Expert Systems with Applications 42 (2015) 3953–3966

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

# An improved shuffled complex evolution algorithm with sequence mapping mechanism for job shop scheduling problems

Fuqing Zhao [a,c,*], Jianlin Zhang [a], Chuck Zhang [b], Junbiao Wang [c]

[a] School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, China
[b] H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
[c] Key Laboratory of Contemporary Design & Integrated Manufacturing Technology, Ministry of Education, Northwestern Polytechnical University, 710072, China

ARTICLE INFO

*Article history:*
Available online 14 January 2015

*Keywords:*
Job shop scheduling
Shuffled complex evolution
Job permutation
Sequence mapping mechanism

ABSTRACT

The job shop problem is an important part of scheduling in the manufacturing industry. A new intelligent algorithm named Shuffled Complex Evolution (SCE) algorithm is proposed in this paper with the aim of getting the minimized makespan. The sequence mapping mechanism is used to change the variables in the continuous domain to discrete variables in the combinational optimization problem; the sequence, which is based on job permutation, is adopted for encoding mechanism and sequence insertion mechanism for decoding. While considering that the basic SCE algorithm has the drawbacks of poor solution and lower rate of convergence, a new strategy is used to change the individual's evolution in the basic SCE algorithm. The strategy makes the new individual closer to best individual in the current population. The improved SCE algorithm (ISCE) was used to solve the typical job shop problems and the results show that the improved algorithm is effective to the job shop scheduling.



## 1. Introduction

The job shop scheduling problem (JSP) is one of the most difficult combinatorial optimization problems which can be described as: There are $n$ jobs to be processed through m machines. Each job must pass through each machine once and once only. Each job should be processed through the machines in a particular order, and there are no precedence constraints among operations of different jobs. Each machine can process only one job at a time, and it cannot be interrupted. Furthermore, the processing time is fixed and known. The objective of the JSP is to find a schedule to minimize the makespan, namely, the process time which is required to finish all jobs. JSP is a NP-hard problem (Cook, 1971), so it can not be exactly solved in a reasonable computation time. Many approximate methods have been developed in the recent years to solve JSP. At present, the method for the job shop scheduling mainly includes two kinds, one of which is exact methods and the other is meta-heuristic algorithm. The exact methods such as linear programming, enumeration, branch and bound are usually used to solve the small-scale scheduling. Recently, a lot of meta-heuristics methods are proposed for the optimization of job shop scheduling problems, these methods can find the optimum in reasonable amount of time and make up for the shortcomings of reaching the optimum in the exact method. Now the methods are used for job shop scheduling problems mainly include tabu search (Meeran & Morshed, 2012), simulated annealing (Elmi, Solimanpur, Topaloglu, & Elmi, 2011), genetic algorithm (Wang, 2003), shuffled frog leaping (Cai & Li, 2010), artificial neural networks (Wang & Qi-di, 2007) and particle swarm optimization (PSO) (Zhang, Sun, Ouyang, & Zhang, 2009; ZHANG, WANG, XU, & JIE, 2012). There are many better features shown when these methods are used for solving high dimensional and complex problems. However, for the job shop scheduling, many of these methods usually trap into local solution and could not get the global optimum, so the research on the job shop scheduling is still an important issue in the field of production scheduling.

Due to the general limitation of exact enumeration methods which can not solve the large scale classical JSSP (Allahverdi, Ng, Cheng, & Kovalyov, 2008; Jain & Meeran, 1999; Mati, Dauzère-Pérès, & Lahlou, 2011), many more recent papers that have developed innovative techniques such as hybrid meta-heuristics (Bilyk, Mönch, & Almeder, 2014; Bożejko, Uchroński, & Wodecki, 2010; Kuo & Cheng, 2013), TPA (team process algorithm) (Li & Chen, 2011), TS/PR (tabu search/path relinking) (Peng, Lü, & Cheng, 2015), PSO with VNS (Tavakkoli-Moghaddam, Azarkish, & Sadeghnejad-Barkousaraie, 2011; Zhao, Tang, Wang, & Jonrinaldi, 2014; Zhao, Tang, Wang, Wang, & Jonrinaldi, 2013), BFA (bacterial foraging algorithm) (Zhao, Jiang, Zhang, & Wang, 2014),

* Corresponding author at: School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, China.
*E-mail addresses:* Fzhao2000@hotmail.com (F. Zhao), 120353171@qq.com (J. Zhang), chuck.zhang@isye.gatech.edu (C. Zhang), 827585311@qq.com (J. Wang).

http://dx.doi.org/10.1016/j.eswa.2015.01.007
0957-4174/

GRASP × ELS approach (Chassaing et al., 2014). It was not surprise with the new techniques emphasis substantial progress was made and in the short time period from 2012–2015, which shall be called as the boom period for the new algorithms to JSSP, some of the most innovating algorithms were formulated.

The job shop problem (JSP) is among the hardest combinatorial problems (Johnson & Garey, 1979). Not only is it complicated, but it is one of the worst NP-complete class members. A large body of literature discusses JSP with meta-heuristic algorithm has been considered in single machine, parallel machine, flow shop and job shop environment. Lin et al. (2010) proposes a new hybrid swarm intelligence algorithm consisting of particle swarm optimization, simulated annealing technique and multi-type individual enhancement scheme to solve the job-shop scheduling problem and prove his algorithm has more robust and efficient than the existing algorithms. The Distributed and Flexible Job-shop Scheduling problem (DFJS) considers the scheduling of distributed manufacturing environments, where jobs are processed by a system of several Flexible Manufacturing Units (FMUs). De Giovanni and Pezzella (2010) extend the gene encoding to include information on job assignment and proposes an Improved Genetic Algorithm to solve the Distributed and Flexible Job-shop Scheduling problem. Wei-ling and Jing (2013) consider the job-shop problem with release dates and due dates, with the objective of minimizing the total weighted tardiness. They combine differential evolution algorithm with the improved critical path algorithm on a disjunctive graph model and presents a hybrid DE (HDE) to solve this kind of problem. Xing, Chen, Wang, Zhao, and Xiong (2010) provides an effective integration between Ant Colony Optimization (ACO) model and knowledge model and proposes a Knowledge-Based Ant Colony Optimization (KBACO) algorithm for the Flexible Job Shop Scheduling Problem (FJSSP). Zhang, Manier, and Manier (2014) consider job shop scheduling problems with transportation constraints and bounded processing times. They use a modified disjunctive graph to represent the whole characteristics and constraints of such considered problems. Compared with classical disjunctive graph, it contains not only processing nodes, but also transportation and storage nodes. The traditional scheduling models consider performance indicators such as processing time, cost and quality as optimization objectives. Salido et al. (2013) study and analyze three important objectives: energy-efficiency, robustness and makespan, and focus the attention in a job-shop scheduling problem where machines can work at different speeds. It can be observed that there exits a clear relationship between robustness and energy-efficiency and a clear trade-off between robustness/energy efficiency and makespan. Adibi, Zandieh, and Amiri (2010) study the dynamic job shop scheduling that considers random job arrivals and machine breakdowns. Considering an event driven policy rescheduling, is triggered in response to dynamic events by variable neighborhood search (VNS) and proposed a new dynamic local search method which is compared with some common dispatching rules that have widely used in the literature for dynamic job shop scheduling problem. Due to the discrete solution spaces of scheduling optimization problems, Sha and Lin (2010) modify the particle position representation, particle movement, and particle velocity of the original PSO algorithm and proposes a multi-objective PSO for solving the job shop problem. Zhang and Wu (2010) propose a hybrid simulated annealing algorithm based on a novel immune mechanism for the job shop scheduling problem with the objective of minimizing total weighted tardiness. The bottleneck jobs existing in each scheduling instance generally constitute the key factors in the attempt to improve the quality of final schedules so that the sequencing of these jobs needs more intensive optimization. Wong, Puan, Low, and Wong (2010) presents an improved bee colony optimization algorithm with Big Valley landscape exploitation (BCBV) as a biologically inspired algorithm to solve the job shop problem. The BCBV algorithm mimics the bee foraging behavior where information of newly discovered food source is communicated via waggle dances. Lei (2010) proposes a random key genetic algorithm (RKGA) for solving the fuzzy job shop scheduling problem with availability constraints which objective is to find a schedule to maximize the minimum agreement index subject to periodic maintenance, non-resemble jobs and fuzzy due-date. Hwang and Choi (2007) propose a workflow-based dynamic scheduling framework, in which a workflow management system (WfMS) serves as a dynamic job-shop scheduler and have developed an algorithm for embedding a discrete-event simulation mechanism into a WfMS, and have implemented a prototype job-shop scheduler. Tavakkoli-Moghaddam et al. (2011) proposes a new multi-objective Pareto archive particle swarm optimization (PSO) algorithm combined with genetic operators as variable neighborhood search (VNS) and presents a new mathematical model for a bi-objective job shop scheduling problem with sequence-dependent setup times and ready times that minimizes the weighted mean flow time and total penalties of tardiness and earliness. Wang and Tang (2011) propose an improved adaptive genetic algorithm (IAGA) for solving the job shop scheduling problem which was Inspired from hormone modulation mechanism, and then the adaptive crossover probability and adaptive mutation probability are designed. Renna (2010) concerns the job shop scheduling problem in cellular manufacturing systems; the schedule is created by a pheromone-based approach. Her proposed approach is carried out by a Multi-agent Architecture and it is compared with a coordination approach proposed in literature used as a benchmark. Liu, Sun, Yan, and Kang (2011) proposes an adaptive annealing genetic algorithm to deal with the job-shop planning and scheduling problem for the single-piece, small-batch, custom production mode. Kachitvichyanukul and Sitthitham (2011) propose a two-stage genetic algorithm (2S-GA) for multi-objective Job shop scheduling problems with three criteria: Minimize makespan, Minimize total weighted earliness, and Minimize total weighted tardiness. Inspired by the decision making capability of bee swarms in the nature, Banharnsakun, Sirinaovakul, and Achalakul (2012) proposes an effective scheduling method based on Best-so-far Artificial Bee Colony (Best-so-far ABC) which biases the solution direction toward the Best-so-far solution rather a neighboring solution and then use the method for solving the job shop problem. Seo and Kim (2010) propose an ant colony optimization algorithm with parameterized search space is developed for job shop problem. The problem is modeled as a disjunctive graph where arcs connect only pairs of operations related rather than all operations are connected in pairs to mitigate the increase of the spatial complexity. Thürer, Godinho Filho, and Stevenson (2013) use the controlled order release in the job shop problem with finite storage space and finds out that the best results are achieved by the workload control order release (WLCOR) rule.

SCE algorithm is a relatively new intelligent optimization method based on population, this algorithm combines shuffled method and make each individual's information shared in the space, so it has the ability of powerful global searching and can avoid being trapped into the local solution. Now the SCE algorithm has been used for solving various problems in many fields (Barakat & Altoubat, 2009; Li, Cheng, Zeng, & Lin, 2011; Tang, Li, & Fan, 2010). Job shop scheduling is a typical combinatorial optimization problem, and there have been also many researches shown the advantages of heuristics algorithm on the job shop scheduling. At present, there are very few researches in the job shop scheduling based SCE algorithm, so in this paper we use the SCE algorithm to get the makespan in the job shop scheduling problem. Because the basic SCE algorithm has the drawbacks of lower convergence rate and poor solution, an improved SCE algorithm (ISCE) is proposed in this paper; this new algorithm changes the strategy of evolution and makes the new individual closer to the best individual in the current population, this strategy improves the quality of solution and speed of convergence, this new algorithm is shown effective in the job shop scheduling from the experiment results.

The remainder of this paper is organized as follows: the SCE algorithm is proposed in Sections 2 and 3 describes the job shop scheduling problem, in Section 4, the job shop scheduling based ISCE algorithm is described and the experiment results are shown in Sections 5 and 6 concludes the paper and provides some proposals for further studies.

