Binary-Representation-Based Genetic Algorithm for Aircraft Arrival Sequencing and Scheduling

Xiao-Bing Hu and Ezequiel Di Paolo

***Abstract*—Arrival sequencing and scheduling (ASS) at airports is an NP-hard problem. Much effort has been made to use permutation-representation-based genetic algorithms (GAs) to tackle this problem, whereas this paper attempts to design an efficient GA based on a binary representation of arriving queues. Rather than using the order and/or arriving time of each aircraft in the queue to construct chromosomes for GAs, this paper uses the neighboring relationship between each pair of aircraft, and the resulted chromosome is a 0-1-valued matrix. A big advantage of this binary representation is a highly efficient uniform crossover operator, which is normally not applicable to those permutation representations. The strategy of receding horizon control (RHC) is also integrated into the new GA to attack the dynamic ASS problem. An extensive comparative simulation study shows that the binary-representation-based GA outperforms the permutation-representation-based GA.**

***Index Terms*—Air traffic control (ATC), arrival scheduling and sequencing, binary representation, genetic algorithm (GAs), receding horizon control (RHC), uniform crossover.**

NOMENCLATURE

AAD	Average airborne delay.
ALT	Assigned landing time.
ASS	Arrival sequencing and scheduling.
ATC	Air traffic control.
FCFS	First-come first-served.
GA	Genetic algorithm.
LTI	Landing time interval.
OCT	Online computational time.
PLT	Predicted landing time.
RHC	Receding horizon control.
TSP	Traveling salesman problem.

I. INTRODUCTION

AIR TRAFFIC control (ATC) has been the hottest topic in transportation research for the past few decades due to the constantly increasing aviation traffic volume around the world [1]. Basically, ATC includes air route traffic control (e.g., confliction detection and resolution [2] and route optimization [3]) and traffic management in terminal areas (such as airport surface control [4], rate control [5], and airport capacity management [6]). Among the terminal traffic problems, aircraft arrival sequencing and scheduling (ASS) is a major issue in the daily airport operations. A simple way to perform ASS is to schedule arrival aircraft on a first-come-first-served (FCFS) basis on a predicted landing time (PLT) at the runway. Although FCFS scheduling establishes a fair order in terms of PLT, it ignores other useful information that can be used to efficiently make use of the capacity of the airport, reduce airborne delays, and/or improve the airport service to airlines [1], [7], [8]. For instance, shifting aircraft position according to a landing time interval (LTI), which is the minimum permissible time interval between two successive landings, can significantly reduce airborne delays during the arriving and landing phase [9]–[16]. Basically, the LTI is a function of the types and relative positions of the two aircraft, as illustrated in Table I, where arriving aircraft are classified into four categories according to speed, capacity, weight, and other technical characteristics. Apparently, the LTIs in Table I are asymmetric. For example, a minimum LTI of 200 s is required for a Boeing 727 to follow a Boeing 747, whereas a minimum LTI of only 72 s needs to be satisfied for the same pair of aircraft in reverse order. The reduction in airborne delay is actually achievable by taking advantage of the asymmetries of the LTIs. However, like a double-edged sword, these asymmetries make the position-shifting-based ASS an NP-hard problem, which has no known algorithm for finding a global optimal solution within a polynomial-bounded amount of time [15]. In the past decades, many methods, such as travelling salesman problem modeling, dynamic programming, expert system, Monte Carlo optimization, and evolutionary computing, have been used to solve the position-shifting-based ASS problem [12]–[20]. This paper aims to shed a little more light on how to design efficient genetic algorithms (GAs) for the dynamic ASS problem.

As a large-scale parallel stochastic searching and optimizing algorithm, GAs are effective in solving NP-hard problems such as the ASS problem (e.g., see [17], [19], [21], and [22]). Although the original GA was developed based on a binary

TABLE I
MINIMUM LTI [15]

S_{ij} (seconds)		Category of following aircraft: j			
		1	2	3	4
Category of leading aircraft: i	1	96	200	181	228
	2	72	80	70	110
	3	72	100	70	130
	4	72	80	70	90

Note: 1=B747; 2=B727; 3=B707; 4=DC9.

Manuscript received December 17, 2006; revised October 22, 2007 and December 4, 2007. This work was supported by the Engineering and Physical Sciences Research Council under Grant EP/C51632X/1. The Associate Editor for this paper was P. Kostek.

