An Improved Discrete Particle Swarm Optimization for Airline Crew Rostering Problem

Ruozhen Zheng
School of Management
Shenzhen University
Shenzhen, China
Jade__Cheng@163.com

***Abstract*—In this paper, an improved Discrete Particle Swarm Optimization (IDPSO) is presented to the Air Crew Rostering Problem for balancing crew cost, workload deviation and cooperation deviation. In IDPSO, a binary particle coding is adopted to generate initial particles. XOR-based updating rules used in updating velocity and position is to accelerate convergence rate. A selective neighbour search is employed at particles with poor performance to keep solutions qualified. Moreover, a refreshing mechanism is applied to overcoming the problem of particles trapped into local optimum and improving the diversity of the swarm. To evaluate IDPSO, computational tests have been performed， and the experiment results have proved its effectiveness.**

Keywords—Air Crew Rostering Problem, Discrete Particle Swarm Optimization, Selective Neighborhood Search, Refreshing Mechanism.

I. INTRODUCTION

Crew scheduling is essential for the airline industry for crewing cost is the second largest part of the total running costs after fuel cost [1]. The crew scheduling problem is an NP-hard optimization problem with different constraints. To reduce complexity, it is usually divided into two phases: crew pairing and crew rostering [2]. In the first problem, a sequence of flight duties is allocated to satisfy the flight requirements at the minimum cost. The crew rostering problem is the assignment of crewmen to the required pairings generated by the crew pairing problem, subject to the certain constraints. In this paper, the focus will be given to the Airline Crew Rostering Problem (ACRP). Given sets of pairings, work rules and crew costs, ACRP is solved for minimizing the cost, workload deviation and cooperation deviation.

There are two main methods to solve the ACRP: mathematical programming and heuristic algorithms approach. In the early decades, researchers used mathematical programming to get the optimal assignment. Ryan described ACRP as a set partitioning problem [3]. However, its constraints are greatly simplified so that it can only be applied to the specific condition, making this approach impractical. Gamache et al. presented an algorithm based on a column generation to assign a personalized roster that consists of pairings, days off, and other activities to airline crew members but this algorithm has a slow convergence speed [2]. Cappanera and Gallo formulated the ACRP as a 0–1 multi-commodity flow problem and some valid inequalities are proposed to tighten the linear programming formula. However, it requires a lot of computational times on large-scale issues [4].

To overcome some drawbacks of the mathematical approach, researchers turn to metaheuristic methods. For instance, Boufaied et al. adopted a cross-repairing based Genetic Algorithm to solve ACRP [5]. The Simulated Annealing technique was used to improve the initial feasible solutions when dealing with ACRP in [6]. According to these methods, relatively good solutions can be obtained within a particular running time. However, solutions can fall into local optimum, and therefore, the quality of solutions can't be guaranteed. Particle Swarm Optimization (PSO) used to be taken in [7] where researchers applied it to ACRP to assign an appropriate workload for each crew member. Nevertheless, researchers did not take enough experiments to prove the method's efficiency and premature convergence problem still existed.

In this study, we want to deal with the existing problems mentioned above and the PSO method is chosen because of its easy implementation. Since ACRP is a discrete problem, we propose an improved Discrete Particle Swarm Optimization (DPSO) to solve ACRP in discrete number space. The main contributions to this improved algorithm are listed as follows:

- A novel binary coding is adopted to better adapt to the discrete ACRP.
- Updating rules for particles' velocity and position are presented to achieve fast convergence.
- A selective neighbourhood search is used to improve solutions.
- A refreshing mechanism is proposed to get a particle away from trapped into the local optimum.

The remainder of this paper is organized as follows: The ACRP modal is described in Section 2, and Section 3 introduces PSO, DPSO and presents the improved algorithm in detail. Following Section 4 gives the experimental results. Finally, Section 5 concludes the whole paper.

