Expert Systems with Applications 39 (2012) 6309–6319

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

Enhanced parallel cat swarm optimization based on the Taguchi method

Pei-Wei Tsai [a], Jeng-Shyang Pan [a,b], Shyi-Ming Chen [c,d,*], Bin-Yih Liao [a]

[a] Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan, ROC
[b] Innovative Information Industry Research Center, Harbin Institute of Technology, Shenzhen Graduate School, Nanshan, Shenzhen City, Guangdong Province, China
[c] Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, ROC
[d] Graduate Institute of Educational Measurement and Statistics, National Taichung University of Education, Taichung, Taiwan, ROC

ARTICLE INFO

Keywords:
Cat swarm optimization
Enhanced parallel cat swarm optimization
Parallel cat swarm optimization
Taguchi method

ABSTRACT

In this paper, we present an enhanced parallel cat swarm optimization (EPCSO) method for solving numerical optimization problems. The parallel cat swarm optimization (PCSO) method is an optimization algorithm designed to solve numerical optimization problems under the conditions of a small population size and a few iteration numbers. The Taguchi method is widely used in the industry for optimizing the product and the process conditions. By adopting the Taguchi method into the tracing mode process of the PCSO method, we propose the EPCSO method with better accuracy and less computational time. In this paper, five test functions are used to evaluate the accuracy of the proposed EPCSO method. The experimental results show that the proposed EPCSO method gets higher accuracies than the existing PSO-based methods and requires less computational time than the PCSO method. We also apply the proposed method to solve the aircraft schedule recovery problem. The experimental results show that the proposed EPCSO method can provide the optimum recovered aircraft schedule in a very short time. The proposed EPCSO method gets the same recovery schedule having the same total delay time, the same delayed flight numbers and the same number of long delay flights as the Liu, Chen, and Chou method (2009). The optimal solutions can be found by the proposed EPCSO method in a very short time.

1. Introduction

In recent years, some artificial intelligence (AI) methods have been presented to solve optimization problems. Chu, Tsai, and Pan (2006) and Chu and Tsai (2007) presented the cat swarm optimization (CSO) method for solving optimization problems by retaining the natural behaviors of cats. Tsai, Pan, Chen, Liao, and Hao (2008) presented a parallel cat swarm optimization (PCSO) method based on the framework of parallelizing the structure of the CSO method. Genetic algorithms (GA) have successfully been used in the internet service (Elliott & Krzymien, 2009) and impedance measurements (Janeiro & Ramos, 2009); particle swarm optimization (PSO) techniques have successfully been used to design antennas (Wu et al., 2009) and to construct parameters in neural network systems (Lin, Chen, & Lin, 2009); ant colony optimization (ACO) techniques have successfully been used to solve the traveling salesman problem (TSP) (Dorigo & Gambardella, 1997) and the routing problem of networks (Pinto, Nägele, Dejori, Runkler, & Sousa, 2009); artificial bee colony (ABC) techniques have successfully been used to solve the lot-streaming flow shop scheduling problem (Pan, Tasgetiren, Suganthan, & Chua, 2011); cat swarm optimization (CSO) techniques have successfully been used to adjust the parameters of the SVM (Lin et al., 2009). Moreover, in the industry, the Taguchi method (Taguchi, Chowdhury, & Taguchi, 2000) has successfully been used for optimizing the product-line design and the process conditions. Tsai, Liu, and Chou (2004) successfully adopted the Taguchi method into the GA's crossover process and presented the hybrid Taguchi-genetic algorithm (HTGA).

Although the parallel cat swarm optimization (PCSO) method presented in Tsai et al. (2008) has the ability to find the near best solution under more strict conditions, its computational speed is not efficient. It is obvious that to reduce the computational time of the PCSO method and to keep high accuracy results with a small population size simultaneously are the desired goals of the PCSO method. Therefore, in this paper, we propose the enhanced parallel cat swarm optimization (EPCSO) method by adopting the orthogonal array of the Taguchi method into the tracing mode process of the PCSO method. The proposed EPCSO method can successfully been used for solving optimization problems.

* Corresponding author. Tel.: +886 2 27376417; fax: +886 2 27301081.
E-mail address: smchen@mail.ntust.edu.tw (S.-M. Chen).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.11.117

2. Parallel cat swarm optimization

Tsai et al. (2008) have proposed the parallel cat swarm optimization (PCSO) method for solving optimization problems. The basic idea of the PCSO method utilizes the major structure of the cat swarm optimization (CSO) method proposed by Chu et al. (2006). The CSO method has two modes, i.e., the seeking mode and the tracing mode, for simulating the behaviors of cats to move the individuals in the solution space. By adjusting the parameter *MR*, the ratio of individuals moved by the seeking process and the tracing process can be controlled, where $MR \in [0,1]$. Some methods for splitting a population into several sub-populations to construct a parallel structure have been presented, such as the island-model genetic algorithm (Whitley, Rana, & Heckendorn, 1999), the parallel genetic algorithm (Abramson & Abela, 1992), the ant colony system with communication strategies (Chu, Roddick, & Pan, 2004) and the parallel particle swarm optimization algorithm with communication strategies (Chang, Chu, Roddick, & Pan, 2005). Each of the sub-populations evolves independently and shares the information they have occasionally. It results in the reducing of the population size for each sub-population and the benefit of cooperation is achieved.

In the PCSO method (Tsai et al., 2008), the individuals are separated into a predefined number of groups in the initial process to construct the virtual parallel space for the individuals. If we let the predefined number of groups be equal to 1, then the PCSO method becomes the CSO method (Chu et al., 2006) due to the fact that there is only one group. The individuals in the same group provide a local near best solution for their group in every generation, and the global near best solution found so far can be discovered by comparing the local near best solutions collected from the parallel groups. The individuals in a group can only access the near best solution discovered by their own group, but when the process of information exchanging is applied, the parallel groups can receive a near best solution from another randomly picked group. The difference between the PCSO method and the CSO method is described as follows. At the beginning of the PCSO method, *N* individuals are created and then they are separated into *G* groups. The calculation of the PCSO method in the tracing mode is different from that of the CSO method and there exists an information exchanging process. The process of the PCSO method is reviewed as follows (Tsai et al., 2008):

Step 1: Create *N* cats, randomly sprinkle the cats into the *M*-dimensional solution space within the constrain ranges of the initial value and randomly collect them into *G* groups. Generate the velocities for each dimension. Set the motion flags of the cats to make them move into the tracing mode or the seeking mode according to the user predefined value of *MR*, where $MR \in [0,1]$. The cats moved by the seeking mode process present higher exploitative capacity. Contrarily, the cats gain a higher explorative ability when they are moved by the tracing mode process. Hence, *MR* affects the ratio of artificial agents to work on the exploitation and exploration.

Step 2: Evaluate the fitness values of the cats, respectively, by taking the coordinates into the fitness function, which represents the benchmark and the characteristics of the problem to be solved. After calculating the fitness values of the cats, respectively, record the coordinate x_{best} and the fitness value of the cat which has the best fitness value found so far.

Step 3: Move the cats by taking the operations in the seeking mode by Eqs. (1)–(4) or the parallel tracing mode by Eqs. (5) and (6) according to the statuses of the motion flags.

Step 4: Reset the motion flags of all cats and separate them into statuses that indicating the seeking or the tracing by repick $[N \times (1 - MR)]$ cats to move in the seeking mode and $(N \times MR)$ cats to move in the parallel tracing mode.

Step 5: Check whether the number of iterations reaches a predefined iteration number. If the condition is satisfied, apply the information exchanging process.

Step 6: Check whether the process satisfies the termination condition. If the process is terminated, output the coordinate which represents the found best solution and **Stop**. Otherwise, go to **Step 2**.