The authors are with the Department of Informatics, University of Sussex, Brighton BN1 9QH, U.K. (e-mail: Xiaobing.Hu@sussex.ac.uk; dr_xiaobinghu@hotmail.co.uk; ezequiel@sussex.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2008.922884

representation of solutions to the problem under consideration (see [23] and [24]), various problem-dependent nonbinary representations were often used in engineering applications of GAs. For example, [19], [21], and [22] used the order of each aircraft in an optional arriving queue to construct chromosomes, whereas [17] even included the assigned landing time (ALT) of each aircraft in the chromosomes. These permutation representations of arriving queues make the chromosomes easy to be understood by human air traffic controllers, academic researchers, or general readers with regard to the underlying physical meanings, but this does not necessarily mean they are also easy for machines, particularly computer systems, to handle. How to design highly efficient evolutionary operators based on permutation representations is always a challenging task, because there is always a high probability that purely random evolutionary operations can make chromosomes invalid/infeasible in terms of the underlying physical meanings. For instance, in an optional arriving queue, each aircraft can be included once and only once, which can hardly be guaranteed by purely random evolutionary operations. Therefore, deterministic measures and heuristic rules often have to be introduced to help the evolutionary operators produce valid/feasible chromosomes. No doubt there is a need to make a good tradeoff between diversifying chromosomes and keeping feasibility, which is not easy to achieve without cost, particularly in the case of permutation-representation-based GAs. Sometimes, the cost is even to give up the crossover operator, simply because it is too difficult to design a good one that not only can efficiently evolve chromosomes but also can effectively avoid invalid/infeasible chromosomes. The GA in [22] and most methods in [19] and [21] did not adopt the crossover operation, and the others had to take some computationally expensive measures to guarantee the feasibility of chromosomes produced by crossover [19].

Although there has long been a strong debate about the usefulness of crossover, there have also been many successful implementations of GAs with crossover playing important roles, particularly in the case of binary representations [24]. The underlying idea of crossover is still sound from an evolutionary point of view; randomly pick out two parent chromosomes and exchange their gene sections, so that the offspring chromosomes can inherit those common genes in the parents, and at the same time, search new possibilities of recombining those different genes in the parents. Normally, key genes and gene sections shared by many fittest chromosomes can be efficiently identified and effectively protected during generations of crossover, which is hardly achievable by mutation. The uniform crossover operator, which generates only one offspring by randomly inheriting genes from either of two parents, is probably the most powerful crossover because it allows the offspring chromosomes to search all possibilities of recombining those genes that are different in the parents [25]. However, it is very difficult, if not impossible, to design an effective uniform crossover operator based on permutation representations for the ASS problem.

This paper attempts to design an efficient GA based on a binary representation rather than a permutation representation to solve the dynamic ASS problem. To this end, the neighboring relationship between each pair of aircraft in an optional arriving queue is used to construct chromosomes, which are therefore formulated as 0-1-valued matrices. Based on the binary matrix, a highly efficient uniform crossover operator is designed to inherit useful gene sections and evolve others. The remainder of this paper is organized as follows: Some relevant previous work is summarized in Section II. The new GA based on a binary representation is given in Section III. An extensive simulation study is reported in Section IV, which is followed by some conclusions in Section V.

II. Previous Work

A. Dynamic ASS

In the real world of air traffic control, ASS is always carried out in a dynamic environment, where the predicted air traffic data and operational conditions at airports vary over time due to various uncertainties and disturbances. Therefore, directly modeling and developing algorithms based on the dynamic feature of the ASS problem could bring advantages. As illustrated in [22] and [26], the strategy of receding horizon control (RHC), which is also known as model predictive control, is useful to model and solve the dynamic ASS problem. Simply speaking, RHC is an N-step-ahead online optimization strategy. Within this framework, decisions are made by looking ahead for N steps in terms of a given cost/criterion, and only the decision for the first step is actually implemented. Then, the implementation result is checked, and a new decision is made by taking account of updated information and looking ahead for another N steps. Fig. 1 illustrates the basic idea of how to apply the RHC strategy to solve the dynamic ASS problem. For more details on RHC and its applications to the ASS, see [22], [26], and [27].

Suppose N_{AC} aircraft are planning to land at an airport during the operating day, and C_i, P_i, and A_i are the category, the PLT, and the ALT of the ith aircraft in the original predicted arrival sequence, respectively. Let $Q(n)$ record the nth aircraft in the optimized arrival sequence, i.e., $Q(n) = i$ means that the ith aircraft in the original predicted arrival sequence turns out to be the nth aircraft in the optimized arrival sequence. With Q, one can calculate A as

$$A_{Q(n)} = \begin{cases} P_{Q(n)}, & n = 1 \\ \max\left(P_{Q(n)}, A_{Q(n-1)} \right. & \\ \qquad \left. + S\left(C_{Q(n-1)}, C_{Q(n)}\right)\right), & n > 1 \end{cases} \tag{1}$$

where $S(i, j)$ is the LTI for an aircraft of category j to follow an aircraft of category i to land. Then, the airborne delay of the ith aircraft in the original predicted arrival sequence is

$$D_i = A_i - P_i, \qquad i = 1, \ldots, N_{\mathrm{AC}}. \tag{2}$$

The objective of the ASS in the operating day is to find an optimal arrival sequence such that the total airborne delay can be minimized, i.e.,

$$\min_{Q(1),\ldots,Q(N_{\mathrm{AC}})} J_1 = \min_{Q(1),\ldots,Q(N_{\mathrm{AC}})} \sum_{i=1}^{N_{\mathrm{AC}}} D_i. \tag{3}$$

