Research Journal of Applied Sciences, Engineering and Technology 5(10): 2997-3003, 2013
DOI:10.19026/rjaset.5.4613
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2013 Maxwell Scientific Publication Corp.
Submitted: September 16, 2012 Accepted: November 01, 2012 Published: March 25, 2013

**Research Article**

# A Discrete Particle Swarm Optimization Algorithm for Gate and Runway Combinatorial Optimization Problem

[1, 2]Jianli Ding and [1, 2]Yong Zhang
[1]College of Computer Science and Technology,
[2]Tianjin Key Lab for Advanced Signal Processing, Civil Aviation University of China, Tianjin 300300, China

**Abstract:** In this study, we set the average taxi time of flight as the objective of the gate and runway assignment problem. We present a gate and runway combinatorial optimization model with several restrictions such as restrictions of gate and runway time, type of aircraft and service. We design a Discrete Particle Swarm Optimization (DPSO) algorithm to solve this problem. Inspired by the genetic algorithm and combined with the neighborhood search, we propose a new location update strategy. Finally, numerical experiments were carried out on two cases where gate supplication is adequate and it's not, experimental results show that the discrete particle swarm algorithm achieved very good results.

**Keywords:** Airport operation, combinatorial optimization model, discrete particle swarm optimization, gate assignment, runway assignment

## INTRODUCTION

In recent years, with the continuous growth of air traffic, the airport's limited resources are becoming a bottleneck to limit the development of aviation industry. In face of the fierce competition and requirement to improve the quality of passenger service, a rational and efficient management and allocation of resources have become an increasing concern of the administrative departments and the airline. Gates and runways are critical resources provided to flights by civil airport and reasonable optimization of the allocation of gates and runways can reduce flight taxiing time, improve the operating efficiency of the airport, reduce airline operating costs and reduce flight delays.

Under normal circumstances, taxi route between gate and runway is fixed. Therefore, in cases not considering the sliding blockage, a specific flight can have different taxiing time among different allocations to gate and runway combinations. This can make ground control agencies, airlines and airport authorities determine the minimum gate and runway allocation scheme to minimize the average flights taxiing time. The optimal program can be determined a day before according to the published flight schedules and passenger reservation. Gate and runway combinational optimization is to determine the optimal gate and the runway combinational assignment scheme, making the flight taxiing time. Traditional gate assignment problem only considers constraints related to gate assignment to determine the flight-to-gate assignment scheme minimizing the objective function, while gates and runways combinatorial optimization should consider constraints related to both gate assignment and runway assignment to make the optimal assignment scheme that minimize the flight taxiing time.

Braaksma and Shortreed (1971) proposed one of the first attempts to use quantitative aproaches to minimize intra-terminal travel through the design of terminals. Babic *et al.* (1984) formulated a 0-1 integer programming model for the AGAP and used the branch and bound method to search the optimal assignment scheme, while not considering transfer passengers.Yan and Chang (1998) proposed a multi-commodity network flow model and used the Lagrangian relaxation method with sub-gradient and a Lagrangian heuristic function to solve the problem. Baron by the simulation analyzed the impact of different gate usage policies for passengers' walking distance (Baron, 1969). Yan and Huo (2001) proposed a bi-objective 0-1 integer programming model to assignment gates. One goal is to minimize passengers' travelling time and the second objective is to minimize passengers' waiting time, because while an aircraft in flight peak time is waiting for available seats, passengers are also waiting for its target flight. Pintea *et al.* (2008) have proposed a gates assignment method which used a hybrid ant colony local search algorithm with the objective of minimizing passengers walking