II. PROBLEM DESCRIPTION

Aircrew rostering is a highly constrained scheduling problem. The constraints have different classifications in previous papers. For example, they can be divided into hard

Supported by the National Natural Science Foundation of China (71971143)

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

constraints which must be satisfied and soft constraints which should be met as many as possible to preserve the quality of the arrangement [2]. Besides, they can be classified into two categories: horizontal constraints and vertical constraints [8]. The former constraints are imposed on a crew member roster, while the latter is imposed on a set of related rosters.

In this study, it is assumed that a set of planned pairings are given in advance, which involve the rest period. Each pairing lasts one or two days and the starting time is the same every day. There are two kinds of crew members: captains and pilots and they fly on the same type of aircraft. Unexpected events or demands are not considered. The planning horizon is seven days (a week). A hard and soft constraints model is taken.

The hard constraints of this paper are as follows:

- Each crew member cannot fly more than 40 hours a week.
- On any given day, each crew member can only be assigned to one pairing.
- Two consecutive pairing duties must be avoided if the former pairing lasts over one day.
- One pairing must be given to one captain and one pilot.

The soft constraints are:

- The total working time for each crew member should be in an equitable way.
- The arrangement should meet the crews' preference toward the partner as much as possible.

A. Notations

For building the ACRP model, different notations are used and they are explained in TABLE I:

B. Mathematical Formulations

There are three objective functions in this study. The objective function (1) and (2) are derived from [11].

1) Crew cost: Equation (1) attempts to minimize the total cost of crew arrangement.

$$f1 = \min\left(\sum_{j=1}^{m}\sum_{d=1}^{sd}\sum_{k=1}^{sk} P_{ijdk}T_{dk}W1_i + \sum_{i=1}^{n}\sum_{d=1}^{sd}\sum_{k=1}^{sk} P_{ijdk}T_{dk}W2_j\right) \quad (1)$$
$$\forall i \in n,\ \ \forall j \in m$$

2) Workload balance: The sum of the deviation of each crew member's actual working hours from the standard working hours is used to describe workload conditions. In (2) a penalty weight ω_1 is introduced to ensure the working time deviation as small as possible. The average deviation of a solution is as follows:

$$f2 = \min\left(\left(\frac{\sum_{i=1}^{n}(WT_i - WT')^2}{n} + \frac{\sum_{j=1}^{m}(WT_j - WT')^2}{m} \right) \times \omega_1 \right) \quad (2)$$

3) Member cooperation deviation: Cooperation between crew members and the harmonious relationship has a great impact on the satisfaction of crew members to rostering results and flight quality. To our knowledge, no published research takes the impact of crew cooperation on the solution into consideration. Therefore, in this paper, a scoring system is proposed to represent the cooperation degree between two picked partners by crew members evaluating each other. Equation (3) is used to minimize the deviation from cooperation degrees between the captain and the pilot who are assigned to a specific pairing. A penalty weight ω_2 is proposed.

$$f3 = \min\left(\sum_{i=1}^{n}\sum_{j=1}^{m}\left(P_{ijdk} \times \frac{\sqrt{(E-EA_{ij})^2 + (E-EB_{ji})^2}}{sk} \right) \times \omega_2\right) \quad (3)$$
$$\forall d \in sd, \forall k \in sk$$

TABLE I. EXPLANATION OF NOTATIONS

Notation	Explanation
n	total number of captains
m	total number of pilots
sd	total rostering period, sd=7
sk	total number of pairings in one day
i	captain index, i=1,2,…,n
j	pilot index, j=1,2,…,m
d	day index in the period, d=1,2,…,7
k	specific pairing k in total pairings on a day, k=1,2,…, sk
$W1_i$	hourly wage of the captain i
$W2_j$	hourly wage of the pilot j
EA_{ij}	evaluation about the captain i to the pilot j, EA_i between 1~10
EB_{ji}	evaluation about the pilot j to the captain i, EB_{ji} between 1~10
T_{dk}	working time of crew members when they are assigned to the pairing k on the day d
FT_{dk}	flight time of crew members when they are assigned to the pairing k on the day d
WT_i	total working time of the captain i
WT_j	total working time of the pilot j
WT'	average working time of all members
P_{ijdk}	$P_{ijdk} = \begin{cases} 1, & \textit{the captain } i \textit{ and the pilot } j \textit{ are assigned to the paring } k \textit{ on the day } d \\ 0, & \textit{otherwise} \end{cases}$

