Ant colony optimization for Cuckoo Search algorithm for permutation flow shop scheduling problem

Yu Zhang, Yanlin Yu, Shenglan Zhang, Yingxiong Luo and Lieping Zhang

College of Mechanical and Control Engineering, Guilin University of Technology, Guilin, People’s Republic of China

ABSTRACT
A Cuckoo Search (CS) algorithm based on ant colony algorithm is proposed for scheduling problem in permutation flow shop scheduling problem (PFSP). When the raised CS algorithm obtains the position of the bird nest to be updated, it is used as a set of initial solution of the ant colony optimization algorithm (ACO), and ACO algorithm search optimization is performed in a very small range. After that, the solution obtained by the ACO search is taken as a new candidate solution, compared with the candidate bird nest according to the fitness degree. When the candidate solution of the ACO search optimization is better than the one generated by the Lévy flight, the latter is replaced. Finally, the CS algorithm is selected, changing the new bird nest position according to the abandonment probability. The updated position tends to be more optimal, which improves the quality of the solution as well as the convergence speed and accuracy of the algorithm. Comparing the performance of the proposed algorithm with the standard Cuckoo one, by testing function, the optimized performance was verified. Finally, the Car benchmark test served as test data, and the performance in the PFSP was compared. The effectiveness and superiority in the algorithm in solving problem were confirmed.

1. Introduction
The permutation flow shop scheduling problem (PFSP), based on constraint of the scheduling problem, further increases the constrained scheduling problem that all workpieces have the same processing order in any machine, which effectively enhances the enterprises’ efficiency (Fernandez-Viagas & Framinan, 2017). Although the process constraint of PFPS is relatively simple, it has confirmed in flexible manufacturing industry that PFSP enjoys more than three machines participating in the production process, which is a typical NP-complete problem, and there is no overall optimization algorithm with polynomial computational complexity (Garey, Johnson, & Sethi, 1976). The methods solved PFSP mainly include the algorithms of exactness, heuristic and intelligence (Xiangbo, Yunlong, & Dingyi, 2017). The first one has much computational complexity and is only suitable for small-scale problems. Then, the second one can construct a solution in short time according to the characteristics of the problem, but it is difficult to guarantee solution’s quality. Finally, with the development of computational intelligence, the third one to address PFSP is widely studied, including genetic algorithm (Abedinnia, Glock, & Brill, 2016), particle swarm optimization (PSO; Wanab, 2013), differential evolution (Coelho, Ayala, & Mariani, 2014), Cuckoo (Xu, Ye, & School, 2015) and various hybrid algorithms (Zobolas, Tarantilis, & Ioannou, 2009).

The Cuckoo Search (CS) belongs to an emerging heuristic algorithm put forward by Professor Xin She Yang and Professor S. Deb (2010) of Cambridge University in 2009. It aims at handling the problem of actually complex optimization by simulating the Lévy flight patterns of brooding behaviour of cuckoos, albatrosses and fruit flies. Due to its simple model, less adjustable parameters and fast convergence speed, CS algorithm has been widely used in various development fields such as artificial intelligence, industrial scheduling, commercial optimization and neural network in recent years (Shehab, Khader, & Al-Betar, 2017). In the iterative process of the standard CS algorithm, the exploration process of Cuckoo’s Optimization and the update formula of new bird nest position are implemented by Lévy random flight strategy. The random jump property of Lévy flight can increase the global search ability allowing for algorithm. Lévy flight is applied to the location update strategy of migration operator to obtain a new bird nest position, which can get rid of the local optimal solution and prevent the algorithm search process from being too premature. At