**Corresponding Author:** Yong Zhang, College of Computer Science and Technology, Civil Aviation University of China, Tianjin 300300, China

distance. Gu and Chung (1999) proposed a genetic algorithm to solve the AGAP problem where the optimization objective is to minimize delay introduced by gate reassignment. Bolat (2000) studied the robustness of gate assignment through analyzing the characteristic of uncertainties of flight delays and its difficulty to accurately estimate. Bard and Yu introduced an integral minimum cost network flow model. This model aims at reconstructing airlines schedules in response to delays by trans-forming the routing problem into a time-based network in which the overall time horizon is divided in discrete periods. The transformation is polynomial with respect to the number of airports and flights. An optimum of the new model corresponds to the optimal solution of the original problem under some slight conditions (Bard *et al.*, 2001). Xu and Bailey (2001) have designed a kind of simple heuristic tabu search algorithm to solve the problem by constructing AGAP (Airport Gate Assignment Problem) as a mixed 0-1 quadratic integer programming problem and then converting it into a 0-1 integer programming problem with a linear objective function and constraints. Ding considers the cases where the total number of flights is greater than the number available gates and makes minimum number of flights not assigned to gate and the three kinds of passengers' the minimum walking distances as the optimization objective. On the basis of neighborhood search technique in Xu and Bailey (2001), the authors proposed a new more efficient neighborhood and a tabu search method to solve AGAP (Ding *et al.*, 2004). Kim *et al.* (2010) have proposed an improved Tabu Search (TS) algorithm with the objective of minimizing the sum of the aircraft movement time and passenger movement time in terminal area. They regarded the movement time of the aircraft in terminal as a kind of movement time of the passengers. Dorndorf *et al.* (2007) gives a detailed overview of the airport gate assignment problem.

The process of arrival and departure of flights is the whole one which needs multi-party cooperation and coordination and gate assignment and runway assignment are two of the most important stages of the process. In this study, we set the average taxi time of flight as the objective to explore the gate and runway assignment problem.

## GATE AND RUNWAY COMBINATIONAL OPTIMIZATION MODEL

**Problem and symbol definition:** Given no taxiway congestion or glide path selection, a single flight's taxiing time between particular gate and runway is fixed. Therefore, a reasonable assignment of gate and runway combination can effectively reduce the average taxi time of the flight. In this study, we assign gate and runway combination to minimize the average taxi time of the flight, for easy discussion, the model assumes that the taxi time between the specific gate and runway is fixed.

Before introducing the model formulation, the notations and symbols used are listed.

**Decision variables:** $x_{ijk}$: assignment variable; 1 if the kth flight is assigned to the ith gate and the jth runway; 0 otherwise.

**Parameters and sets:**

| | | |
|---|---|---|
| $F$ | : | Set of flights need to be assigned $\{f_1, f_2, \ldots\ldots, f_p\}$ |
| $G$ | : | Set of gates available to flights $\{g_1, g_2, \ldots\ldots, g_n\}$ |
| $R$ | : | Set of runways available to flights $\{r_1, r_2, \ldots\ldots, r_m\}$ |
| $P$ | : | Set of gate and runway combinatorial pairs $\{p_1, p_2, \ldots\ldots, p_{n*m}\}$ |
| $M$ | : | A sufficiently large number |
| $t^{buff}$ | : | A buffer time between two consecutive flights assigned to the same gate |
| $t_i^{in}$ | : | Arrival time of the ith flight |
| $t_i^{out}$ | : | Departure time of the ith flight |
| $t_{ij}$ | : | Taxi time from the ith gate to jth runway |
| $b_{ij}$ | : | The time interval between flight i and flight j that are consecutive flights assigned to the same gate |
| $h_k$ | : | The passenger number of flight $k$ |

**Constraints of gate and runway:** Gate can be categorized into three types such as large one, medium one and small one according to the type of aircraft or near gate and far gate according to the location of gate, or cargo gate and exclusive gate according to the usage characteristics of gate. Airport operation monitoring department assigns the appropriate gate according to the current state of gate and aircraft type. Let flight $f_i$ be assigned to gate $g_i$, G(i) represents the set of gates available to the ith flight, here follows:

$$g_i \in G(i)$$

If the time of two flights overlaps, these two flights cannot be assigned to the same gate and runway combinatorial pair. The constraint follows:

$$(t_i^{out} - t_j^{in})(t_j^{out} - t_i^{in}) \le M(2 - x_{pqj} - x_{pqi})$$
$$i \ne j, \forall i, j \in F$$

In general case an aircraft serving a flight may be assigned to different gates and runways for arrival and departure processing and for optional intermediate parking; however, this study assigns such an aircraft to the same gate and runway combinatorial pair.