## 2. SCE algorithm

### 2.1. Introduction of SCE algorithm

The SCE algorithm is proposed by Duan, Gupta, and Sorooshian (1993) for the calibration of rain-runoff models and identification of parameters of aquifer formation, and it has been extensively investigated to optimize engineering problems, especially for the choice of parameter's optimization in the hydrology field, and becomes the preferred method in this field (Chu, Gao, & Sorooshian, 2010; Kuczera, 1997; Sorooshian, Duan, & Gupta, 1993). Its advantages include: (1) can quickly reach the global optimum in the multimodal search space; (2) the powerful global search can avoid being trapped into local optimum; (3) low parameter sensitivity and parameter interdependence; (4) can build the model of high dimensional problem easily.

SCE algorithm combines the advantage of simplex optimization method, competitive evolution and complex shuffling, which not only makes it with the feature of effectiveness and robustness, but also has great performance on efficiency and flexibility. Deterministic search strategy can better guide the directions of searching global optimum. From the existed literature we can learn that the SCE algorithm with better performance while solving nonlinear and non-convex high dimensional problems, meanwhile its convergence and robustness is well represented.

There are few parameters in the SCE algorithm, mainly include: $m$ (the number of individuals in each complex), $q$ (the number of individuals in each sub-complex), $p$ (the number of complexes), $p_{\min}$ (the minimum number of individuals in the complex), $\alpha$ (the number of iteration in each sub-complex), $\beta$ (the number of iteration in each complex). The choice of parameters is important for the convergence and convergence rate of SCE algorithm. In this paper, we use the expression proposed by Duan, Sorooshian, and Gupta (1994) as formula (1) under experimental calibration, where $n$ is the dimension of problems.

$$\begin{aligned} m &= 2n+1 \\ q &= n+1 \\ \beta &= m \\ \alpha &= 1 \\ p &= p_{\min} \end{aligned} \tag{1}$$

### 2.2. The procedure of SCE algorithm

**Step 1:** Initialize the parameter $p$ and $m$, and then compute the initial population size $s = p * m$.
**Step 2:** Randomly generate $s$ individuals $x\{x_1, x_2, \ldots, x_s\}$ in the feasible space, and then respectively compute their function values $f_i$ (objective function has to be minimized).
**Step 3:** Sort $s$ individuals with their function value ascending, and store them in an array $D$, $\mathrm{D} = \{(x_i, f_i), i = 1, 2, \ldots, s\}$, where $i = 1$ represents the smallest function values in the population.
**Step 4:** Partition array $D$ into $p$ complexes, noted $A_1, A_2, \ldots, A_p$, and each complex contains m individuals, where $A_k = [(x_j^k, f_j^k) \quad | x_j^k = x_{k+p(j-1)}, f_j^k = f_{k+p(j-1)}, \ j = 1, \ldots, m]$.
**Step 5:** Evolve each complex $A_k$ independently according to the CCE algorithm.
**Step 6:** Replace $A_1, A_2, \ldots, A_p$ into array $D$ and resort these individuals as their function value ascending, then stored them in array $D, D = \{A_k, k = 1, \ldots, p\}$.
**Step 7:** If the results meet the criteria, then stop; otherwise go to step 4.

### 2.3. The procedure of CCE algorithm

**Step 1:** Initialize the parameters $q, \alpha, \beta$, where $q$ is the number of sub-complex, $\alpha$ is the number of evolution in each sub-complex, and $\beta$ is the number of evolution in each complex.
**Step 2:** Assign each individual in $A_k$ with a triangular probability distribution $p_i = \{2(m+1-i)/ \quad m(m+1)\}$, where $i = 1, \ldots, m, \mathrm{x}_1^k$ has highest probability $p_1 = 2/(m+1), \mathrm{x}_m^k$ has the lowest probability $p_m = 2/\{m(m+1)\}$.
**Step 3:** Randomly choose $q$ individuals from $A_k$ under the probability distribution. Then store these individuals and their position in array $B = \{(u_i, v_i), i = 1, \ldots, q\}$ and $L$, where $v_i$ is the function value of $u_i$.
**Step 4:** Generate offspring. The detailed procedure is as follows:

(1) Sort array $B$ and $L$ so that the $q$ individuals are in ascending order of function values, and then compute the centroid of best $q-1$ points in subcomplex using the expression $g = [1/(q-1)]\sum_{j=1}^{q-1} u_j$.
(2) Compute new individual using expression $r = 2gu_q$, where $r$ is the reflection of $u_q$ and $u_q$ is the worst individual in $q$ individuals. The detailed structure of reflection $r$ is shown in Fig. 1, where $g$ is the centroid individual of $b$ and $t$, $b$ and $t$ represents the best $q-1$ points in subcomplex.
(3) If $r$ is in the feasible space $H$, then compute its function value $f_r$, go to (4); otherwise randomly generate an individual $z$ in $H$, compute its value $f_z$, set $r = z$ and $f_r = f_z$.
(4) If $f_r < f_q$, then replace individual $u_q$ by $r$, go to (6); otherwise generate a new individual $c$, set $c = 0.5(q + u_q)$ and compute its function value $f_c$. The detailed structure of contraction $c$ is shown in Fig. 2, where $g$ is the centroid individual of $b$ and $t, b$ and $t$ represents the best $q-1$ points in subcomplex.
(5) If $f_c < f_q$, then replace individual $u_q$ by $c$, go to (6); otherwise randomly generate an individual $z$ in $H$, compute its function value $f_z$, replace $u_q$ by $z$.
(6) Repeat (1)–(5) $\alpha$ times.

**Step 5:** Replace the parents with the offspring in array $B$ using their initial locations stored in array $L$ and then resort these individuals as their function value ascending.
**Step 6:** Repeat step 2-step 6 $\beta$ times.

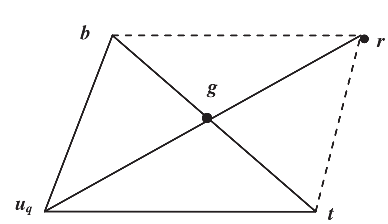

**Fig. 1.** The structure of reflection.

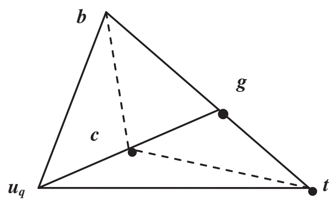


**Fig. 2.** The structure of contraction.

The flow chart for the SCE algorithm is shown in Fig. 3.

### 2.4. Simulation test of SCE

The experiment is a MATLAB simulation test, taken on the computer of Windows7 operation system, Core i5 CPU and 2G RAM. In order to test the performance of SCE, 18 benchmark functions are used, in which the mean, the standard deviation, the best value and the worst value of 20 independent runs regard as evaluative

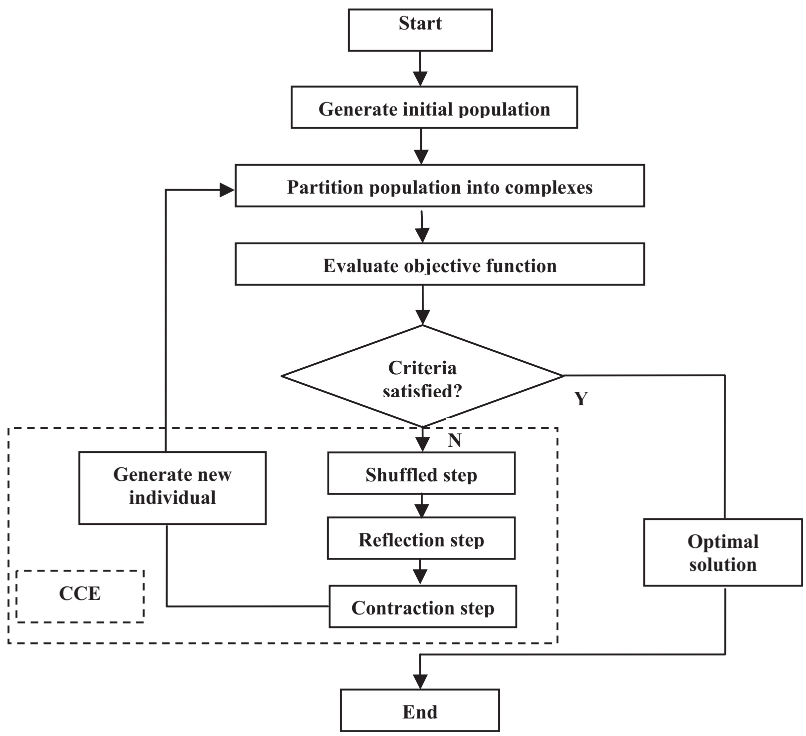


**Fig. 3.** The flow chart of SCE algorithm.

**Table 1**
Benchmark functions.

| Function | Mathematical representation |
|---|---|
| $f_1(x), [-100, 100]^n$ | $f_1(x) = \sum_{i=1}^n x_i^2$ |
| $f_2(x), [-10, 10]^n$ | $f_2(x) = \sum_{i=1}^n \mid x_i \mid + \prod_{i=1}^n \mid x_i \mid$ |
| $f_3(x), [-100, 100]^n$ | $f_3(x) = \sum_{i=1}^n (\sum_{j=1}^i x_j^2)$ |
| $f_4(x), [-100, 100]^n$ | $f_4(x) = \max_{1 \leqslant i \leqslant n} \mid x_i \mid$ |
| $f_5(x), [-30, 30]^n$ | $f_5(x) = \sum_{i=1}^n (\lfloor x_i + 0.5 \rfloor)^2$ |
| $f_6(x), [-1.28, 1.28]^n$ | $f_6(x) = (\sum_{i=1}^n i x_i^4) + random[0, 1)$ |
| $f_7(x), [-100, 100]^n$ | $f_7(x) = \sum_{i=1}^n \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$ |
| $f_8(x), [-500, 500]^n$ | $f_8(x) = \sum_{i=1}^n - x_i \sin\left(\sqrt{x_i}\right)$ |
| $f_9(x), [-5.12, 5.12]^n$ | $f_9(x) = \sum_{i=1}^n (x_i^2) - 10\cos(2\pi x_i) + 10$ |
| $f_{10}(x), [-32, 32]^n$ | $f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}\right) + e - \exp\left(\frac{1}{n}\sum_{i=1}^n \cos(2\pi x_i)\right) + 20$ |
| $f_{11}(x), [-600, 600]^n$ | $f_{11}(x) = \frac{1}{4000}\sum_{i=1}^n x_i^2 + 1 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right)$ |
| $f_{12}(x), [-50, 50]^n$ | $f_{12}(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^n u(x_i, 10, 100, 4)$ |
| $f_{13}(x), [-50, 50]^n$ | $f_{13}(x) = \frac{1}{10}\left\{10\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2\left[1 + \sin^2(3\pi x_{i+1})\right]\right\} + \frac{1}{10}(x_n - 1)^2(1 + \sin^2(2\pi x_n)) + \sum_{i=1}^n u(x_i, 5, 100, 4)$ |
| $f_{14}(x), [-10, 10]^2$ | $f_{14}(x, y) = 0.5 + \frac{\left(\sin\sqrt{x^2+y^2}\right)^2 - 0.5}{(1+0.001(x^2+y^2))^2}$ |
| $f_{15}(x), [-65.536, 65.536]^2$ | $f_{15}(x, y) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})}\right]^{-1}$ |
| $f_{16}(x), [-5, 5]^2$ | $f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$ |
| $f_{17}(x), [-5, 10] \times [0, 15]$ | $f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$ |
| $f_{18}(x), [-2, 2]^2$ | $f_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2)(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$ |