CONTACT Lieping Zhang zlp_gx_gl@163.com

© 2018 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.
the same time, the random update of bird nest position can also advance the algorithm’s population diversity. However, on the other hand, because of the overall position, the update position cannot step towards the optimal way as the iterative algorithm progresses, which will cause the algorithm’s optimization process to converge slowly, especially in the vicinity of the optimal solution. The search efficiency will be very low, which may make the population oscillate back and forth in the vicinity of the optimal solution, jump and cannot converge, and even lost phenomenon. Luus and other scholars pointed out that an algorithm cannot achieve the best results between the search ranges and results. Generally, the local search algorithm can be mixed to effectively promote the capability of the algorithm (Luus & Jaakola, 2010). Based on these factors, many scholars have raised various improvements for the CS algorithm. For example, Amirhossein et al. came up with the PSO algorithm to boost the all performance of the Cuckoo algorithm. Hybrid algorithm of PSO and CS (Ghodrati & Lotfi, 2012) and Jiang, Qi, Meiling, and Zhang (2014) put forward a separate local search strategy in the optimization process before entering the next iteration. Nie, Liu, Wei, and Yang (2013) worked out a concept of introducing elite individuals, local search strategy for elite individuals and the aging mechanism of leaders. Besides, Yin, Qiu, and Gao (2018) proposed a meta-heuristic algorithm based on GA and CS algorithm to solve the problem of gene expression data cluster. Chen, Yu, Zhou, and Meng (2018) presented the CS algorithm combined with Broyden Fletcher–Goldfarb–Shanno (BFGS) algorithm (CS-BFGS) to identify time-dependent boundary conditions for 2-D transient heat conduction problems in functional gradient materials. And Binh, Hanh, Quan, and Dey (2016) came up with ways to shorten the computation time and stability of NP problem by two nature-based algorithms, namely Improved Cuckoo Search and Chaotic Flower Pollination algorithm. Moreover, Faris, Sheta, and Öznergiz (2016) presented a new hybrid approach on the basis of multigene genetic programming and CS algorithms for developing three rigorous models for roll force, torque and slab temperature in the hot rolling industrial process at the Eregli Iron and Steel Factory in Turkey.

The ant colony optimization algorithm (ACO) is a kind of swarm intelligence optimization algorithm put forward by Italian learner M. Dorigo and others to observe the foraging behaviour of ants in nature (He, Chen, & Zhao, 2006). The ACO has many advantages, such as strong robustness, good local optimization, global search performance, parallel computing and easy to combine with other optimization methods (Sóbester & Forrester, 2010). Cao, Zhang, Guan, and Zhou (2018) proposed a quantum-inspired ACO algorithm that integrates ACO and quantum computing to predict the links in networks. Patil and Kadhane (2018) raised ACO built on hybrid algorithm for selective harmonic elimination in single phase of seven levels in multilevel inverter with reduced switches. The time depended vehicle routing problem with time windows is considered. A multi-type ant system algorithm hybridized with ant colony system and max–min ant system algorithms is brought forward (Deng, Zhu, Li, & Zheng, 2018). Chebli, Elakkary, and Sefiani (2018) used a multi-criterion ACO based on Pareto optimality to give the conflict character of closed-loop system performance parameters.

Aiming at the shortcomings of standard CS algorithm, this paper proposes a CS based on ACO. The algorithm utilizes ACO algorithm to optimize the position update strategy of CS, making the candidate solution evolve towards the optimal one with the iteration, so as to address the problems of slow convergence speed and low convergence precision caused by the random position update of CS algorithm. On this basis, the algorithm is applied to settle the optimization design problem of PFSP, and the effectiveness of the proposed algorithm is verified by Car benchmark class.

2. Mathematical description of PFSP

The basic process of PFSP can be described as follows (Wanliang & Qidi, 2007): n components are processed on m sets of equipment. Each component needs to go through K process steps. The time, spent by component i to complete this process on device j, is fixed, assuming that it is $t_{ij}(i = 1, 2, \ldots, n)$. The problem to be solved in scheduling is to give an optimal sequence of process elements for the component process flow, so that the whole process takes the shortest time. In PFSP, it is impossible to have only one rather than many components to process and it is not unique. In other words, it needs to be processed separately by different equipment. Each process demands to be processed on different equipment, and the processing time spent on different equipment is also different. It can be seen that taking different component processing sequences on the device for the same production process leads to different results, and they are so different.

General PFSP satisfies the following specific conditions: (1) pipelined processing of n components on M device; (2) all components in the processing sequence are the same; (3) the sequence of components processed by each device is the same; (4) the preparation and processing time of all components consumed on all devices is known in advance; and (5) all equipment can only process one component when machining. The goal of solving PFSP is to find the optimal scheduling scheme.
under the five specific conditions mentioned above, even if is the shortest during the whole process.

If the criterion of scheduling is the maximum completion time, this PFSP problem can be described by the triple method of scheduling, that is, \( F_m|prmu|C_{\text{max}} \). Among them, \( F_m \) refers to the flow shop scheduling model with \( m \) equipment; and \( prmu \), referring to the processing sequence of each component on any equipment, is consistent; \( C_{\text{max}} \) indicates that the maximum time of the entire production process; let \( t_{jk,i} \), \( j \) be the processing time that component \( j_k \) spends in device \( i \) (the reserve time of each component is stipulated to be included in the processing time); \( C_{jk,i} \) is the completion time of component \( j_k \) on device \( i \); and \( \pi \in \{\Omega|j_1,j_1,\ldots,j_n\} \) is the processing order of components; while \( \Omega \) represents the total set of sequential sequence of all components. For the component processing, the following constraints are set: the process of component processing cannot be interrupted or abandoned; each equipment is unique; and each component is unique. By setting all components to be processed in the order of device \( 1 \sim m \), the following formulae can be used to calculate the completion time of each component on each device (Yanfeng, 2012). That is:

\[
C_{j1,i} = \sum_{r=1}^{i} t_{j1,r}, i = 1, 2, \ldots, m
\]

\[
C_{jk,1} = \sum_{s=1}^{k} t_{jk,s}, k = 1, 2, \ldots, n
\]

\[
C_{jk,i} = \max\{C(jk,i-1), C(jk-1,i)\} + t_{jk,i}
\]

\[
C_{\text{max}}(\pi) = C_{\text{max}}(j_n,m)
\]

\[
\pi* = \arg \{C_{\text{max}}(\pi) = C(j_n,m)\} \rightarrow \text{min}
\]

\[
\forall \pi \in \Omega
\]

According to them, Formulae (1)–(3) are the recursive equations for calculating the completion time; Formula (4) is the maximum completion time of the batch of jobs; Formula (5) is the optimal scheduling scheme for minimizing the maximum completion time; while \( n \) and \( m \) respectively indicate the number of components and devices. The scheduling process is shown in Figure 1.

3. Cuckoo Search algorithm optimized by ant colony AC algorithm

3.1. Standard Cuckoo algorithm

The standard CS algorithm is inspired by the behaviour of cuckoo breeding. It simulates the process of CS for bird nest and laying eggs in nature by introducing the Lévy flight mechanism of birds, which belongs to the global intelligent optimization algorithm. Yang and Deb proposed three ideal states of the algorithm (Johnzén, 2014):

(1) The cuckoo only has one egg in each generation’s reproduction process, and randomly chooses a bird nest to hatch.

(2) Before finding a better bird nest, the current best bird nest will be used as the next generation’s host to enter the next breeding.

(3) The total number of bird nests available for each generation is constant, and it is probable that the eggs laid by cuckoo are found by the host as \( P_a \). If the bird’s eggs are perceived by the host who will give up them or the entire bird nest and find a suitable location to build other new bird nest.

Among them, \( P_a \) means that host eggs maybe not their children, \( P_a \in [0, 1] \), to which most of the researchers set it as 0.25 (Naik, Nath, Wunnava, Sahany, & Panda, 2015).

Standard CS algorithm realizes that the search of cuckoo bird nest is simulated by three basic processes: candidate population generation, selection and random migration. In terms of the above three ideal states, the cuckoo flight finding route and position in the algorithm

![Figure 1. Gantt chart for scheduling \( F_m|prmu|C_{\text{max}} \) problem of permutation flow shop.](image)
generate candidate populations by the following formula:

$$X_{t}^{i+1} = X_{t}^{i} + \alpha \oplus L(\beta) \quad (6)$$

In the (6), $X_{t}^{i+1}$ and $X_{t}^{i}$ respectively refer to the position vectors of the bird nest of $i (i = 1, 2, \ldots, n, n$ is the bird nest number) in the $t + 1$ and $t$ generation ($X_{t}^{i} = x_{i,1}^{t}, x_{i,2}^{t}, \ldots, x_{i,d}^{t}$); $d$ is the dimension of each bird nest; and $j$ represents any $j$ dimension, $j = 1, 2, \ldots, d$; as $k$ serves as algorithm algebra; $t = 1, 2, \ldots, t_{c}$, $t_{c}$ are algebras for convergence of algorithms); $\alpha$ represents step length factor, which is used to control the range of random search, a constant greater than zero, and it can take different values for different situations; $\oplus$ is the point-to-point multiplication; $L(\beta)$ indicates the random search route of Lévy flight; $\alpha \oplus L(\beta)$ refers to the flight step of Lévy flight, representing the flying distance from all bird nests (feasible solutions) of generation $i$ to $i + 1$ by Lévy random distribution.

The relationship between the random optimization circuit and time $t$ of Lévy flight obeys the Lévy probability distribution as shown in formula (7), where $\beta$ is a power co-efficiency and $\mu$ is a random number with normal distribution. Formula (7) shows that the optimal path of CS algorithm is composed of frequent short jumps and occasional long jumps. This optimization method can make the algorithm obtain greater search space and more easily jump out of the local optimum.

