Soft Comput
DOI 10.1007/s00500-016-2062-9

METHODOLOGIES AND APPLICATION

A new cuckoo search algorithm with hybrid strategies for flow shop scheduling problems

Hui Wang[1,2] · **Wenjun Wang**[3] · **Hui Sun**[2] · **Zhihua Cui**[4] · **Shahryar Rahnamayan**[5] · **Sanyou Zeng**[6]

Abstract Cuckoo search (CS) is a recently developed meta-heuristic algorithm, which has shown good performance on many continuous optimization problems. In this paper, we present a new CS algorithm, called NCS, for solving flow shop scheduling problems (FSSP). The NCS hybridizes four strategies: (1) The FSSP is a typical NP-hard problem with discrete characteristics. To deal with the discrete variables, the smallest position value (SPV) rule is employed to convert continuous solutions into discrete job permutations; (2) To generate high quality initial solutions, a new method based on the Nawaz-Enscore-Ham (NEH) heuristic is used for population initialization; (3) A modified generalized opposition-based learning (GOBL) is utilized to accelerate the convergence speed; and (4) To enhance the exploitation, a local search strategy is proposed. Experimental study is conducted on a set of Taillard's benchmark instances. Results show that NCS obtains better performance than the standard CS and some other meta-heuristic algorithms.

Keywords Cuckoo search (CS) · Flow shop scheduling problem · Makespan · Discrete optimization

Communicated by V. Loia.

✉ Hui Wang
huiwang@whu.edu.cn

Wenjun Wang
wangwenjun881@126.com

Hui Sun
sun_hui2006@163.com

Zhihua Cui
zhihuacui@gmail.com

Shahryar Rahnamayan
shahryar.rahnamayan@uoit.ca

Sanyou Zeng
sanyouzeng@gmail.com

1 School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China

2 School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, China

3 School of Business Administration, Nanchang Institute of Technology, Nanchang 330099, China

4 School of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan 030024, China

5 Department of Electrical, Computer, and Software Engineering, University of Ontario Institute of Technology (UOIT), 2000 Simcoe Street North, Oshawa, ON, L1H 7K4, Canada

6 School of Computer Science, China University of Geosciences, Wuhan 430074, China

1 Introduction

Flow shop scheduling problem (FSSP) plays an important role in manufacturing systems. Good scheduling techniques can significantly improve the production efficiency. To achieve a good position in the market competition, more effective scheduling methods are always needed. The permutation flow shop scheduling problem (PFSSP) is one of the most popular production scheduling problems, which can be regarded as a simplified version of FSSP. In the PFSSP, each machine can process only one job at a time. The given machine sequence is same for all jobs, and the sequence of jobs on machines is also the same. According to literature (Michael and David 1979), the PFSSP has been proved to be NP-hard. For the significance in both theory and engi-

Published online: 12 February 2016

neering applications, different kinds of approaches have been proposed to solve the PFSSP.

To solve scheduling problems, there are some classical methods. In Johnson (1954), a heuristic method was designed for a simple PFSSP, which only consisted of two machines. Bansal (1977) proposed a branch and bound (B&B) algorithm to minimize the sum of completion times. Results show that the B&B is very effective. In Croce et al. (2002), an improved B&B was proposed for the two machine total completion time. In Ignall and Schrage (1965), the B&B was applied to some other flow shop scheduling problems.

In the past several years, some meta-heuristic algorithms have been proposed to solve the flow shop scheduling problem. Liu et al. (2007) designed a memetic PSO algorithm (called PSOMA) for the PFSSP with the objective to minimize the maximum completion time. In PSOMA, both PSO-based search and local search operators are used to achieve a balance between the global and local search. Moreover, the PSOMA employs some adaptive local search strategies to perform exploitation. Yang and Deb (2009) proposed an alternate two phases particle swarm optimization (called ATPPSO) to solve the FSSP. The ATPPSO designed two processes named the attractive process and the repulsive process, which execute alternatively during the search. Zhang et al. (2010a) presented an extended ATPPSO (called I-ATPPSO), which combines the PSO with genetic operators and annealing strategy. In I-ATPPSO, each particle consists of two phases, the attractive phase and the repulsive phase. In Zhang et al. (2010b), a circular discrete PSO (CDPSO) was applied to solve the FSSP. In CDPSO, a particle similarity and swarm activity metric are defined. Results show that CDPSO outperforms the other two algorithms. Tasgetiren et al. (2006) presented a discrete artificial bee colony (ABC) for minimizing the total flow-time in permutation flow shops. In Li and Yin (2012), another discrete ABC algorithm with composite mutation strategies was proposed for the PFSSP. According to the no free lunch theorem (Wolpert and Macready 1997), no single mutation operation would be appropriate for diverse instances of a particular problem. Therefore, a set of mutation strategies including swap, insert, inverse, and adjacent exchange is utilized to avoid premature convergence. In Marichelvam (2012), an improved hybrid cuckoo search (IHCS) algorithm was proposed for the PFSSP. The IHCS also used the NEH heuristic for population initialization. Results show that the IHCS performs better than an ant colony optimization meta-heuristic (MHD-ACS). Li and Yin (2013a) presented a hybrid CS (HCS) to solve the PFSSP. To improve the local search ability, a fast local search operator is used. Simulation results show the effectiveness of the HCS. Differential evolution (DE) is another population-based search algorithm, which has shown good search abilities on many optimization problems (Cui et al. 2016; Lin et al. 2015a). Li and Yin (2013b) proposed a modified opposition-based DE (ODDE) to solve the PFSSP. In ODDE, the NEH heuristic combined with random initialization is employed. Moreover, opposition-based learning (OBL) (Tizhoosh 2005) is used for population initialization and generation jumping to improve the global search ability. Similar to Li and Yin (2013b), Zhao et al. (2015) combined the shuffled complex evolution with OBL to solve the PFSSP. The OBL aims to improve the population quality and accelerate the convergence rate. There are 29 classical instances used in the experiments. Simulation results show the effectiveness of the proposed approach. Lin et al. (2015b) proposed a hybrid backtracking search algorithm (HBSA) to solve the PFSSP. In HBSA, some improved strategies including crossover and mutation strategies and simulated annealing (SA) are employed to obtain good solutions. Results show the efficiency of HBSA. In Rahman et al. (2015), a real time strategy is designed to solve dynamic PFSSP. For multi-objective flexible job shop scheduling problems, some efficient multi-objective algorithms are required (Chen et al. 2010; Lin and Chen 2013; Liang et al. 2015). Karthikeyan et al. (2015) designed a hybrid discrete firefly algorithm to solve multi-objective flexible job shop scheduling problems. Experiments on some famous benchmark instances show that the proposed algorithm is feasible and effective approach.