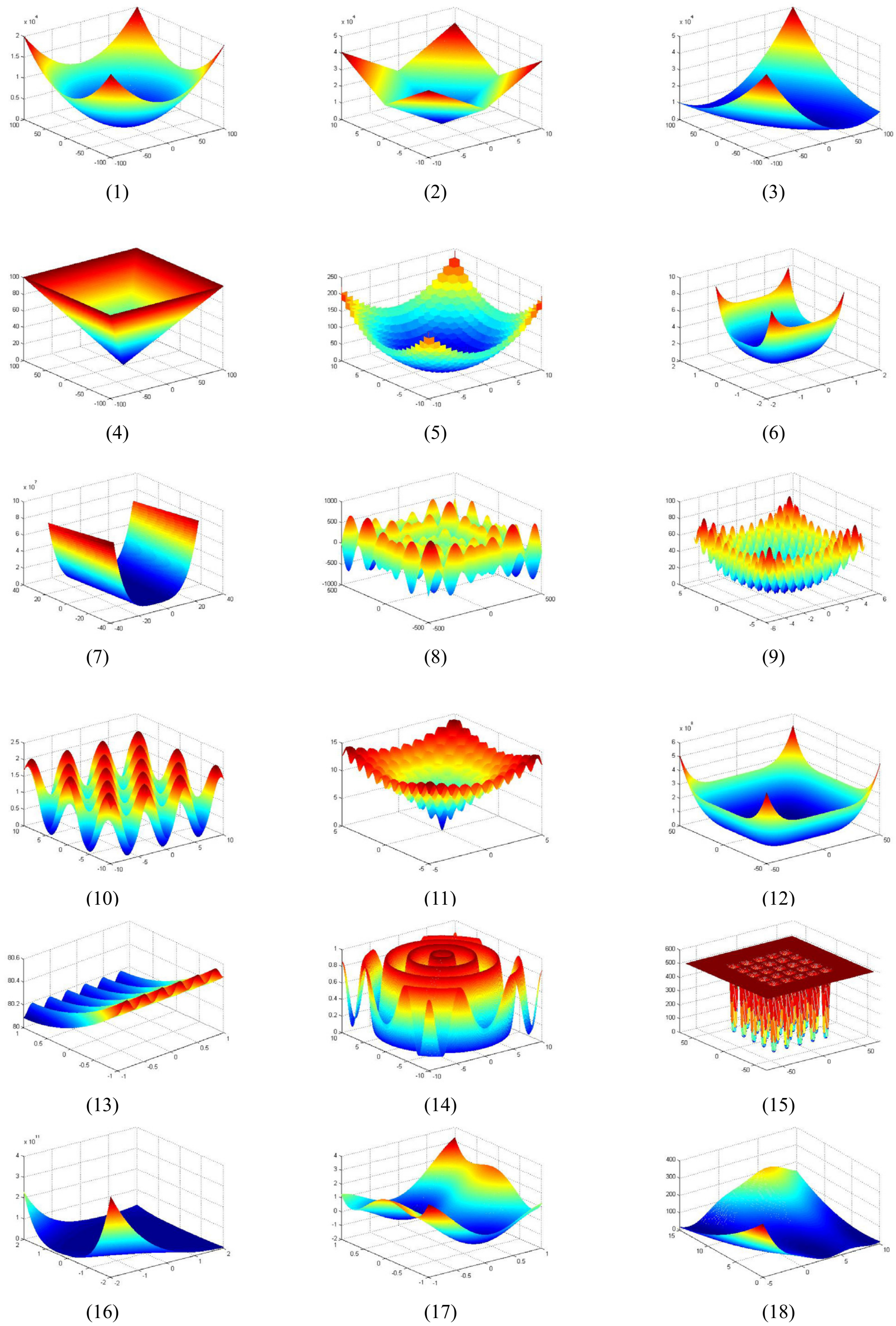


**Fig. 4.** 3D view of benchmark functions $f_1 - f_{18}$.

results of accuracy and the average CPU time represents algorithm efficiency, as shown in Table 1, the dimensions of the functions are set at 30-dimensional. Fig. 4 is the 3D view of the test functions, where $f_1$–$f_6$ are unimodal functions, $f_7$–$f_{18}$ are multimodal functions.

The parameters of these algorithms are listed as follows,

SCE: $p = 2$.
GA: population $N = 30$, the crossover probability constant $C = 0.92$ and mutation probability constant $M = 0.05$.

PSO: population $N = 30$, inertia factor $w = 0.4$ and learning factor constant $c_1 = c_2 = 2$.

The results listed in Table 2 are the average mean, best, worst solution and CPU times of each algorithm obtained from 20 runs on each benchmark function.

It can be seen from Table 2, due to the random local search ability of SCE, the quality of solutions of SCE in high-dimensional unimodal function $f_1$–$f_3$ are superior to PSO which means SCE is suitable for solving unimodal problems. From the solutions in $f_4$–$f_6$, it can be seen that SCE is not suitable for high dimensional multimodal problems, which is easily trapped into local optimal solution. Especially in f6, the PSO can hardly get rid of local optimum. SCE outperforms GA and PSO on all the benchmark functions especially for the multimodal functions $f_4$–$f_6$. This means the differential mutation operation in SCE can effectively avoid the local

**Table 2**
Comparison of experiment result.

| Functions | Algorithms | Mean | Std dev | Best | Worst | Time (s) |
|---|---|---|---|---|---|---|
| *Experimental results of function* | | | | | | |
| $f_1$ | **SCE** | 2.904e−32 | 3. 149e−32 | 1. 267e−33 | 1. 031e−33 | 0.890 |
| | **GA** | 6. 893e−13 | 2. 621e−13 | 3. 149e−13 | 1. 714e−12 | 3.351 |
| | PSO | 3. 214e−24 | 3. 645e−24 | 3. 722e−25 | 1. 321e−22 | 1. 345 |
| $f_2$ | **SCE** | 5. 138e−21 | 4. 884e−21 | 1. 130e−21 | 1. 197e−21 | 1. 471 |
| | **GA** | 3. 144e−08 | 6. 521e−08 | 2. 140e−08 | 4. 466e−08 | 4. 869 |
| | PSO | 1. 958e−15 | 8. 721e−16 | 6. 855e−16 | 3. 143e−15 | 2. 872 |
| $f_3$ | **SCE** | 1. 118e−29 | 1. 483e−29 | 4. 688e−29 | 7. 166e−29 | 1. 403 |
| | **GA** | 6. 642e−12 | 2. 658e−12 | 2. 388e−12 | 1. 539e−11 | 2. 699 |
| | PSO | 4.525e−22 | 9.454e−22 | 1.158e−23 | 4.106e−21 | 1. 656 |
| $f_4$ | **SCE** | 3. 835e−31 | 3. 343e−31 | 6. 395e−32 | 1. 352e−30 | 2. 752 |
| | **GA** | 1. 214e+00 | 1. 911e−01 | 8. 196e−01 | 1. 738e+00 | 4. 355 |
| | PSO | 3. 891e−10 | 1. 535e−09 | 4. 389e−12 | 5. 290e−09 | 3. 942 |
| $f_5$ | **SCE** | 0.003e+00 | 0.001e+00 | 0.000e+00 | 0.009e+00 | 0.323 |
| | **GA** | 0.005e+00 | 0.003e+00 | 0.000e+00 | 0.012e+00 | 0.969 |
| | PSO | 0.004e+00 | 0.002e+00 | 0.000e+00 | 0.011e+00 | 0. 643 |
| $f_6$ | **SCE** | 1. 533e−02 | 2. 861e−03 | 8. 827e−03 | 1. 699e−02 | 0.231 |
| | **GA** | 1. 382e−02 | 3. 533e−03 | 1. 719e−02 | 2. 636e−02 | 1.303 |
| | PSO | 4.211e−03 | 2. 624e−03 | 1. 214e−03 | 1.037e−02 | 0.296 |
| $f_7$ | **SCE** | 1. 202e+01 | 5. 753e−01 | 9. 692e+00 | 1.113e+01 | 2.201 |
| | **GA** | 2. 342e+01 | 2. 657e−01 | 2. 138e+01 | 2. 848e+01 | 3. 417 |
| | PSO | 4. 658e+01 | 5.407e+01 | 2. 266e+01 | 2.484e+02 | 3.341 |
| $f_8$ | **SCE** | −1.239e+04 | 1. 686e−12 | −1. 625e+04 | −1.371e+04 | 4.069 |
| | **GA** | −8.201e+03 | 2. 295e+02 | −8. 617e+03 | −8.112e+03 | 4. 861 |
| | PSO | −7.072e+03 | 5. 278e+02 | −8.345e+03 | −6.019e+03 | 4. 452 |
| $f_9$ | **SCE** | 0.001e+00 | 0.001e+00 | 0.000e+00 | 0.005e+00 | 2.999 |
| | **GA** | 1. 231e+02 | 6.045e+00 | 1. 319e+02 | 1.014e+02 | 4. 743 |
| | PSO | 3. 629e+00 | 3. 629e+00 | 6. 496e+00 | 1. 089e+01 | 4.152 |
| $f_{10}$ | **SCE** | 4. 144e−15 | 0.002e+00 | 4.441e−15 | 4.589e−15 | 1. 365 |
| | **GA** | 2. 026e−07 | 2. 729e−08 | 1. 759e−07 | 2. 188e−07 | 4. 829 |
| | PSO | 4.233e−15 | 0.000e+00 | 4.441e−15 | 4.556e−15 | 2. 718 |
| $f_{11}$ | **SCE** | 0.002e+00 | 0.003e+00 | 0.000e+00 | 0.004e+00 | 1. 806 |
| | **GA** | 5. 036e−11 | 1. 246e−10 | 1. 205e−12 | 6. 627e−10 | 3. 458 |
| | PSO | 0.004e+00 | 0.003e+00 | 0.000e+00 | 0.007e+00 | 1. 678 |
| $f_{12}$ | **SCE** | 1. 057e−32 | 2. 081e−32 | 1.570e−32 | 1.665e−32 | 2. 868 |
| | **GA** | 4. 891e−12 | 2. 913e−12 | 2. 911e−12 | 1. 610e−11 | 5. 513 |
| | PSO | 1.603e−31 | 3.332e−32 | 1. 512e−31 | 2.601e−31 | 3. 689 |
| $f_{13}$ | **SCE** | 1. 035e−32 | 2.811e−48 | 1.350e−32 | 1.499e−32 | 2. 137 |
| | **GA** | 6.802e−12 | 2. 933e−12 | 3. 407e−12 | 1. 206e−11 | 4.049 |
| | PSO | 2. 165e−30 | 1.611e−30 | 1.375e−30 | 7. 671e−30 | 3.711 |
| $f_{14}$ | **SCE** | 0.001e+00 | 0.001e+00 | 0.000e+00 | 0.004e+00 | 0. 152 |
| | **GA** | 4. 495e−04 | 1. 342e−03 | 0.000e+00 | 9. 671e−03 | 0.299 |
| | PSO | 7.607e−03 | 3. 034e−04 | 8.206e−04 | 9. 192e−03 | 0.281 |
| $f_{15}$ | **SCE** | 9.908e−01 | 9.908e−01 | 9.908e−01 | 10.211e−01 | 0.193 |
| | **GA** | 1. 424e+00 | 1. 213e+00 | 9.908e−01 | 5. 952e+00 | 0. 607 |
| | PSO | 3. 392e+00 | 1. 393e+00 | 1. 619e+00 | 8. 483e+00 | 0.575 |
| $f_{16}$ | **SCE** | −1. 203e+00 | 2. 822e−16 | −1. 203e+00 | −1. 203e+00 | 0.060 |
| | **GA** | −1. 203e+00 | 2. 022e−16 | −1.032e+00 | −1.051e+00 | 0.143 |
| | PSO | −1.023e+00 | 2.272e−16 | −1.023e+00 | −1.055e+00 | 0.077 |
| $f_{17}$ | **SCE** | 3.979e−01 | 0.002e+00 | 3.979e−01 | 4.224e−01 | 0.178 |
| | **GA** | 4.011e−01 | 0.001e+00 | 3.979e−01 | 4.353e−01 | 0.239 |
| | PSO | 4.013e−01 | 0.002e+00 | 3.979e−01 | 4.561e−01 | 0.211 |
| $f_{18}$ | **SCE** | 3.012e+00 | 3. 952e−16 | 3.000e+00 | 3.981e+00 | 0. 144 |
| | **GA** | 3.011e+00 | 3.056e−16 | 3.000e+00 | 3.348e+00 | 0.283 |
| | PSO | 3.012e+00 | 3. 952e−16 | 3.000e+00 | 3.575e+00 | 0.251 |

optimum problem and has more advantage in solving high dimensional multimodal problems.