In practice, the two flights assigned to the same gate must be separated by a certain time interval, i.e., buffer time. Figure 1 shows the situation where the time interval of two consecutive flights assigned to the same gate is less than the buffer time. A buffer time between two consecutive flights that are assigned to the same

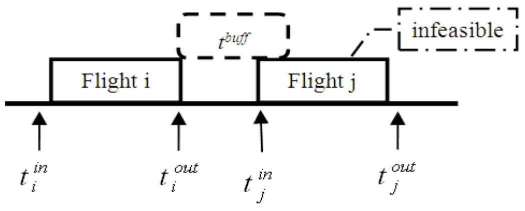

Fig. 1: Infeasible gate assignment with insufficient time interval

gate may improve the robustness of the assignment for there is some spare time for the situation where the first flight is delayed. The constraint follows:

$$b_{ij} \geq t^{buff}$$

Based on the above analysis and the mathematical definition, the objective function of gate and runway combinatorial optimization model is as follows:

$$g(f) = \min\{\alpha z_1(f) + \beta z_2(f)\}$$

The above formula expresses the objective of minimizing the linear function with weight factor $\alpha$ and $\beta$ where $\alpha \geq 0$, $\beta \geq 0$.

With respect to flight $f_k$, we get a weight by assigning this flight to gate and runway combinatorial pair $p_{ij}$. In this study, we let the taxi time as this kind of weight, i.e., $C_k = t_{ij}$. Hence, here is the definition of average taxi time of flight:

$$z_1 = \sum_{k=1}^{p}\sum_{i=1}^{n}\sum_{j=1}^{m} t_{ij} x_{ijk} h_k \Big/ \sum_{k=1}^{p} h_k$$

It's quite difficulty to assign two consecutive flights to the same gate without violate buffer time constraint when it nearly comes to the extreme of gates capacity at the peak times. Therefore, the buffer time constraint is relaxed in this study, but not the overlap constraint, so that two consecutive flights with less time interval than the buffer time can be assigned to the same gate and then we get the difference between $t^{buff}$ and $b_{ij}$ which can be used to evaluate the assignment. However, it is desirable to keep the number of this kind of assignment as low as possible. Hence, there is:

$$z_2 = \sum_{i=1}^{p} \sum_{j=2,j>i}^{p} \max\{t^{buff} - b_{ij}, 0\} * y_{ij}$$

where,

$$y_{ij} = \sum_{k=1}^{n} (\sum_{l \in R} x_{kli}) * (\sum_{l \in R} x_{klj})$$

In addition, flights to which no available gates can be allocated should be assigned to the apron. However, the apron, as a special gate, does not contain the time overlap constraint and the aircrafts' type constraint for flights assigned to the apron. We assume that the taxiing time between apron and runway is a fixed value which is significantly higher than the ones between gate and runway.