Subject to:

Equation (4) imposes a limit that the flight time of the crew member for one week cannot exceed 40 hours.

$$\sum_{d=1}^{sd}\sum_{k=1}^{sk}(P_{ijdk} \times FT_{dk}) \leq 40, \forall i \in n, \forall j \in m \quad (4)$$

Equation (5) and (6) ensure that only one pairing is assigned for each crew member on each day.

$$\sum_{j=1}^{m}\sum_{k=1}^{sk} P_{ijdk} = 1, \forall i \in n, \quad \forall d \in sd \tag{5}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{sk} P_{ijdk} = 1, \quad \forall j \in m, \quad \forall d \in sd \tag{6}$$

Equation (7) avoids two consecutive pairing duties when former pairing consumes over 24 hours (a day).

$$\sum_{i=1}^{n}\sum_{j=1}^{m} P_{ijdk} - \sum_{i=1}^{n}\sum_{j=1}^{m} P_{ij(d+1)k} \leqslant 1 \quad if \ FT_{dk} \geqslant 24, \forall d \in sd, \forall k \in sk \tag{7}$$

Equation (8) is proposed to make one pairing given to one captain and one pilot.

$$\sum_{i=1}^{n}\sum_{j=1}^{m} P_{ijdk} = 1, \quad \forall d \in sd, \forall k \in sk \tag{8}$$

III. Related works and The Improved Algorithm

A. Particle Swarm Optimization

Standard Particle Swarm Optimization (PSO) is proposed by Kennedy and Eberhart based on imitations of natural animals' collaborating and swarming phenomena [9]. Taking foraging behaviours of birds, for example, each bird in the swarm is flying within the space to search for food. Similarly, each particle with an initial stochastic setting $X_i = (x_{i1}, x_{i2}, x_{i3}, ..., x_{in})$ is a solution to the problem to be solved and its velocity $V_i = (v_{i1}, v_{i2}, v_{i3}, ..., v_{in})$ is randomly initialized in the search space. Particles update their position and velocities, according to (9) and (10).

$$V_{id} = \omega V_{id} + c_1 rand(p_{best} - X_{id}) + c_2 rand(g_{best} - X_{id}) \tag{9}$$

$$X_{id} = X_{id} + V_{id} \tag{10}$$

ω is the inertia weight, regulating new velocities based on older ones. *rand* refers to a random number, ranging from 0 to 1. c_1 and c_2, the learning factors, determine the ratios a particle learned from its personal and swarm's global position. p_{best} and g_{best} are respective individual and global best position.

B. Discrete Particle Swarm Optimization

Based on PSO, Kennedy and Eberhart proposed Discrete Particle Swarm Optimization (DPSO) in 1997, which is operating in discrete space [10]. The concept of the algorithm remained unchanged, but in DPSO, particles search the solution in a binary space. p_{best}, g_{best} and X_i are integer in [0,1]. V_{id} refers to the probability of X_{id} taking the value 1. $s(V_{id})$ can be used to accomplish this modification. The following rules define the resulting change in position:

$$s(V_{id}) = 1/[1 + \exp(-V_{id})] \tag{11}$$

$$X_{id} = \begin{cases} 1, & rand < s(V_{id}) \\ 0, & otherwise \end{cases} \tag{12}$$

PSO and DPSO receive widespread attention in scientific research and engineering practice. In recent years DPSO has been widely investigated in the field of network problems. With the structural features of the network, researchers usually adopt topology-based local searches to speed up exploitation [12]. And according to the characteristics of ACRP, we proposed an improved DPSO algorithm abbreviated as IDPSO.