Cuckoo search (CS) is a recently proposed optimization algorithm developed by Yang and Deb (2009), which simulates the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds. Preliminary studies show that CS outperforms some existing algorithms such as genetic algorithm (GA) and particle swarm optimization (PSO) (Kennedy and Eberhart 1995). In this paper, we propose a new CS algorithm with hybrid strategies, called NCS, to solve the flow shop scheduling problem. In the NCS, the smallest position value (SPV) rule is employed to convert continuous solutions into discrete job permutations. To generate good initial solutions, a new method based on the NEH is used for population initialization. A modified generalized opposition-based learning (GOBL) on discrete variables is used to accelerate the convergence speed. Moreover, a local search strategy is employed to enhance the exploitation. The proposed NCS is different from IHCS (Marichelvam 2012) and HCS (Li and Yin 2013a). The IHCS only used the NEH heuristic for population initialization. For HCS, the NEH heuristic and a local search operator are employed. Experimental study is conducted on a set of Taillard's benchmark instances. Results show that NCS obtains better performance than the standard CS, IHCS, and some other meta-heuristic algorithms.

The rest of the paper is organized as follows. The problem descriptions are formulated in Sect. 2. The standard CS algorithm and its brief review are presented in Sect. 3. The proposed CS algorithm is described in Sect. 4. Experimental

results are given in Sect. 5. Finally, the work is concluded in Sect. 6.

2 Problem descriptions

The flow shop scheduling problem (FSSP) can be described as follows. There are n jobs ($i = 1, 2, \ldots, n$) and m machines ($j = 1, 2, \ldots, m$). Each job j will be sequentially processed on m machines: $1, 2, \ldots, m$. Assume that $t_{i,j}$ is the processing time of job i on machine j, and $O_{j,k}$ is the processing operation of job j on machine k. Each machine can process only one job at a time. The sequence in which the jobs are to be processed is the same for each machine. A schedule can be represented as a permutation $\pi = \{\pi_1, \pi_1, \ldots, \pi_n\}$ of jobs which can be mapped into a schedule defining completion times for all operations. The completion time C_j for each job j is the completion of the last operation $O_{j,m}$, $C_j = C_{j,m}$. The FSSP aims to minimize the makespan in this paper. Therefore, the FSSP can be defined by (Zhang and Sun 2009)

$$C_{1,1} = t_{1,1}, \tag{1}$$

$$C_{j,1} = C_{j-1,1} + t_{j,1}, \quad j = 2, \ldots, n, \tag{2}$$

$$C_{1,k} = C_{1,k-1} + t_{1,k}, \quad k = 2, \ldots, m, \tag{3}$$

$$C_{j,k} = \max\{C_{j-1,k}, C_{j,k-1}\}, \quad j = 2, \ldots, n, k = 2, \ldots, m, \tag{4}$$

$$C_{\max} = \max\{C_{n,m}\}, \tag{5}$$

$$f = \max\{C_{\max}\}. \tag{6}$$

Thus, the objective of this paper is to find a job permutation π to minimize the function f.

3 Cuckoo search

Optimization problems arise in a variety of engineering fields, such as structural design, scheduling, economic dispatch, and portfolio investment. With the rapid development of economy, the optimization problems become more and more complex, and more effective optimization algorithms are required. In the past several years, different kinds of nature-inspired optimization algorithms have been designed, such as PSO (Kennedy and Eberhart 1995), firefly algorithm (FA) (Yang 2010), artificial bee colony (ABC) (Karaboga 2005), ant colony optimization (ACO) (Dorigo et al. 1996), and cuckoo search (CS) (Yang and Deb 2009). Among these algorithms, CS is a recently developed swarm intelligence algorithm, which has shown good performance on many optimization problems (Yang and Deb 2010; Basu and Chowdhury 2013).

Algorithm 1: The Standard CS

```
Randomly initialize a population of N host nests;
Calculate the fitness value of each initial solution;
while t ≤ MaxGen do
    Get a cuckoo (say i) randomly by Lévy flights;
    Evaluate the fitness value of f_i;
    Randomly choose a nest among N (say j);
    if f_i is better than f_j then
        Replace j by the new solution;
    end
    Abandon a fraction (p_a) of worse nests, and build ones
    via Lévy flights;
    Update the global best solutions;
    t++;
end
```

In the CS, a new solution X_i for the ith cuckoo is generated by the following Lévy flight (Yang and Deb 2009).

$$X_i(t+1) = X_i(t) + \alpha \oplus L\acute{e}vy, \tag{7}$$

where $\alpha > 0$ is the step size which should be related to the scales of the problem of interest. The product $\oplus$ means entry-wise multiplications. The Lévy flight is a random walk, in which the step length is determined by Lévy distribution (Yang and Deb 2009).

$$\text{Lévy} \sim u = t^{-\lambda}, (1 < \lambda < 3), \tag{8}$$

It is known that the Lévy distribution has an infinite variance with an infinite mean. Therefore, the consecutive jumps of a cuckoo form a random walk process which obeys a power-length distribution with a heavy tail (Yang and Deb 2009). Some new solutions should be generated by the Lévy walk around the global best solution found so far. This will accelerate the local search. However, a substantial fraction of the new solutions should be generated by far field randomization and whose locations should be far enough from the current best solution. This is helpful to avoid falling into local optima (Yang and Deb 2010).

The basic steps of the CS algorithm can be summarized in Algorithm 1, where N is the population size, f is the fitness evaluation function, $p_a \in [0, 1]$ is the probability of discovering an alien egg, t is the generation index, and $MaxGen$ is the maximum number of generations.