$$Levy(\beta) \sim \mu = t^{\beta}, (1 < \beta \leq 3) \quad (7)$$

In order to practice the CS algorithm, the literature Cuckoo Search (CS) via Lévy Flights method is used to simulate the flight path of $Levy(\beta)$ flight. That is the step length, as shown in the following formula:

$$Levy(\beta) = \mu / |\nu|^{\beta} \quad (8)$$

### 3.2. Improvement and implementation of standard CS algorithm

The proposed algorithm uses ACO algorithm to optimize the position updating strategy of CS algorithm, and improves the quality of candidate solutions generated by Lévy flight through the local search ability of ACO algorithm, so that the candidate solutions can evolve in the direction of optimal solution with iteration. When the CS algorithm finds that the optimal solution is every iterated, in other words, when Lévy flight is to obtain the bird nest position to be updated, it is introduced as a set of initial solutions into the ACO search algorithm as well as the initial one of the ACO algorithm. The ACO algorithm is searched within an very small range, and the solution obtained from this search is regarded as a new candidate solution. According to the fitness comparison with the candidate bird nest generated by Lévy flight, the preferred operator is executed again. When the candidate solution of ACO search is better than the one generated by Lévy flight, it is replaced. Then the CS algorithm is used to choose the optimal operator and the position is substituted in light of the abandonment probability $P_{a}$. By selecting the candidate solution, the quality of the solution is improved, and the updated position tends to be closer to the optimal solution. Thus, the search performance of the algorithm gets improved, and the convergence speed and precision of the CS algorithm are also improved.

For ensuring the searching performance of ACOCS algorithm after fusion, it is important to prevent the searching process of ACO algorithm from being too long and improve the quality of Lévy flight candidate solution effectively. This paper proposes Formula (9) as the searching radius of ACO algorithm in the range of feasible solution. At the same time, the solution set of CS algorithm is taken as the initial solution of ACO algorithm, and the optimization process is carried out in the search area with the radius $d_{i}$ of the initial solution as the centre.

$$d_{i} = \frac{|n_{\text{worst}} - n_{\text{best}}|}{2} \quad (9)$$

In (9), $d_{i}$ refers to the search radius of ACO algorithm in generation $i$; $n_{\text{worst}}$ indicates to the current worst bird nest location; while $n_{\text{best}}$ is the current best one.

In line with the idea of improvement, the specific implementation process of the algorithm can be described as Figure 2.

#### 3.3. The simulation and result analysis of improved algorithm

##### 3.3.1. Setting up simulation environment and parameter

The simulation test environment of this paper contains: Windows 10 operating system, Inter Core CPU T6400 2.10 GHz, 2.00 GB memory, MATLAB 2014b simulation software. The test algorithm is standard CS algorithm and the ACOCS algorithm put forward in this paper. The initial number of birds is $N = 25$, the initial set abandonment probability is $P_{a} = 0.25$, and the step factor is $\alpha = 1$. The end condition of the algorithm is the number of searches $T_{\text{max}} = 3000$ or the precision of the optimal solution reaches the error limitation. Test with six common test functions is shown in Table 1, and the basic features of the six common test functions are shown in Table 2.
### 3.3.2. Improved algorithm simulation and result analysis

The minimum, maximum, average and variance of dimension 5 and 15 optimization results were used as performance evaluation indexes to compare the performance of CS and ACOCS. The simulation results are shown in Table 3. The performance indicators in the table are the average values of the algorithm running 30 times separately.

Seen from Table 3, all the optimization data of ACOCS algorithm are better than those of CS algorithm whether the dimension is 5 or 15. At the same time, it can be seen that both CS algorithm and ACOCS algorithm can completely converge at the end of the number of times when the dimension $N = 5$, but the convergence speed of ACOCS algorithm is faster than CS algorithm, and the quality of the optimal solution is better than CS algorithm. When the dimension $N = 15$, the CS algorithm can completely converge except when the Griewank and Schwefel test functions were used, the other four test functions cannot achieve fully convergence, and each function of ACOCS algorithm can totally converge. Therefore, compared with CS algorithm, ACOCS algorithm can find that the optimal solution is faster and its quality is better. Especially for complicated functions, ACOCS search performance is significantly better than CS.

### 4. PFSP solution based on ACOCS algorithm

#### 4.1. Solution algorithm coding

As the solution space of PFSP belongs to the discrete distribution, CS algorithm is generally used in continuous optimization problems. In order to use the improved CS algorithm to solve the PFSP and obtain the optimal scheduling scheme in the production process, a reasonable coding method is needed to represent the scheduling problem, that is, the sequential combination of shop scheduling. In view of the characteristics of PFSP, this paper selects random key coding method with ascending order principle (Tasgetiren, Sevkli, & Gencyilmaz, 2007). Unifying the cuckoo's continuous position vector $X_i = [x_{i1}, x_{i2}, \ldots, x_{in}]$ in the search for bird nest and the components' processing procedure $\pi = (j_1, j_2, \ldots, j_n)$ on all equipment in the production process, some certain dimension vector of each bird nest position represents a process in the component processing. The nest position is made up of $n$-dimension vector representing

