An Application of Multiobjective Genetic Algorithm to Intelligent-Based Flight Scheduling

Tung-Kuan Liu* (劉東官) ,Da-Yuan Jou** (周大源),
and Chung-Nan Lee** (李宗南)

*Department of Mechanical & Automation Engineering, National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan, R. O. C

E-Mail: tkliu@ccms.nkfust.edu.tw
TEL: (07) 601-1000 Ext. 2224

**Department of Computer Science and Engineering, National SunYat-Sen University, Kaohsiung, Taiwan, R. O. C

E-Mail: {joudy, cnlee}@mail.cse.nsysu.edu.tw
TEL: (07) 525-4335

Abstract

This paper discusses and develops a methodology to solve the problem of flight scheduling, which is proved to be a multi-objective optimization problem. A multi-objective optimization problem has many constraints and performance indices to be considered simultaneously. Unfortunately, the constraints and performances are usually conflicting. To deal with such problems, in this research we employ the Method of Inequalities (MOI) to formulate the objective functions and present them in a vector form. We also employ a multi-objective rank-based genetic approach as the optimization method. Furthermore, to overcome the conflicts of the objective functions, which consist of available ground holding time, take-off place, maintaining issues, *etc.* simultaneously, a 2-degree freedom design method is used to find feasible solutions. We have successfully applied the proposed method to solve a practical problem of flight scheduling for a domestic airline company.

***Keywords*: Method of Inequalities, genetic algorithm, multi-objective optimization, 2-degree of freedom design, flight scheduling**

1.Introduction

Flight scheduling is extremely important in resource planning in airline carriers [1]. Its main process is to generate a daily flight schedule that determines which aircraft should be assigned to which flight. When planners want to generate daily flight schedules, they should consider various constraints and performance indices, for example, available aircrafts, available airports, ground-holding time, maintenance rules, and fueling constraints, and so on. Most airline carriers may have several performance indices, such as minimizing the fleet size, minimizing the total cost, and maximizing the benefits. Since the flight scheduling will affect the total cost of airline carriers, airline carriers should make the highest benefit under the lowest cost as possible. Therefore, the best way to achieve this goal is to optimize the flight scheduling.

Practically, the flight scheduling process consists of two phases [14]: a schedule construction phase and a schedule evaluation phase. In the construction phase, the planners propose a draft flight schedule based on various limitations such as the number of aircrafts, the number of maximal flights, *etc*. In the schedule evaluation phase, the planners evaluate the draft schedule, and adjust it to match all constraints repeatedly. Also, the total cost and the minimization of fleet size should be considered.

Several models of flight scheduling have been proposed for decades. Most of these models are proposed with various objective functions and constraints [3][4][5][6][7], such as minimizing the fleet size, or maximizing total profits. Since this kind of problem has been proved to be NP-complete [8], some typical exact solution techniques [3][6][9][10][7], such as branch-and-bound method, and cutting aircraft method are often used. Lee [10] proposed a Lagrangian technique to reduce the computational complexity. Also, the Dantzig-Wolfe decomposition [4], which is a kind of linear programming solution technique, has been added to the Lagrangian multiplier technique to solve the problem of flight scheduling. Furthermore, some other models were also proposed, like integer linear programming model [12], multi-criteria model [9], mixed integer programming model [13], and minimum cost network flow models. Yan *et al.* [14] proposed a decision support framework for multi-fleet routing and multi-stop flight scheduling.

However, most of these researches use deterministic method to solve the problem of flight scheduling. These methods often assign flights to the aircrafts step by step. It is unpractical when the number of aircrafts and flights become large. Furthermore, if one cannot find out feasible solutions, one may be confused whether there exists no feasible solutions or the method is incorrect. Consequently, an effective way is extremely important to solve the problems. Therefore, we employ a heuristic method, multi-objective genetic algorithm to solve the problem of flight scheduling.

Genetic algorithm, which is proposed by John Holland in 1962 [15], exploits a novel field of multi-objective optimization. The mathematical framework was developed in the late 1960s in Holland's [16]. This kind of algorithms is stochastic optimization algorithms that are originally motivated by the mechanisms of natural selection and evolution genetics. By using Darwinian's survival-of-the-fittest strategy, genetic algorithms can prevent unfit characteristics and use random information exchange, with exploitation of knowledge contained in old trial solutions, to affect a search mechanism with surprising power and efficiency. This algorithm has been successfully used in many fields. With the characteristics of maintaining various candidate solutions at the same time, GA is very suitable for solving the multi-objective optimization problems. More details about GA may be found in Goldberg's [11].