Since the development of CS, it has been applied to different optimization problems. Bhandari et al. (2014) proposed a hybrid CS for image processing (Chen et al. 2015; Li et al. 2015; Xia et al. 2014a, b; Zheng et al. 2015). In Behnasr and Jazayeri-Rad (2015), CS was used to optimize the robust data-driven soft sensor based on support vector regression (Gu et al. 2015a, b). Navimipour and Milani (2015) applied CS to cloud computing (Fu et al. 2015; Ren et al. 2015; Xia et al. 2015). In Sajwan et al. (2014), CS and other swarm intelligence algorithms are used for web usage mining in recommender system (Ma et al. 2015). Goel et al. (2013) proposed a biogeography based CS algorithm for

classification (Liang et al. 2016; Wen et al. 2015). In Dhivya and Sundarambal (2011), CS was used to aggregate data in the sensor network (Shen et al. 2015; Xie and Wang 2014). In Elazim and Ali (2016), CS was used to optimize the power system stabilizers. Compared to GA and the conventional method, CS achieved better performance. Naik and Panda (2016) designed an adaptive CS (ACS) for face recognition. The ACS is almost parameter free, because it eliminates the Lévy step. Huang et al. (2015) proposed a hybrid CS called TLCS, which introduced teaching-learning into CS. Simulation study on some well-known engineering optimization problems show that the TLCS is very effective. Djelloul et al. (2015) proposed a quantum based CS to solve the graph coloring problem (GCP). Experiments show that the new approach obtains encouraging results on the standard DIMACS benchmark.

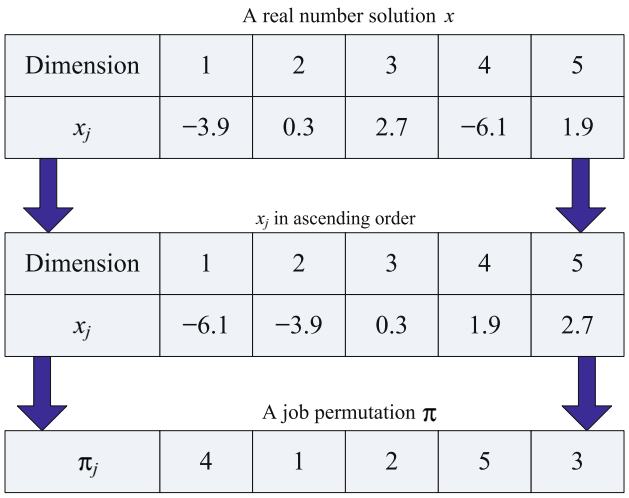

Fig. 1 An example for the SPV rule

4 Proposed approach

In this section, we present a new CS (NCS) algorithm for the flow shop scheduling problem. The detailed descriptions of the NCS are given as follows.

4.1 Solution representation

The standard CS algorithm was originally designed to solve continuous optimization problems, while the FSSP is a discrete problem. Thus, the standard CS cannot be directly used to solve the FSSP. To apply CS to FSSP, one of the key issues is to construct a relationship between real number solutions and job sequences. To address this issue, some different solution representation methods have been proposed, such as the largest ranked value (LRV) (Liang et al. 2011), the smallest position value (SPV) (Tasgetiren et al. 2006), and the largest order value (LOV) (Qian et al. 2008).

In this paper, the SPV rule is utilized (Tasgetiren et al. 2006). The SPV is a simple method, which has been successfully applied to various production scheduling problems. Let each index of the dimensions of a continuous solution represent a job from $J = \{1, 2, \ldots, n\}$. Then, n indexes denote n different jobs. Assume that $X = \{x_1, x_2, \ldots, x_n\}$ is a continuous solution. By sorting the position values of X in ascending order, a job permutation π is obtained. Figure 1 presents an example of the SPV rule.

4.2 Population initialization

Population initialization plays an important role in the performance of CS and other stochastic search algorithms. To achieve a good initial population, a new method based on Nawaz-Enscore-Ham (NEH) (Nawaz et al. 1983) heuristic is employed The NEH heuristic is a famous method for solving flow shop scheduling problems (Li and Yin 2013a; Marichelvam 2012). The main idea of the NEH heuristic is that the high processing time on all machines should be scheduled as early in the sequence as possible. The detailed steps of the NEH are listed as follows (Nawaz et al. 1983).

- Calculate the total processing time of each job on all m machines. Sort the jobs in terms of the total processing time in non-increasing order. Then, we get a permutation $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$;
- The first two jobs of π are taken and the two partial possible permutations of these two jobs are evaluated. Then, the better partial permutation is chosen as the current one.
- Take the job π_j, $j = 3, 4, \ldots, n$, and find the best partial permutation by inserting it in all possible positions of the partial permutation of jobs that have been already scheduled. The best permutation would be selected for the next iteration.

Based on the above steps, we can get a good job permutation π. To suitable for the CS algorithm, the π should be converted to a continuous solution X. Let π be a job permutation, and $X = \{x_1, x_2, \ldots, x_n\}$ be a continuous solution. Then, the conversion can be defined by

$$x_{\pi_j} = \frac{(x_{\max} - x_{\min})}{n} \cdot j - x_{\max}, \quad j = 1, 2, \ldots, n, \tag{9}$$

where π_j is the jth job for a given job π, n is the number of jobs, $x_{\max} = 1$, and $x_{\min} = -1$. As seen, x_1 is the smallest value among $\{x_1, x_2, \ldots, x_n\}$, while x_n is the largest one. According to the SPV rule, the dimension index of the smallest value achieves the first job in π.

The NEH heuristic can only generate one initial solution. To generate multiple initial solutions, a new method is used

by the suggestions of Li and Yin (2013a). In the initial population, the $10\,\% \cdot N$ solutions are generated by the NEH heuristic, and the rest of the $90\,\% \cdot N$ solutions are randomly initialized.

4.3 Generalized opposition-based learning

Opposition-based learning (OBL) (Tizhoosh 2005) is an effective method for accelerating the convergence speed of population-based search algorithms. It has been successfully applied to DE (Rahnamayan et al. 2008) and PSO (Wang et al. 2011b). The main idea of OBL is the simultaneous evaluation of the current solution and its corresponding opposite solution to achieve a better approximation of the current candidate solution. However, the GOBL is usually used for continuous optimization problems. In this paper, we apply it to discrete optimization problems.