C. Improved Discrete Particle Swarm Optimization Algorithm

In order to apply the IDPSO, the problem solution is represented as one particle, and in this section, the coding of particles is redefined. Three main components of the proposed algorithm are described in detail, respectively. Pseudo codes for IDPSO are shown in TABLE II.

TABLE II. Pseudocode for IDPSO

Begin:	
1	Initial position and velocity of each particle
2	For (iteration)
3	For (particle)
4	Compute the current fitness value
5	Compute the personal and global best position and store them
6	Update velocities and positions of the particles by (15) and (16)
7	End (for particle)
8	Selective neighborhood search
9	Refresh the swarm
10	End (for iteration)

1) Particle Coding Strategy: ACRP is a discrete problem so it may be challenging to convert the rostering arrangement of ACRP to a particle vector. In that case, we adopt a novel particle coding strategy to complete swarm initialization and reduce the complexity of the solution representation.

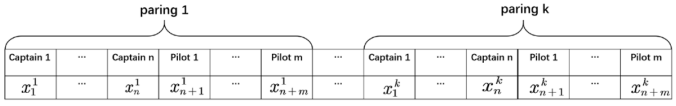

Fig. 1. Binary particle coding

As shown in Fig.1, each particle has (n+m)*k fix-length binary bits (n refers to the number of captains and m refers to the number of pilots. k is the total number of pairings given in advance). And it can be divided into different groups of captains and pilots. The value of each dimension x_i^k is a binary decision variable. When a pairing k is assigned to crew member i, x_i^k=1; otherwise it is 0. With this setting, constraint (8) is

ensured. The initial population is generated with this coding modal and it must be subject to formula (13) and (14) in order to satisfy the hard constraint (5) and (6). For instance, in Fig.2, we can see that pairing 1 is assigned to captain 1 and pilot m because $x_1^1=1, x_{n+m}^1=1$ while other decision variables are equal to 0 in pairing 1. And a penalty function is adopted to satisfy remaining constraints.

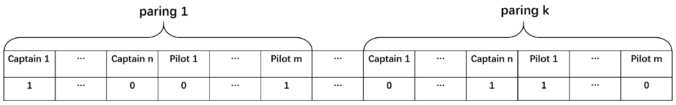

Fig. 2. An example of the binary particle coding

$$\sum_{i=1}^{n} x_i^{k'}=1, \quad \forall k' \in k \tag{13}$$

$$\sum_{i=1}^{m} x_{n+i}^{k'}=1, \quad \forall k' \in k \tag{14}$$

2) Updating Rules: Updating velocity and position are the most significant procedures in PSO. Since in the original DPSO, the velocity of each dimension determines the probability that one bit of position takes value one or zero, the position of each particle is dynamic and ephemeral. ACRP is highly constrained so that it is difficult to generate particles satisfying all the constraints in the original DPSO and the efficiency will be significantly diminished. In that case, velocity and position updating rules are developed in IDPSO. They are updated by the following equations:

$$v_{id}(t+1)=(p_{id}(t)\oplus x_{id}(t))+(g_{id}(t)\oplus x_{id}(t)) \tag{15}$$

$$x_{id}(t+1)=\begin{cases}|x_{id}(t)-1|, & v_{id}(t+1)=2\\ 0/1, & v_{id}(t+1)=1\\ x_{id}(t), & v_{id}(t+1)=0\end{cases} \tag{16}$$

$\oplus$ refers to exclusive OR operation. v_{id} is the velocity of d^{th} bit in particle i. p_i is particle i's best previous position. g_i is the global best previous position. x_i refers to particle i's current position.

