12 and 13. It is obvious that although the Pareto solution set found by the INSGA-II algorithm and the HC algorithm overlap, they all dominate the standard CS algorithm. It can be seen from Figure 12 that the Pareto front of the HCS algorithm is at the bottom left of the other two algorithms, indicating that the Pareto solution set of the HC algorithm completely controls the Pareto solution set of the other two algorithms. As the size of the problem increases, compared with the standard CS algorithm and INSGA-II algorithm, the distance between the Pareto front of the HC algorithm and the Pareto front of the other two algorithms increases, and the superiority of optimization is more and more obvious.

The comparison results between the INSGA-II and HCS from running times are shown in Figure 14. We can clearly see that the running time of INSGA-II is much longer than HCS when the size of instances is large, and INSGA-II get fewer effective solution compare to HCS. Therefore, the HCS algorithm has better

### Table 6. Statistical results of HCS and CS.

| Problem size | HCS | CS |   |
|--------------|-----|----|---|
|              | R\_NDS\_HCS | NDS\_NUM\_HCS | R\_NDS\_CS | NDS\_NUM\_CS |
| 10_5         | 1    | 3  | 0.025 | 0.05 |
| 20_5         | 1    | 1  | 0    | 0    |
| 30_5         | 1    | 4  | 0    | 0    |
| 40_5         | 1    | 4.6| 0    | 0    |
| 50_10        | 1    | 3.05| 0    | 0    |
| 60_10        | 1    | 2.95| 0    | 0    |
| 70_10        | 1    | 1.9 | 0    | 0    |
| 90_20        | 1    | 4.85| 0    | 0    |
| 90_30        | 1    | 4.8 | 0    | 0    |
| 100_30       | 1    | 5.9 | 0    | 0    |

### Table 7. Statistical results of HCS and INSGA-II.

| Problem size | HCS | INSGA-II |
|--------------|-----|----------|
|              | R\_NDS\_HCS | NDS\_NUM\_HCS | R\_NDS\_INSGA-II | NDS\_NUM\_INSGA-II |
| 10_5         | 1    | 3  | 0.96  | 2.85 |
| 20_5         | 0.8  | 2.6| 0.35  | 0.25 |
| 30_5         | 0.9875 | 3.95 | 0.175 | 0.6 |
| 40_5         | 0.91 | 3.95| 0.3   | 1.35 |
| 50_10        | 0.925 | 2.35 | 0.3   | 0.85 |
| 60_10        | 0.66 | 1.9 | 0.5   | 1    |
| 70_10        | 0.8  | 1.6 | 0.575 | 1.1 |
| 90_20        | 0.82 | 3.5 | 0.379 | 1.15 |
| 90_30        | 0.8375 | 4.4 | 0.2975 | 0.85 |
| 100_30       | 0.752 | 3.55 | 0.59  | 1.7 |
solution speed when solving large-size instances problem.

To sum up, HCS algorithm is more efficient than INSGA-II algorithm and CS algorithm in solving MOPFSP of all the above problem sizes.

At the same time, the HCS algorithm was used to investigate the wire and cable production scheduling problem of a wire and cable factory in Jiangsu Changzhou. The production process of the factory’s pre-formed cable consists of five steps: monofilament drawing, monofilament annealing, conductor strand- ing, insulation extrusion, and cable formation. The five links are processed on five specific machines, and the processing machines in each link can set the processing speed by adjusting the gear. In recent years, the company has actively responded to green production,
energy saving, and energy reduction, and considered both economic indicators (makespan) and environmental indicators (TCE) in its production process. Obviously, the production scheduling problem of this pre-formed cable is a typical MOPFSP. At present, the production scheduling of this cable factory is manually ordered and scheduled by the dispatcher based on experience after number the work piece. This paper uses the actual production data of 30 types of cables produced by the factory as test examples, and uses the HCS algorithm to run a 1.5 s solution. The target value comparison of HCS algorithm is shown in Figure 15.

Furthermore, the dispatcher is asked to give a manual dispatching plan within 5 min. The HCS algorithm obtained the scheduling scheme S1–S4. The scheduling scheme T1 obtained by dispatcher through experience is shown in Table 8. As can be seen from Table 8, S1–S4 are significantly better than T1. The results show that the HCS algorithm can solve practical MOPFSP problems quickly and effectively.