Algorithm 2: The GOBL Operation for FSSP

```
Update [a_j(t), b_j(t)] according to Eq. 12;
for i = 1 to N do
    Generate a random value for k;
    Generate x̌_i according to Eq. 13;
    Apply the SPV to convert the x̌_i into a job permutation π̌^i;
    Calculate the makespan of the π̌^i;
end
Select N fittest job permutations from all π and π̌ as the new current population;
```

Let x be a continuous solution in the population. Its opposite solution $\check{x}$ is defined by (Rahnamayan et al. 2008)

$$\check{x} = a + b - x, \tag{10}$$

where $x \in [a, b]$.

Based on the OBL, an extended version called generalized OBL (GOBL) is proposed as follows (Wang et al. 2011a).

$$\check{x}_{i,j} = k \cdot [a_j(t) + b_j(t)] - x_{i,j}, \tag{11}$$

$$a_j(t) = \min(x_{i,j}(t)),\ b_j(t) = \max(x_{i,j}(t)), \tag{12}$$

$$i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, n,$$

where $x_{i,j}$ is the jth vector of the ith solution, $\check{x}_{i,j}$ is the opposite vector of $x_{i,j}$, $a_j(t)$ and $b_j(t)$ are the minimum and maximum values of the jth dimension in current search space, respectively, and N is the population size.

The main steps of GOBL for flow shop scheduling problem are described in Algorithm 2, where $\check{\pi}^i$ is the corresponding job permutation of the opposite solution $\check{x}_i$.

4.4 Local search

Some previous studies have proven that local search is helpful to obtain good solutions for solving scheduling problems (Wang and Tang 2012; Mladenović and Hansen 1997; Tasgetiren et al. 2007). There are some popular local search operators, such as swap, insert, and inverse. In this paper, the above three local search operators are employed. Based on these operations, we can conduct local search to achieve more accurate solutions.

For the swap, two jobs at different positions w and z in solution π are exchanged. By conducting the operation, a new solution $\pi' = \text{swap}(\pi, w, z)$ is obtained. Figure 2 illustrates the process of the swap operator.

For the insert, a job in solution π is removed from its current position w and inserted into a different position z. After this operation, we can get a new solution $\pi' = \text{insert}(\pi, w, z)$. Figure 3 describes of the insert operator.

For the inverse, jobs between two different positions w and z in solution π are inversed. we can obtain a new solution $\pi' = \text{inverse}(\pi, w, z)$ after this operation. Figure 4 presents the process of the inverse operator.

The main steps of the proposed local search are inspired by Wang and Tang (2012), which are described in Algorithm

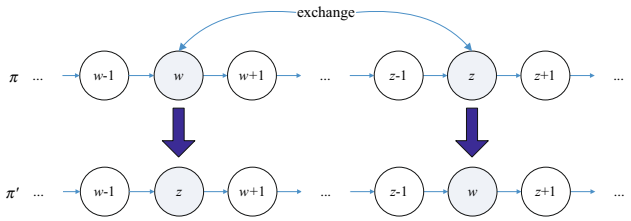

Fig. 2 The swap operator used in the local search

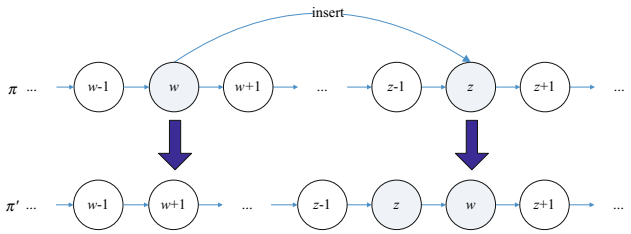

Fig. 3 The insert operator used in the local search

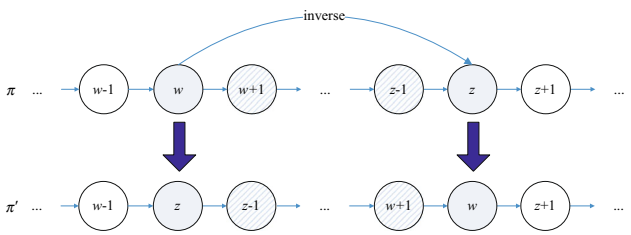

Fig. 4 The inverse operator used in the local search

3, where $\pi^\star$ is the global best job permutation found so far. The computational time complexity of local search operators is very high. So, we only conduct the local search on the global best job permutation.

Algorithm 3: The Local Search Operation

```
Let π* be the global best job permutation found so far;
Set π = π* and q = 1;
while q ≤ n × (n − 1) do
    h = 1;
    while h ≤ 3 do
        if h == 1 then
            Randomly select two different positions w and z;
            Conduct the operation π' = swap(π, w, z);
        end
        if h == 2 then
            Randomly select two different positions w and z;
            Conduct the operation π' = insert(π, w, z);
        end
        if h == 3 then
            Randomly select two different positions w and z;
            Conduct the operation π' = inverse(π, w, z);
        end
        if f(π') < f(π) then
            π = π';
            h = 1;
        end
        else
            h++;
        end
    end
    if f(π) < f(π*) then
        π* = π;
    end
    q++;
end
```

4.5 Framework of NCS

Algorithm 4: The Proposed NCS

```
Initialize population based on the modified NEH method;
while t ≤ MaxGen do
    Generate X_i by Lévy flights according to Eq. 7;
    Convert X_i to a job permutation π^i based on the SPV rule;
    Calculate the makespan of the π^i;
    Randomly choose a solution (say X_j) from the population, and its corresponding job permutation is π^j;
    if π^i is better than π^j then
        Replace X_j by the new solution X_i;
        Replace π^j by the new job permutation π^i;
    end
    Abandon a fraction (p_a) of worse nests, and build ones via Lévy flights;
    Convert the new solution to job permutations and calculate their makespan;
    Update the global best solutions;
    if rand(0, 1) ≤ p_o then
        Execute the GOBL operation (Algorithm 2);
    end
    Execute the local search (Algorithm 3);
    Update the global best solutions;
    t++;
end
```

The main steps of the proposed NCS algorithm are described in Algorithm 4, where $rand(0, 1)$ is a random value in the range [0, 1], p_o is the probability of the GOBL, and $MaxGen$ is the maximum number of generations.