The velocity of the given particle is based on the disparity with the global best particle and personal best particle. When it comes to (16), it is clear that the value of velocity determines the position. If x_{id} differs from both p_{id} and g_{id}, v_{id} is equal to 2. If x_{id} is different from p_{id} (g_{id}), and is the same as g_{id} (p_{id}), v_{id} is equal to 1. While x_{id} is the same as p_{id} and g_{id}, v_{id} is equal to 0. The higher v_{id} is, the larger the disparity is. As shown in Fig.3, when v_{id} comes to 2, x_{id} gets reversed to be the same as p_{id} and g_{id}. If v_{id}=0, it indicates that x_{id}, p_{id}, g_{id} share the same value in this node and no measure is taken to change the former position node. However, if v_{id}==1, global best position and personal best position transmit different position signals. It is worth mentioning that velocity equal to 1 appears in pairs. When this happens, randomly assign one of the corresponding positions to 1, while another is assigned to 0. Fig.3 shows an example of position updating.

According to this way, on the one hand, particles can get the information from global and personal best positions to reinforce the search toward the potential optimal solution, and on the other hand, the random selection strategy makes particles diversify the solutions. The searching ability and convergence rate of the algorithm is greatly enhanced.

3) Selective Neighborhood Search: The neighbourhood search strategy is to search for a better solution in a neighbourhood, which is commonly used in solving hard optimization problems [11]. In the IDPSO, a selective neighbourhood search strategy is putting forward based on the coding of particles.

$v_i(t+1)$	2	2	0	0	0	…	1	0	1
$x_i(t)$	1	0	0	0	0	…	1	0	0
possible $x_i(t+1)$	0	1	0	0	0	…	1	0	0
possible $x_i(t+1)$	0	1	0	0	0	…	0	0	1

Fig. 3. An example of position updating

The biggest difference between the selective neighbourhood search and the original one is that the former strategy is just applied to part of particles in the swarm. It won't consume too much computational time when improving the quality of solutions. In this strategy, first, the fitness value of each particle is computed, and the particles are arranged in descending order according to the fitness values. Particles with large fitness values will be sorted out. And then the neighbourhood search is exposed to this swarm.

Since the particle coding is taken, before neighborhood search, the particle vector needs to be transformed into a matrix, as shown in Fig.4.

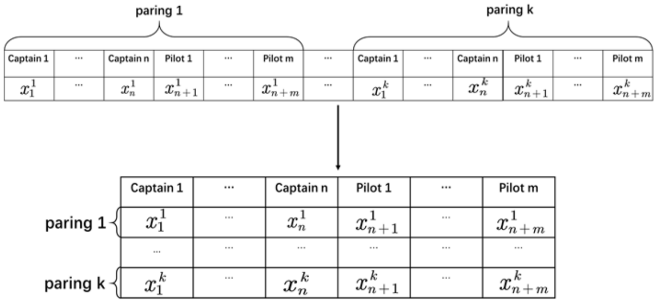

Fig. 4. Transforming a particle vector

There are many ways to generate neighbours such as "swap", "insert", "fractional insert" and so forth. "Swap" is implemented to construct a neighbourhood in this method. As shown in Fig.5, take columns randomly in the matrix and swap

their locations in the matrix. But it is worth noting that specific columns exchange themselves within corresponding captain and pilot sequences to keep the solution valid. Then restore the vector and recompute the particle value. If the new solution is better, the position will be updated by the new one.

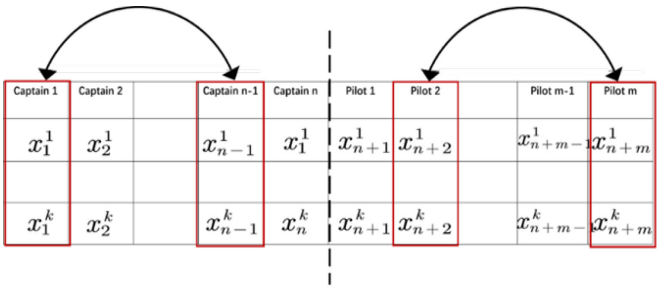

Fig. 5. "Swap" operation

4) Refreshing Mechanism: With velocity and position updating, the convergence speed is greatly improved, but the problem of particles trapped in local optimum remains. If there is a relatively good particle in the swarm, it will make the rest of particles approach it, which lowers the diversity of the swarm. Hence, a refreshing mechanism is proposed.