## 3. Job shop scheduling problem

### 3.1. The description of the job shop scheduling

Job shop scheduling (JSS) is a typical kind of combinatorial optimization problem, and has been proved as NP hard. In the job shop scheduling, the set $m$ represents the number of machines, the set $n$ represents the number of jobs need to be operated on the machines, a job $i$ contains several operations $(o_{i1}, o_{i2}, \ldots, o_{im})$, each job has its own processing route; that is to say each job has a sequence of operation on the machines. $o_{ij}$ represents the $i$th job with its $j$th operation. Each operation $o_{ij}$ has a fixed processing time, and each job can be processed by only one machine at a time and cannot be interrupted until it being finished. $C_i$ is the finish time on job $i$ and would not take other factors into account. The objective of job shop scheduling is to find a better performance on the condition of some existing constraints.

### 3.2. The mathematic model of job shop scheduling

The job shop scheduling problem can be described as: $n$ jobs need to be operated on $m$ machines; each job has a processing sequence and fixed time on each machine. The mathematic model of job shop scheduling with the aim of getting the minimized makespan modeled as formula (2):

$$\min \quad \max_{1 \leqslant k \leqslant m}\{\max_{1 \leqslant i \leqslant n} c_{ik}\} \tag{2}$$

$$\begin{aligned} s.t.\ \ & c_{ik} - p_{ik} + M(1 - a_{ihk}) \geqslant c_{ih}, \quad i = 1,2,\ldots,n, \quad h,k = 1,2,\ldots,m \\ & c_{jk} - c_{ik} + M(1 - x_{ijk}) \geqslant p_{jk}, \quad i,j = 1,2,\ldots,n, \quad k = 1,2,\ldots,m \\ & c_{ik} \geqslant 0, \quad i = 1,2,\ldots,n, \quad k = 1,2,\ldots,m \end{aligned}$$

$$x_{ijk} = 0 \text{ or } 1, \quad i,j = 1,2,\ldots,n, \quad k = 1,2,\ldots,m \tag{3}$$

$$a_{ihk} = 0 \text{ or } 1, \quad i = 1,2,\ldots,n, \quad h,k = 1,2,\ldots,m \tag{4}$$

where $c_{ik}$ is the finish time of $i$th job on the $k$th machine, $p_{ik}$ is the process time of $i$th job on the $k$th machine, $M$ is a large enough positive number, $a_{ihk}$ represents the job $i$ will be processed on machine $h$ before machine $k$, $x_{ijk}$ represents the job $i$ will be processed on machine $k$ before job $j$.

## 4. The job shop scheduling based on the SCE algorithm

### 4.1. Encoding scheme

The encoding scheme based on the job permutation is adopted for job shop scheduling. For a $m \times n$ job shop scheduling, its searching space is created in $m \times n$ dimensions, and the position of an individual is represented with $m \times n$ real numbers. Its coding sequence is made up of job's number with the length of $m \times n$, the sequence shows the order of jobs be processed on the machine and a job's sequence is represented by its job's number. For a encoding sequence [2 1 1 1 3 2 2 3 3], where 1, 2, 3 respectively represents the job of J1, J2, J3, and the order represents the operation would be processed on the given machine.

SCE is an optimization algorithm used for continuous domain problem; while job shop scheduling is a typical combinatorial optimization problem, and its domain is discrete. So the SCE algorithm is not directly used for job shop scheduling, in order to solve this issue, a variable sequence mapping mechanism is proposed in this paper.

A sequence of real numbers $[k_1, k_2, \ldots, k_{mn}]$ in the domain $[0, 1]$ was randomly generated, which represents the solution in the feasible domain. Then marks the position according to the value from largest to smallest, through the transformation, the integer series $[k_1, k_2, \ldots, k_{mn}]$ can be transformed to an integer series $[\lambda_1, \lambda_2, \ldots, \lambda_{mn}]$, where $\lambda_1$ represents a job index in the series $[k_1, k_2, \ldots, k_{mn}]$. Furthermore, the $n$th smallest integers can be marked 1, and by this reflection, the $n$th largest integers can be marked$m$. The series can be marked as $[w_1, w_2, \ldots, w_{mn}]$. The corresponding series is a feasible operation permutation in the job shop scheduling problem.

A detailed example to illustrate the procession for a $3 \times 3$ job shop scheduling is shown in Fig. 5. Suppose that a feasible individual in the continuous space is [0.021, 0.154, 0.362, 0.946, 0.016, 0.401, 0.110, 0.302, 0.786], as shown in sequence 1 in the Fig. 4, it can be encode to an integer series [8, 6, 4, 1, 9, 3, 7, 5, 2] as sequence 2 according to their position index. Then according their values in the integer series, it can be transformed to integer series [3, 2, 2, 1, 3, 1, 3, 2, 1], because the 8, 9, 7 is the 3th biggest integers and 1, 3, 2 is 3th smallest integers in the sequence 2. The sequence 3 is corresponding to an operation sequence 4.

### 4.2. Decoding scheme

The procedure is as follows:

- **Step 1:** First change the encoded sequence into an order operation table.
- **Step 2:** Use the operation table and their constraints to compute the finish time on each job.
- **Step 3:** Generate the scheduling program.

From the procedure above, it can be drawn that the decoding scheme could generate an activity scheduling. In Table 3, the sequence constraints such as the time spending on each machine and processing order with each job was illustrated. For example, the job J1 can be processed on the order of M1, M2, M3 with the time of 3, 3, 2, the job J2 can be processed on the order of M1, M3, M2 with the time of 1, 5, 3, the job J3 can be processed on the order of M2, M1, M3 with the time of 3, 2, 3. According to the procedure of decoding scheme and constraint in Table 3, the operation sequence [2 1 1 1 3 2 2 3 3] represents job J2 which can be processed on machine M1 with the 1 unit time, job J1 can be processed on machine M1 with 3 unit time. The operation sequence [2 1 1 1 3 2 2 3 3] can generate an order operation table $[O_{211}, O_{111}, O_{122}, O_{133}, O_{223}, O_{232}, O_{312}, O_{321}, O_{333}]$, where $O_{ijk}$ represents the $j$th operation of job $i$ would be processed on machine$k$. The operation process of sequence [2 1 1 1 3 2 2 3 3] are scheduled as Fig. 6.

### 4.3. The fitness function

From the mathematical model of job shop scheduling above, with the aim of minimizing the makespan, the function expression can noted as *fitness*, as Eq. (5):

$$fitness = \min\{\max C\} \quad C \text{ is the finish time of jobs} \tag{5}$$

### 4.4. The job shop scheduling based on ISCE algorithm

The basic SCE algorithm has the drawbacks of poor solution and lower convergence rate while dealing with the complex problems, the local search strategy of SCE algorithm mainly depends on the CCE algorithm, by means of changing the evolution strategy of CCE algorithm, makes the new individual closer to the best individual in the current population, this strategy improves the quality of

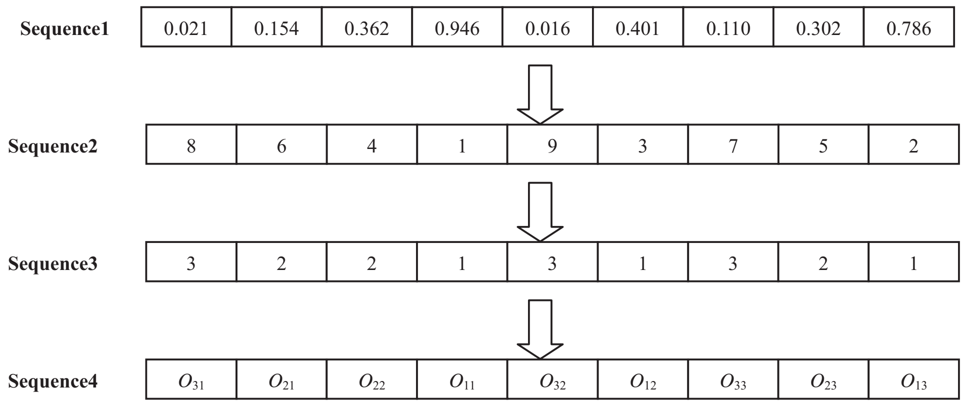


**Fig. 5.** The process of sequence change.

**Table 3**
Example of 3 × 3 job shop scheduling problem.

| Job | Time | | | Machine | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | M1 | M2 | M3 |
| J1 | 3 | 3 | 2 | 1 | 2 | 3 |
| J2 | 1 | 5 | 3 | 1 | 3 | 2 |
| J3 | 3 | 2 | 3 | 2 | 1 | 3 |

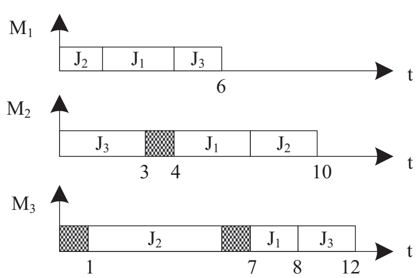


**Fig. 6.** The schedule of sequence [ 2 1 1 1 3 2 2 3 3 ].

solution and rate of convergence. The detail information is as below:

**The Basic SCE Algorithm:**

$$\text{Reflection}: \quad X_{ref} = 2 * Ce - Xw \tag{6}$$

$$\text{Contraction}: \quad X_{con} = \frac{(Xw + Ce)}{2} \tag{7}$$

**The ISCE Algorithm:**

$$\begin{aligned} &\text{Reflection}: \quad St = t * Ce + (1 - t) * Xb \quad 0 < t < 1 \\ &\qquad X_{ref} = 2 * St - Xw \end{aligned} \tag{8}$$

$$\begin{aligned} &\text{Contraction}: \quad St = t * Ce + (1 - t) * Xb \quad 0 < t < 1 \\ &\qquad X_{con} = \frac{2}{3} * St + \frac{1}{3} * Xw \end{aligned} \tag{9}$$

Where $Xw$ is the worst individual in the population, $Ce$ is the centroid individual, $Xb$ is the best individual, $X_{ref}$ is the reflection and $X_{con}$ is the contraction.

**The procedure of job shop scheduling algorithm based improved SCE is as follows:**

**Step 1:** Initialize the parameters: the number of complexes ($p$), the number of individuals in a complex ($m$), and then compute the population size $s = p * m$.

**Step 2:** Respectively generate the scheduling sequence according to real sequence from step 1, compute their function value.

**Step 3:** Sort these scheduling sequences as their function value ascending, and then partition them into complexes.

**Step 4:** Check for convergence, if satisfied, then output the results; otherwise, turn to step 5.

**Step 5:** Evolve each complex as follows:

(1) Initialize the parameters: the number of scheduling sequences in a sub-complex ($q$), the number of evolutions of sub-complex ($\alpha$), the number of evolutions of complex ($\beta$). Assign the triangular probability distribution $p_i$ for each sequence, where $p_i = \{2(m + 1 - i)/m(m + 1)\}, i = 1, \ldots, m$.

(2) Choose $q$ scheduling sequences as parents from the current population according to triangular probability, compute the worst scheduling sequence $Xw$ and the centroid of left sequences $Ce$.

(3) Compute the reflection according to Eq. (8), if this sequence is in feasible space and its fitness is lower than the worst sequence $Xw$, replace $Xw$ with this sequence, turn to step (6); otherwise turn to step (4).

(4) Compute the contraction according to Eq. (9), if the fitness of this sequence is lower than the worst sequence $Xw$, replace $Xw$ with this sequence, turn to step (6); otherwise turn to step (5).