2.1. The seeking mode process

In the seeking mode, the cat moves slowly and conservatively. It observes the environment before it moves. Four essential factors are defined in the seeking mode, i.e., Seeking Memory Pool (*SMP*), Seeking Range of the selected Dimension (*SRD*), Counts of Dimension to Change (*CDC*) and Self-Position Considering (*SPC*). *SMP* is used to define the size of the seeking memory for each cat to indicate the points sought by the cat. The cat will pick a point from the memory pool according to the rules described later; *SRD* declares the mutative ratio for the selected dimensions; *CDC* discloses how many dimensions will be varied. In the seeking mode, if a dimension is selected to mutate, the difference between the new value and the old one cannot be out of the range defined by *SRD*; *SPC* is a Boolean variable, which decides whether the point in which the cat is already standing will be one of the candidates to move to. No matter the value of *SPC* is true or false, the value of *SMP* will not be influenced. These factors are all playing important roles in the seeking mode. The process of the seeking mode is reviewed as follows (Tsai et al., 2008):

Step 1: Generate *j* copies of cat_k, where $j = SMP$. If the value of *SPC* is true, let $j = SMP - 1$ and return the present position as one of the candidates.

Step 2: According to *CDC*, plus/minus *SRD* percents of the current value randomly and replace the old one for all copies according to Eqs. (1)–(3):

$$M = Modify \cup (1 - Modify), \tag{1}$$

$$|Modify| = CDC \times M, \tag{2}$$

$$x_{jd} = \begin{cases} x_{jd}, & d \notin Modify, \\ (1 + rand \times SRD) \times x_{jd}, & d \in Modify, \end{cases}\Bigg|_{d=1,2,\ldots,M} \quad \forall j, \tag{3}$$

where *Modify* denotes the elements of the selected dimensions, whose values will be modified, and *rand* is a random variable in the range $[0,1]$.

Step 3: Calculate the fitness values of all candidate points, respectively.

Step 4: If all the fitness values are not exactly equal, calculate the selecting probability P_i of each candidate point, shown as follows:

$$P_i = \begin{cases} 1, & \text{if } FS_{\max} = FS_{\min}, \\ \frac{|FS_i - FS_b|}{FS_{\max} - FS_{\min}}, & \text{where } 0 < i < j, \quad \text{otherwise}. \end{cases} \tag{4}$$

If the goal of the fitness function is to find the minimum solution, then let $FS_b = FS_{\max}$. Otherwise, let $FS_b = FS_{\min}$, where $FS_{\max}$ denotes the largest *FS* in the candidates and $FS_{\min}$ denotes the smallest one.

Step 5: Randomly pick the point to move from the candidate points and replace the position of cat_k.

2.2. The parallel tracing mode process

The parallel tracing mode process is reviewed as follows (Tsai et al., 2008):

Step 1: Update the velocities for every dimension $v_{k,d}(t)$ for the cat_k at the current iteration, shown as follows:

$$v_{k,d}(t) = v_{k,d}(t-1) + r_1 \cdot c_1 \cdot [x_{l_{best},d}(t-1) - x_{k,d}(t-1)], \quad \text{where } d = 1,2,\ldots,M \tag{5}$$

where t denotes the iteration number, $x_{l_{best},d}(t-1)$ denotes the position of the cat which has the best fitness value at the previous iteration in the group that cat_k belongs to and M denotes the dimension of the solution space.

Step 2: Check whether the velocities are in the range of maximum velocity. The new velocity is bounded to the maximum velocity in case the new velocity is over-range.

Step 3: Update the position of cat_k according to Eq. (6):

$$x_{k,d}(t) = x_{k,d}(t-1) + v_{k,d}(t). \tag{6}$$

2.3. The information exchanging process

The information exchange process forces the sub-populations to exchange their information and achieves somehow the cooperation. It defined a parameter *ECH* to control the exchanging of the information between sub-populations. The information exchanging process is applied once per *ECH* iterations. The process of information exchanging consists of the following four steps (Tsai et al., 2008):

Step 1: Pick up a group of sub-populations sequentially and sort the individuals in this group according to their fitness values.

Step 2: Randomly select a local best solution from an unrepeatable group.

Step 3: The individual whose fitness value is the worst in the group is replaced by the selected local best solution.

Step 4: Repeatedly perform Step 1 to Step 3 G times to let every group receives a local best solution from the others.

3. The Taguchi method

The Taguchi method (Taguchi et al., 2000) is an important tool for robust design. It is widely used for optimizing the product-line design and the process conditions due to the fact that it can provide high quality products with low development costs (Tsai et al., 2004). One of the major tools in the Taguchi method is called the orthogonal array, which is adopted by the proposed EPCSO method. In the EPCSO method, only the two-level orthogonal array is used to take part in the process. In the following, we briefly review the concept of two-levels orthogonal arrays from Taguchi et al. (2000). An array is called orthogonal due to that each column indicates a value of a considered factor, and the factors listed in the orthogonal array can be evaluated independently. Every row in an orthogonal array represents a set of parameters for one run of the experiment. The two-levels orthogonal array can be described as follows:

$$L_n(2^{n-1}), \tag{7}$$

where $n = 2^k$, k is a positive integer which is greater than 1, and $n-1$ denotes the number of columns in the two-levels orthogonal array. For example, assume that we have two sets of solutions with 7 parameters in our design and assume that we want to find the optimal combination of their values. Then, the $L_8(2^7)$ orthogonal array is shown in Table 1.

The elements in the orthogonal array shown in Table 1 indicate which parameter value should be taken into the experiment on a specific considered factor. The values "0" and "1" in the elements of the orthogonal array shown in Table 1 indicate which factor's value should be used in a run of the experiment, where the value "0" in Table 1 means that the factor's value should be taken from the first set of the solution, and the value "1" in Table 1 means that the factor's value should be taken from the other set of the solution. Therefore, from Table 1, we can see that the 5th run of the experiment takes the values of the factors A, B, E and F from the first solution set and takes the values of the factors C, D and G from the second solution set to compose the run of the experiment. The orthogonal array provided by the Taguchi method (2000) can reduce the runs of the experiment efficiently. The case given in Table 1 requires 2^7runs of the experiment to explore the full combination, whereas the Taguchi method in this example reduces the experiment to 8 runs only.

4. The proposed enhanced parallel cat swarm optimization (EPCSO) method

In Tsai et al. (2008), we have proposed the PCSO method by organizing the artificial agents into a predefined number of groups. When the agent moves in the process of the parallel tracing mode, it collects the local best information found so far by its own group and use the information to update its velocity. Although it presents higher accuracies and faster convergence than the CSO method (Chu et al., 2006) under the conditions of a few number of artificial agents and a short limited computational time, its computational time could be further improved in case that the number of artificial agents increases due to the fact that the number of artificial agents moving in the process of parallel seeking mode is much more than those move in the tracing mode in the original design. In this paper, in order to overcome the drawbacks of the CSO method (2006) and the PCSO method (2008), we adopt the Taguchi method's orthogonal array into the process of the parallel tracing mode process to propose the enhanced parallel cat swarm optimization (EPCSO) method.

The orthogonal array in the Taguchi method is quite useful for reducing the run of the experiment. When the artificial agent moves in the new designed parallel tracing mode, it accesses not only the local near best solution in its own group, but also utilizes the information comes from the global near best solution overall groups. The operation of the enhanced parallel tracing mode is described as follows:

Step 1: Generate two sets of candidate velocities, which are denoted by the symbols $cv_{1,d}$ and $cv_{2,d}$, respectively, where

$$cv_{1,d}(t) = v_{k,d}(t-1) + r_1 \cdot c_1 \cdot [x_{g_{best},d}(t-1) - x_{k,d}(t-1)], \quad \text{where } d = 1,2,\ldots,M, \tag{8}$$

$$cv_{2,d}(t) = v_{k,d}(t-1) + r_1 \cdot c_1 \cdot [x_{l_{best},d}(t-1) - x_{k,d}(t-1)], \quad \text{where } d = 1,2,\ldots,M, \tag{9}$$

Table 1
The $L_8(2^7)$ orthogonal array.