**Conclusion**

Nowadays, with the increasingly severe environmental problems in the manufacturing industry, the green manufacturing has received considerable attention. Although the flow shop scheduling problem has been widely studied in recent years, but the research on multi-objective flow shop scheduling with green indicators is still relatively limited. In this research a hybrid cuckoo algorithm is proposed to solve the green MOPFSP with environmental indicators, which meets the current needs of ecological environment governance. The main contributions of this paper are as follows:

1. The LOV rule is used to transform the individuals in the HCS algorithm from the real number vector to the workpiece ordering, so that it can be easily searched in the solution space of MOPFSP;

2. An adaptive step size control factor is designed to control the search range of the evolution phase of the algorithm and makes the algorithm easy to find high-quality individuals in the later stages of evolution;

3. An multi-neighborhood local search method is proposed for detailed search of high-quality solution regions, so that the high-quality solution area found by the global search of the HCS algorithm can be searched more carefully;

**Table 8. Scheduling plan.**

| Plan | Work piece scheduling sort | $C_{\text{max}}$ | TCE  |
|------|-----------------------------|------------------|------|
|      | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 |                |      |
| T1   | 16 17 18 19 20 21 22 23 24 25 10 9 11 12 | 1108            | 9764930 |
| S1   | 3 6 9 20 30 5 29 19 23 16 13 12 1 17 | 1035            | 9750170 |
| S2   | 18 12 28 24 10 5 15 2 27 9 26 4 11 12 | 1039            | 9749536 |
| S3   | 16 12 27 2 24 10 5 15 9 26 21 29 23 28 | 1056            | 9744074 |
| S4   | 18 28 9 26 15 5 10 24 27 12 2 4 8 19 | 131042          | 9748265 |

**Figure 15.** HCS target value comparison.
The HCS algorithm uses the proposed adaptive step size control factor and multi-neighborhood local search to balance the global and local search of the algorithm, and improves the performance and convergence speed of the algorithm. The results of simulation and algorithm comparison show that HCS algorithm can solve MOPFSP quickly, and its performance is better than CS algorithm and INSGA-II algorithm. The effectiveness of the HCS algorithm in solving MOPFSP is verified.

Some future works may be done from following aspects: (1) Regarding the future research of cuckoo algorithm in complex production scheduling, we can consider extending it to more complex scheduling problems than MOPFSP, especially the uncertain green scheduling problem. (2) It is interesting to present some heuristic algorithms to generate initial population instead of random procedure used in this paper so as to further improve the algorithms performance.

Acknowledgements
The authors would like to thank the referees for their helpful comments and suggestions.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This paper is supported by National Science Foundation of China (No. 51875171), and the Fundamental Research Funds for the Central Universities (2019B21614).