(5) Randomly generate a right sequence in the feasible space, and compute its function value, then replace the worst sequence $Xw$ with this sequence.

(6) Repeat (2)–(5) $\beta$ times.

**Step 6:** Replace the parents with the new sequence and sort these sequences as their function value ascending.

**Step 7:** Check for convergence, if satisfied, then stop; otherwise, turn to step 3.

### *4.5. Computational complexity of ISCE algorithm for job shop scheduling*

Suppose there are $N$ jobs, $M$ machines in the job shop scheduling, the number of complexes is $p$, the number of individuals in a complex is m, the number of individuals in a sub-complex is $q$, the number of evolutions of sub-complex is $\alpha$, the number of evolutions of complex is $\beta$. From the process of the ISCE algorithm for job shop above we can learn that this algorithm is made up of four parts. The first part is to compute the function value, its complexity is $O(pmMN)$; the second part is to sort the whole individuals, in the worst case its complexity is $O(pm * pm)$; the third part is evolve the complex, that is the complexity of CCE algorithm. The CCE algorithm is mainly made up of two parts, one is sorting for $q$

individuals and the other is sorting for $m$ individuals, their complexity respectively is $O(q^2)$ and $O(m^2)$ in the worst case, its complexity is $O(\beta) * (O(q^2) + O(m^2))$ in the $\beta$ times evolution, that is $O(m^2)$; the fourth part is shuffling, its complexity is $O(pm * pm)$.

According to the analysis above, combines the Eq. (1), the computational complexity of this algorithm under the condition of $k$ times evolution can be:

$$\begin{aligned} O(p,m,q,\alpha,\beta,M,N) &= O(k) * (O(pmMN) + O(pm * pm) + O(m^2) \\ &\quad + O(pm * pm)) \\ &= O(p(2MN+1)MN) + O(p^2M^2N^2) \\ &\quad + O((2MN+1)^2) + O(p^2(2MN+1)^2) \\ &\approx O(M^2N^2) \end{aligned}$$

## 5. The simulation and analysis of the performance for ISCE to JSSP

### 5.1. Benchmark results

We consider 73 JSSP benchmarks instances from OR-Library (Beasley, 1990) which is known in this field to test the efficiency of ISCE. These instances can be classified into the following three classes: (1) FT06, FT10, FT20 were designed by Fisher and Thompson (1963); (2) the instance LA01-LA40 were designed by Lawrence (1984); (3) the instance TA01-TA30 were designed by Taillard (1993). The best known lower and upper bounds of Taillard's instances are provided on Taillard's web site (http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/jobshop.dir/best_lb_up.tx).

The algorithm is written by matlab and tested on the machine of Window XP Professional, 1.73 GHz CPU and 2G memory, and runs 20 times, the stop criteria is the times of function evaluation meets $\max n > 10000$. The results are shown in the Table 4.

From the Table 4 we can learn that the ISCE algorithm has good results on the instance of FT06, LA01, LA05-06, LA08-14, and get the optimal solution respectively. Figs. 7–9 are the process of their iteration in solving the solution.

The data in Table 5 is the comparison of ISCE algorithm and basic SCE algorithm for job shop scheduling, we can learn that the basic SCE algorithm only get the best solution on the problems LA01, LA05, LA10, LA14 and could not get optimum on the other problems. Fig. 10 shows the difference of ISCE algorithm and basic SCE algorithm on these typical job shop scheduling. Meantime on the condition of same solution, the time consuming of ISCE algorithm on these problems is less than basic SCE algorithm besides the LA01, it can be seen that the ISCE algorithm is more effective on the job shop scheduling than basic SCE algorithm.

**Table 4**
The results of ISCE algorithm for job shop scheduling.

| Instance | Scale | Optimum | Improved SCE algorithm | | |
|---|---|---|---|---|---|
| | | | Solution | Parameter ($p$) | Iteration |
| FT06 | $6 \times 6$ | 55 | **55** | 9 | 10 |
| LA01 | $10 \times 5$ | 666 | **666** | 12 | 10 |
| LA05 | $10 \times 5$ | 593 | **593** | 2 | 11 |
| LA06 | $15 \times 5$ | 926 | **926** | 5 | 11 |
| LA08 | $15 \times 5$ | 863 | **863** | 13 | 5 |
| LA09 | $15 \times 5$ | 951 | **951** | 3 | 14 |
| LA10 | $15 \times 5$ | 958 | **958** | 2 | 12 |
| LA11 | $20 \times 5$ | 1222 | **1222** | 9 | 5 |
| LA12 | $20 \times 5$ | 1039 | **1039** | 2 | 16 |
| LA13 | $20 \times 5$ | 1150 | **1150** | 3 | 11 |
| LA14 | $20 \times 5$ | 1292 | **1292** | 2 | 13 |

Bold values are the optimum which has been obtained by various algorithms till now.

### 5.2. Results and comparison with other algorithms

For FT and LA instances, we run the ISCE 20 times independently. Table 6 lists the computational results obtained by ISCE. It includes problem name (Instance), problem size (size, number of jobs, number of operations), the population size (NP), the best known solution (BKS), the best value (Best), the worst value (Max), the average makespan (Mean), the running times (CPU time). In Table 6, the solutions boldface represents that the better obtain the best known. It can be seen from Table 6 that ISCE can find the best known solution with 22 instances.

Table 7 gives the comparison of our algorithms with other novel heuristic algorithms which are TLBO (Baykasoğlu, Hamzadayi, &

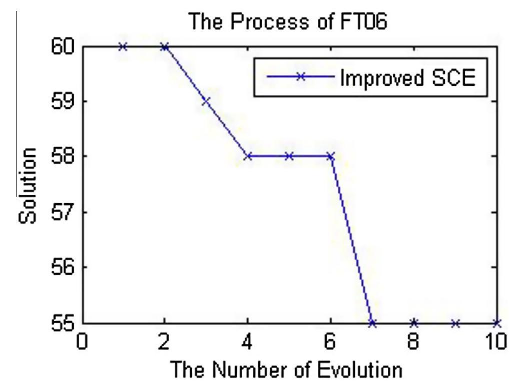


**Fig. 7.** The process of iteration of FT06.

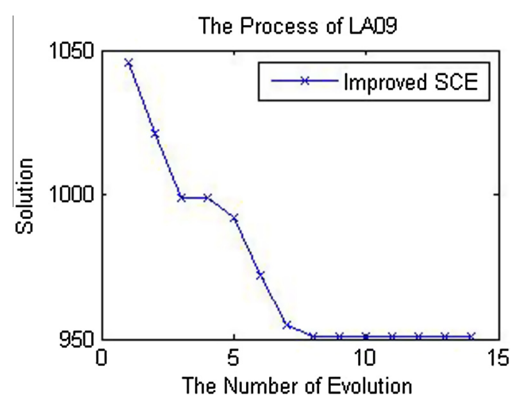


**Fig. 8.** The process of iteration of LA09.

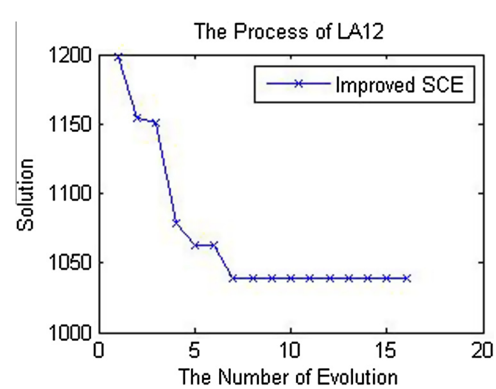


**Fig. 9.** The process of iteration of LA12.

**Table 5**
The comparison of improved SCE and basic SCE for job shop scheduling.

| Instance | Scale | Optimum | Basic SCE algorithm | | Improved SCE algorithm | |
|---|---|---|---|---|---|---|
| | | | Solution | Time (s) | Solution | Time (s) |
| FT06 | 6 × 6 | 55 | **57** | 4.211 | **55** | 2.312 |
| LA01 | 10 × 5 | 666 | 666 | **3.704** | 666 | **2.661** |
| LA05 | 10 × 5 | 593 | 593 | **0.312** | 593 | **0.124** |
| LA06 | 15 × 5 | 926 | **939** | 0.503 | **926** | 0.391 |
| LA08 | 15 × 5 | 863 | **892** | 12.091 | **863** | 11.263 |
| LA09 | 15 × 5 | 951 | **956** | 5.176 | **951** | 4.472 |
| LA10 | 15 × 5 | 958 | 958 | **0.224** | 958 | **0.154** |
| LA11 | 20 × 5 | 1222 | **1258** | 1.975 | **1222** | 1.632 |
| LA12 | 20 × 5 | 1039 | **1074** | 2.708 | **1039** | 2.115 |
| LA13 | 20 × 5 | 1150 | **1211** | 9.063 | **1150** | 8.662 |
| LA14 | 20 × 5 | 1292 | 1292 | **0.112** | 1292 | **0.074** |

Bold values are the optimum which has been obtained by various algorithms till now.