---

#### Table 1. Six common test functions.

| Function name | Expression |
|---------------|------------|
| Ackley        | $f_1(x) = -c_1 \exp \left( -0.2 \sqrt{\frac{1}{n} \sum_{j=1}^{n} x_j^2} - \exp \left[ \frac{1}{n} \sum_{j=1}^{n} \cos(2\pi x_j) \right] \right) + 20 + e$ |
| Griewank      | $f_2(x) = \frac{1}{1000} \left( \sum_{i=0}^{n-1} x_i^2 - \sum_{i=0}^{n-1} \cos \left( \frac{x_i}{\sqrt{1 + i}} \right) + 1 \right)$ |
| Rastrigin     | $f_3(x) = \sum_{i=1}^{n} \left( x_i^2 - 10 \cos(2\pi x_i) + 10 \right)$ |
| Rosenbrock    | $f_4(x) = \sum_{i=1}^{n} \left( 100(x_i - x_i^*)^2 + (x_i - 1)^2 \right)$ |
| Schwefel      | $f_5(x) = \sum_{i=1}^{n} \left( \sum_{j=1}^{m} x_i^j \right)^2$ |
| Sphere        | $f_6(x) = \sum_{i=1}^{n} x_i^2$ |

#### Table 2. Basic characteristics of six functions used for testing.

| Test function | Search domain | Global minimum | Optimal fitness value |
|---------------|---------------|----------------|-----------------------|
| Ackley        | $-15 \leq x_i \leq 30, i = 1, 2, \ldots, n$ | $x^* = (0, \ldots, 0)$ | $f(x^*) = 0$ |
| Griewank      | $-600 \leq x_i \leq 600, i = 1, 2, \ldots, n$ | $x^* = (0, \ldots, 0)$ | $f(x^*) = 0$ |
| Rastrigin     | $-10 \leq x_i \leq 10, i = 1, 2, \ldots, n$ | $x^* = (1, \ldots, 1)$ | $f(x^*) = 0$ |
| Rosenbrock    | $-5.12 \leq x_i \leq 5.12, i = 1, 2, \ldots, n$ | $x^* = (0, \ldots, 0)$ | $f(x^*) = 0$ |
| Schwefel      | $-10 \leq x_i \leq 10, i = 1, 2, \ldots, n$ | $x^* = (0, \ldots, 0)$ | $f(x^*) = 0$ |
| Sphere        | $-10 \leq x_i \leq 10, i = 1, 2, \ldots, n$ | $x^* = (0, \ldots, 0)$ | $f(x^*) = 0$ |
the processing order of a group of components in all devices. Such a position directly manifests a group of scheduling methods. In accordance with this encoding method, the corresponding scheduling solution can be calculated by counting the fitness of bird nest. Meanwhile, the coding method based on this multidimensional vector procedure is compatible with CS algorithm structure, which can avoid modifying the evolution operator of CS algorithm because of the coding method of scheduling problem.

4.2. Solving algorithm implementation process

Based on the mathematical model of PFSP and the proposed coding method, the implementation process of PFSP based on ACOCS algorithm is as follows:

Step 1: Setting parameters and population initialization; setting the population size popnum, step factor \( \alpha \), discovery probability \( P_a \), the maximum number of iterations \( N_{\text{max}} \), and then initialize the population according to the attributes of the problem.

Step 2: The cuckoo randomly picks the bird nest location \( X_i \) \( (i = 1, 2, \ldots, m) \), and computes the fitness of each bird nest (scheduling scheme) on the basis of the given encoding method to the order of component processing to find the best bird nest location \( X^* \).

Step 3: Candidate population is generated based on step size. In Formula (6), if the offspring exceeds the constraint range of the problem, the invalid solution should be handled properly.

Step 4: Adopting one-to-one selection strategy, selecting the fitness individuals from the parent and offspring as the parents of the next generation.

Step 5: Executing the random migration operator. From the updated population of step 5, a certain number of inferior h bodies are selected to prepare mutations in line with probability \( P_a \).

Step 6: Lévy flight is used to randomly determine the candidate bird nest location for the ill-fitted bird nest location in Step 5.

Step 7: The candidate bird nest location is searched locally by ACO algorithm, which is chose according to the fitness, in which the bird nest is retained as the new candidate bird nest location.

Step 8: Choosing the best choice operator again. For the individuals who perform mutation operations, compared the fitness of the mutated individuals with that of the bird nest individuals is to update. If the mutated individuals have higher fitness after mutation, they will be concluded in the population. Otherwise, the individuals before mutation will be retained and step 3 will be jumped.