5 Experimental study

5.1 Experiment setup

In this section, we present an experimental study on the performance of the proposed approach. Experiments are conducted on a set of Taillard's benchmark instances (Taillard 1990). The problems size is from 20 × 5 to 500 × 20.

To compare the performance of NCS, there are seven algorithms involved as follows.

- The standard CS.
- Improved hybrid CS (IHCS) (Marichelvam 2012).
- An alternate two phases PSO (ATPPSO) (Zhang and Sun 2009).
- A hybrid ATPPSO (I-ATPPSO) (Zhang et al. 2010a).
- Genetic algorithm (GA) (Nearchou 2004).
- A novel PSO (NPSO) (Lian et al. 2008).
- The proposed NCS.

The parameter settings of the above six algorithms are described as follows. For the standard CS, IHCS, and NCS, the population size N and p_a is set to 50 and 0.25, respectively. For NCS, the probability of GOBL p_o is set to 0.1. The parameters of GA, NPSO, ATPPSO and I-ATPPSO are set according to descriptions of Zhang et al. (2010a). The $MaxGen$ is set to 500 for the standard CS IHCS, and NCS. For other three algorithms, the $MaxGen$ is set to 900 (Zhang et al. 2010a).

For each algorithm, each test instance is conducted ten trials, and the average relative difference (ARD) is calculated as follows (Zhang et al. 2010a):

$$\text{ARD} = \frac{100 \times (C^{\text{opt}} - C^A)}{C^{\text{opt}}}, \tag{13}$$

where C^A is the makespan obtained by the NCS algorithm or other compared algorithms, and C^{opt} is the known minimum makespan for the problem or the lowest known upper bound for Taillard's instances. The ARD can measure the performance of an algorithm. A smaller ARD means that the algorithm is better.

All algorithms are encoded in VC++ 6.0 and run on an Intel Core i7-4510U CPU 2.60 GHz with 8.0 GB Memory in the Windows 7 Operating System.

5.2 Comparison of NCS with the standard CS and IHCS

In this section, we present the comparison of NCS with the standard CS. Table 1 lists the computational results achieved by the standard CS, IHCS, and NCS, where "Mean" indicates the mean makespan, and "ARD" is the average relative difference. The better results are shown in bold. As seen, the

Table 1 Results achieved by CS, IHCS and NCS

Problems	Size	CS		IHCS		NCS	
		Mean	ARD	Mean	ARD	Mean	ARD
Ta010	20 × 5	1127.6	1.77	1117.1	0.82	**1108**	**0.00**
Ta020	20 × 10	1629.4	2.41	1618.2	1.71	**1606**	**0.94**
Ta030	20 × 20	2226.2	2.21	2220.4	1.95	**2184**	**0.28**
Ta040	50 × 5	2789.3	0.26	2784.1	0.08	**2782**	**0.00**
Ta050	50 × 10	3260.2	6.37	3168.4	3.37	**3131.2**	**2.16**
Ta060	50 × 20	4045.6	7.71	3908.5	4.06	**3860.6**	**2.78**
Ta070	100 × 5	5346.1	0.45	5339.3	0.33	**5326**	**0.08**
Ta080	100 × 10	6065.5	3.77	5912.6	1.16	**5891.4**	**0.79**
Ta090	100 × 20	7042.8	9.46	6641.2	3.22	**6602.8**	**2.62**
Ta100	200 × 10	11422.5	7.00	10789.2	1.07	**10734**	**0.55**
Ta110	100 × 20	12603.4	11.65	11762.4	4.2	**11633.6**	**3.06**
Ta120	500 × 20	27486.5	3.89	26973.6	1.95	**26897.2**	**1.66**
Total average			4.75		1.99		**1.24**

NCS outperforms the standard CS and IHCS on all twelve test instances. The total average ARD of NCS is 1.24 %, which is better than the standard CS (4.75 %) and IHCS (1.99 %).

Fig. 5 lists the convergence plots of the standard CS, IHCS and NCS on Ta010-Ta060. It can be seen that NCS converges faster than the standard CS and IHCS. By combining the NEH heuristic, IHCS shows faster convergence speed than the standard CS. It seems that IHCS converges faster at the beginning of the search. As the iteration increases, the IHCS can hardly find more accurate solutions. By embedding a local search operator into IHCS, it may achieve better performance. That is why NCS performs better than IHCS.

5.3 Comparison of NCS with GA, NPSO, ATPPSO and I-ATPPSO

In this section, the proposed NCS is compared with GA, NPSO, ATPPSO and I-ATPPSO on the twelve test instances. Table 2 presents the computational results obtained by GA, NPSO, ATPPSO, I-ATPPSO, and NCS, where "Mean" indicates the mean makespan, and "ARD" is the average relative difference. The best results for each test instances are shown in bold. Results of GA, NPSO, ATPPSO and I-ATPPSO are taken from Table 4 in the literature (Zhang et al. 2010a). The parameter setting of these algorithms can be found in Zhang et al. (2010a).

As shown in Table 2, the proposed NCS achieves the best results in terms of the overall solution quality. The NCS obtains the smallest total average ARD (1.24 %), which is better than GA (4.31 %), ATPPSO (1.94 %), NPSO (2.37 %), and I-ATPPSO (1.43 %). NCS outperforms other four algorithms on all test instances except for Ta010 and Ta050. For Ta010, both NCS and I-ATPPSO can find the global optimum, while I-ATPPSO achieves better solutions than NCS on Ta050.

To compare the performance of multiple algorithms on all test instances, we conduct Friedman test according to the suggestions of García et al. (2010). Table 3 shows the average ranking of GA, ATPPSO, NPSO, I-ATPPSO, and NCS. The best ranking is shown in bold. As seen, the performance of the five algorithms ranks as follows: NCS, I-ATPPSO, ATPPSO, NPSO, and GA. The best ranking is obtained by the proposed NCS algorithm. It demonstrates that NCS is the best one among the five algorithms.

To compare the performance differences between NCS and the other four algorithms, we conduct a Wilcoxon test based on the achieved ARD (García et al. 2010). Table 4 shows the resultant p-values when comparing among NCS and the other four algorithms. The p-values below 0.05 are shown in bold. From the results, it can be seen that NCS is significantly better than GA, NPSO, ATPPSO, and I-ATPPSO.