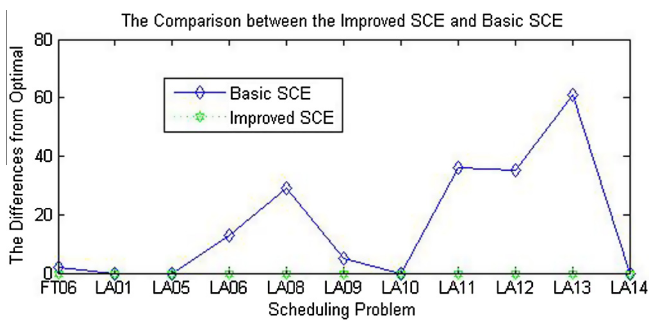


**Fig. 10.** The comparison between improved SCE and basic SCE.

**Table 6**
Results using ISCE for FT and LA problems.

| Instance | Size | NP | Rate | BKS | ISCE | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Best | Max | Mean | CPU times |
| FT06 | 6,6 | 20 | 0.05 | 55 | **55** | 55 | 55.0 | 2.312 |
| FT10 | 10,10 | 20 | 0.05 | 930 | 937 | 967 | 951.3 | 328.602 |
| FT20 | 20,5 | 20 | 0.05 | 1165 | 1180 | 1202 | 1187.1 | 418.034 |
| LA01 | 10,5 | 20 | 0.05 | 666 | **666** | 666 | 666.0 | 2.661 |
| LA02 | 10,5 | 20 | 0.05 | 655 | **655** | 676 | 665.2 | 22.092 |
| LA03 | 10,5 | 20 | 0.05 | 597 | **597** | 605 | 600.5 | 18.354 |
| LA04 | 10,5 | 20 | 0.05 | 590 | **590** | 604 | 597.4 | 24.081 |
| LA05 | 10,5 | 20 | 0.05 | 593 | **593** | 593 | 593.0 | 0.124 |
| LA06 | 15,5 | 20 | 0.05 | 926 | **926** | 926 | 926.0 | 0.391 |
| LA07 | 15,5 | 20 | 0.05 | 890 | **890** | 890 | 890.0 | 5.082 |
| LA08 | 15,5 | 20 | 0.05 | 863 | **863** | 863 | 863.0 | 11.263 |
| LA09 | 15,5 | 20 | 0.05 | 951 | **951** | 951 | 951.0 | 4.472 |
| LA10 | 15,5 | 20 | 0.05 | 958 | **958** | 958 | 958.0 | 0.154 |
| LA11 | 20,5 | 20 | 0.05 | 1222 | **1222** | 1222 | 1222.0 | 1.632 |
| LA12 | 20,5 | 20 | 0.05 | 1039 | **1039** | 1039 | 1039.0 | 2.115 |
| LA13 | 20,5 | 20 | 0.05 | 1150 | **1150** | 1150 | 1150.0 | 8.662 |
| LA14 | 20,5 | 20 | 0.05 | 1292 | **1292** | 1292 | 1292.0 | 0.074 |
| LA15 | 20,5 | 20 | 0.05 | 1207 | **1207** | 1209 | 1207.7 | 12.287 |
| LA16 | 10,10 | 20 | 0.10 | 945 | 956 | 982 | 977.1 | 87.901 |
| LA17 | 10,10 | 20 | 0.10 | 784 | **784** | 797 | 787.3 | 77.459 |
| LA18 | 10,10 | 20 | 0.10 | 848 | 849 | 873 | 861.5 | 92.182 |
| LA19 | 10,10 | 20 | 0.10 | 842 | 847 | 879 | 867.5 | 93. 320 |
| LA20 | 10,10 | 20 | 0.10 | 902 | **902** | 921 | 915.4 | 81. 416 |
| LA21 | 15,10 | 20 | 0.10 | 1046 | 1056 | 1077 | 1065.5 | 475. 617 |
| LA22 | 15,10 | 20 | 0.10 | 927 | 936 | 947 | 943.5 | 599. 903 |
| LA23 | 15,10 | 20 | 0.10 | 1032 | 1036 | 1053 | 1045.6 | 255. 103 |
| LA24 | 15,10 | 20 | 0.10 | 935 | 937 | 956 | 945.2 | 613. 172 |
| LA25 | 15,10 | 20 | 0.10 | 977 | 980 | 993 | 984.2 | 651. 542 |
| LA26 | 20,10 | 20 | 0.10 | 1218 | 1219 | 1280 | 1258.2 | 849. 796 |
| LA27 | 20,10 | 20 | 0.10 | 1235 | 1238 | 1259 | 1247.6 | 927. 642 |
| LA28 | 20,10 | 20 | 0.10 | 1216 | 1221 | 1274 | 1255.2 | 926.702 |
| LA29 | 20,10 | 20 | 0.10 | 1152 | 1160 | 1189 | 1178.2 | 906. 491 |
| LA30 | 20,10 | 20 | 0.10 | 1355 | 1371 | 1390 | 1387.6 | 896.276 |
| LA31 | 30,10 | 10 | 0.05 | 1784 | **1784** | 1786 | 1785.1 | 228.163 |
| LA32 | 30,10 | 10 | 0.05 | 1850 | **1850** | 1853 | 1852.2 | 599.369 |
| LA33 | 30,10 | 10 | 0.05 | 1719 | **1719** | 1719 | 1722.4 | 135.671 |
| LA34 | 30,10 | 10 | 0.05 | 1721 | 1723 | 1746 | 1728.6 | 869.605 |
| LA35 | 30,10 | 10 | 0.05 | 1888 | **1888** | 1891 | 1887.4 | 559.361 |
| LA36 | 15,15 | 15 | 0.05 | 1268 | 1286 | 1299 | 1274.4 | 607.311 |
| LA37 | 15,15 | 15 | 0.05 | 1397 | 1405 | 1435 | 1467.5 | 594.304 |
| LA38 | 15,15 | 15 | 0.05 | 1196 | 1221 | 1256 | 1248.6 | 578.122 |
| LA39 | 15,15 | 15 | 0.05 | 1233 | 1258 | 1300 | 1278.8 | 668.165 |
| LA40 | 15,15 | 15 | 0.05 | 1222 | 1235 | 1266 | 1280.1 | 862.554 |

Köse, 2014), EDA (He, Zeng, Xue, & Wang, 2010), MA(PR) (Hasan, Sarker, Essam, & Cornforth, 2009), RD-ACS (Rossi & Dini, 2007), PSO-priority (Sha & Hsu, 2006), LAGA (Ombuki & Ventresca, 2004), AIS (Coello, Rivera, & Cortes, 2003), Beam search (Sabuncuoglu & Bayiz, 1999), GA (Della Croce, Tadei, & Volta, 1995), PGA (Dorndorf & Pesch, 1995) and SBI (Adams, Balas, & Zawack, 1988). For the small instance FT06 and LA01-LA15, almost all the algorithms can find the best known solution. For relatively large problems LA16-LA40, the solutions obtained by the proposed algorithm are superior to or equal to the results of other compared algorithms. Among those instances, LA31, LA32, LA33, LA35 can be achieved the best known solution. Although TLBO achieves the best known solution in the instances of FT10 and FT20, from LA01 to LA40, with the larger scale of LA problems, the TLBO performs weaker than ISCE. In Table 7, it also can be seen that the results of ISCE are better than EDA, RD-ACS and PSO-priority. This shows that the combined algorithm of Simplex and Evolution Algorithm develops the advantages of these two combined algorithms, and enhances the performance of the proposed algorithm. Meanwhile, the neighborhood search also plays an important role to improve the quality of the solutions.

For TA problems, ISCE was compared with other two typical evolutionary algorithms which are genetic algorithm (GA) and particle swarm optimization (PSO). The parameters of PSO are in accordance with the reference (Lin et al., 2010). For GA, it adopts the operation-based representation to encode a chromosome. The POX crossover method and bit flip mutation is used as reproduction operators. The probability of mutation and crossover are all set to 0.5. The population size is set to 30. The other settings of the above algorithms keep consistent with the proposed algorithm. Each instance is executed by ISCE, PSO and GA for 20 times independently, respectively. Table 8 reports the computational results obtained by ISCE, PSO and GA. Table 8 includes problem name (Instances), problem size (size, number of jobs, number of operations), the best known solution (BKS) and the best known lower and upper bounds (LB, UB). The results of the best solution (Best), the mean relative error (MRE, $MRE = 100 \times (MRE - BKS\ (or\ \ UB))/BKS\ (or\ \ UB)$) and the running time (CPU times) of ISCE, PSO and GA are also shown in Table 8. The graphical representation in Figs. 11 and 12 shows the comparison of TA problems results and average computational time obtained from ISCE with PSO

**Table 7**
Comparisons of the results between ISCE and other approaches.

| Instance | Size | BKS | ISCE | Baykasoğlu et al. (2014) TLBO | He et al. (2010) EDA | Hasan et al. (2009) MA(PR) | Sha and Hsu (2006) PSO-priority |
|---|---|---|---|---|---|---|---|
| FT06 | 6,6 | 55 | **55** | 55 | 55 | – | 55 |
| FT10 | 10,10 | 930 | 937 | 938 | 937 | – | 1007 |
| FT20 | 20,5 | 1165 | 1180 | 1165 | 1184 | – | 1242 |
| LA01 | 10,5 | 666 | **666** | 666 | 666 | 667 | 681 |
| LA02 | 10,5 | 655 | **655** | 655 | – | 655 | 694 |
| LA03 | 10,5 | 597 | **597** | 597 | – | 617 | 633 |
| LA04 | 10,5 | 590 | **590** | 607 | – | 606 | 611 |
| LA05 | 10,5 | 593 | **593** | 593 | – | 593 | 593 |
| LA06 | 15,5 | 926 | **926** | 926 | 926 | 926 | 926 |
| LA07 | 15,5 | 890 | **890** | 890 | – | 890 | 890 |
| LA08 | 15,5 | 863 | **863** | 864 | – | 863 | 863 |
| LA09 | 15,5 | 951 | **951** | 951 | – | 951 | 953 |
| LA10 | 15,5 | 958 | **958** | 958 | – | 958 | 958 |
| LA11 | 20,5 | 1222 | **1222** | 1222 | 1222 | 1222 | 1222 |
| LA12 | 20,5 | 1039 | **1039** | – | – | 1039 | 1039 |
| LA13 | 20,5 | 1150 | **1150** | – | – | 1150 | 1150 |
| LA14 | 20,5 | 1292 | **1292** | – | – | 1292 | 1292 |
| LA15 | 20,5 | 1207 | **1207** | – | – | 1207 | 1232 |
| LA16 | 10,10 | 945 | 956 | 946 | 945 | 994 | 1006 |
| LA17 | 10,10 | 784 | **784** | – | – | 785 | 833 |
| LA18 | 10,10 | 848 | 849 | – | – | 861 | 901 |
| LA19 | 10,10 | 842 | 847 | – | – | 896 | 895 |
| LA20 | 10,10 | 902 | **902** | – | – | 967 | 963 |
| LA21 | 15,10 | 1046 | 1056 | 1091 | 1071 | 1090 | 1201 |
| LA22 | 15,10 | 927 | 936 | – | – | 985 | 1046 |
| LA23 | 15,10 | 1032 | 1036 | – | – | 1043 | 1146 |
| LA24 | 15,10 | 935 | 937 | – | – | 986 | 1082 |
| LA25 | 15,10 | 977 | 980 | – | – | 1077 | 1107 |
| LA26 | 20,10 | 1218 | 1219 | – | 1257 | 1303 | 1409 |
| LA27 | 20,10 | 1235 | 1238 | 1256 | – | 1328 | 1437 |
| LA28 | 20,10 | 1216 | 1221 | – | – | 1328 | 1434 |
| LA29 | 20,10 | 1152 | 1160 | – | – | 1267 | 1359 |
| LA30 | 20,10 | 1355 | 1371 | – | – | 1363 | 1517 |
| LA31 | 30,10 | 1784 | **1784** | 1784 | 1789 | 1784 | 1886 |
| LA32 | 30,10 | 1850 | **1850** | – | – | 1850 | 2000 |
| LA33 | 30,10 | 1719 | **1719** | – | – | 1719 | 1832 |
| LA34 | 30,10 | 1721 | 1723 | – | – | 1740 | 1876 |
| LA35 | 30,10 | 1888 | **1888** | – | – | 1898 | 2027 |
| LA36 | 15,15 | 1268 | 1286 | 1332 | 1292 | 1389 | 1357 |
| LA37 | 15,15 | 1397 | 1405 | – | – | 1544 | 1494 |
| LA38 | 15,15 | 1196 | 1221 | – | – | 1367 | 1338 |
| LA39 | 15,15 | 1233 | 1258 | – | – | 1342 | 1343 |
| LA40 | 15,15 | 1222 | 1235 | 1241 | – | 1357 | 1311 |

| Rossi and Dini (2007) RD-ACS | Ombuki and Ventresca (2004) LSGA | Coello et al. (2003) AIS | Sabuncuoglu and Bayiz (1999) Beam search | Della Croce et al. (1995) GA | Dorndorf and Pesch, 1995 PGA | Adams et al. (1988) SBI |
|---|---|---|---|---|---|---|
| – | – | – | – | – | – | 55 |
| – | – | 941 | 1016 | 946 | 960 | 1015 |
| – | – | – | – | 1178 | 1249 | 1290 |
| 666 | – | 666 | 666 | 666 | 666 | 666 |
| 665 | – | 655 | 704 | 666 | 681 | 720 |
| 604 | – | 597 | 650 | 666 | 620 | 623 |
| 611 | – | 590 | 620 | – | 620 | 597 |
| 593 | – | 593 | 593 | – | 593 | 593 |
| 926 | – | 926 | 926 | 926 | 926 | 926 |
| 890 | – | 890 | 890 | – | 890 | 890 |
| 863 | – | 863 | 863 | – | 863 | 868 |
| 951 | – | 951 | 951 | – | 951 | 951 |
| 958 | – | 958 | 958 | – | 958 | 959 |
| 1222 | – | – | 1222 | 1222 | 1222 | 1222 |
| 1039 | – | – | 1039 | – | 1039 | 1039 |
| 1150 | – | – | 1150 | – | 1150 | 1150 |
| 1292 | – | – | 1292 | – | 1292 | 1292 |
| 1212 | – | – | 1207 | – | 1237 | 1207 |
| 978 | 959 | 945 | 988 | 979 | 1008 | 1021 |
| 792 | 792 | 785 | 827 | – | 809 | 796 |
| 861 | 857 | 848 | 881 | – | 916 | 891 |
| 862 | 860 | 848 | 882 | – | 880 | 875 |
| 907 | 907 | 907 | 948 | – | 928 | 924 |
| 1134 | 1097 | – | 1154 | 1097 | 1139 | 1172 |
| 992 | 980 | – | 985 | – | 998 | 1040 |
| 1032 | 1032 | – | 1051 | – | 1072 | 1061 |