**Gate and runway combinatorial optimization model:** The gate and runway combinatorial assignment problem can be summarized in the following optimization model:

$$\min \sum_{f|f \in F} (\alpha z_1(f) + \beta z_2(f)) \tag{1}$$

subjects to:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ijk} = 1, (1 \leq k \leq p) \tag{2}$$

$$g_i \in G(i) \tag{3}$$

$$(t_i^{out} - t_j^{in})(t_j^{out} - t_i^{in}) \leq M(2 - x_{pqj} - x_{pqi}) \quad i \neq j, \forall i, j \in F \tag{4}$$

$$z_1 = \sum_{k=1}^{p}\sum_{i=1}^{n}\sum_{j=1}^{m} t_{ij} x_{ijk} h_k \Big/ \sum_{k=1}^{p} h_k \tag{5}$$

$$z_2 = \sum_{i=1}^{p} \sum_{j=2,j>i}^{p} \max\{t^{buff} - b_{ij}, 0\} * y_{ij} \tag{6}$$

where,

$$y_{ij} = \sum_{k=1}^{n} (\sum_{l \in R} x_{kli}) * (\sum_{l \in R} x_{klj})$$

## A DISCRETE PARTICLE SWARM OPTIMIZATION ALGORITHM SOLVING MODEL

Particle Swarm Optimization (PSO) Algorithm was first proposed by Eberhart and Kennedy (1995) to solve the optimization problem of continuous functions. Kennedy and Eberhart (1997) proposed a discrete binary particle swarm optimization algorithm for solving problems with discrete change of variables in the solution space. Position vector in this version is the binary vector, but velocity vector remains continuous floating-point vector. Each dimension of the velocity vector expressed the probability of the corresponding dimension values of the position vector being 1 or 0. Subsequently, there are different discrete versions of particle swarm optimization having been proposed for different problems. In this study, we proposed a discrete particle swarm algorithm for solving the gate and runway combinatorial optimization problem through analyzing of the characteristics of this problem and used a new form of the position vector and position update strategy.

**Neighborhood search method:** In this study, we use the neighborhood search method that consists of three different moves, namely the Insert Move, the Exchange Interval Move and the Apron Exchange Move. The Insert Move is basic and was given a detailed analysis in Xu and Bailey (2001), the Exchange Interval Move and the Apron Exchange Move were proposed by Ding *et al.* (2004) and a detailed analysis can be found in their work. The experimental results show that the three moves have a good performance. Here, we just give a brief description for reference:

- **Insert move:** Move a single flight to a gate other than the one it currently assigns.
- **Interval exchange move:** Exchange two flight intervals in the current assignment. A flight interval consists of one or more consecutive flights in one gate.
- **Apron exchange move:** Exchange one flight which has been assigned to the apron with a flight that is assigned to a gate currently.

**Position vector encoding:** In this study, the position vector is an *m*+1 dimensional vector, where *m* is the number of gates. Each dimension of the position vector consists of flight the set of flights allocated to the corresponding gate. The flights assigned to the same gate, in accordance with the allocation to the runway, can be divided into multiple sub-sequences. Obviously, the encoding scheme has a certain hierarchy. In continuous version and the binary version of particle swarm algorithm, the position vector of each dimension can be considered a value in the real number domain, while each dimension of the position vector in this study can be considered a value in the collection domain. Mark $\{S_1 \ldots S_i \ldots S_m, S_{m+1}\}$ for a assignment scheme, then $S_i$, the *i*-th dimension of position vector, denotes the collection of flights assigned to the *i*-th gate which, according to the allocation to the runway, can be divided into multiple sub-sets.

**Position updating strategy:** In this study, the way of a particle updating its position is as follows:

$$X_i^{t+1} = (k * \omega \otimes N(X_i^t)) \oplus (c_1 \otimes C_{a,b}(X_i^t, pB_i^t)) \oplus (c_2 \otimes C_{a,b}(X_i^t, gB^t))$$

where,

| | |
|---|---|
| $N(X^t_i)$ | : The neighborhood of particle's position $X^t_i$ which is described in detail in the first section of his chapter |
| $pB^t_i$ | : The $i^{th}$ particle's individual extreme value which represents the best value that a particle has obtained |
| $gB^t$ | : The global extreme value of the swarm which represents the best value that all the particles have obtained |
| $\omega$ | : The inertial weight and $\omega \in [0,1]$ |
| k | : The attenuation factor used to control the impaction of the inertia weight on the position update operation |
| $c_1$ | : The "cognition" factor which coordinates the pace by which a particle flies to $pB_{ti}$ and $c_1 \in [0, 1]$ |
| $c_2$ | : The "social" factor which coordinates the pace by which a particle flies to $gB^t_i$ and $c_2 \in [0, 1]$ |