Step 9: Turning to step 10 when the maximum number of searches is reached, otherwise turning to step 3 for the next round of search.

Step 10: Algorithm outputs the best solution and its scheduling plan.

4.3. Scheduling simulation experiment and result analysis

To achieve the abundant benchmark data of PFSP, this paper will use 8 operators Car1,Car2,...,Car8 of Car benchmark instance class as experimental data to test. The common evaluation indexes in Car benchmark class are BRE (best relative error), ARE (average relative error),

| Test function | Dimension | Algorithm | Minimum value | Maximum value | Average value | Number of iterations | Success rate% |
|---------------|-----------|-----------|---------------|---------------|---------------|---------------------|---------------|
| Ackley        | 5         | CS        | 7.2451e−8     | 1.0002e−6     | 9.776e−7      | 492                 | 100           |
|               | 15        | ACOCS     | 3.2018e−9     | 9.1213e−7     | 9.1936e−8     | 261                 | 100           |
|               |           | ACOCS     | 2.2425e−8     | 9.1784e−7     | 5.1908e−7     | 850                 | 98            |
|               | 15        | ACOCS     | 8.3512e−13    | 0.3131e−6     | 9.5075e−9     | 371                 | 100           |
| Griewank      | 5         | CS        | 9.0873e−19    | 9.0316e−16    | 6.0407e−17    | 564                 | 100           |
|               | 15        | ACOCS     | 0             | 0             | 0             | 265                 | 100           |
|               |           | ACOCS     | −8.0653e−17   | 5.0333e−13    | 3.0191e−16    | 1205                | 100           |
|               |           | ACOCS     | 0             | 0             | 0             | 547                 | 100           |
| Rastrigin     | 5         | CS        | −2.1109e−22   | 3.2250e−16    | 9.2048e−19    | 624                 | 100           |
|               | 15        | ACOCS     | −1.1563e−24   | 8.1616e−19    | 3.3231e−23    | 215                 | 100           |
|               |           | ACOCS     | 1.1704e−15    | 2.6907e−14    | 8.0017e−15    | 1149                | 99            |
|               | 15        | ACOCS     | 8.6907e−18    | 4.2793e−12    | 1.4873e−16    | 578                 | 100           |
| Rosenbrock    | 5         | CS        | 4.0950e−19    | 3.9232e−16    | 2.5620e−18    | 751                 | 100           |
|               |           | ACOCS     | 0             | 0             | 0             | 397                 | 100           |
|               |           | ACOCS     | 3.4246e−12    | 9.7286e−9     | 1.2712e−10    | 1615                | 99            |
|               | 15        | ACOCS     | 0             | 0             | 0             | 802                 | 100           |
| Schwefel      | 5         | CS        | 4.0139e−18    | 9.0286e−16    | 1.6033e−17    | 672.230             | 100           |
|               |           | ACOCS     | 2.0055e−20    | 7.0117e−17    | 1.6092e−19    | 433.0212            | 100           |
|               | 15        | ACOCS     | −5.0638e−15   | 9.0614e−12    | 3.1370e−14    | 1574.210            | 100           |
| Sphere        | 5         | CS        | 7.8338e−18    | 8.1878e−13    | 0.8539e−14    | 1124.301            | 100           |
|               |           | ACOCS     | 6.3383e−33    | 9.9416e−29    | 3.6319e−31    | 1057.123            | 100           |
|               | 15        | ACOCS     | 1.0860e−37    | 7.2690e−31    | 1.5209e−35    | 626.390             | 100           |
|               |           | ACOCS     | 6.2534e−14    | 9.1574e−13    | 2.2407e−13    | 2328.900            | 99            |
|               |           | ACOCS     | 2.0217e−16    | 7.0035e−14    | 0.1791e−14    | 1681.621            | 100           |
WRE (worst relative error), SR (success rate) and RE (relative error), which are expressed as the following formulae (Carlier, 1978):

\[
\begin{align*}
\text{BRE} &= \frac{\min(C_{\max} - C^*)}{C^*} \times 100\% \\
\text{ARE} &= \frac{\text{avg}(C_{\max} - C^*)}{C^*} \times 100\% \\
\text{WRE} &= \frac{\max(C_{\max} - C^*)}{C^*} \times 100\% \\
\text{RE} &= \frac{C_{\max} - C^*}{C^*} \times 100\%
\end{align*}
\]

(10) (11) (12) (13)

Among these formulas, \(C_{\max}\) is the minimum maximum completion time of the scheme found by the algorithm; and \(C^*\) is the lower bound of the problem.