(continued on next page)

**Table 7** (*continued*)

| Rossi and Dini (2007) RD-ACS | Ombuki and Ventresca (2004) LSGA | Coello et al. (2003) AIS | Sabuncuoglu and Bayiz (1999) Beam search | Della Croce et al. (1995) GA | Dorndorf and Pesch, 1995 PGA | Adams et al. (1988) SBI |
|---|---|---|---|---|---|---|
| 988 | 1001 | – | 992 | – | 1014 | 1000 |
| 1042 | 1031 | 1022 | 1073 | – | 1014 | 1048 |
| 1281 | 1295 | – | 1269 | 1231 | 1278 | 1304 |
| 1324 | 1306 | – | 1316 | – | 1378 | 1325 |
| 1308 | 1302 | 1277 | 1373 | – | 1327 | 1256 |
| 1301 | 1280 | 1248 | 1252 | – | 1336 | 1294 |
| 1405 | 1406 | – | 1435 | – | 1411 | 1403 |
| 1784 | 1784 | – | 1784 | 1784 | – | 1784 |
| 1853 | 1850 | – | 1850 | – | – | 1850 |
| 1719 | 1719 | – | 1719 | – | – | 1719 |
| 1751 | 1758 | – | 1780 | – | – | 1721 |
| 1891 | 1888 | 1903 | 1888 | – | – | 1888 |
| 1349 | 1357 | 1323 | 1401 | 1305 | 1373 | 1351 |
| 1481 | 1494 | – | 1503 | – | 1498 | 1485 |
| 1307 | 1338 | 1274 | 1297 | – | 1296 | 1280 |
| 1304 | 1343 | 1270 | 1369 | – | 1351 | 1321 |
| 1296 | 1311 | 1258 | 1347 | – | 1321 | 1326 |

Bold values are the optimum which has been obtained by various algorithms till now.

**Table 8**
Results using ISCE for TA problems.

| Instance | Size | BKS or (LB, UB) | ISCE | | | PSO | | | GA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Best | MRE | CPU times | Best | MRE | CPU times | Best | MRE | CPU times |
| TA01 | 15,15 | 1231 | 1231 | 4. 729 | 1656.4 | 1421 | 15.763 | 2306.1 | 1376 | 12.306 | 5441.9 |
| TA02 | 15,15 | 1244 | 1244 | 4. 461 | 1632.5 | 1401 | 12.873 | 2595.1 | 1400 | 11.310 | 5631.1 |
| TA03 | 15,15 | 1218 | 1218 | 5. 358 | 213.6 | 1392 | 14.262 | 2714.1 | 1369 | 13.745 | 5343.9 |
| TA04 | 15,15 | 1175 | 1175 | 6. 185 | 724.1 | 1348 | 15.098 | 2266.5 | 1343 | 14.256 | 5392.1 |
| TA05 | 15,15 | 1224 | 1238 | 6. 142 | 1164. 7 | 1409 | 15.203 | 2416.7 | 1366 | 11. 178 | 5806.2 |
| TA06 | 15,15 | 1238 | 1251 | 4. 481 | 741.3 | 1410 | 13.949 | 2327.2 | 1397 | 13.139 | 5908.7 |
| TA07 | 15,15 | 1227 | 1237 | 5. 488 | 1056.1 | 1377 | 12.734 | 2403.3 | 1395 | 14.412 | 5714.1 |
| TA08 | 15,15 | 1217 | 1217 | 3. 841 | 1181.2 | 1366 | 12.474 | 2657.6 | 1331 | 9.811 | 5842.2 |
| TA09 | 15,15 | 1274 | 1291 | 4. 805 | 1429.2 | 1456 | 14.738 | 2419.9 | 1437 | 13.832 | 5961.1 |
| TA10 | 15,15 | 1241 | 1262 | 6. 205 | 1481.3 | 1427 | 15.628 | 2583.6 | 1384 | 12.424 | 5708.3 |
| TA11 | 20,15 | (1323,1357) | 1357 | 9.213 | 3015.2 | 1479 | 16.374 | 5451.8 | 1504 | 14.203 | 9418.3 |
| TA12 | 20,15 | (1351,1367) | 1366 | 8. 241 | 2721.5 | 1494 | 17.112 | 5414.6 | 1516 | 15.175 | 8442.3 |
| TA13 | 20,15 | (1282,1342) | 1331 | 9. 349 | 2586.1 | 1562 | 16.099 | 5453.1 | 1553 | 16.261 | 9385.1 |
| TA14 | 20,15 | 1345 | 1366 | 6. 211 | 2527.4 | 1476 | 17.402 | 5537.1 | 1576 | 21.309 | 9036.5 |
| TA15 | 20,15 | (1304,1339) | 1355 | 9. 093 | 2488.6 | 1483 | 18.025 | 5524.5 | 1511 | 14.481 | 8785.2 |
| TA16 | 20,15 | (1304,1367) | 1367 | 9. 702 | 2334.3 | 1461 | 14.006 | 5586.2 | 1489 | 16.032 | 9961.5 |
| TA17 | 20,15 | 1462 | 1462 | 6.098 | 2491.2 | 1576 | 15.211 | 5609.3 | 1575 | 14.041 | 9050.2 |
| TA18 | 20,15 | (1369,1396) | 1428 | 8. 364 | 2513.1 | 1510 | 15.094 | 5512.3 | 1560 | 14.551 | 9789.6 |
| TA19 | 20,15 | (1304,1332) | 1332 | 9.901 | 3485.5 | 1425 | 14.072 | 5631.2 | 1503 | 15.048 | 8850.1 |
| TA20 | 20,15 | (1318,1348) | 1348 | 10. 083 | 3096.2 | 1396 | 16.247 | 5620.1 | 1476 | 17.012 | 9291.2 |
| TA21 | 20,20 | (1573,1642) | 1649 | 9. 527 | 5005.2 | 1746 | 17.281 | 9013.4 | 1843 | 18.883 | 13860.4 |
| TA22 | 20,20 | (1542,1600) | 1600 | 9.252 | 4672.3 | 1710 | 16.035 | 9250.4 | 1769 | 17.051 | 13817.2 |
| TA23 | 20,20 | (1474,1557) | 1518 | 10.512 | 5104.3 | 1652 | 19.616 | 9367.5 | 1742 | 18.853 | 13780.6 |
| TA24 | 20, 20 | (1606,1644) | 1616 | 10.056 | 4850.2 | 1780 | 21.012 | 9541.7 | 1850 | 19.915 | 13931.2 |
| TA25 | 20,20 | (1518,1595) | 1552 | 10.312 | 4672.4 | 1672 | 17.212 | 9042.4 | 1738 | 16.315 | 14021.5 |
| TA26 | 20,20 | (1558,1643) | 1651 | 10.312 | 5026.2 | 1724 | 17.801 | 9581.1 | 1808 | 17.112 | 13851.4 |
| TA27 | 20,20 | (1617,1680) | 1731 | 9.901 | 5218.3 | 1851 | 17.208 | 9473.4 | 1890 | 13.332 | 13930.1 |
| TA28 | 20,20 | (1591,1603) | 1661 | 10.203 | 4806.1 | 1741 | 22.011 | 9363.2 | 1801 | 14.211 | 14212.5 |
| TA29 | 20,20 | (1525,1625) | 1681 | 10.208 | 5051.3 | 1791 | 15.254 | 9591.3 | 1821 | 14.012 | 13321.3 |
| TA30 | 20,20 | (1485,1584) | 1605 | 10.213 | 4981.2 | 1730 | 15.339 | 9311.4 | 1803 | 15.117 | 14008.2 |
| MRE | – | – | – | 8.011 | – | – | 13.603 | – | – | 15.519 | – |

and GA. From Table 8 and Fig. 12, we know the results obtained by the proposed algorithm are better than these two typical algorithms. Although ISCE does not obtain the best known solutions in present the number of generations for large problems, the evolutionary trend of the population does not stagnate. That is to say ISCE can further optimize to obtain the better solution. Meanwhile, in Fig. 11, the running time of ISCE is also superior to other algorithms. The population size of GA and PSO must keep a certain scale, otherwise they are easily trapped in local optimum, but the large population will increase the running time. ISCE has strong disturbance capacity, so even if the population is relatively small, it can boost the searching and readily escape the local optimum.

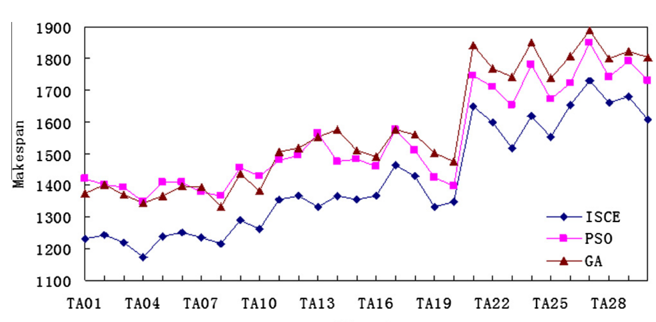


**Fig. 11.** Makespan of ISCE algorithm compared with PSO and GA for TA01-TA30.

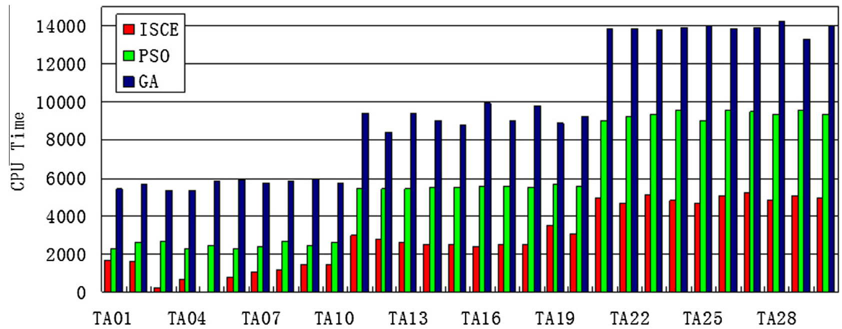


**Fig. 12.** CPU time of ISCE algorithm compared with PSO and GA for TA01-TA30.

## 6. Conclusion

This paper discusses the application of SCE algorithm and ISCE algorithm on the job shop scheduling with the aim of getting the makespan. The results show that the ISCE algorithm is more effective that basic SCE algorithm in the quality of solution and rate of convergence. Moreover, by means of using variable sequence mapping mechanism, the variable from the continuous space is reasonable corresponding to the variable from the discrete space. From the results we can get that the ISCE algorithm is effective on the job shop scheduling and the ISCE algorithm for other large-scale scheduling problems is worthy of studies. Furthermore, developing novel and effective neighborhood structures for the JSP and FJSP in the local search is meaningful and challenging. Real world problems usual involve flexible processing plan and fuzzy constraints, and the investigation on the fuzzy FJSP is our objective in near future. Meanwhile, it is also a promising direction of applying ISCE to other kinds of combination optimization problems.

However, the performance of ISCE algorithm on the large-scale job shop scheduling problem fluctuated in different situation, especially with the increase of scale in the job shop scheduling. In certain TA scheduling problems, the ISCE algorithm shows big MRE result, although it is more precise than most of the newly algorithm.

Further research will be conducted in three directions. The first one will consist in improving the performance of ISCE algorithm and verify its efficiency on large-scale scheduling problem, then applying the improved ISCE to other kinds of combination optimization problems. The second one is to try to combine ISCE with other meta-heuristics for scheduling problem. The third one is to develop novel and effective neighborhood structures for the JSP and FJSP in the local search, there are more flexible processing plan and fuzzy constraints involved in application problems.

## Acknowledgement

This work was financially supported by the National Natural Science Foundation of China under Grant No. 51365030. It was also supported by the General and Special Program of the Postdoctoral Science Foundation of China, the Science Foundation for Distinguished Youth Scholars of Lanzhou University of Technology, Lanzhou Science Bureau Project under Grant Nos. 2012M521802, 2013T60889, J201405, and 2013-4-64, respectively.