Experiment number	Considered factors						
	A	*B*	*C*	*D*	*E*	*F*	*G*
1	0	0	1	0	1	1	0
2	1	0	0	0	0	1	1
3	0	1	0	0	1	0	1
4	1	1	1	0	0	0	0
5	0	0	1	1	0	0	1
6	1	0	0	1	1	0	0
7	0	1	0	1	0	1	0
8	1	1	1	1	1	1	1

M denotes the dimension of the solution space, $x_{g_{best}}$ denotes the global near best solution found so far, $x_{l_{best}}$ denotes the local near best solution of the group, l represents the group that x_k belongs to, r_1 is a random value in the range of $[0,1]$, and c_1 is a constant. Use the candidate velocities and the Taguchi orthogonal array to create a series of velocity sets, shown as follows:

$$v_{set_s,d}(t) = \begin{cases} cv_{1,d}(t), & \text{if the element in the orthogonal array is "0"}, \\ cv_{2,d}(t), & \text{otherwise}, \end{cases} \tag{10}$$

where $d = 1,2,\ldots,M$, M denotes the dimension of the solution space, and s is the index of the velocity set.

Step 2: Take one velocity set to update the original velocity $v_{k,d}(t)$ of the artificial agent each time, shown as follows:

$$v_{k,d}(t) = \begin{cases} v_{\max}, \text{if } [v_{k,d}(t-1) + v_{set_s,d}(t)] \\ \text{exceeds the maximum velocity}, \\ v_{k,d}(t-1) + v_{set_s,d}(t), \text{otherwise}, \end{cases} \tag{11}$$

update the position $x_{k,d}(t)$ of the artificial agent, shown as follows:

$$x_{k,d}(t) = x_{k,d}(t-1) + v_{k,d}(t) \tag{12}$$

and calculate its fitness value for later use. Accumulate the fitness values contributed by the column factors and take the most adaptive factors to compose the latest velocity.

Step 3: Move the artificial agent with the latest velocity by Eq. (12) to update its position.

Let us consider the orthogonal array shown in Table 1. Assume that the fitness function is shown as follows:

$$f(X) = \sum_{d=1}^{M} \frac{1}{x_d} \tag{13}$$

and assume that the goal is to minimize the fitness value. The current position X of the artificial agent and the velocity of the artificial agent in the previous run are assumed to be the values shown in Table 2. First, the candidate velocity sets $cv_{1,d}$ and $cv_{2,d}$ are generated, as shown in Table 3. Therefore, eight solution sets will be created and the dimension M of the solution space for this example is 7. Then, the eight solution sets are composed with the elements of $cv_{1,d}$ and $cv_{2,d}$ based on the values shown in Table 1. The experimental results and the accumulated fitness values are shown in Table 4. The final velocity of the agent in this run of the experiment is shown in Table 5.

The proposed EPCSO method has the same operation as the CSO method (Chu et al., 2006) for the process of the seeking mode. The flowchart of the proposed EPCSO method is shown in Fig. 1. The PCSO method (Tsai et al., 2008) only limits the artificial agents, who move in the parallel tracing mode, to access the local near best solution of which group they belong. Although this constrain divides the solution space into several parallel spaces, the efficiency of improving the accuracy and the searching speed of the PCSO method is not promoted. In this paper, in order to improve the efficiency of the PCSO method, we further adopt the Taguchi orthogonal array into process of the parallel tracing mode of the proposed EPCSO method. The enhanced parallel tracing mode utilizes the orthogonal array to run the test samples

Table 2
The coordinate and the original velocity of the artificial agent.

Current position	Considered factors						
	A	B	C	D	E	F	G
X	0	0	0	0	0	0	0
$v_{k,d}(t-1)$	1	1	1	1	1	1	1

Table 3
The candidate velocity sets.

Candidate velocity	Considered factors						
	A	B	C	D	E	F	G
cv_1	2	1	3	2	1	0	1
cv_2	3	3	1	0	2	1	2

constructed by the updated velocities with the distances between the local near best solution to the artificial agent and the global near best solution to the artificial agent. It makes the enhanced parallel tracing mode returns the searching result more stable and the searching efficiency is improved.

In the proposed EPCSO method, the number G of groups is defined as follows:

$$G = 2^n | n \in N^0 \tag{14}$$

and the total population size N is defined as follows:

$$N = 2^m \times G | m \in N^0, \tag{15}$$

where N^0 is a set of natural number including zero.

In Fig. 1, MR is a user defined value in the range of $[0,1]$. If MR is equal to 0, then it implies that all the cats will move into the seeking mode; if it is equal to 1, then it implies that all the cats will move into the tracing mode. In our experiment, MR is set to 0.1 to let 10% cats to move into the tracing mode and to let 90% cats to move into the seeking mode.

5. Experimental results

In order to evaluate the accuracy of the proposed EPCSO method, a series of experiments are taken with five familiar benchmark functions, shown as follows:

$$f_1(X) = \sum_{d=1}^{D} x_d^2 \Big| \text{Initial Range} : [-100,\ 100]^D, \tag{16}$$

$$f_2(X) = \sum_{d=1}^{D} \frac{x_d^2}{4000} - \prod_{d=1}^{D} \cos\left(\frac{x_d}{\sqrt{d}}\right) + 1 \Big| \text{Initial Range} : [0,\ 600]^D, \tag{17}$$

$$f_3(X) = \left\{ \begin{matrix} -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{d=1}^{D} x_d^2}\right) - \exp\left(\frac{1}{D}\sum_{d=1}^{D}\cos(2\pi x_d)\right) \\ +20 + \exp(1) \end{matrix} \right\} \Big| \text{Initial Range} : [-32,\ 32]^D, \tag{18}$$

$$f_4(X) = \sum_{d=1}^{D} \left(x_d^2 - 10\cos(2\pi x_d) + 10\right) \Big| \text{Initial Range} : [-5,\ 5]^D, \tag{19}$$

$$f_5(X) = \sum_{d=1}^{D} \left(\sum_{b=1}^{d} x_b\right)^2 \Big| \text{Initial Range} : [-100,\ 100]^D. \tag{20}$$

All the benchmark functions are evaluated according to the conditions shown in Chen and Li (2007) with 2000 iterations and repeated 25 runs. The number of dimensions of the solution space are set to 30 and 100, respectively, for evaluating the performance in different dimensional situations. Because the parallelism structure is constructed by 4 parallel groups, the size of the swarm for the proposed EPCSO method and the PCSO method (Tsai et al., 2008) are set to 16. The parameters setting for PSO-type algorithms can refer to (Chen & Li, 2007) and the parameters setting of the proposed EPCSO method and the PCSO method (Tsai et al., 2008) are shown in Table 6.

Table 4
The accumulated fitness values.

Experiment number	Updated coordinates of the artificial agent							Fitness value
	A	*B*	*C*	*D*	*E*	*F*	*G*	
1	0 + 1 + 2	0 + 1 + 1	0 + 1 + 1	0 + 1 + 2	0 + 1 + 2	0 + 1 + 1	0 + 1 + 1	3.00
2	0 + 1 + 3	0 + 1 + 1	0 + 1 + 3	0 + 1 + 2	0 + 1 + 1	0 + 1 + 1	0 + 1 + 2	2.67
3	0 + 1 + 2	0 + 1 + 3	0 + 1 + 3	0 + 1 + 2	0 + 1 + 2	0 + 1 + 0	0 + 1 + 2	2.83
4	0 + 1 + 3	0 + 1 + 3	0 + 1 + 1	0 + 1 + 2	0 + 1 + 1	0 + 1 + 0	0 + 1 + 1	3.33
5	0 + 1 + 2	0 + 1 + 1	0 + 1 + 1	0 + 1 + 0	0 + 1 + 1	0 + 1 + 0	0 + 1 + 2	4.17
6	0 + 1 + 3	0 + 1 + 1	0 + 1 + 3	0 + 1 + 0	0 + 1 + 2	0 + 1 + 0	0 + 1 + 1	3.83
7	0 + 1 + 2	0 + 1 + 3	0 + 1 + 3	0 + 1 + 0	0 + 1 + 1	0 + 1 + 1	0 + 1 + 1	3.33
8	0 + 1 + 3	0 + 1 + 3	0 + 1 + 1	0 + 1 + 0	0 + 1 + 2	0 + 1 + 1	0 + 1 + 2	3.17
Accumulated fitness value of cv_1	13.33	13.67	12.66	11.83	13.50	14.16	13.49	
Accumulated fitness value of cv_2	13.00	12.66	13.67	14.50	12.83	12.17	12.84	
The chosen candidate	cv_2	cv_2	cv_1	cv_1	cv_2	cv_2	cv_2	

Table 5
The final velocity of the agent.