The above equation consists of 3 parts where the meaning of $\oplus$ operator is to select any one operand. The detailed description of each part is the following:

The first component represents the reflection of a particle on its own velocity, that is:

$$E_i^t = k * \omega \otimes N(X_i^t) = \begin{cases} N(X_i^t) & rand() < k * \omega \\ X_i^t & else \end{cases}$$

where, $N(X^t_i)$ represents the velocity of a particle.

The second component is the cognition part of the particle representing the private thinking of the particle itself, that is:

$$S_i^t = c_1 \otimes C_{a,b}(X_i^t, pB_i^t) = \begin{cases} C_{a,b}(X_i^t, pB_i^t) & rand() < c_1 \\ X_i^t & else \end{cases}$$

The third component is the social part of the particle representing the collaboration among particles, that is:

$$X_i^{t+1} = c_2 \otimes C_{a,b}(X_i^t, gB^t) = \begin{cases} C_{a,b}(X_i^t, gB^t) & rand() < c_2 \\ X_i^t & else \end{cases}$$

For the $C_{a,b}$ (X, Y) operator in position update strategy, inspired by the crossover operator of the genetic algorithm, the output of the $C_{a,b}$ (X, Y) operation consists of the a-th and b-th dimensions from Y and the other dimensions from X. Usually, the output may contain duplicate flights or miss out some flights, so the following adjustment is needed. For easy discussion, let's denote the output of the $C_{a,b}$ (X, Y) operation as X'. First, keep the a-th and b-th dimensions of X' unchanged; then, delete repeated flights in other dimensions; finally, insert all the missed flights into the (m+1)-th dimension of X'.

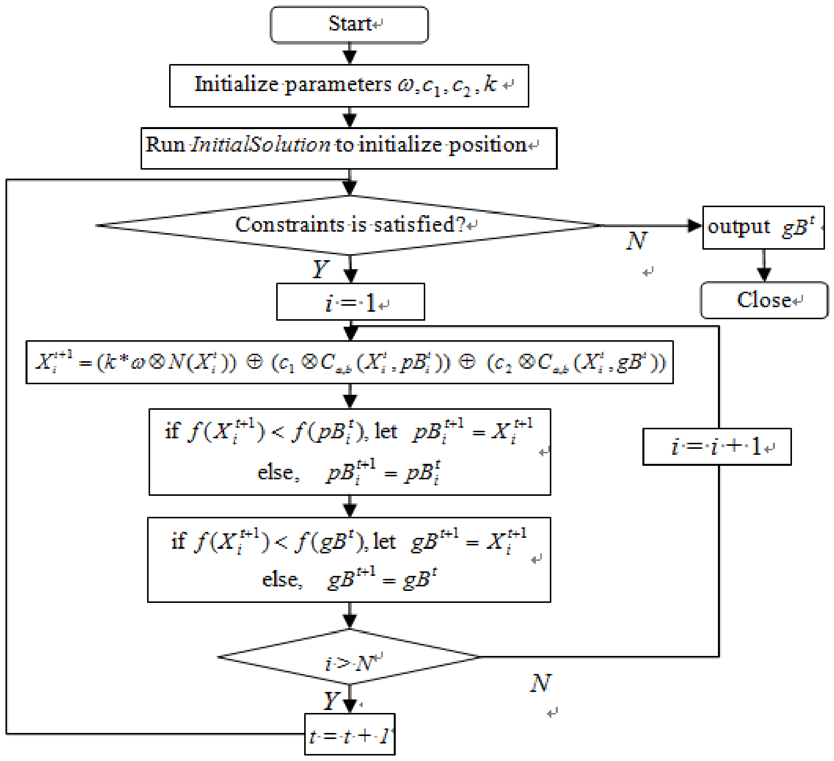
Start
Initialize parameters $\omega, c_1, c_2, k$
Run *InitialSolution* to initialize position
Constraints is satisfied?
N
output $gB^t$
Close
Y
$i = 1$
$X_i^{t+1} = (k * \omega \otimes N(X_i^t)) \oplus (c_1 \otimes C_{a,b}(X_i^t, pB_i^t)) \oplus (c_2 \otimes C_{a,b}(X_i^t, gB^t))$
if $f(X_i^{t+1}) < f(pB_i^t)$, let $pB_i^{t+1} = X_i^{t+1}$ else, $pB_i^{t+1} = pB_i^t$
$i = i + 1$
if $f(X_i^{t+1}) < f(gB^t)$, let $gB^{t+1} = X_i^{t+1}$ else, $gB^{t+1} = gB^t$
$i > N$
N
Y
$t = t + 1$