Liu [21] proposed a Multi-objective Genetic Algorithm (MGA) to the application of control systems design. In this research, MGA is applied to solve the problem of flight scheduling. First, the Method of Inequalities [19] is used to formulate various constraints and performance indices. We also use a rank-based selection method company with the auxiliary vector performance index. Furthermore, similar to the multiple thread problems in computer science when a multi-objective problem is to be optimized, the deadlock problems may occur since some of the objectives are conflicting. In order to prevent such problem, it uses a 2-degree of freedom design method to find feasible solutions instead of traditional 1-degree of freedom design method. The method relaxes one of the objectives and optimizes the other conflicting objective alternatively.

We have applied the proposed method to solve flight-scheduling problems. The results show that it can solve the problem under the limitation of the number of aircrafts instead of the minimization of the number of aircrafts that are usually employed in traditional method. It provides the airline carriers a greater chance to obtain a neo-optimal flight schedule and help to decrease the cost of flight scheduling.

The remainder of this paper is organized as follows. Section 2 introduces the problem of flight scheduling and formulates the problem. Specific constraints are also described in this section, too. The proposed methods are presented in Section 3. Experimental results are presented in Section 4. Discussions are given in Section 5, and conclusions are given in the last section.

2. The Problem of Flight Scheduling and Mathematical Models

In this section, we first give the problem statement of flight scheduling.

Suppose that the available set of α aircraft is AC,

$$AC = \{ac_i \mid i = 1,2,...,\alpha\} \quad \text{Eq. (2.1)}$$

Let the available set of ω airports be AP,

$$AP = \{ap_i \mid i = 1,2,...,\omega\} \quad \text{Eq. (2.2)}$$

Each entry in AP contains various information, such as the airport name, a key to identify whether it is a fueling station, a key to identify whether it is a maintenance station, and a key to identify whether it is an off-land station or an in-land station. Thus, it can be formulated as a vector, defined as follows:

$$ap_i = (name^{(i)}, fuel^{(i)}, m^{(i)}, off^{(i)}) \quad \text{Eq. (2.3)}$$

where $name^{(i)}$ is the name of airport i, and

$$fuel^{(i)} = \begin{cases} 1 & \text{if } ap_i \text{ has a fuel station} \\ 0 & \text{otherwise} \end{cases}$$

$$m^{(i)} = \begin{cases} 1 & \text{if } ap_i \text{ has a maintenance station} \\ 0 & \text{otherwise} \end{cases}$$

$$off^{(i)} = \begin{cases} 1 & \text{if } ap_i \text{ is an off} - \text{land station} \\ 0 & \text{if } ap_i \text{ is an in} - \text{land station} \end{cases}$$

for all $i \le \omega$.

In each daily flight schedule, the maximal number of flight of assigned to each aircraft should not exceed the number listed in the laws and regulations. Let β represent the maximal number of flights that each aircraft is allowed in one day, and then the set of daily flights, or timetable F, defined as:

$$F = \{f_i \mid i = 1,2,...,\gamma\} \quad \text{Eq. (2.4)}$$

where $\gamma \le \alpha\beta$. Since each entry in F contains various information, which can be defined as a vector:

$$f_i = (id^{(id)}, t_D^{(id)}, t_A^{(id)}, ap_D^{(id)}, ap_A^{(id)}) \quad \text{Eq. (2.5)}$$

where
$$\begin{cases} id & : \text{flight identifier} \\ t_D^{(id)} & : \text{departure time of } ap_D^{(id)} \\ t_A^{(id)} & : \text{arrival time of } ap_A^{(id)} \\ ap_D^{(id)} & : \text{departure airport} \\ ap_A^{(id)} & : \text{arrival airport} \end{cases}$$

and $ap_D^{(i)}, ap_A^{(i)} \in AP$. The four fields $t_D^{(id)}$, $t_A^{(id)}$, $ap_D^{(id)}$, and $ap_A^{(id)}$ are indexed by the identifier id of each entry in F.