The refreshing mechanism is to adopt an evaluation function to evaluate particles and the evaluation function is formulated as follows: (P_i is particle i)

$$E(P_i) = fit(P_i) \times e^{-\left(\frac{1}{S(P_i)}\right)} \tag{17}$$

The evaluation function consists of two terms: one term $fit(P_i)$ is the particle's fitness value and the other $S(P_i)$ is to measure the degree of similarity between particles. Jaccard's coefficient of similarity is commonly used to measure the similarity of two binary sets. It is described as (18).

$$Jaccard(X,Y) = \frac{X \cap Y}{X \cup Y} \tag{18}$$

X and Y are two different sets.

For particles are in binary space and particle coding modal is taken, in IDPSO, the Jaccard similarity coefficient between particles P_i and P_j is redefined as (19).

$$Similarity(P_i,P_j) = \frac{M_{11}}{SUM_1} \tag{19}$$

M_{11} refers to the total number of bits where corresponding positions of two particles are set to value 1. SUM_1 is the number of required staff in one plan horizon (sum of bits valued in 1 in a particle).

According to (19), the similarity degrees between any given particles can be computed. The average similarity degree $S(P_i)$ is the final similarity evaluation in the swarm, defined as follows:

$$S(P_i) = \frac{\sum_{j=1}^{N} Similarity(P_i,P_j)}{N} \tag{20}$$

N refers to the number of particles.

Thus, it is concluded that the evaluation function is to measure the difference between particles and the fitness value of particles. In this refreshing mechanism, particles will be ranked in descending order by evaluation values. It aims to find particles with large evaluation value and half of the particles ranged in the front will be eliminated. A particle with a high similarity degree and large fitness value is more likely to be removed. And to keep the population size unchanged, these particles will be regenerated. Using this method, the swarm diversity can be improved and particle quality can be guaranteed to some extent.

IV. Experiments and Results

A. Experimental Settings

The experiments are based on distributing 12 captains and pilots to 21 pairings in 7 days, which will be executed ten times. The data is originated from [13]. The salaries of crew members are determined by reference to the standard of the Chinese air industry. It is assumed that every crew member is available during the planning horizon. And to measure the effectiveness of the novel algorithm, the other four algorithms DPSO, Genetic Algorithm (GA), Differential Evolution (DE) and Simulated Annealing (SA) are taken to compare with IDPSO on crew rostering. The average solution values will be recorded.

In this paper, we implemented IDPSO with the following parameters:

TABLE III. Parameter and its value

Parameter	Value
Iteration	750
The number of particles	50
Dimensions of each particle	12*21=252
ω_1 in (2)	10
ω_2 in (3)	1000

B. Experiment Results

In this part, simulation experiments are adopted to measure the efficiency of algorithms. TABLE IV is about the best average results of ten independent runs. Mean values reflect the overall performance of each algorithm. From TABLE IV, it is clear that the average fitness value of the proposed IDPSO is 5491.57. Thus, IDPSO excels other algorithms for solving ACRP. From the best and the worst objective results, we can see that in IDPSO, two values get close to each other, which means that our algorithm is robust and can keep the solution qualified.

According to TABLE IV, in IDPSO, the deviation of the crew member's workload (W.D) is 165.278, much smaller than others. And we can see that SA performs well in minimizing the cooperation deviation (C.D), but the C.D of crew rostering from IDPSO is the second smallest. It is concluded that

solutions found from IDPSO are feasible solutions to balance the crew cost, workload and member cooperation.