Candidate velocity	Considered factors						
	A	*B*	*C*	*D*	*E*	*F*	*G*
$v_{k,d}(t)$	3	3	3	2	2	1	2

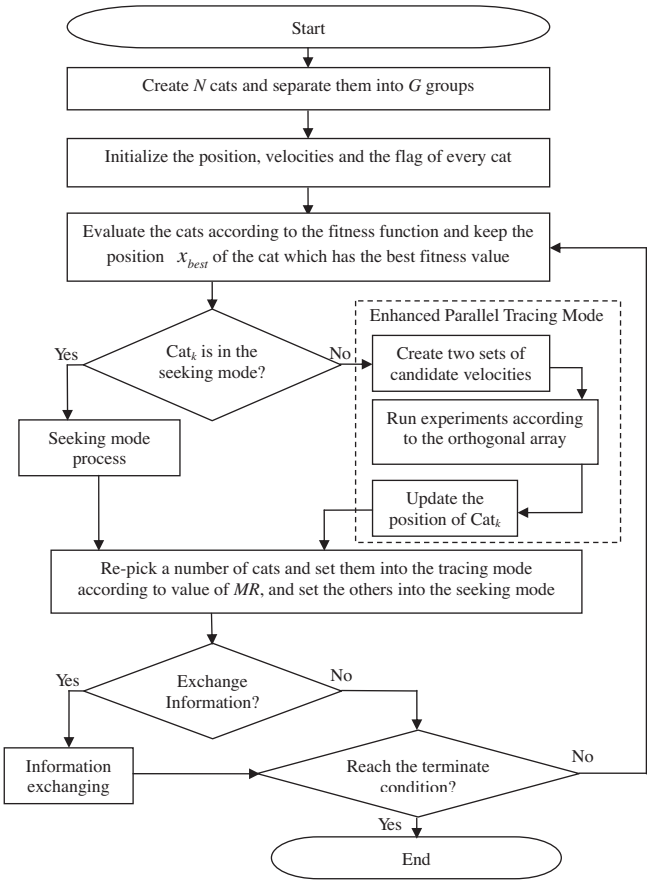

Fig. 1. The flowchart of the proposed EPCSO method.

The experimental results of the proposed EPCSO method are compared with that of the PCSO method (Tsai et al., 2008) and the ones of PSO-type optimization algorithms (Chen & Li, 2007), i.e., the typical PSO method with linearly decreasing inertial weight (the PSO-LDIW method), the PSO method with controllable random exploration velocity (the PSO-CREV method) for $\xi = 0$ and $\xi \neq 0$, the continuous-time model of PSO (the GCPSO method), the PSO method with a "mutation" and time-varying acceleration coefficient (the MPSO-TVAC method), the CPSO-H_K method and the PSO-DVM method, as shown from Tables 7–11, where the objective is to minimize the values of the benchmark functions. The experiment is run on an Intel Core 2 2.4G CPU with the GCC compiler for C with a Freebsd 7.1-Release operating system. A comparison of the experimental results of the proposed EPCSO method and the other methods is shown in Fig. 2.

The average fitness values are all accumulated and are shown in Fig. 2. The objective of the optimization is to minimize the outcome of the test functions, where the lower outcome represents better result. From Fig. 2, we can see that the outcomes of the proposed EPCSO method and the PCSO method (Tsai et al., 2008) are comparatively too small to be plotted in Fig. 2 since they are more than 5 times better than the one obtained by the PSO-H_6 method (Chen & Li, 2007). Moreover, the computational speed of the proposed EPCSO method is 16.5% faster than the PCSO method (Tsai et al., 2008). In summary, the proposed EPCSO gets higher accuracy than the other methods based on the PSO method (Chen & Li, 2007) to find the near best solution and the computation time of the proposed EPCSO method is less than the PCSO method (Tsai et al., 2008).

6. Apply the proposed EPCSO method to solve the aircraft schedule recovery problem

In this section, we apply the proposed EPCSO method to solve the aircraft schedule recovery problem and compare the experimental results of the proposed method with the HTGA method (Liu et al., 2009). The aircraft schedule recovery is necessary when the established aircraft schedule has to be changed due to some inevitable reasons, e.g., the weather or mechanical problems. It can be dealt with by swapping aircraft, delaying the flights, dispatching other aircrafts to support or taking other actions. However, considering the practical conditions and the ideal situation, the trade-off between delaying the flights, swapping the aircrafts, minimizing the total delay time and minimizing the maximum delay lead to a NP-hard problem. The flight schedule can be described by as follows:

$$S = \{s_{i,j} | s_{i,j} =< d_{i,j}, a_{i,j}, \hat{p}_{i,j}, \bar{p}_{i,j}, \hat{t}_{i,j}, \bar{t}_{i,j} >\}, \quad 1 \leqslant i \leqslant M, 1 \leqslant j \leqslant N, \tag{21}$$

where S denotes the flight schedule, $s_{i,j}$ denotes the assignment of the jth flight to the ith aircraft, $d_{i,j}$ denotes the duty name of $s_{i,j}$, $a_{i,j}$ denotes the aircraft number, $\hat{p}_{i,j}$ denotes the origin of $s_{i,j}$, $\bar{p}_{i,j}$ denotes the destination of $s_{i,j}$, $\hat{t}_{i,j}$ denotes the departure time from $\hat{p}_{i,j}$, $\bar{t}_{i,j}$ denotes the arrival time at airport $\bar{p}_{i,j}$, M denotes the number

Table 6
Parameter setting for the PCSO method and the proposed EPCSO method.

Methods	*SMP*	*SPC*	*SRD* (%)	*CDC* (%)	*MR* (%)	c_1
PCSO method (Tsai et al., 2008)	5	True	20	80	10	2
The proposed EPCSO method	3	True	20	80	10	2

Table 7
The experimental results of $f_1(X)$.

Number of dimensions	Methods	Population size	Average function value	Standard deviation
30	The PCSO method (Tsai et al., 2008)	16	0	0
	The PSO-LDIW method (Chen & Li, 2007)	20	1.764×10^{-4}	5.127×10^{-5}
	The PSO-CREV method ($\xi \neq 0$) (Chen & Li, 2007)	20	3.568×10^{-8}	4.622×10^{-10}
	The PSO-CREV method ($\xi = 0$) (Chen & Li, 2007)	20	1.184×10^{-1}	8.889×10^{-2}
	The GCPSO method (Chen & Li, 2007)	20	1.049×10^{-20}	3.882×10^{-21}
	The MPSO-TVAC method (Chen & Li, 2007)	20	2.112×10^{-1}	7.165×10^{-2}
	The CPSO-H_6 method (Chen & Li, 2007)	20	0	0
	The PSO-DVM method (Chen & Li, 2007)	20	1.354×10^{-2}	6.385×10^{-4}
	The proposed EPCSO method	16	0	0
100	The PCSO method (Tsai et al., 2008)	16	0	0
	The PSO-LDIW method (Chen & Li, 2007)	20	1.930×10^{3}	1.464×10^{2}
	The PSO-CREV method ($\xi \neq 0$) (Chen & Li, 2007)	20	1.272×10^{-4}	1.610×10^{-6}
	The PSO-CREV method ($\xi = 0$) (Chen & Li, 2007)	20	4.439×10^{3}	3.631×10^{2}
	The GCPSO method (Chen & Li, 2007)	20	10.0099	4.5058
	The MPSO-TVAC method (Chen & Li, 2007)	20	8.7721	5.4689
	The CPSO-H_6 method (Chen & Li, 2007)	20	6.957×10^{-30}	4.396×10^{-30}
	The PSO-DVM method (Chen & Li, 2007)	20	7.671×10^{-1}	2.511×10^{-2}
	The proposed EPCSO method	16	2.000×10^{-16}	3.000×10^{-16}

Table 8
The experimental results of $f_2(X)$.