According to the definitions above, the flight schedule can be represented as a 2-dimensional matrix, denoted as S:

$$S = \begin{bmatrix} s_{11} & s_{12} & \cdots & \cdots & s_{1\beta} \\ s_{21} & \ddots & & & \vdots \\ \vdots & & s_{ij} & & \vdots \\ s_{\alpha-1,1} & & & \ddots & s_{\alpha-1,\beta} \\ s_{\alpha 1} & \cdots & \cdots & \cdots & s_{\alpha\beta} \end{bmatrix} \quad \text{Eq. (2.6)}$$

where $s_{ij} = \begin{cases} \text{an identifier of a } flight \in F & \text{, actual flight} \\ 0 & \text{, a dummy} \end{cases}$

Eq. (2.7)

for all $i \le \alpha, j \le \beta$.

The first thing one should do is to assign the duties into the flight schedule S to generate a draft flight schedule. Here, s_{ij} means a flight identifier for the jth duty of the *i*th aircraft.

As described earlier, the problem contains various objectives. Some are constraints, and others are cost evaluation. Let the set of objectives *OBJ* be written as

$$OBJ = \{obj_i \mid i = 1,2,\ldots,\pi\} \qquad \text{Eq.(2.8)}$$

where π is the number of objectives. Here we let the value of π be 6. The six objectives are discussed as follows:

1. The departure time of each flight assigned to an aircraft should be 25 minutes latter than the arrival time of its previous flight.
2. The origin airport of each flight assigned to an aircraft should be the same as the destination airport of its previous flight.
3. Each aircraft should not fly longer than the fueling period.
4. Each aircraft should not fly longer than the maintenance period.
5. The destination of last flight of each aircraft in a daily flight schedule should be an in-land airport.
6. The number of aircraft should be equal to or less than available number of aircrafts.

3. Multiobjective GA Approaches

We integrate genetic algorithm with a number of methods to solve this problem. First, we employ the Method of Inequalities (MOI) to represent our objectives. Second, since the number of objective is more than one, a better method, rank-based evaluation method [20] is used to evaluate each candidate solution. To prevent the problem of conflicting objectives during the search procedure, a 2-degree of freedom design method is used to solve the deadlock problem. Finally, we integrate these methods in the genetic algorithm to solve the problem of flight scheduling.

3.1 Method of Inequalities

In general, most optimization problems have an objective function with a single value that should be maximal or minimal. However, some problems cannot be combined into a single value. Suppose that the *i*th objective function is ϕ_i, these objective functions may have the following form:

$$\min_{p\in P}\{\phi_i(\mathbf{p}),\ i = 1,2,\cdots,\pi\} \qquad \text{Eq. (3.1)}$$

where ***p*** is the set of tunable parameters, and π is the number of objectives.

Lots of problems include various design objectives that should be optimized simultaneously. The objectives are usually conflicting in most case. It is natural to represent these objectives in the form of inequality constraints on multiple performance indices. The specification can be represented by inequalities as follows:

$$\phi_i(\mathbf{p}) \le \varepsilon_i,\ (i = 1,2,\ldots,\pi) \qquad \text{Eq. (3.2)}$$

where $\phi_i(\mathbf{p})$ is a *scalar performance index* depending on the tunable parameter vector *p*, *P* is the set of possible **p**, and ε_i is an admissible performance bound specified by the designer.

In multi-objective optimization, a Pareto optimal solution consisting of many non-dominated solutions is considered to be a wise choice [21]; a Utopian solution, which is the best in all dimensions, can be considered as a special case of Pareto optimal solutions. For Pareto optimization, define the performance index

$$\Phi(\mathbf{p}) \equiv [\phi_1(\mathbf{p})\ \ \phi_2(\mathbf{p})\ \ \cdots\ \ \phi_\pi(\mathbf{p})]' \qquad \text{Eq. (3.3)}$$

which is assumed to have Pareto optimal solutions.