Fig. 2: Hybrid discrete particle swarm optimization algorithm flow chart

In the PSO system, $pB_i^t$ and $gB^t$ transit their own messages to the particle itself or other particles by the position update strategy, respectively, to lead the particles to the local optimum and global optimum direction of evolution. The particles are mainly concentrated in and near the area of $pB_i^t$ and $gB^t$ which drive particle swarm gradually to evolve in the direction of the optimal solution. $pB_i^t$ and $gB^t$ have a significant impact on the optimization performance of PSO; but, PSO tends to make excessive exploration of $pB_i^t$ and $gB^t$ which leads to premature convergence of the algorithm in local optimal solution. In PSO, the inertia weight controls the balance of global and local search (Shi and Eberhart, 1998). Therefore, we, by introducing the attenuation factor $k$ to adjust the inertia weight, give the particles more opportunity to explore at their own pace in the beginning of the algorithm. With the increase in the number of iterations, we gradually increase the impact of p $B_i^t$ and $pB^t$ on the particle and finally we improve the algorithm both local and global search capability.

**Initial population:** Ding proposed a greedy algorithm to minimize the number of un-gated flights and used it to produce the initial solution of tabu search algorithm which improved the efficiency of tabu search algorithm (Ding *et al*., 2004). In this study, the above greedy algorithm (initial Soloution) is used to get the initial position, the basic steps are:

```
Begin
    Sort flights by departure times $t_i^{out}$ ($1 \le i \le n$).
    Let $g_k$ ($1 \le k \le m$) represents the earliest available
    time (the departure time of last flight) of gate $k$.
    Set $g_k = -1$ for all $k$.
    For each flight $i$,
Begin
    Find gate $k$ such that $g_k < t_i^{in}$ and $g_k$ is maximized.
    If such $k$ exists, then assign flight $i$ to gate $k$, update
    $g_k = t_i^{out} + t^{buff}$.
    If $k$ does not exist, then assign flight $i$ to the apron.
  End
End
```

**A discrete particle swarm optimization algorithm:** After the above analysis, we present the discrete particle swarm optimization algorithm illustrated in Fig. 2.

The algorithm parameters are set as follows: $\omega = 0.509$, $c_1 = 0.429$, $c_2 = 0.638$, attenuation factor $k = 1/\log(10 * N)$ where $N$ is the number of iterations of the algorithm.