Number of dimensions	Methods	Population size	Average function value	Standard deviation
30	The PCSO method (Tsai et al., 2008)	16	5.986×10^{-4}	2.548×10^{-3}
	The PSO-LDIW method (Chen & Li, 2007)	20	3.171×10^{-2}	1.331×10^{-2}
	The PSO-CREV method ($\xi \neq 0$) (Chen & Li, 2007)	20	3.087×10^{-3}	8.008×10^{-4}
	The PSO-CREV method ($\xi = 0$) (Chen & Li, 2007)	20	3.288×10^{-2}	5.504×10^{-3}
	The GCPSO method (Chen & Li, 2007)	20	1.053×10^{-2}	1.832×10^{-3}
	The MPSO-TVAC method (Chen & Li, 2007)	20	4.482×10^{-2}	1.186×10^{-2}
	The CPSO-H_6method (Chen & Li, 2007)	20	1.545×10^{-2}	2.843×10^{-3}
	The PSO-DVM method (Chen & Li, 2007)	20	3.156×10^{-2}	1.594×10^{-2}
	The proposed EPCSO method	16	3.515×10^{-3}	7.193×10^{-3}
100	The PCSO method (Tsai et al., 2008)	16	4.090×10^{-3}	8.916×10^{-3}
	The PSO-LDIW method (Chen & Li, 2007)	20	5.5343	2.759×10^{-1}
	The PSO-CREV method ($\xi \neq 0$) (Chen & Li, 2007)	20	9.000×10^{-4}	3.463×10^{-5}
	The PSO-CREV method ($\xi = 0$) (Chen & Li, 2007)	20	12.7656	9.140×10^{-1}
	The GCPSO method (Chen & Li, 2007)	20	6.541×10^{-1}	6.927×10^{-2}
	The MPSO-TVAC method (Chen & Li, 2007)	20	2.967×10^{-1}	4.796×10^{-2}
	The CPSO-H_6 method (Chen & Li, 2007)	20	3.844×10^{-3}	1.098×10^{-3}
	The PSO-DVM method (Chen & Li, 2007)	20	2.404×10^{-1}	7.072×10^{-3}
	The proposed EPCSO method	16	1.940×10^{-3}	5.369×10^{-3}

of aircrafts, and N denotes the number of flights assigned to the aircraft.

The recovered flight schedule is denoted by S' with the same definition for all the elements. The constraints and the objectives of the flight schedule recovery are defined by five formulas, shown as follows (Liu et al., 2009):

(1) The total flight delay time is defined as follows:

$$\phi_1(S,S') = \sum_{i=1}^{M}\sum_{j=1}^{N} x_{i,j}, \quad \text{where} \quad x_{i,j} = \begin{cases} 0, & \text{if } \hat{t}'_{i,j} \leqslant \hat{t}_{i,j}, \\ \hat{t}'_{i,j} - \hat{t}_{i,j}, & \text{otherwise}, \end{cases} \tag{22}$$

(2) The flight duty swap is defined as follows:

$$\phi_2(S,S') = \sum_{i=1}^{M}\sum_{j=1}^{N} x_{i,j},$$
$$\text{where } x_{i,j} = \begin{cases} 1, & \text{if } [(j=1) \wedge (a'_{i,j} \neq i)] \vee \left[(j \neq 1) \wedge \left(a'_{i,j-1} \neq a'_{i,j}\right)\right], \\ 0, & \text{otherwise}. \end{cases} \tag{23}$$

(3) The delayed time variance is shown as follows:

$$\phi_3(S,S') = \tfrac{1}{M\times N}\sum_{i=1}^{M}\sum_{j=1}^{N}(x_{i,j} - \omega)^2,$$
$$\text{where } \omega = \tfrac{1}{M\times N}\sum_{i=1}^{M}\sum_{j=1}^{N} x_{i,j} \text{ and } x_{i,j} = \begin{cases} 0, & \text{if } \hat{t}'_{i,j} \leqslant \hat{t}_{i,j}, \\ \hat{t}'_{i,j} - \hat{t}_{i,j}, & \text{otherwise}. \end{cases} \tag{24}$$

Table 9
The experimental results of $f_3(X)$.

Number of dimensions	Methods	Population size	Average function value	Standard deviation
30	The PCSO method (Tsai et al., 2008)	16	3.600×10^{-15}	1.200×10^{-15}
	The PSO-LDIW method (Chen & Li, 2007)	20	1.547×10^{-1}	9.573×10^{-2}
	The PSO-CREV method ($\xi \neq 0$) (Chen & Li, 2007)	20	7.706×10^{-3}	5.292×10^{-5}
	The PSO-CREV method ($\xi = 0$) (Chen & Li, 2007)	20	2.2372	1.467×10^{-1}
	The GCPSO method (Chen & Li, 2007)	20	1.4912	3.776×10^{-1}
	The MPSO-TVAC method (Chen & Li, 2007)	20	1.557×10^{-2}	4.019×10^{-3}
	The CPSO-H_6 method (Chen & Li, 2007)	20	1.517×10^{-14}	6.702×10^{-16}
	The PSO-DVM method (Chen & Li, 2007)	20	2.655×10^{-2}	8.484×10^{-4}
	The proposed EPCSO method	16	6.400×10^{-15}	3.700×10^{-15}
100	The PCSO method (Tsai et al., 2008)	16	8.870×10^{-12}	2.235×10^{-11}
	The PSO-LDIW method (Chen & Li, 2007)	20	8.5619	1.802×10^{-1}
	The PSO-CREV method ($\xi \neq 0$) (Chen & Li, 2007)	20	1.092×10^{-2}	8.576×10^{-4}
	The PSO-CREV method ($\xi = 0$) (Chen & Li, 2007)	20	11.9619	2.014×10^{-1}
	The GCPSO method (Chen & Li, 2007)	20	6.7431	4.506×10^{-1}
	The MPSO-TVAC method (Chen & Li, 2007)	20	3.6326	1.778×10^{-1}
	The CPSO-H_6 method (Chen & Li, 2007)	20	4.566×10^{-1}	1.245×10^{-1}
	The PSO-DVM method (Chen & Li, 2007)	20	1.549×10^{-1}	4.491×10^{-3}
	The proposed EPCSO method	16	7.598×10^{-10}	6.765×10^{-10}

Table 10
The experimental results of $f_4(X)$.

Number of dimensions	Methods	Population size	Average function value	Standard deviation
30	The PCSO method (Tsai et al., 2008)	16	3.210×10^{0}	7.245×10^{0}
	The PSO-LDIW method (Chen & Li, 2007)	20	1.412×10^{2}	7.714×10^{0}
	The PSO-CREV method ($\xi \neq 0$) (Chen & Li, 2007)	20	5.922×10^{1}	2.687×10^{0}
	The PSO-CREV method ($\xi = 0$) (Chen & Li, 2007)	20	9.781×10^{1}	9.220×10^{0}
	The GCPSO method (Chen & Li, 2007)	20	1.038×10^{2}	8.361×10^{0}
	The MPSO-TVAC method (Chen & Li, 2007)	20	7.623×10^{1}	7.372×10^{0}
	The CPSO-H_6 method (Chen & Li, 2007)	20	1.848×10^{0}	3.876×10^{-1}
	The PSO-DVM method (Chen & Li, 2007)	20	1.304×10^{1}	7.837×10^{0}
	The proposed EPCSO method	16	8.643×10^{0}	1.012×10^{1}
100	The PCSO method (Tsai et al., 2008)	16	3.459×10^{0}	3.241×10^{0}
	The PSO-LDIW method (Chen & Li, 2007)	20	7.095×10^{2}	3.436×10^{1}
	The PSO-CREV method ($\xi \neq 0$) (Chen & Li, 2007)	20	2.631×10^{2}	1.611×10^{1}
	The PSO-CREV method ($\xi = 0$) (Chen & Li, 2007)	20	5.242×10^{2}	2.448×10^{1}
	The GCPSO method (Chen & Li, 2007)	20	4.329×10^{2}	2.716×10^{1}
	The MPSO-TVAC method (Chen & Li, 2007)	20	3.256×10^{2}	1.616×10^{1}
	The CPSO-H_6 method (Chen & Li, 2007)	20	1.282×10^{2}	3.245×10^{0}
	The PSO-DVM method (Chen & Li, 2007)	20	3.771×10^{2}	3.150×10^{1}
	The proposed EPCSO method	16	1.546×10^{1}	1.484×10^{1}

Table 11
The experimental results of $f_5(X)$.