ORCID iD
Wenbin Gu https://orcid.org/0000-0001-5918-8800

References
1. Fang K, Uhan N, Zhao F, et al. A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. J Manuf Syst 2011; 30(4): 234–240.
2. Wu XL and Cui Q. Multi-objective flexible flow shop scheduling problem with renewable energy. Comput Integr Manuf Syst 2018; 24: 2792–2807.
3. Fang K, Nelson A, et al. Flow shop scheduling with peak power consumption constraints. Ann Oper Res 2013; 206(1): 115–145.
4. Yang XS and Deb S. Multi-objective cuckoo search for design optimization. Comput Oper Res 2013; 40: 1616–1624.
5. Yenisey MM and Yagmahan B. Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. Omega 2014; 45: 119–135.
6. Li X and Ma S. Multi-objective discrete artificial bee colony algorithm for multi-objective permutation flow shop scheduling problem with sequence dependent setup times. IEEE Trans Eng Manage 2017; 64(2): 149–165.
7. Wang X and Tang L. A machine-learning based memetic algorithm for the multi-objective permutation flowshop scheduling problem. Comput Oper Res 2017; 79: 60–77.
8. Luo H, Du B, Huang GQ, et al. Hybrid flow shop scheduling considering machine electricity consumption cost. Int J Prod Econ 2013; 146: 423–439.
9. Ding JY, Song SJ, Wu C, et al. Carbon efficient scheduling of flow shops by multi-objective optimization. Eur J Oper Res 2016; 248: 758–771.
10. Liu CH and Huang DH. Reduction of power consumption and carbon footprints by applying multi-objective optimization via genetic algorithms. Int J Prod Res 2014; 52: 337–352.
11. Ding J, Song S and Wu C. Carbon efficient scheduling of flow shops by multi-objective optimization. Eur J Oper Res 2015; 248: 1–14.
12. Tang D. Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization. Comput Indus Eng 2015; 81: 82–95.
13. Lu C. Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm. J Clean Prod 2017; 144: 228–238.
14. Wang WJ and Tian GD. Dual-objective program and improved artificial bee colony for the optimization of energy-conscious milling parameters subject to multiple constraints. J Clean Prod 2020; 245: 118–129.
15. Fu YP and Tian GD. Stochastic multi-objective modeling and optimization of an energy-conscious distributed permutation flow shop scheduling problem with the total tardiness constraint. J Clean Prod 2019; 226: 515–525.
16. Yang XS and Deb S. Cuckoo search via Lévy flights. In: Proceedings of world congress on nature & biologically inspired computing, India, 29 December 2009, pp.210–214. IEEE Publications.
17. Wang LJ, Zhong YW and Yin YL. Orthogonal crossover cuckoo search algorithm with external archive. J Comput Res Dev 2015; 52: 2496–2507.
18. Shefi S and Chaudhuri SS. Cuckoo search algorithm using Lévy flights: a review. Mod Educ Comput Sci 2013; 12: 10–15.
19. Li X and Yin M. A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem. Int J Prod Res 2013; 51: 4723–4754.
20. Marichel V, Prabaharan T and Yang X. Improved cuckoo search algorithm for hybrid flow shop scheduling. Appl Soft Comput J 2014; 19: 93–101.
21. Alaa S and Allobaidi A. Two improved cuckoo search algorithms for solving the flexible job shop scheduling problem. Int J Percept Cogn Comput 2016; 2: 25–31.
22. Wang H, Wang W, Sun H, et al. A new cuckoo search algorithm with hybrid strategies for flow shop scheduling problem. Soft Comput 2017; 21: 4297–4307.
23. Zhang YH, Wang L and WU Q-D. Dynamic adaptation cuckoo search algorithm. *Control Decis* 2014; 29: 617–622.
24. Lu C, Gao L, Li X, et al. Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm. *J Clean Prod* 2017; 144: 228–238.
25. Wang H, Fu Y and Huang M. A species based multi-objective evolutionary algorithm for multi-objective flow shop scheduling problem. In: *2015 IEEE congress on evolutionary computation*, Sendai, Japan, 25–28 May 2015, pp.3243–3247, IEEE.
26. Hu ML, Wan YC, Wang MW, et al. Band selection based on chaotic cuckoo search algorithm for hyperspectral image. *Microelectron Comput* 2018; 35: 124–129.
27. Nasiri MM, Yazdan PR and Jolai F. Simulation optimization approach for optimizing jobs dispatching rule in an open shop scheduling problem. *Int J Comput Integr Manuf* 2017; 30: 1239–1252.
28. Bai D, Zhang Z and Zhang Q. Flexible open shop scheduling problem to minimize makespan. *Comput Oper Res* 2015; 67: 207–215.
29. Li X and Li M. Multi-objective local search algorithm-based decomposition for multi-objective permutation flow shop scheduling problem. *IEEE Trans Eng Manage* 2015; 62: 544–557.
30. Sha DY and Hsu CY. A new particle swarm optimization for the open shop scheduling problem. *Comput Oper Res* 2008; 35: 3243–3261.
31. Zheng H and Zhou Y. A novel cuckoo search optimization algorithm base on gauss distribution. *J Comput Inform Syst* 2012; 8: 1–8.
32. Zhu XH and Wang N. Cuckoo search algorithm with RNA crossover operation for PID control of overhead cranes. *J Zhejiang Univ* 2017; 51: 1397–1404.
33. Mang YH and Wu YQ. Two-dimensional Renyi gray entropy image threshold selection based on chaotic cuckoo search optimization. *CAAI Trans Intell Syst* 2018; 13(1): 152–158.
34. Zhang HL, Zhang XJ, He ZD, et al. The study onimage matching method based on cuckoo search. *J Zhengzhou Univ* 2017; 49(4): 51–56.
35. Daniel E, Anitha J and Gnanaraj J. Optimum laplacian wavelet mask based medical image using hybrid cuckoo search-grey wolf optimization algorithm. *Knowl Based Syst* 2017; 131: 58–69.
36. Tao T, Zhang J, Xin KL, et al. Optimal valve control in water distribution systems based on cuckoo search. *J Tongji Univ* 2016; 44: 600–604.
37. Wang R. An improved non-dominated sorting genetic algorithm for multi objective problem. *Math Probl Eng* 2016; 2006: 1–7.
38. Qian B, Li Z, Hu Rong, et al. A hybrid differential evolution algorithm for the multi objective reentrant job-shop scheduling problem. In: *IEEE international conference on control and automation (ICCA)*, Hangzhou, China, 12–14 June 2013, pp.485–489, IEEE.