In Fig. 6, the convergence curves within 750 iterations are presented. It is shown that all curves of algorithms decrease at the beginning. Algorithm GA, IDPSO, PSO and SA seem to already converge after a few hundred iterations. When it comes to IDPSO, its value reduces dramatically within 100 iterations and the result it gets is much better than others. It shows excellent-searching ability and convergence rate. And the curve has tended to be stabilized in the following iterations.

TABLE IV. PARAMETER AND ITS VALUE

Alg.	Value			W.D	C.D
	AVG	BEST	WORST		
IDPSO	**5.49e+03**	**5.30e+03**	**5.59e+03**	**1.65e+02**	4.34e+00
DPSO	6.19e+03	6.06e+03	6.30e+03	3.09e+02	5.48e+00
DE	5.97e+03	5.86e+03	6.03e+03	2.15e+02	5.68e+00
SA	6.58e+03	6.51e+03	7.55e+03	6.86e+02	**3.43e+00**
GA	6.53e+03	6.20e+03	6.64e+03	4.04e+02	5.73e+00

Based on the curves of GA, DPSO and DE, we can assume that they are trapped into local optimum so that they converged before they obtained a good solution. As for DE, it may have the potential to search a better result in future iterations, but its convergence speed is too slow. In contrast, others have already gotten converged so that it is challenging to apply it to practical ACRP. Fig.7 is the final crew rostering schedule proposed by IDPSO.

From these results, it is confirmed that IDPSO is a relatively competitive algorithm compared with the other four algorithms with a fast convergence rate and good searching ability.

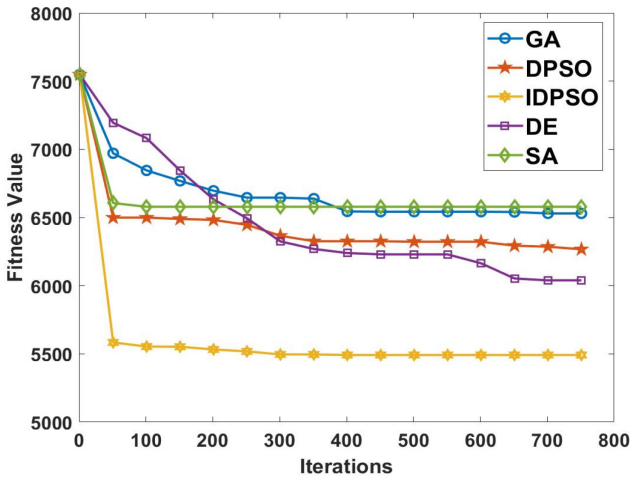

Fig. 6. Convergence curve of algorithms

V. CONCLUSION

An improved Discrete Particle Swarm Optimization algorithm (IDPSO) is developed for solving the Air Crew Rostering Problem. In the context of IDPSO, 0-1 integer coding model is taken to generate legal particles. Updating rules of velocity and position is redesigned based on XOR operation to accelerate the convergence rate. And using the selective neighbourhood search strategy to update particles with poor performance, the quality of solutions will be improved during the whole search. Furthermore, due to stagnation in classic PSO, a refreshing mechanism is added to IDPSO to enhance the diversity of the swarm and keep particles' exploration. To test IDPSO's effectiveness, IDPSO and other four well-known algorithms DPSO, GA, DE, SA, are selected and tested over a small size of instances. According to the experiment results, it is clear that IDPSO is superior to other algorithms. Therefore, it is concluded that IDPSO is an efficient algorithm in dealing with ACRP.

IDPSO is just tested on small-scale ACRP in this study. In terms of future work, we will apply the IDPSO to solve more ACRP instances and other realistic conditions will be investigated for improving the quality of the solutions. Additionally, it may be possible to employ the IPSO algorithm at solving different types of rostering problems.