Number of dimensions	Methods	Population size	Average function value	Standard deviation
30	The PCSO method (Tsai et al., 2008)	16	7.500×10^{-15}	3.060×10^{-14}
	The PSO-LDIW method (Chen & Li, 2007)	20	6.140×10^{1}	8.260×10^{0}
	The PSO-CREV method ($\xi \neq 0$) (Chen & Li, 2007)	20	2.067×10^{0}	3.163×10^{-1}
	The PSO-CREV method ($\xi = 0$) (Chen & Li, 2007)	20	5.848×10^{3}	6.090×10^{2}
	The GCPSO method (Chen & Li, 2007)	20	3.191×10^{1}	1.662×10^{1}
	The MPSO-TVAC method (Chen & Li, 2007)	20	3.628×10^{1}	3.949×10^{0}
	The CPSO-H_6 method (Chen & Li, 2007)	20	1.544×10^{-14}	8.474×10^{-15}
	The PSO-DVM method (Chen & Li, 2007)	20	4.930×10^{0}	3.866×10^{-1}
	The proposed EPCSO method	16	2.648×10^{-12}	8.916×10^{-12}
100	The PCSO method (Tsai et al., 2008)	16	8.672×10^{-2}	4.059×10^{-1}
	The PSO-LDIW method (Chen & Li, 2007)	N/A	N/A	N/A
	The PSO-CREV method ($\xi \neq 0$) (Chen & Li, 2007)	N/A	N/A	N/A
	The PSO-CREV method ($\xi = 0$) (Chen & Li, 2007)	N/A	N/A	N/A
	The GCPSO method (Chen & Li, 2007)	N/A	N/A	N/A
	The MPSO-TVAC method (Chen & Li, 2007)	N/A	N/A	N/A
	The CPSO-H_6 method (Chen & Li, 2007)	N/A	N/A	N/A
	The PSO-DVM method (Chen & Li, 2007)	N/A	N/A	N/A
	The proposed EPCSO method	16	9.874×10^{0}	3.733×10^{1}

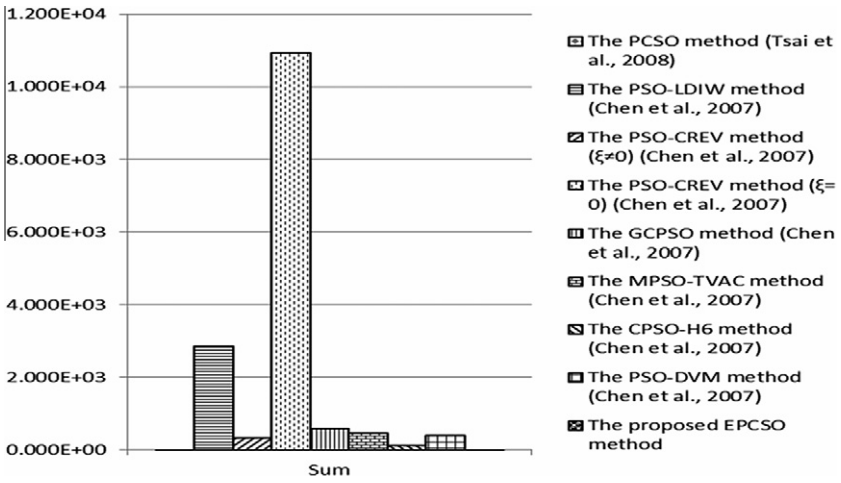

Fig. 2. A comparison of the experimental results for different methods.

Table 12
The MD-90 aircraft schedule (Liu et al., 2009).

MD90#c1											
Dep. airport	KHH	TSA	KHH	TSA	KHH	TSA	KHH	TSA	KHH	TSA	KHH
Arv. airport	TSA	KHH	TSA	KHH	TSA	KHH	TSA	KHH	TSA	KHH	TSA
Flight ID	802	803	810	811	818	817	824	823	830	833	840
Dep. time	0630	0750	0910	1050	1220	1405	1530	1700	1830	2000	2130
Arv. time	0720	0840	1000	1140	1310	1455	1620	1750	1920	2050	2220
MD90#c2											
Dep. airport	KHH	TSA	KHH	TSA	KHH	TSA	KHH	TSA	KHH	TSA	
Arv. airport	TSA	KHH	TSA	KHH	TSA	KHH	TSA	KHH	TSA	KHH	
Flight ID	804	805	812	813	820	819	826	825	832	835	
Dep. time	0720	0840	1010	1210	1330	1500	1630	1800	1930	2100	
Arv. time	0810	0930	1100	1300	1420	1550	1720	1850	2020	2150	
MD90#c3											
Dep. airport	TSA	KHH	TSA	KHH	TSA	KHH	TSA	KHH	TSA	KHH	TSA
Arv. airport	KHH	TSA	KHH	TSA	KHH	TSA	KHH	TSA	KHH	TSA	KHH
Flight ID	801	806	807	814	815	822	821	828	829	836	841
Dep. time	0650	0810	0940	1105	1300	1430	1555	1730	1900	2030	2230
Arv. time	0740	0900	1030	1155	1350	1520	1645	1820	1950	2120	2320
MD90#c4											
Dep. airport	TSA	TTT	TSA	KNH	TSA	KNH	TSA	MZG	TSA		
Arv. airport	TTT	TSA	KNH	TSA	KNH	TSA	MZG	TSA	TNN		
Flight ID	851	852	0681	0682	0683	0684	0609	0610	897		
Dep. time	0635	0750	1035	1200	1410	1535	1715	1830	2000		
Arv. time	0725	0840	1130	1255	1505	1630	1800	1915	2045		
MD90#c5											
Dep. airport	TSA	TTT	TSA	TTT	TSA	TTT	TSA	TTT	TSA	KHH	
Arv. airport	TTT	TSA	TTT	TSA	TTT	TSA	TTT	TSA	KHH	TSA	
Flight ID	853	854	855	856	857	858	861	862	831	838	
Dep. time	0755	0920	1045	1215	1340	1455	1625	1750	1930	2100	
Arv. time	0845	1010	1135	1305	1430	1545	1715	1840	2020	2150	
MD90#c6											
Dep. airport	TSA	TNN	TSA	TNN	TSA	TNN	TSA	TNN	TSA	TNN	
Arv. airport	TNN	TSA	TNN	TSA	TNN	TSA	TNN	TSA	TNN	TSA	
Flight ID	881	884	883	886	887	890	891	894	895	898	
Dep. time	0740	0900	1010	1125	1250	1430	1550	1720	1845	2005	
Arv. time	0825	0945	1055	1210	1335	1515	1635	1805	1930	2050	
MD90#c7											
Dep. airport	TNN	TSA	MZG	TSA	MZG	TSA	TNN	TSA	TNN	TSA	KHH
Arv. airport	TSA	MZG	TSA	MZG	TSA	TNN	TSA	TNN	TSA	KHH	TSA
Flight ID	882	603	602	605	606	889	892	893	896	8133	842
Dep. time	0750	0905	1020	1135	1300	1425	1600	1730	1900	2030	2200
Arv. time	0835	0950	1105	1220	1345	1510	1645	1815	1945	2120	2250