6 Conclusions

This paper presents a new cuckoo search algorithm with local search (NCS) for solving the permutation flow shop scheduling problem. To handle the discrete variables, the SPV rule is employed to convert continuous solutions into discrete job permutations. For population initialization, the NEH heuristic is used to generate high quality initial solutions. The modified GOBL for discrete problems is utilized to acceler-

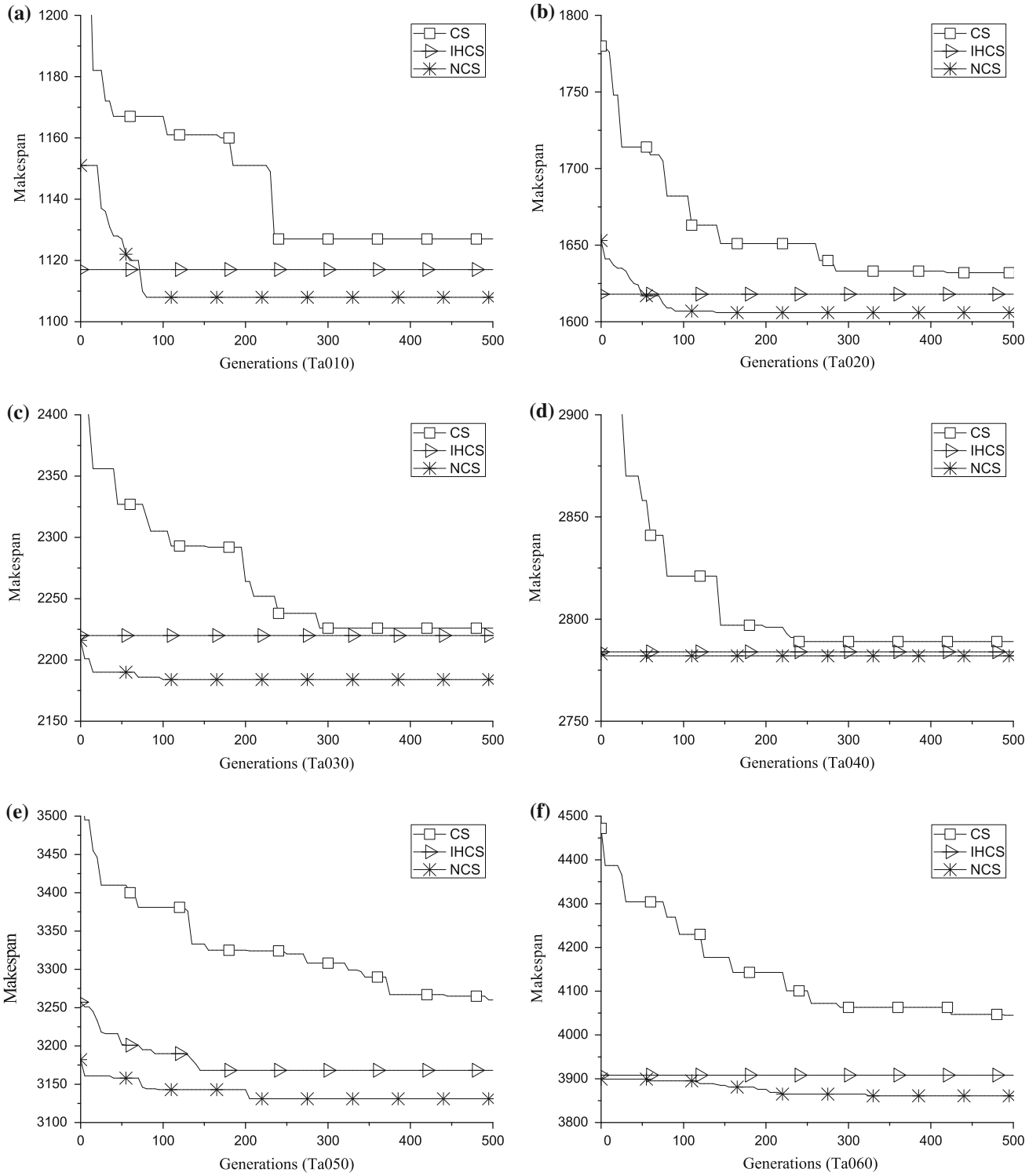

Fig. 5 The convergence characteristics of CS, IHCS and NCS on Ta010-Ta060

ate the convergence speed during the search. To improve the exploitation ability of CS, a local search strategy is employed. Experiments are conducted on twelve representative Taillard's benchmark instances. Simulation results show that the proposed NCS is significantly better than the standard CS, IHCS, GA, NPSO, ATPPSO, and IATPPSO in terms of the quality of solutions.

Table 2 Results achieved by GA, NPSO, ATPPSO, I-ATPPSO, and NCS

Problems	Size	GA		NPSO		ATPPSO		I-ATPPSO		NCS	
		Mean	ARD	Mean	ARD	Mean	ARD	Mean	ARD	Mean	ARD
Ta010	20 × 5	1135.6	2.49	1115.4	0.67	1110.4	0.22	**1108**	**0.00**	**1108**	**0.00**
Ta020	20 × 10	1632.8	2.63	1621	1.89	1608.3	1.09	1608.8	1.12	**1606**	**0.94**
Ta030	20 × 20	2237.5	2.73	2215.1	1.70	2193.6	0.72	2184.7	0.31	**2184**	**0.28**
Ta040	50 × 5	2815.7	1.21	2783.2	0.04	2782.5	0.02	2782.2	0.01	**2782**	**0.00**
Ta050	50 × 10	3225.5	5.24	3171.7	3.48	3156.1	2.97	**3129.5**	**2.10**	3131.2	2.16
Ta060	50 × 20	4008.6	6.73	3959.8	5.43	3903.2	3.92	3881.3	3.34	**3860.6**	**2.78**
Ta070	100 × 5	5372.3	0.95	5345.8	0.45	5344.9	0.43	5335.4	0.25	**5326**	**0.08**
Ta080	100 × 10	6056.3	3.62	5914.5	1.19	5900.1	0.94	5898.6	0.92	**5891.4**	**0.79**
Ta090	100 × 20	6910.3	7.40	6723.3	4.50	6690.6	3.99	6604.7	2.65	**6602.8**	**2.62**
Ta100	200 × 10	11025.6	3.28	10796.9	1.14	10846.2	1.60	10744.5	0.65	**10734**	**0.55**
Ta110	200 × 20	12269.5	8.70	11832.1	4.82	11783	4.39	11707.5	3.72	**11633.6**	**3.06**
Ta120	500 × 20	28245	6.76	27282	3.12	27246.5	2.98	27017.7	2.12	**26897.2**	**1.66**
Total average			4.31		2.37		1.94		1.43		**1.24**