## COMPUTATIONAL RESULTS

In this study, the size of each test instance is expressed with $p{*}n{*}m$, where p is the number of flights, $n$ is the number of gates and $m$ is the number of runways. This study assumes that $m$ is a constant and

Table 1: Comparation result of the first data set

| | Results of ITS | | | Results of DPSO | | |
|---|---|---|---|---|---|---|
| Prob. size | Avg. (min) | Best (min) | Cpu time (sec) | Avg. (min) | Best (min) | Cpu time (sec) |
| 100*15 | 5.77 | 5.76 | 91.0 | 5.76 | 5.74 | 55.7 |
| 200*20 | 6.45 | 6.44 | 158.3 | 6.45 | 6.44 | 84.0 |
| 300*30 | 6.32 | 6.31 | 229.3 | 6.31 | 6.31 | 104.0 |
| 400*40 | 6.18 | 6.17 | 276.0 | 6.17 | 6.15 | 220.3 |
| 500*50 | 6.24 | 6.23 | 354.3 | 6.21 | 6.20 | 397.3 |
| 600*60 | 6.43 | 6.42 | 458.7 | 6.42 | 6.42 | 712.7 |
| 700*70 | 6.58 | 6.58 | 558.7 | 6.56 | 6.55 | 1649.0 |
| 800*80 | 6.88 | 6.87 | 675.7 | 6.82 | 6.82 | 3864.3 |
| 900*90 | 6.74 | 6.73 | 787.3 | 6.71 | 6.70 | 1997.3 |
| 1000*90 | 6.95 | 6.97 | 886.8 | 6.99 | 6.98 | 2547.5 |

Table 2: Comparation result of the second data set

| | Results of ITS | | | Results of DPSO | | |
|---|---|---|---|---|---|---|
| Prob. size | Avg. (min) | Best (min) | Cpu time (sec) | Avg. (min) | Best (min) | Cpu time (sec) |
| 100*9 | 6.67 | 6.53 | 174.3 | 6.60 | 6.52 | 138.0 |
| 200*16 | 7.28 | 7.25 | 167.3 | 7.06 | 7.03 | 188.0 |
| 300*25 | 7.00 | 6.97 | 223.7 | 6.99 | 6.97 | 587.3 |
| 400*30 | 7.02 | 7.02 | 300.3 | 7.01 | 6.98 | 738.0 |
| 500*36 | 7.45 | 7.44 | 388.3 | 7.47 | 7.46 | 774.3 |
| 600*45 | 7.26 | 7.23 | 463.0 | 7.25 | 7.23 | 2601.7 |
| 700*52 | 7.46 | 7.41 | 600.7 | 7.44 | 7.42 | 1170.0 |
| 800*59 | 7.46 | 7.45 | 625.0 | 7.47 | 7.46 | 2576.0 |
| 900*65 | 7.66 | 7.65 | 757.2 | 7.69 | 7.68 | 4607.6 |
| 1000*71 | 7.82 | 7.81 | 826.5 | 7.86 | 7.86 | 5925.8 |

set its value to be 3. So the size of the problem can be simplified as $p*n$. Let $f_i$ is the $i$-th flight and then we choose a random value from interval [0, 900] as the value of $t_i^{in}$ and one from interval [$t_i^{in}$ + 40, $t_i^{out}$ + 60] as the value of $t_i^{out}$. With the observation of actual airport operations about flights, we choose a random value from interval [5, 30] as the value of $t_{ij}$. The value of $t^{buff}$ is 15. Weight factors $\alpha$ and $\beta$ are set to be 1 separately.

In this study, we use the above approach to generate two sets of experimental data; each experimental data consists of 10 test cases and for each case we separately run 10 times ITS algorithm proposed in Xu and Bailey (2001) and the discrete particle swarm algorithm. The first set of experimental data does not have flights unassigned to gates, which corresponds to the situation of an adequate supply of gates. The second set of experimental data contains flights unassigned to gates, which corresponds to the situation of a short supply of gates in the peak of the flight times. Table 1 and 2 show the results of the experimental tests.

From the Table 1, we see that DPSO algorithm provides better solutions than ITS. The DPSO algorithm takes a relatively long running time. However, Even if we increased the number of iterations in the ITS method, we are not likely to get results as good as those from the DPSO algorithm. This shows that in sufficient supply of gates case, the discrete particle swarm algorithm is superior to the ITS algorithm. From the Table 2, we see that DPSO performs not better in the second data set than the first one. But the DPSO algorithm still provides better solutions than ITS, which indicates that in insufficient supply of gates case, the discrete particle swarm algorithm is still superior to the ITS algorithm.