The parameters of the two algorithms for comparison are the initial discovery probability \(P_0\) of the standard CS algorithm is 0.25, i.e. the probability of the bird’s bird nest being abandoned is 0.75; the step factor is \(\alpha = 1\); The initial discovery probability \(P_0\) of the ACOCS algorithm refers to 0.25; the population number of each algorithm is set as \(\text{popnum} = 15\); the maximum number of iterations \(T_{\max} = 1000\); \(P\) is the type of Car; \(C^*\) is the corresponding problem. In the minimum completion time, \(n\) is the number of components involved in the process, and \(m\) is the number of devices. Each algorithm repeats 30 times to optimize. The simulation results are shown in Table 4.

With regard to the experimental results in Table 4, ACOCS algorithm can find the best solution each time in the Car benchmark class of the two algorithms, and the search process is faster than CS algorithm, reflecting that ACOCS algorithm has good global search performance. Besides, according to ARE and BRE data, ACOCS algorithm is also higher than CS algorithm in the quality of the solution. It can be seen that the efficiency of CS algorithm is effectively improved after local search of candidate solutions of CS algorithm through ACO algorithm.

5. Conclusion

Aiming at addressing the problem of the quality of candidate solutions produced by the randomness of CS algorithm location update strategy, a CS algorithm based on ACO optimization is proposed, which utilizes ACO algorithm with strong local search ability to re-optimize the new candidate solution before CS algorithm generating its replacement evolution, so that the candidate solution moves towards optimal solution. The simulation results of six test functions show that the quality of candidate solutions generated by this algorithm is higher than that of CS algorithm, which can enhance the convergence speed and accuracy of CS algorithm. From this, the ACOCS algorithm is applied to the PFSP solution. The ascending order random key coding method is used to realize the PFSP solution coding while the Car benchmark class is used to test the PFSP solution. The test data show that the scheduling performance of the ACOCS algorithm is better than that of the standard CS algorithm.

The number of parameters of CS algorithm is relatively small. In this paper, the selection of appropriate parameters is based on empirical values. How to formulate appropriate parameters and selection methods to expand the application scope of the algorithm to reduce its limitations on specific problems is what we need to further study.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Abedinnia, H., Glock, C. H., & Brill, A. (2016). New simple constructive heuristic algorithms for minimizing total flow-time in the permutation flowshop scheduling problem. Computers & Operations Research, 74(C), 165–174.

Binh, H. T. T., Hanh, N. T., Quan, L. V., & Dey, N. (2016). Improved cuckoo search and chaotic flower pollination optimization algorithms for maximizing area coverage in wireless sensor networks. Neural Computing & Applications, 30(7), 1–13.

Cao, Z., Zhang, Y., Guan, J., & Zhou, S. (2018). Link prediction based on quantum-inspired ant colony optimization. Scientific Reports, 8, 1–11.

Carlier, J. (1978). Scheduling with disjunctive constraints. Operations Research, 12(4), 333–350.

Chebli, S., Elakkary, A., & Sefiani, N. (2018). Ant colony optimization based on pareto optimality: Application to a congested router controlled by PID regulation. Systems Science & Control Engineering An Open Access Journal, 6, 360–369.

Chen, H.-L., Yu, B., Zhou, H.-L., & Meng, Z. (2018). Identification of transient boundary conditions with improved cuckoo search x. Engineering Analysis with Boundary Elements, 95, 124–141.

Coelho, L. D. S., Ayala, H. V. H., & Mariani, V. C. (2014). A self-adaptive chaotic differential evolution algorithm using gamma distribution for unconstrained global optimization. Applied Mathematics & Computation, 234(10), 452–459.

Table 4. Test results of two algorithms for Car series benchmark.

| P      | N/m | Car   | Algorithm | BRE   | ARE   | WRE   | SR/% |
|--------|-----|-------|-----------|-------|-------|-------|------|
| Car1   | 11/5| 7038  | CS        | 0     | 0.17  | 0.54  | 90   |
|        |     |       | ACOCS     | 0.09  | 0.21  | 100   |      |
| Car2   | 13/4| 7166  | CS        | 0     | 0.74  | 3.59  | 80   |
|        |     |       | ACOCS     | 0.05  | 0.79  | 100   |      |
| Car3   | 12/5| 7312  | CS        | 0.59  | 1.24  | 2.16  | 30   |
|        |     |       | ACOCS     | 0.32  | 0.95  | 1.67  | 60   |
| Car4   | 14/4| 8003  | CS        | 0     | 0.64  | 1.59  | 60   |
|        |     |       | ACOCS     | 0.35  | 1.27  | 80    |      |
| Car5   | 10/6| 7720  | CS        | 0.17  | 1.17  | 1.94  | 70   |
|        |     |       | ACOCS     | 0.49  | 1.07  | 100   |      |
| Car6   | 8/9  | 8505  | CS        | 0     | 0.76  | 1.43  | 60   |
|        |     |       | ACOCS     | 0.41  | 0.87  | 100   |      |
| Car7   | 7/7  | 6590  | CS        | 0.07  | 0.36  | 70    |      |
|        |     |       | ACOCS     | 0     | 0     | 100   |      |
| Car8   | 8/8  | 8366  | CS        | 0.93  | 2.14  | 60    |      |
|        |     |       | ACOCS     | 0.2   | 0.71  | 90    |      |
Deng, Y., Zhu, W., Li, H., & Zheng, Y. (2018). Multiple ant system algorithm for the time dependent vehicle routing problem with time windows. *Journal of Systems Engineering and Electronics*, 29(3), 625–638.