To discuss the Pareto optimal solutions, we use the following notations for vector inequalities. For the vector inequality of two vectors **x** and **y** with elements denoted as x_i and y_i, one can write $\mathbf{x} \le \mathbf{y}$ if and only if for each element, inequality $x_i \le y_i$ holds, or $\mathbf{x} \ge \mathbf{y}$ if and only if for each element, inequality $x_i \ge y_i$ holds. However, this is rarely the case in multi-objective optimization. For the most case, one can use the dominated relation, *e.g.* **x** dominates **y**, denoted as $\mathbf{x} \prec \mathbf{y}$ if and only if **x** < **y**, and the relations beyond this is called non-dominated. With the notation $\mathbf{x} \prec \mathbf{y}$, a vector **p** is defined as a Pareto optimal solution if and only if there exists no vector $\hat{\mathbf{p}}(\neq \mathbf{p})$ satisfying $\Phi(\hat{\mathbf{p}}) \prec \Phi(\mathbf{p})$. Note that a utopian solution is a special case of Pareto optimal solution.

A vector of auxiliary performance indices related to the inequality performance specifications is defined as

$$\lambda_i(\mathbf{p},\varepsilon_i) \equiv \begin{cases} 0 & (\text{if}\,\phi_i(\mathbf{p}) \le \varepsilon_i) \\ \phi_i(\mathbf{p}) - \varepsilon_i & (\text{if}\,\phi_i(\mathbf{p}) > \varepsilon_i) \end{cases}, i = 1,2,\ldots,\pi$$

Eq. (3.4)

and an auxiliary vector index is defined as

$$\Lambda(\mathbf{p},\varepsilon) = [\lambda_1(\mathbf{p},\varepsilon_1), \lambda_2(\mathbf{p},\varepsilon_2), \cdots, \lambda_M(\mathbf{p},\varepsilon_M)]' \quad \text{Eq. (3.5)}$$

where $\boldsymbol{\varepsilon} \equiv [\varepsilon_1 \quad \varepsilon_2 \quad \cdots \quad \varepsilon_M]'$ is a set of admissible performance bound, which can be obtained by training.

By using above methods, we may formulate the objectives described in Section 2 in the form of inequalities.

$$OBJ(p,\varepsilon) = [obj_1(p,\varepsilon_1), obj_2(p,\varepsilon_2) \cdots, obj_M(p,\varepsilon_M)]'$$

Eq (3.6)

$$obj_i(\mathbf{p},\varepsilon_i) \equiv \begin{cases} 0 & (\text{if}\,\phi_i(\mathbf{p}) \le \varepsilon_i) \\ obj_i(\mathbf{p}) - \varepsilon_i & (\text{if}\,\phi_i(\mathbf{p}) > \varepsilon_i) \end{cases}, \quad i = 1,2,\ldots,M$$

Eq (3.7)

3.2 Improved Rank-based Multiobjective Method

One of the major problems of applying MGA to multi-objective optimization is the computation complexity of multi-criterion fitness. Especially when the number of objectives gets large, the calculation of multi-objective fitness becomes tedious.

In this paper, we employ a computation procedure proposed by Liu [21] for nondominated sorting. Suppose that two vectors with non-negative elements satisfy the relation that the vector **x** dominates the vector of *y*, or $\mathbf{x} \prec \mathbf{y}$ if and only if the inequality

$$\sum_{i=1}^{M} x_i \le \sum_{i=1}^{M} y_i \qquad \text{Eq. (3.8)}$$

holds. In general, the value of rank of each vector can be determined by using the number of candidate solution that it dominates in the current iteration.

Eq. (3.8) can be applied to the improved rank-based fitness assignment method. Suppose that the set of vectors Π,

$$\Pi = \{\Lambda_1, \Lambda_2, \cdots, \Lambda_N\} \qquad \text{Eq. (3.9)}$$

where N is the size of Π .The set Π contains N vectors in M dimensions,

$$\Lambda_i = \{\lambda_{i1}, \lambda_{i2}, \cdots, \lambda_{iM}\} \qquad \text{Eq. (3.10)}$$

The procedure of improved rank-based fitness assignment method is shown as follows:

Input: a set of vectors $\Pi = \{\Lambda_1, \Lambda_2, \cdots, \Lambda_N\}$, a counter *K*, temporary set Q, G,

Output: a set of vectors $\Pi = \{\Lambda_1, \Lambda_2, \cdots, \Lambda_N\}$, which is assigned fitness values to each Λ according to its rank.