## CONCLUSION

This study focus on the gate and runway combinatorial assignment problem and the objective is to minimize the average taxi time of the flight. We proposed a gate and runway combinatorial optimization model and design a hybrid discrete particle swarm optimization algorithm to solve this model. We use a neighborhood search method as the particle's thinking of its own speed and inspired by the crossover operator in the genetic algorithm, we use the similar operation to the crossover operator as the self-perception and social cognition part in position update strategy. Finally, numerical experiments show that, compared to the ITS algorithm, the discrete particle swarm optimization achieves better results within a reasonable period of time.

## ACKNOWLEDGMENT

This study is supported by the National Nature Science Foundation of China (60879015, 61139002) and Technology Fund of Civil Aviation (MHRD 201013).

## REFERENCES

Babic, O., D. Teodorovic and V. Tosic, 1984. Aircraft stand assignment to minimize walking. J. Transport. Eng., 110: 55-66.

Bard, J., G. Yu and M. Argüello, 2001. Optimizing aircraft routings in response to groundings and delays. HE Trans., 33: 931-947.

Baron, P., 1969. A simulation analysis of airport terminal operations. Transport. Res., 3: 481-491.

Bolat, A., 2000. Procedures for providing robust gate assignments for arriving aircrafts. Eur. J. Oper. Res., 120: 63-80.

Braaksma, J.P. and J.H.W.F. Shortreed, 1971. Improving airport gate usage with critical path method. Transport. Eng. J. ASCE, 97: 187-203.

Ding, H., A. Lim, B. Rodrigues and Y. Zhu, 2004. Aircraft and gate scheduling optimization at airports. Proceedings of the 37th Annual Hawaii International Conference on System Sciences, pp: 74-81, ISBN: 0-7695-2056-1.

Dorndorf, U., A. Drexl, Y. Nikulin and E. Pesch, 2007. Flight gate scheduling: State-of-the-art and recent developments. Omega, 35: 326-334.

Eberhart, R.C. and J. Kennedy, 1995. A new optimizer using particle swarm theory. Proceedings of the 6th International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp: 39-43.

Gu, Y. and C. Chung, 1999. Genetic algorithm approach to aircraft gate reassignment problem. ASCE J. Transport. Eng., 125: 384-389.

Kennedy, J. and R.C. Eberhart, 1997. Discrete binary version of the particle swarm algorithm. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Orlando, Florida, USA, 5: 4104-4108.

Kim, S.H., E. Feron and J.P. Clarke, 2010. Airport gate assignment that minimizes passenger flow in terminals and aircraft congestion on ramps. AIAA-2010-7693, AIAA Guidance, Navigation and Control Conference. Toronto, Ontario, Aug. 2-5.

Pintea, C.M., P.C. Pop, C. Chira and D. Dumitrescu, 2008. A hybrid ant-based system for gate assignment problem. HAIS, Proceedings of the 3rd International Workshop on Hybrid Artificial Intelligence Systems, Springer-Verlag Berlin, Heidelberg, LNAI 5271, pp: 273-280.

Shi, Y. and R.C. Eberhart, 1998. A modified particle swarm optimizer. Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage, AK, pp: 69-73.

Xu, J. and G. Bailey, 2001. The airport gate assignment problem: Mathematical model and a tabu search algorithm. Proceeding of the 34th Hawaii International Conference on System Sciences, ISBN: 0-7695-0981-9.

Yan, S. and C. Huo, 2001. Optimization of multiple objective gate assignments. Transport. Res., 35A: 413-432.

Yan, S. and C.M. Chang, 1998. A network model for gate assignment. J. Adv. Transport., 32: 176-189.