## References

Adams, J., Balas, E., & Zawack, D. (1988). Shifting bottleneck procedure for jop shop scheduling. *Management Science, 34*, 391–401.
Adibi, M., Zandieh, M., & Amiri, M. (2010). Multi-objective scheduling of dynamic job shop using variable neighborhood search. *Expert Systems with Applications, 37*, 282–287.
Allahverdi, A., Ng, C. T., Cheng, T. C. E., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research, 187*, 985–1032.
Banharnsakun, A., Sirinaovakul, B., & Achalakul, T. (2012). Job shop scheduling with the best-so-far ABC. *Engineering Applications of Artificial Intelligence, 25*, 583–593.
Barakat, S. A., & Altoubat, S. (2009). Application of evolutionary global optimization techniques in the design of RC water tanks. *Engineering Structures, 31*, 332–344.
Baykasoğlu, A., Hamzadayi, A., & Köse, S. Y. (2014). Testing the performance of teaching–learning based optimization (TLBO) algorithm on combinatorial problems: Flow shop and job shop scheduling cases. *Information Sciences, 276*, 204–218.
Beasley, J. E. (1990). OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society, 41*, 1069–1072.
Bilyk, A., Mönch, L., & Almeder, C. (2014). Scheduling jobs with ready times and precedence constraints on parallel batch machines using metaheuristics. *Computers & Industrial Engineering, 78*, 175–185.
Bożejko, W., Uchroński, M., & Wodecki, M. (2010). Parallel hybrid metaheuristics for the flexible job shop problem. *Computers & Industrial Engineering, 59*, 323–333.
Cai, L.-w., & Li, Xia (2010). Optimization of job shop scheduling based on shuffled frog leaping algorithm. *Journal of Shenzhen University Science and Engineering, 27*(5), 391–395.
Chassaing, M., Fontanel, J., Lacomme, P., Ren, L., Tchernev, N., & Villechenon, P. (2014). A GRASP × ELS approach for the job-shop with a web service paradigm packaging. *Expert Systems with Applications, 41*, 544–562.
Chu, W., Gao, X., & Sorooshian, S. (2010). Improving the shuffled complex evolution scheme for optimization of complex nonlinear hydrological systems: Application to the calibration of the Sacramento soil-moisture accounting model. *Water Resources Research*, 46.
Coello, C. A. C., Rivera, D. C., & Cortes, N. C. (2003). Use of an artificial immune system for job shop scheduling. In *Artificial immune systems* (pp. 1–10). Springer.
Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on theory of computing* (pp. 151–158). ACM.
De Giovanni, L., & Pezzella, F. (2010). An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. *European Journal of Operational Research, 200*, 395–408.
Della Croce, F., Tadei, R., & Volta, G. (1995). A genetic algorithm for the job shop problem. *Computers & Operations Research, 22*, 15–24.
Dorndorf, U., & Pesch, E. (1995). Evolution based learning in a job shop scheduling environment. *Computers & Operations Research, 22*, 25–40.
Duan, Q., Gupta, V. K., & Sorooshian, S. (1993). Shuffled complex evolution approach for effective and efficient global minimization. *Journal of Optimization Theory and Applications, 76*, 501–521.
Duan, Q., Sorooshian, S., & Gupta, V. K. (1994). Optimal use of the SCE-UA global optimization method for calibrating watershed models. *Journal of Hydrology, 158*, 265–284.
Elmi, A., Solimanpur, M., Topaloglu, S., & Elmi, A. (2011). A simulated annealing algorithm for the job shop cell scheduling problem with intercellular moves and reentrant parts. *Computers & Industrial Engineering, 61*, 171–178.
Fisher, H., & Thompson, G. (1963). *Probabilistic learning combinations of local job-shop scheduling rules*. Englewood Cliffs, NJ: Prentice-Hall, pp. 225–251.
Hasan, S. M. K., Sarker, R., Essam, D., & Cornforth, D. (2009). Memetic algorithms for solving job-shop scheduling problems. *Memetic Computing, 1*, 69–83.
He, X.-J., Zeng, J.-C., Xue, S.-D., & Wang, L.-F. (2010). An efficient estimation of distribution algorithm for job shop scheduling problem. In *1st swarm, evolutionary and memetic computing conference, SEMCCO 2010, December 16, 2010 – December 18, 2010. LNCS* (Vol. 6466, pp. 656–663). Chennai, India: Springer-Verlag.
Hwang, H. C., & Choi, B. K. (2007). Workflow-based dynamic scheduling of job shop operations. *International Journal of Computer Integrated Manufacturing, 20*, 557–566.
Jain, A. S., & Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research, 113*, 390–434.
Johnson, D. S., & Garey, M. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco: Freeman & Co.

Kachitvichyanukul, V., & Sitthitham, S. (2011). A two-stage genetic algorithm for multi-objective job shop scheduling problems. *Journal of Intelligent Manufacturing, 22*, 355–365.
Kuczera, G. (1997). Efficient subspace probabilistic parameter optimization for catchment models. *Water Resources Research, 33*, 177–185.
Kuo, R. J., & Cheng, W. C. (2013). Hybrid meta-heuristic algorithm for job shop scheduling with due date time window and release time. *International Journal of Advanced Manufacturing Technology, 67*, 59–71.
Lawrence, S. (1984). Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. In USA: Carnegie Mellon University.
Lei, D. (2010). Fuzzy job shop scheduling problem with availability constraints. *Computers & Industrial Engineering, 58*, 610–617.
Li, Gang, Cheng, Chun-tian, Zeng, Yun, & Lin, Jian-Yi (2011). Research on short-term electricity load forecasting model using support vector machine based on SCE-UA algorithm. *Journal of Dalian University of Technology, 51*(3).
Li, Y., & Chen, Y. (2011). An effective TPA-based algorithm for job-shop scheduling problem. *Expert Systems with Applications, 38*, 2913–2918.
Lin, T.-L., Horng, S.-J., Kao, T.-W., Chen, Y.-H., Run, R.-S., Chen, R.-J., et al. (2010). An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications, 37*, 2629–2636.
Liu, M., Sun, Z.-j., Yan, J.-w., & Kang, J.-s. (2011). An adaptive annealing genetic algorithm for the job-shop planning and scheduling problem. *Expert Systems with Applications, 38*, 9248–9255.
Mati, Y., Dauzère-Pérès, S., & Lahlou, C. (2011). A general approach for optimizing regular criteria in the job-shop scheduling problem. *European Journal of Operational Research, 212*, 33–42.
Meeran, S., & Morshed, M. (2012). A hybrid genetic tabu search algorithm for solving job shop scheduling problems: A case study. *Journal of Intelligent Manufacturing, 23*, 1063–1078.
Ombuki, B. M., & Ventresca, M. (2004). Local search genetic algorithms for the job shop scheduling problem. *Applied Intelligence, 21*, 99–109.
Peng, B., Lü, Z., & Cheng, T. C. E. (2015). A tabu search/path relinking algorithm to solve the job shop scheduling problem. *Computers & Operations Research, 53*, 154–164.
Renna, P. (2010). Job shop scheduling by pheromone approach in a dynamic environment. *International Journal of Computer Integrated Manufacturing, 23*, 412–424.
Rossi, A., & Dini, G. (2007). Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method. *Robotics and Computer-Integrated Manufacturing, 23*, 503–516.
Sabuncuoglu, I., & Bayiz, M. (1999). Job shop scheduling with beam search. *European Journal of Operational Research, 118*, 390–412.
Salido, M. A., Escamilla, J., Barber, F., Giret, A., Tang, D., & Dai, M. (2013). Energy-aware parameters in job-shop scheduling problems. In *GREEN-COPLAS 2013: IJCAI 2013 workshop on constraint reasoning, planning and scheduling problems for a sustainable future* (pp. 44–53).
Seo, M., & Kim, D. (2010). Ant colony optimisation with parameterised search space for the job shop scheduling problem. *International Journal of Production Research, 48*, 1143–1154.
Sha, D., & Lin, H.-H. (2010). A multi-objective PSO for job-shop scheduling problems. *Expert Systems with Applications, 37*, 1065–1070.
Sha, D. Y., & Hsu, C.-Y. (2006). A hybrid particle swarm optimization for job shop scheduling problem. *Computers and Industrial Engineering, 51*, 791–808.
Sorooshian, S., Duan, Q., & Gupta, V. K. (1993). Calibration of rainfall–runoff models: Application of global optimization to the Sacramento soil moisture accounting model. *Water Resources Research, 29*, 1185–1194.
Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research, 64*, 278–285.
Tang, H.-S., Li, P.-F., & Fan, C.-X. (2010). Shuffled complex evolution algorithm for structural system identification. *Journal of Vibration Engineering, 23*, 243–248.
Tavakkoli-Moghaddam, R., Azarkish, M., & Sadeghnejad-Barkousaraie, A. (2011). A new hybrid multi-objective Pareto archive PSO algorithm for a bi-objective job shop scheduling problem. *Expert Systems with Applications, 38*, 10812–10821.
Thürer, M., Godinho Filho, M., & Stevenson, M. (2013). Coping with finite storage space in job shops through order release control: An assessment by simulation. *International Journal of Computer Integrated Manufacturing, 26*, 830–838.
Wang, L. (2003). *Shop scheduling with genetic algorithms*. Beijing: Tsinghua University Press.
Wang, L., & Tang, D.-B. (2011). An improved adaptive genetic algorithm based on hormone modulation mechanism for job-shop scheduling problem. *Expert Systems with Applications, 38*, 7243–7250.
Wang, W.-l., & Qi-di, W. (2007). *Production scheduling intelligent algorithm and application*. Beijing: Science Press.
Wei-ling, W., & Jing, Y. (2013). A hybrid differential evolution algorithm for job shop scheduling problem to minimize the total weighted tardiness. In *2013 International conference on management science and engineering (ICMSE)* (pp. 294–300). IEEE.
Wong, L.-P., Puan, C. Y., Low, M. Y. H., & Wong, Y. W. (2010). Bee colony optimisation algorithm with big valley landscape exploitation for job shop scheduling problems. *International Journal of Bio-Inspired Computation, 2*, 85–99.
Xing, L.-N., Chen, Y.-W., Wang, P., Zhao, Q.-S., & Xiong, J. (2010). A knowledge-based ant colony optimization for flexible job shop scheduling problems. *Applied Soft Computing, 10*, 888–896.
Zhang, C.-S., Sun, J.-G., Ouyang, D.-T., & Zhang, Y.-G. (2009). A self-adaptive hybrid particle swarm optimization algorithm for flow shop scheduling problem. *Chinese Journal of Computers, 32*, 2137–2146.
ZHANG, J., WANG, W.-l., XU, X.-l., & JIE, J. (2012). Hybrid particle-swarm optimization for multi-objective flexible job-shop scheduling problem. *Control Theory & Applications, 6*, 005.
Zhang, Q., Manier, H., & Manier, M.-A. (2014). A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints. *International Journal of Production Research, 52*, 985–1002.
Zhang, R., & Wu, C. (2010). A hybrid immune simulated annealing algorithm for the job shop scheduling problem. *Applied Soft Computing, 10*, 79–89.
Zhao, F., Jiang, X., Zhang, C., & Wang, J. (2014). A chemotaxis-enhanced bacterial foraging algorithm and its application in job shop scheduling problem. *International Journal of Computer Integrated Manufacturing*, 1–16.
Zhao, F. Q., Tang, J. X., Wang, J. B., & Jonrinaldi (2014). An improved particle swarm optimization with decline disturbance index (DDPSO) for multi-objective job-shop scheduling problem. *Computers & Operations Research, 45*, 38–50.
Zhao, F. Q., Tang, J. X., Wang, J. Z., Wang, J. B., & Jonrinaldi (2013). Application of PSO with different typical neighbor structure to complex job shop scheduling problem. *Applied Mathematics & Information Sciences, 7*, 499–503.