Step 1: Sort vectors Λ_i in Π from the least to the largest according to $\sum_{j=1}^{M} \lambda_{ij}$. Let $K \equiv 1$.

Step 2: Let temporary non-dominated set $\mathbf{Q} \equiv \varnothing$, temporary dominated set $\mathbf{G} \equiv \varnothing$.

Step 3: Let the first point Λ_1 of Π be a criterion.

Step 4: Remove the dominated points Λ_i by checking whether or not the following conditions are satisfied.

$$\max g_{ij} \le 0 \wedge (\exists_j)(g_{ij} < 0),$$

where g_{ij} are the components of $\mathbf{g}_i = \Lambda_1 - \Lambda_i$, $j = 1,2,3,\ldots,M$, and $i = 2,3,\ldots,N$.

Step 5: Remove the criterion Λ_1 and the dominated points Λ_i from Π. Save Λ_1 to the temporary non-dominated set Q, Λ_i to the temporary dominated set G.

Step 6: If $\Pi \ne \varnothing$ then go to Step 3.

Step 7: Rank the points of Q as K. Let $\Pi \equiv$ G and $K \equiv K+1$.

Step 8: If $\Pi \ne \varnothing$ then go to Step 2.

Step 9: The fitness of each vector $V_N(i)$ is given by the following linear function

$$V_N(i) = \frac{2(f_{\max} - 1)}{N-1}(R_{\max} + 1 - rank(i)) \quad \text{Eq. (3.11)}$$

where $rank(i)$ is the rank of ith vector. Each $rank(i)$ will be less than or equal to N. $R_{\max}$ is the maximal value of $rank(i)$, and $f_{\max}$ is considered as a GA diversity maintaining parameter that is specified by a designer. Here, we let $1 < f_{\max} \le 2$.

According to above procedure, we may assign a rank to each candidate solution, and obtain the fitness values of each vector, or candidate solution.

3.3 2-Degree of Freedom Design

The deadlock problem can happen for multiobjective optimization problems, due to objectives are often conflicting.

In general, the factors that may cause deadlock problems are shown in the following:

1. Mutual exclusion: There exists at least one non-sharable resource. In other words, any resource can only used by one process at a time. After the resource is released, other processes can use this resource.
2. Hold and Wait: There exist at least one-process that holds a resource and wait for another resource that is hold by another process.
3. Non-preemptive: Each resource has no priority; consequently, a resource may be released after the process that holds it completes the job.
4. Circular wait: There must exist a set of processes {*Proc* 0, *Proc* 1… *Proc n*} that wait for resources, where *Proc* 0 waits for the resource that Proc1 holds, *Proc* 1 waits for the resource that Proc2 holds, … , *Proc n*-1 waits for the resource that *Proc n* holds, and *Proc n* waits for the resource that *Proc* 0 holds.

Since we may generate several candidate solutions by a random procedure, it is possible to generate infeasible solutions that do not match the objectives described in Section 2. These infeasible schedules should be adjusted until they match the objectives. However, the first objective, called the time objective and the second objective, called the airport objective may not be optimized simultaneously. Then a deadlock problem occurs.

In order to solve the deadlock problem, the method of multi-degree of freedom is employed. In this case, the method of 2-degree of freedom is used. It may find solutions to meet one constraint first, and then to meet the other constraint, repeatedly. To achieve the goal of 2-degree of freedom, we should divide time constraint and airport constraint, $Proc_{time\&airport}$, into two parts. In other words, one is a procedure only used to solve airport constraint, denoted as $Proc_{place}$, and the other is only used to solve time constraint, denoted as $Proc_{time}$. Thus, we alternatively use the procedures $Proc_{time}$ and $Proc_{airport}$ instead. Therefore, we should first optimize for one constraint, and then optimize the other constraint. Figure 1 represents the block diagram of 1-degree of freedom design method and Figure 2 represents the 2-degree of freedom design method.