Faris, H., Sheta, A. F., & Öznergiz, E. (2016). MGP-CC: A hybrid multigene GP-Cuckoo Search method for hot rolling manufacture process modelling. *Systems Science & Control Engineering An Open Access Journal*, 4(1), 1–16.

Fernandez-Viagas, V., & Framinan, J. M. (2017). A beam-search-based constructive heuristic for the PFSP to minimise total flowtime. *Computers & Operations Research*, 81, 167–177.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2), 117–129.

Ghodrati, A., & Lotfi, S. (2012). A hybrid CS/PSO algorithm for global optimization. Asian conference on intelligent information and database systems (pp. 89–98).

He, Y., Chen, D., & Zhao, W. (2006). Ensemble classifier system based on ant colony algorithm and its application in chemical pattern classification. *Chemometrics & Intelligent Laboratory Systems*, 80(1), 39–49.

Jiang, H., Qi, R., Meiling, N., & Zhang, W. (2014). Modified Cuckoo Search algorithm and its application to optimization in multiple-effect evaporation. *Computers & Applied Chemistry*, 31(11), 1363–1368.

Johnzén, C. (2014). Cuckoo search: Recent advances and applications. *Neural Computing & Applications*, 24(1), 169–174.

Luus, R., & Jaakola, T. H. I. (2010). Optimization by direct search and systematic reduction of the size of search region. *Aiche Journal*, 19(4), 760–766.

Naik, M., Nath, M. R., Wunnava, A., Sahany, S., & Panda, R. (2015). A new adaptive Cuckoo Search algorithm. IEEE international conference on recent trends in information systems (pp. 1–5).

Nie, H., Liu, B., Wei, X. Y., & Yang, H. H. (2013). Improved Cuckoo search algorithm for resource-constrained project scheduling. *Journal of Guilin University of Technology*, 33(3), 529–563.

Patil, S. D., & Kadwane, S. G. (2018). Hybrid optimization algorithm applied for selective harmonic elimination in multilevel inverter with reduced switch topology. *Microsystem Technologies*, 24(8), 1–7.

Shehab, M., Khader, A. T., & Al-Betar, M. A. (2017). A survey on applications and variants of the Cuckoo Search algorithm. *Applied Soft Computing*, 61, 1041–1059.

Sőbester, A., & Forrester, A. (2010). Ant colony optimization. *Alphascript Publishing*, 28(3), 1155–1173.

Tasgetiren, M. F., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research*, 177(3), 1930–1947.

Wanab, Z. (2013). A hybrid intelligent algorithm by combining particle swarm optimization with chaos searching technique for solving nonlinear bilevel programming problems. *Swarm & Evolutionary Computation*, 8, 26–32.

Wanliang, W., & Qidi, W. (2007). Intelligent algorithm for production scheduling and its application. Beijing: Science Press.

Xiangbo, Q., Yunlong, Z., & Dingyi, Z. (2017). Dual population collaborative learning algorithm for solving PFSP. *Control and Decision*, 32(1), 12–20.

Xu, Y., Ye, C., & School, B. (2015). Cuckoo Search algorithm for solving permutation flow-shop scheduling problems. *Mathematical Theory & Applications*, 35(3), 89–96.

Yanfeng, L. (2012). Several intelligent algorithms for permutation flow shop scheduling problem. Xi’an: Xi’an Electronic and Science University.

Yang, X. S., & Deb, S. (2010). Cuckoo Search via levy flights. *Mathematics*, 6(2), 210–214.

Yin, L., Qiu, J., & Gao, S. (2018). Biclustering of gene expression data using Cuckoo Search and genetic algorithm. *International Journal of Pattern Recognition & Artificial Intelligence*, 32(11), 1850039-1–1850039-31.

Zobolas, G. I., Tarantilis, C. D., & Ioannou, G. (2009). Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm. *Computers & Operations Research*, 36(4), 1249–1267.