Table 3 Results achieved by Friedman test

Algorithms	Rankings
NCS	**1.13**
I-ATPPSO	1.96
ATPPSO	3.00
NPSO	3.92
GA	5.00

The best ranking (with the lowest ranking value) is shown in bold

Table 4 Results achieved by Wilcoxon test

NCS vs.	p-values
GA	**2.22e–03**
NPSO	**2.22e–03**
ATPPSO	**2.21e–03**
I-ATPPSO	**9.89e–03**

The p-values below 0.05 are shown in bold

The GOBL is usually used for continuous optimization problems, while we apply it to handle discrete variables in this paper. Results show that the GOBL also works well on discrete optimization problems. For the parameter p_o, we use an empirical value. Our experiments show that a dynamical p_o may be more suitable for solving different kinds of problems. This will be investigated in the future work.

Acknowledgments This work is supported by the Priority Academic Program Development of Jiangsu Higher Education Institutions, the Humanity and Social Science Foundation of Ministry of Education of China (No. 13YJCZH174), the National Natural Science Foundation of China (Nos. 61305150 and 61261039), and the Natural Science Foundation of Jiangxi Province (No. 20142BAB217020).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Abd Elazim SM, Ali ES (2016) Optimal power system stabilizers design via cuckoo search algorithm. Int J Electr Power Energy Syst 75:99–107

Andreas C (2004) Nearchou. The effect of various operators on the genetic search for large scheduling problems. International Journal of Production Economics 88(2):191–203

Bansal SP (1977) Minimizing the sum of completion times of n jobs over m machines in a flowshopa branch and bound approach. AIIE Trans. 9(3):306–311

Basu M, Chowdhury A (2013) Cuckoo search algorithm for economic dispatch. Energy 60:99–108

Bhandari AK, Singh VK, Kumar A, Singh GK (2014) Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using kapurs entropy. Expert Syst Appl 41(7):3538–3560

Chen J, Lin Q, Hu Q (2010) Application of novel clonal algorithm in multiobjective optimization. International Journal of Information Technology & Decision Making 9(02):239–266

Chen B, Shu H, Coatrieux G, Chen G, Sun X, Coatrieux JL (2015) Color image analysis by quaternion-type moments. J Math Imaging Vis 51(1):124–144

Croce FD, Ghirardi M, Tadei R (2002) An improved branch-and-bound algorithm for the two machine total completion time flow shop problem. Eur J Oper Res 139(2):293–301

Cui L, Li G, Lin Q, Chen J, Lu N (2016) Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations. Comput Oper Res 67:155–173

Dhivya M, Sundarambal M (2011) Cuckoo search for data gathering in wireless sensor networks. Int J Mob Commun 9(6):642–656

Djelloul H, Layeb A, Chikhi S (2015) Quantum inspired cuckoo search algorithm for graph colouring problem. Int J Bio-Inspired Comput 7(3):183–194

Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B Cybern 26(1):29–41

Fu Z, Sun X, Liu Q, Zhou L, Shu J (2015) Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. IEICE Trans Commun E98–B(1):190–200

García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Inf Sci 180(10):2044–2064

Goel S, Sharma A, Bedi P (2013) Novel approaches for classification based on cuckoo search strategy. Int J Hybrid Intell Syst 10(3):107–116

Gu B, Sheng VS, Tay KY, Romano W, Li S (2015a) Incremental support vector learning for ordinal regression. IEEE Trans Neural Netw Learn Syst 26(7):1403–1416

Gu B, Sheng VS, Wang Z, Ho D, Osman S, Li S (2015b) Incremental learning for ν-support vector regression. Neural Netw 67:140–150

Huang J, Gao L, Li X (2015) An effective teaching-learning-based cuckoo search algorithm for parameter optimization problems in structure designing and machining processes. Appl Soft Comput 36:349–356

Ignall E, Schrage L (1965) Application of the branch and bound technique to some flow-shop scheduling problems. Oper Res 13(3):400–412

Johnson SM (1954) Optimal two and three stage production schedules with setup times included. Naval Res Logist Q 1(1):61–68

Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department

Karthikeyan S, Asokan P, Nickolas S, Page T (2015) A hybrid discrete firefly algorithm for solving multi-objective flexible job shop scheduling problems. Int J Bio-Inspired Comput 7(6):386–401

Kennedy J, Eberhart R (1995) Particle swarm optimization. IEEE Int Conf Neural Netw 4:1942–1948

Li J, Li X, Yang B, Sun X (2015) Segmentation-based image copy-move forgery detection scheme. IEEE Trans Inf Forensics Secur 10(3):507–518

Lian Z, Gu X, Jiao B (2008) A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan. Chaos Solitons Fractals 35(5):851–861

Liang J, Pan Q-K, Tiejun C, Wang L (2011) Solving the blocking flow shop scheduling problem by a dynamic multi-swarm particle swarm optimizer. Int J Adv Manuf Technol 55(5–8):755–762

Liang Z, Song R, Lin Q, Du Z, Chen J, Ming Z, Yu J (2015) A double-module immune algorithm for multi-objective optimization problems. App Soft Comput 35:161–174

Liang Z, Sun J, Lin Q, Du Z, Chen J, Ming Z (2016) A novel multiple rule sets data classification algorithm based on ant colony algorithm. Appl Soft Comput 38:1000–1011

Lin Q, Zhu Q, Huang P, Chen J, Ming Z, Yu J (2015a) A novel hybrid multi-objective immune algorithm with adaptive differential evolution. Comput Oper Res 62:95–111

Lin Q, Gao L, Li X, Zhang C (2015b) A hybrid backtracking search algorithm for permutation flow-shop scheduling problem. Comput Ind Eng 85:437–446