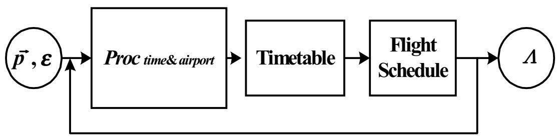

Figure 1. The block diagram of 1-degree of freedom design method

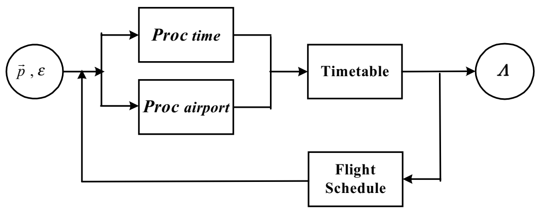

Figure 2. The block diagram of a 2-degree of freedom design method

3.4 Multiobjective Optimization with Genetic Algorithm

Integrating the Method of Inequalities, Improved Rank-Based Fitness Assignment Model, with the 2-degree of freedom design method, and the genetic algorithm are used to optimize the flight schedules. The steps of this algorithm is shown as follows:

Encoding: The model is also a 2-dimensional form, which is the same as Eq. (2.6) in Section 2. Thus, each row represents the flights assigned to an aircraft, and each element represents a flight. All flights contain the information of flight identifier, departure time, arrival time, departure airport, and arrival airport. As described earlier, for the purpose of convenience computing, we assign dummy duties to a chromosome (or a flight schedule) to make all chromosomes have the same length.

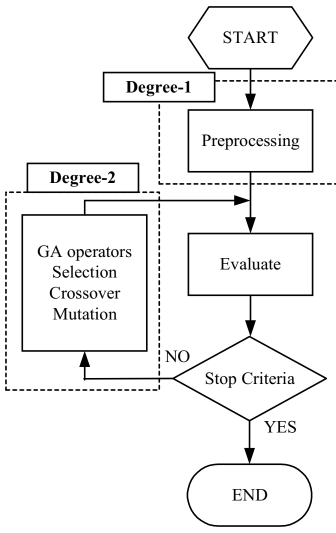

Figure 3. The 2-degree of freedom design method with MGA

Preprocessing: Since there may be some conflicts between the time and place constraints, we propose to relax some of them at first and then tune the others. In order to optimize the time constraint, we arrange the flights assigned to each aircraft according to the departure time. This will enhance the validity of time constraint.

Selection: We use the improved rank-based selection operation. According to the rank value, which is computed by the method in Subsection 3.2, the individuals with better evaluation value will have more chance to be selected.

Crossover: We use a single-point crossover method. First, we translate the 2-dimensional chromosome into a one-dimensional string. In order to illustrate our crossover method, we use a simplified chromosome. Each uppercase letter represents a flight duty. The parents and the dividing point are on the left side; the generated offspring are on the right side.

Mutation: A two-point mutation method. First, select two flight duties are selected randomly. Then, their positions are exchanged.

4. Experimental Results

In this section, applying them to solve a practical problem of flight scheduling validates the proposed methods. As shown in Figure 4, there are two off-land airports among 7 airports. The airline routes belong to a hub-and-spoke network. All arcs in this figure represent available routes. The numbers of available aircraft and airports are 9 and 7, respectively. The maximal number of flights assigned to each aircraft is no more than 12.

The main parameters of genetic algorithms are as follows: Population size is 100, the maximal number of generation is 5000, crossover rate is 1, mutation rate 0.2, and reproduction rate is 0.8. The experiment is run on an Intel Pentium 266 MHz PC. Our algorithm is implemented in Borland C++ Builder 4.0

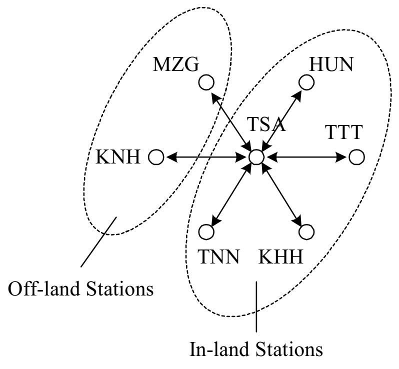

Figure 4.The Hub-and-Spoke Network of all flights

As we described earlier, the problem of flight scheduling has several conflicting objectives to optimize simultaneously. Six objectives, described in Eq. (2.8), are considered in this experiment. For more detailed design, we find all the objectives can be separate into two kinds of constraints and one kind of performance: 1) constraints of time: obj_1, obj_3 and obj_4, 2) constraints of place: obj_2, obj_5 and 3) cost