(4) The delayed flight number objective is defined as follows:

$$\phi_4(S,S') = \sum_{i=1}^{M}\sum_{j=1}^{N} x_{i,j}, \quad \text{where } x_{i,j} = \begin{cases} 0, & \text{if } \hat{t}'_{i,j} \leqslant \hat{t}_{i,j}, \\ 1, & \text{otherwise.} \end{cases} \tag{25}$$

(5) The objective of the number of long-delayed flights over 30 min is defined as follows:

$$\phi_5(S,S') = \sum_{i=1}^{M}\sum_{j=1}^{N} x_{i,j}, \quad \text{where } x_{i,j} = \begin{cases} 1, & \text{if } \left(\hat{t}'_{i,j} - \hat{t}_{i,j}\right) \geqslant 30, \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

To integrate the constraints and the objectives into one formula for the proposed EPCSO method to use as the benchmark function, we use the linear-sum to convert them into a single one. Therefore, the aircraft recovery problem can be reformulated as follows:

$$\text{Minimize } \lambda(S,S') = \sum_{x=1}^{5} \phi_x(S,S'), \quad \text{subject to } s_{i,j} \in S, s'_{i,j} \in S', \quad 1 \leqslant i \leqslant M,\ 1 \leqslant j \leqslant N \tag{27}$$

and the objective is to minimize the outcome of Eq. (27).

The MD-90 aircraft schedule which comes from the daily domestic airline plans in Taiwan is shown in Table 12 (Liu et al., 2009), where the route diagram is shown in Fig. 3 (Liu et al., 2009). In this case, seven aircrafts are assigned with totally 72 flights per day. The closure time of the "TSA" airport is set between 14:00 and 15:00. It results in 39 flight duties are influenced due to the closure and the aircraft schedule of theses flights are required to be rearranged.

The closure of the "TSA" airport is announced at 14:00. Therefore, flights 817, 0683 and 889 are not allowed to take off due to the fact that the reopen time is not predictable. Although flights 822, 858 and 890 are not currently in the closure airport, they also are not allowed to take off. In spite of flight 820 is on the air heading to the "TSA" airport, the airport is closed before it arrives. There are three options for the flight 820: The one is keep circling around the "TSA" airport and wait for the reopening to land; another one is changing its destination to land at some other airport; the last one is returning back to the departure airport. In our experiment, the flight 820 is designed to choose option one.

In our experimental design, every cat represents a recovered part of the aircraft schedule, which means that the cat carries the information of 38 influenced flights. The aircraft number correlating to the flight is taken to be the elements for coding. Thus, the 38 influenced flights provide 38 elements for coding. The flight ID, the departure airport, the arrival airport and the flight time are bound as the reference information to the location of the element represented in the cat. To decide the departure time of every flight, the flights will be sorted based on the assigned aircraft and its original departure time. The departure time of the flights to an aircraft will be assigned right after the moment when the aircraft is available to take off. Therefore, the departure time of every flight should be at least 30 min later then the arrival time of the flight in front of it. Fig. 4 shows an example of the coding result of a cat. In Fig. 4, the first row indicates the flight IDs which represent the corresponding information to the elements in the second row; the elements in the second row are the coding result of a cat. In this example, flight 817 is assigned to the aircraft MD-90#c3, flight 824 is assigned to the aircraft MD-90#c1, and so on so forth.

The initialization of the cat is generated based on the original schedule with a direct delay. Therefore, all the initial populations are feasible, but not optimal solutions. The total delay time represented by the initial solution is 2280 min and it is very poor comparing to the final result found by the proposed EPCSO method. The parameters of the proposed EPCSO method are shown in Table 13. The solution space is consisted of the aircraft numbers. The sequence of the elements is correlated with the flights, which implies the departure airport, arrival airport and the flight time. The departure time is dependent on the original departure sequence and the available take off moment of the assigned aircraft.

Eq. (27) is used as the benchmark function for the proposed EPCSO method to evaluate the recovered aircraft schedule by considering the objectives shown in Eqs. (22)–(26). Based on the coding scheme and the reasons mentioned above, the aircraft schedule recovery becomes a combinatorial problem. The movement of the proposed EPCSO method does not affect the element values directly, but influences the location indices, and the operations of both modes are slightly adjusted to fit the characteristic of the solution space. The flights shown in Table 12 are divided vertically into clusters with at most the number of elements equals to the same number of aircrafts in the movement process. Once an element is selected to be modified in the seeking mode, the suitable candidate locations are selected by Eq. (28) from its cluster, where e denotes the currently assigned aircraft of the selected element and l_i is a valid location index. When the element is moved from a location index to another, the elements permute after it in the same row will be moved accordingly. Fig. 5(a) is an example of the seeking mode movement; Fig. 5(b) is the result after the movement. Assume that the selected element to be moved is 825 (i.e.,

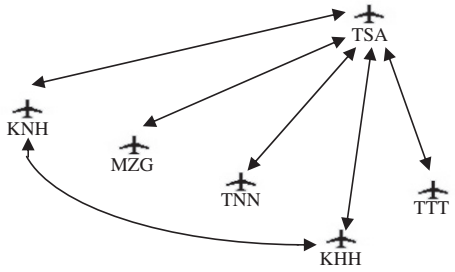

Fig. 3. The route diagram (Liu et al., 2009).

Flight ID	817	824	823	830	833	840	....	842
Assigned Aircraft	3	1	5	1	2	6		7

Fig. 4. An example of the coding result for a cat.

Table 13
Parameters of the proposed EPCSO method for the aircraft schedule recovery problem.

SMP	SPC	SRD	CDC (%)	MR (%)	c_1
3	True	1	20	10	2

817	824	823	.....	840
819	826	825	.....	ψ
822	821	828	.....	841
889	892	893	.....	842

(a)

817	824	825	.....	ψ
819	826	823	.....	840
822	821	828	.....	841
889	892	893	.....	842

(b)

Fig. 5. The operation of the movement in the proposed EPCSO method.