Lin Q, Chen J (2013) A novel micro-population immune multiobjective optimization algorithm. Comput Oper Res 40(6):1590–1601

Liu B, Wang L, Jin Y-H (2007) An effective pso-based memetic algorithm for flow shop scheduling. IEEE Trans Sys Man Cybern Part B Cybern 37(1):18–27

Li X, Yin M (2012) A discrete artificial bee colony algorithm with composite mutation strategies for permutation flow shop scheduling problem. Scientia Iranica 19(6):1921–1935

Li X, Yin M (2013a) A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem. Int J Prod Res 51(16):4732–4754

Li X, Yin M (2013b) An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. Adv Eng Softw 55:10–31

Ma T, Zhou J, Tang M, Tian Y, Al-dhelaan A, Al-rodhann M, Lee S (2015) Social network and tag sources based augmenting collaborative recommender system. IEICE Trans Inf Syst E98–D(4):902–910

Marichelvam MK (2012) An improved hybrid cuckoo search (ihcs) metaheuristics algorithm for permutation flow shop scheduling problems. Int J Bio-Inspired Comput 4(4):200–205

Masoud B, Hooshang J-R (2015) Robust data-driven soft sensor based on iteratively weighted least squares support vector regression optimized by the cuckoo optimization algorithm. J Nat Gas Sci Eng 22:35–41

Michael RG, David SJ (1979) Computers and intractability: a guide to the theory of NP-completeness. WH Freeman & Co., San Francisco

Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24(11):1097–1100

Naik MK, Panda R (2016) A novel adaptive cuckoo search algorithm for intrinsic discriminant analysis based face recognition. Appl Soft Comput 38:661–675

Navimipour NJ (2015) Milani FS (2015) Task scheduling in the cloud computing based on the cuckoo search algorithm. Int J Model Optim 5(1):44

Nawaz M (1983) E Emory Enscore, and Inyong Ham. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11(1):91–95

Qian B, Wang L, Rong H, Wang W-L, Huang D-X, Wang X (2008) A hybrid differential evolution method for permutation flow-shop scheduling. Int J Adv Manuf Technol 38(7–8):757–777

Rahman HF (2015) Ruhul Sarker, and Daryl Essam. A real-time order acceptance and scheduling approach for permutation flow shop problems. Eur J Oper Res 247(2):488–503

Rahnamayan S, Tizhoosh HR, Salama M (2008) Opposition-based differential evolution. Evol Comput IEEE Trans 12(1):64–79

Ren Y, Shen J, Wang J, Han J, Lee S (2015) Mutual verifiable provable data auditing in public cloud storage. J Internet Technol 16(2):317–323

Sajwan M, Acharya K, Bhargava S (2014) Swarm intelligence based optimization for web usage mining in recommender system. International Journal of Computer Applications Technology and Research 3(2):119–124

Shen J, Tan H, Wang J, Wang J, Lee S (2015) A novel routing protocol providing good transmission reliability in underwater sensor networks. J Internet Technol 16(1):171–178

Taillard E (1990) Some efficient heuristic methods for the flow shop sequencing problem. Eur J Oper Res 47(1):65–74

Tasgetiren MF, Liang Y-C, Sevkli M, Gencyilmaz G (2006) Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem. Int J Prod Res 44(22):4737–4754

Tasgetiren MF, Liang Y-C, Sevkli M, Gencyilmaz G (2007) A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. Eur J Oper Res 177(3):1930–1947

Tizhoosh HR (2005) Opposition-based learning: A new scheme for machine intelligence. In: International conference on computational intelligence for modelling control and automation IEEE, pp 695–701

Wang H, Wu Z, Rahnamayan S (2011a) Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. Soft Comput 15(11):2127–2140

Wang H, Wu Z, Rahnamayan S, Liu Y, Ventresca M (2011b) Enhancing particle swarm optimization using generalized opposition-based learning. Inf Sci 181(20):4699–4714

Wang X, Tang L (2012) A discrete particle swarm optimization algorithm with self-adaptive diversity control for the permutation flowshop problem with blocking. Appl Soft Comput 12(2):652–662

Wen X, Shao L, Xue Y, Fang W (2015) A rapid learning algorithm for vehicle classification. Inf Sci 295:395–406

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Xia Z, Wang X, Sun X, Liu Q, Xiong N (2014a) Steganalysis of LSB matching using differences between nonadjacent pixels. Multimed Tools Appl. doi:10.1007/s11042-014-2381-8

Xia Z, Wang X, Sun X, Wang B (2014b) Steganalysis of least significant bit matching using multi-order differences. Secur Commun Netw 7(8):1283–1291

Xia Z, Wang X, Sun X, Wang Q (2015) A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans Parallel Distrib Syst. doi:10.1109/TPDS.2015.2401003

Xie S, Wang Y (2014) Construction of tree network with limited delivery latency in homogeneous wireless sensor networks. Wireless personal communications 78(1):231–246

Yang X-S, Deb S (2009) Cuckoo search via Lévy flights. In: World congress on nature and biologically inspired computing IEEE, pp 210–214

Yang X-S (2010) Engineering optimization: an introduction with metaheuristic applications. Wiley, New York

Yang X-S, Deb S (2010) Engineering optimisation by cuckoo search. Int J Math Model Numer Optim 1(4):330–343

Zhang C, Ning J, Ouyang D (2010a) A hybrid alternate two phases particle swarm optimization algorithm for flow shop scheduling problem. Comput Ind Eng 58(1):1–11

Zhang J, Zhang C, Liang S (2010b) The circular discrete particle swarm optimization algorithm for flow shop scheduling problem. Expert Syst Appl 37(8):5827–5834

Zhang C, Sun J (2009) An alternate two phases particle swarm optimization algorithm for flow shop scheduling problem. Expert Syst Appl 36(3):5162–5167

Zheng Y, Jeon B, Xu D, Wu QM, Zhang H (2015) Image segmentation by generalized hierarchical fuzzy C-means algorithm. J Intel Fuzzy Syst 28(2):961–973

Zhao F, Zhang J, Wang J, Zhang C (2015) A shuffled complex evolution algorithm with opposition-based learning for a permutation flow shop scheduling problem. Int J Comput Integr Manuf 28(11):1220–1235