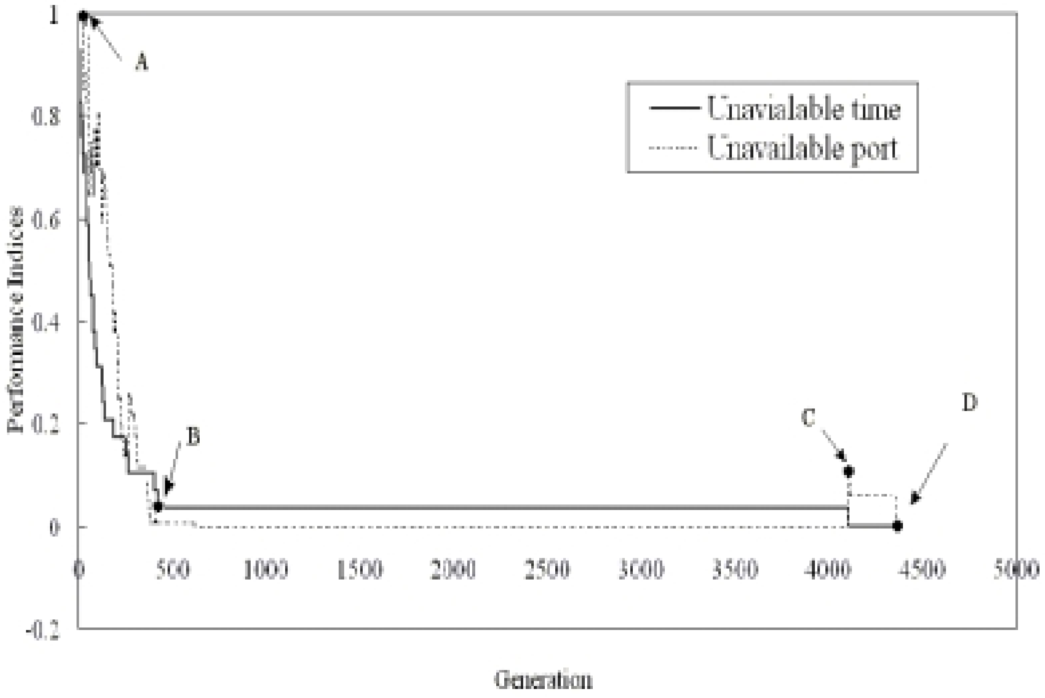

Figure 5. The convergence of violations of the identical airports and available time

performances, obj_6. Especially, obj_1 and obj_2 representing the available ground-holding time and the identical departure and arrival airport constraints are very difficult to be satisfied simultaneously. In this research we design the schedule with two steps: first considering obj_1 and obj_2 and then optimizing the others.

Figure 5 presents the convergence of the two constraint indices (unavailable denaturing time and identical airport), respectively. In this figure, the constraint indices are normalized. At the beginning of the experiment, the number of violations of available time and identical airport are extremely high. As iteration increases, they are decreased simultaneously. Finally, the program obtains a feasible solution near the 4500th generation. The feasible timetable we generated has no violations of available time and identical airport. Therefore, the value of unavailable time and airport are both zero. Furthermore, to optimize the remaining constraints and performance, obj_3 to obj_6, we use the pervious result as the initial value and run the program again.

Resolutely, this system can provide a mechanism for 2-degree of freedom design method and the objectives can be added to the problem easily. In other words, planners can deal with many objectives step by step. Moreover, since we limited the number of aircrafts previously, it makes a great convenience for planner to generate a reasonable solution.

5. Discussion

As shown in Figure 5, we may obtain that both unavailable time and unavailable airport converge at point A. Since the number of unavailable airport converges naturally, unavailable time converges manually by the preprocessing procedure, the number of unavailable airports will converge more smoothly than unavailable time. After point B, both the constraints converge very smooth.

In the period between B and C, it causes a deadlock problem. In other words, this becomes a *premature* solution since all ground holding time match the ground holding time constraint, but the airports do not match the airport constraint. By the proposed method, we may solve the deadlock problem by a 2-degree of freedom design method. Since this is caused by two conflicting objectives, we relax the ground holding constraint first. Therefore, at point C the unavailable time may increase. At the same time, unavailable number of airport may decrease. Hence, the 2-degree of freedom certainly solve the deadlock problem. At point D, both constraints decrease to 0, and it finds out a feasible solution. According to the experimental results, we verify that the proposed method can overcome the deadlock problem, and find out a feasible solution.