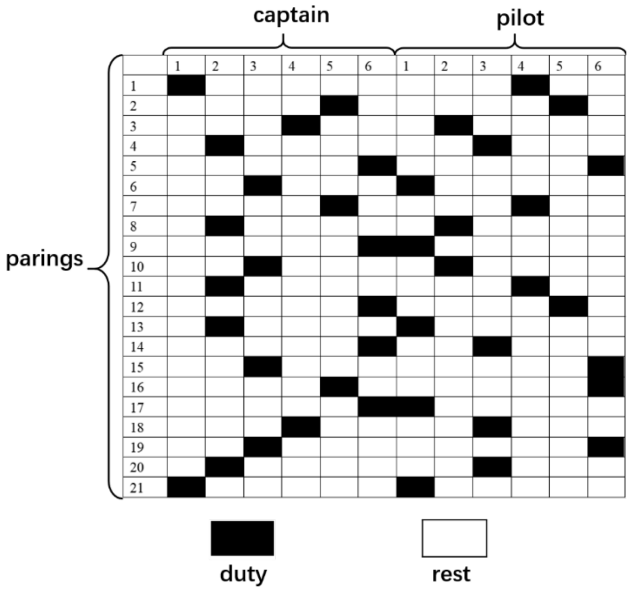

Fig. 7. Crew rostering schedule

ACKNOWLEDGMENT

The work described in this paper is supported by the National Natural Science Foundation of China (71971143).

REFERENCES

[1] Maenhout, B. and Vanhoucke, M., 2010. A hybrid scatter search heuristic for personalized crew rostering in the airline industry. *European Journal of Operational Research*, 206(1), pp.155-167.

[2] Gamache, M., Soumis, F., Marquis, G. and Desrosiers, J., 1999. A Column Generation Approach for Large-Scale Aircrew Rostering Problems. *Operations Research*, 47(2), pp.247-263.

[3] Ryan, D., 1992. The Solution of Massive Generalized Set Partitioning Problems in Aircrew Rostering. *The Journal of the Operational Research Society*, 43(5), p.459.

[4] Cappanera, P. and Gallo, G., 2004. A Multicommodity Flow Approach to the Crew Rostering Problem. *Operations Research*, 52(4), pp.583-596.

[5] Boufaied, C., Trabelsi, R., Masri, H. and Krichen, S., "A construction of rotations-based rosters with a Genetic Algorithm," *2016 IEEE Congress*

on Evolutionary Computation (CEC), Vancouver, BC, 2016, pp.2389-2394.

[6] Lučić, P. and Teodorović, D., 1999. Simulated annealing for the multi-objective aircrew rostering problem. *Transportation Research Part A: Policy and Practice*, 33(1), pp.19-45.

[7] Limlawan, V., Kasemsontitum, B. and Jeenanunta, C., "Airline crew rostering problem using particle swarm optimization," *2011 IEEE International Conference on Quality and Reliability*, Bangkok, 2011, pp.501-505.

[8] Cappanera, P. and Gallo, G., 2004. A Multicommodity Flow Approach to the Crew Rostering Problem. *Operations Research*, 52(4), pp.583-596.

[9] Kennedy, J., Eberhart, R., "Particle swarm optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, 1995, pp.1942-1948 vol.4.

[10] Kennedy, J., Eberhart, R., "A discrete binary version of the particle swarm algorithm," *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, Orlando, FL, USA, 1997, pp.4104-4108 vol.5.

[11] Souai, N. and Teghem, J., 2009. Genetic algorithm based approach for the integrated airline crew-pairing and rostering problem. *European Journal of Operational Research*, 199(3), pp.674-683.

[12] Phung, M., Quach, C., Dinh, T. and Ha, Q., 2017. Enhanced discrete particle swarm optimization path planning for UAV vision-based surface inspection. *Automation in Construction*, 81, pp.25-33.

[13] Zhang, Z., Zhou, M. and Wang, J., "Construction-Based Optimization Approaches to Airline Crew Rostering Problem," in *IEEE Transactions on Automation Science and Engineering.*