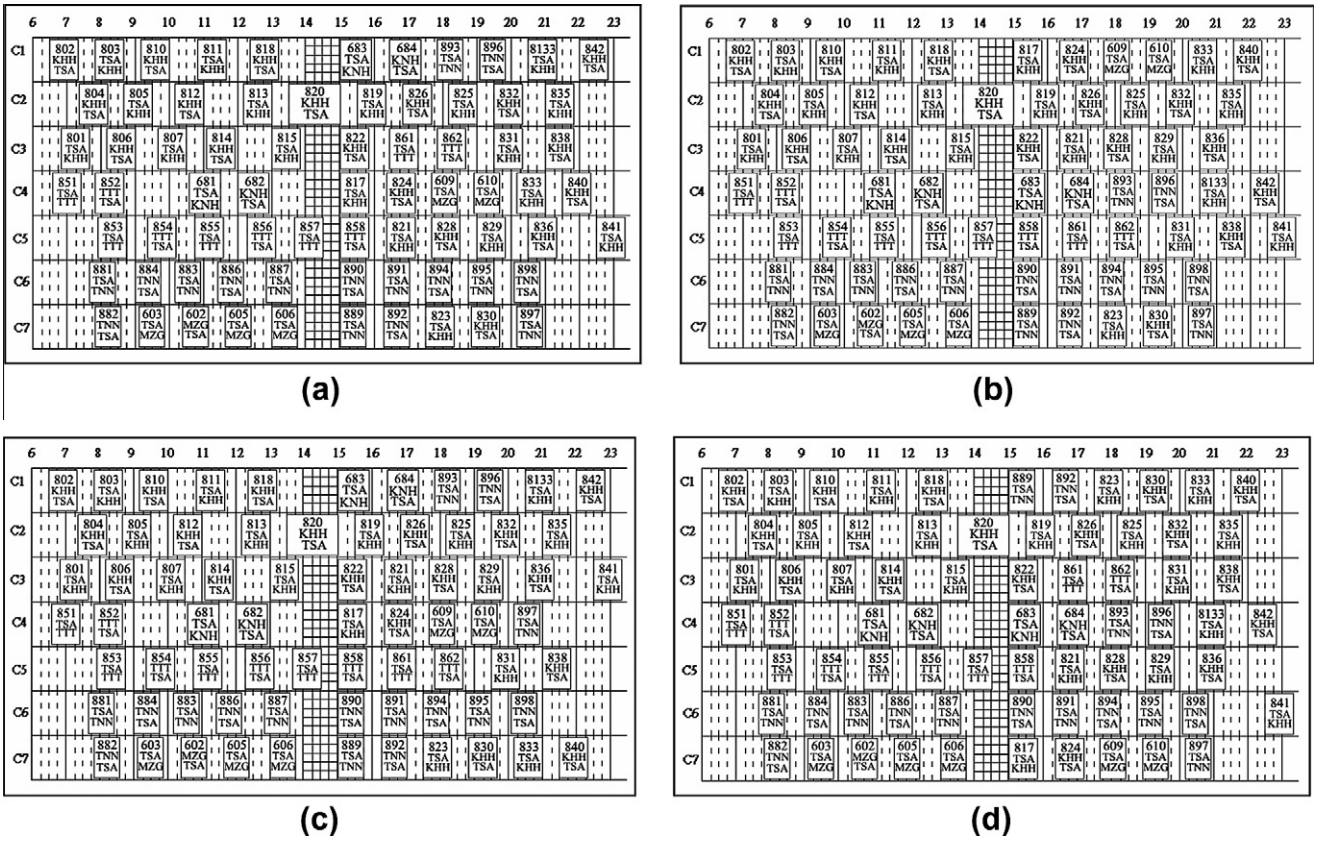

Fig. 6. The recovery schedules found by the proposed EPCSO method.

$e = 825$), the valid candidate positions, in this case, are the location of 823 and 893. After the seeking operation, the selected element and the elements locate after it are swapped with the calculated location. The tracing operation is also brought to the location index to insure every movement still retains the consistency between the destination of the former flight and the origin of the current flight.

$$L = \{l_i | \hat{p}'_{e,j} = \bar{p}'_{i,j-1}\}, \quad 1 \leqslant i \leqslant M,\ 1 \leqslant j \leqslant N,\ 1 \leqslant l_i \leqslant M. \tag{28}$$

In our experiment, the population size of the proposed EPCSO method is set to 16 and the populations are divided into four groups. The maximum iteration is set to 2000 and the experiment is repeated with 25 runs. The experiment is run on an Intel Core 2 2.4G CPU with the GCC compiler for C with a Freebsd 7.1-Release operating system. The experimental results show that the optimal recovery schedules can be found in every run by delay 24 flights with totally delay 595 min, and only 5 flights will be delayed relatively longer (more than half hour). To fully run the 2000 iterations costs less than 4.6 s and the optimum solutions can be found with 24 iterations in average. Fig. 6 shows four of the recovered schedules found by the proposed EPCSO method, where the darker zone is the disturbance period of the closure of the airport "TSA". The proposed EPCSO method gets different recovered schedules with stable fitness values at final in all 25 runs. For all 25 runs, the standard deviation of the final fitness values is 4.547×10^{-13} and the total delay time is stabilized on 595 min. Fig. 5 shows the recovery results delay 24 flights with totally 595 min delay, and 5 flights are delayed more than 30 minutes. The standard deviation of the statistical total delay time for every recovery schedule is 14.

7. Conclusions

In this paper, we have presented the enhanced parallel cat swarm optimization (EPCSO) method for solving optimization problems. In this paper, five test functions are used to evaluate the accuracy of the proposed EPCSO method. The experimental results show that the proposed EPCSO method gets higher accuracies with less computational time than the existing methods. We also have applied the proposed method to solve the aircraft schedule recovery problem. Aircraft schedule recovery contains many essentialities and should be solved within a short period of time due to the disrupted aircraft schedule implies departure time delay for many flights. The experimental results show that the proposed EPCSO method can provide the optimum recovered aircraft schedule in a very short time. The proposed EPCSO method gets the same recovery schedule having the same total delay time, the same delayed flight numbers and the same number of long delay flights as the Liu et al.'s method (2009). The optimal solutions can be found by the proposed EPCSO method in a very short time, i.e., 24 iterations in average.

Acknowledgement

The authors thank Mrs. Szu-Ping Hao, Department of Mechanical Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan, for her help during this work.

References

Abramson, D. & Abela, J. (1992). A parallel genetic algorithm for solving the school timetabling problem. In *Proceedings of 15 Australian computer science conference* (pp. 1–11), Hobart, Australia.

Chang, J. F., Chu, S. C., Roddick, J. F., & Pan, J. S. (2005). A parallel particle swarm optimization algorithm with communication strategies. *Journal of Information Science and Engineering, 21*(4), 809–818.

Chen, X., & Li, Y. M. (2007). A modified PSO structure resulting in high exploration ability with convergence guaranteed. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, 37*(5), 1271–1289.

Chu, S. C., Tsai, P. W., & Pan, J. S. (2006). Cat swarm optimization. In *Proceedings of the 9th Pacific rim international conference on artificial intelligence* (pp. 854–858), Guilin, China.

Chu, S. C., Roddick, J. F., & Pan, J. S. (2004). Ant colony system with communication strategies. *Information Sciences, 167*(1–4), 63–76.

Chu, S. C., & Tsai, P. W. (2007). Computational intelligence based on behaviors of cats. *International Journal of Innovative Computing, Information and Control, 3*(1), 163–173.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*(1), 53–66.

Elliott, R. C., & Krzymien, W. A. (2009). Downlink scheduling via genetic algorithms for multiuser single-carrier and multicarrier MIMO systems with dirty paper coding. *IEEE Transactions on Vehicular Technology, 58*(7), 3247–3262.

Janeiro, F. M., & Ramos, P. M. (2009). Impedance measurements using genetic algorithms and multiharmonic signals. *IEEE Transactions on Instrumentation and Measurement, 58*(2), 383–388.

Lin, C. J., Chen, C. H., & Lin, C. T. (2009). A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, 39*(1), 55–68.

Liu, T. K., Chen, C. H., & Chou, J. H. (2009). Optimization of short-haul aircraft schedule recovery problems using a hybrid multiobjective genetic algorithm. *Expert Systems with Applications, 37*(3), 2307–2315.

Pan, Q. K., Tasgetiren, M. F., Suganthan, P. N., & Chua, T. J. (2011). A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information Sciences, 181*(12), 2455–2468.

Pinto, C., Nägele, A., Dejori, M., Runkler, T. A., & Sousa, J. M. C. (2009). Using a local discovery ant algorithm for bayesian network structure learning. *IEEE Transactions on Evolutionary Computation, 13*(4), 767–779.

Taguchi, G., Chowdhury, S., & Taguchi, S. (2000). *Robust engineering*. New York: McGraw-Hill.

Tsai, P. W., Pan, J. S., Chen, S. M., Liao, B. Y., & Hao, S. P. (2008). Parallel cat swarm optimization. In *Proceedings of the seventh international conference on machine learning and cybernetics* (pp. 3328–3333), Kunming, China.

Tsai, J. T., Liu, T. K., & Chou, J. H. (2004). Hybrid Taguchi-genetic algorithm for global numerical optimization. *IEEE Transactions on Evolutionary Computation, 8*(4), 365–377.

Whitley, D., Rana, S., & Heckendorn, R. B. (1999). The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology, 7*(1), 109–125.

Wu, H., Geng, J., Jin, R., Qiu, J., Liu, W., Chen, J., et al. (2009). An improved comprehensive learning particle swarm optimization and its application to the semiautomatic design of antennas. *IEEE Transactions on Antennas and Propagation, 57*(10), 3018–3028.