6. Conclusions

We have proposed a multi-objective Genetic algorithm to solve the problem of flight scheduling. By using this algorithm, we have obtained feasible flight schedules under the limitation of fixed number of aircrafts instead of just minimizing the number of aircraft. When perturbation occurs, this system uses a mechanism for decision support.

The experimental results show that the proposed algorithm can solve the deadlock problem caused by conflicting objectives in a 2-degree of freedom design method. Consequently, it can decrease the cost of flight scheduling.

References

[1] G. Yu, *Operations Research in the Airline Industry*, Kluwer Academic Publishers, 1998.

[2] S. Y. Yan and H. F. Young, “ A Decision Support Framework for Multi-Fleet Routing and Multi-Stop Flight Scheduling,” *Transportation Research,* Vol. 30, No.5, pp. 379-398, 1996.

[3] A. Levin, “Some Fleet Routing and Flight scheduling Problems for Air Transportation Systems,” Flight Transportation Laboratary, Report R-68-5, Massachusetts Institute of Technology, MA, 1969.

[4] R. W. Simpson, “Scheduling and Routing Models for Airline Systems,” Flight Transportation Laboratory, Report R-68-3, Massachusetts Institute of Technology, MA, 1969.

[5] W. Hyman and L. Gordon, “Commercial Airline Scheduling Tecfhnique,” *Transportation Research*, Vol 12, pp. 23-29, 1978.

[6] A. Levin, “Scheduling and Fleet Routing Models for Transportation Systems,” *Transportation Science*, Vol. 5, pp.232-255, 1971

[7] D. Teodorovic, *Airline Operations Research*, Gordon and Breach Science, New York, 1988.

[8] M. R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman & Company, San Francisco, 1979.

[9] D. Teodorovic and S. Gubernic, “Optimal Dispatching Strategy on an Airline Network after a Schedule Perturbation,” *European Journal on Operations Research*, Vol 15. pp. 178-182, 1984.

[10] B. C. Lee, “Routing Problem with Service Choices,” Flight Transportation Laboratory, Report R86-4, Massachusetts Institute of Technology, MA, 1986.

[11] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[12] J. Abara, “Applying Integer Linear Programming to Flight Scheduling Problem”, *Interfaces*, Vol. 19, pp. 20-28, 1989.

[13] A. Balakrishman, T. Chien, and R. Wong, “Selecting Aircraft Routes for Long-haul Operations: a formulation and solution method,” *Transportation Res.* Vol. 24B, pp. 57-72, 1990.

[14] S. Y. Yan and H. F. Young, “A Decision Support Multi-Fleet Routing and Multi-Stop Flight Scheduling”, *Transportation Research*, A, Vol. 30, No. 5, pp. 379-398, 1996.

[15] J. H. Holland, “Outline of Control Parameters for Genetic Algorithms,” *Journal of the Association for Computer Machinary,* Vol. 3, pp. 297-314, 1962.

[16] J. H. Holland, *“Adaptive in Neural and Artificial Systems,”* Ann Arbor, MI: University of Michigan Press, 1975.

[17] A. I. Jarrah, G. Yu, N. Krishnumurthy, and A.Rakshit, “A Decision Support Framework for Airline Flight Cancellations and Delays,” *Transportation Science*, Vol. 27, pp. 266-280, 1993.

[18] V. Zakian and U. AI-Naib, “Design of Dynamical and Control Systems by the Method of Inequalities,” *Proceedings of IEE*, Vol. 120, No.11, pp. 1421-1427, 1973.

[19] V. Zakian, “Perspectives on the Principle of Matching and the Method of Inequalities,” *International Journal Control*, Vol. 65, No.1, pp. 147-175, 1996.

[20] B. C. Levin, “Routing Problem with Service Choice,” Flight Transportation Laboratory, Report R86-4. Massachusetts Institute of Technology, MA.

[21] T. K. Liu, *Application of Multi-objective Genetic Algorithms to Control System Design,* Doctorial Thesis of Graduate School of Sciences, Tohoku University, Japan, 1997.