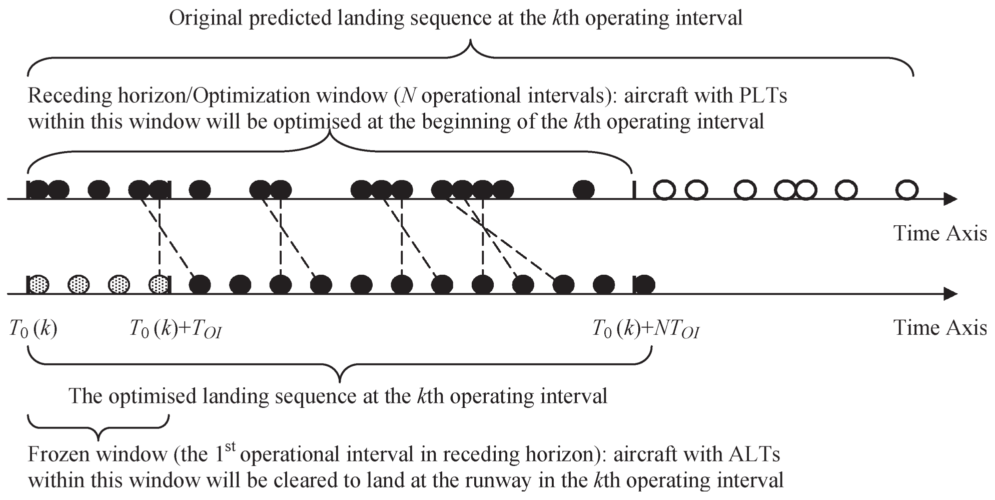
Original predicted landing sequence at the kth operating interval
Receding horizon/Optimization window (N operational intervals): aircraft with PLTs within this window will be optimised at the beginning of the kth operating interval
Time Axis
Time Axis
$T_0(k)$
$T_0(k)+T_{OI}$
$T_0(k)+NT_{OI}$
The optimised landing sequence at the kth operating interval
Frozen window (the 1st operational interval in receding horizon): aircraft with ALTs within this window will be cleared to land at the runway in the kth operating interval

Fig. 1. RHC for dynamic ASS.

In the dynamic ASS, the minimization problem (3) needs to be modified to apply the RHC strategy. First, we need to distinguish two sets of variables: one set of real variables in the implementation and the other set of computational variables predicted/calculated in the receding horizon. The real variables are indicated by (k), which means that the value of the associated variable is measured or actually implemented for the current operational interval k. The computational variables are indicated by $(k+i|k)$, which means that the value of the associated variable for the $(k+i)$th operational interval is predicted/calculated at the beginning of the current operational interval k, where $i=0,\ldots,N_H-1$, and N_H is the number of time intervals included in a receding horizon. The computational variables for/during the whole receding horizon are indicated by $(\cdot|k)$. Second, we need to the replace the objective function J_1 in (3) with

$$J_2(\cdot|k) = \sum_{i=1}^{M_{\mathrm{AC}}(\cdot|k)} \alpha(j)D_i(\cdot|k) + \beta\left(N_{\mathrm{AC}}(\cdot|k) - M_{\mathrm{AC}}(\cdot|k)\right)$$
$$(k+j-1)T_{\mathrm{OI}} < P_i(\cdot|k) \le (k+j)T_{\mathrm{OI}}$$
$$j = 1,\ldots,N_H \tag{4}$$

where $M_{\mathrm{AC}}(\cdot|k)$ is the number of aircraft whose $A_i(\cdot|k)$ are within the receding horizon, and $0<\alpha(j)\le 1$ is a weight applicable to those aircraft whose $P_i(\cdot|k)$ are within the $(k+j-1)$th operational interval, T_{OI} is the length of an operational interval, and $\beta>0$ is a terminal weight to punish those aircraft that are not covered by the receding horizon. The objective function J_2 in (4) is improved from what was used in [22] for the dynamic ASS. First, the weights $\alpha(j)$ may vary over the receding horizon, normally decreasing as j increases, which means that decisions based on the farther future information contribute less to J_2. Second, the terminal penalty is independent of the delays applied to those aircraft whose $A_i(\cdot|k)$ are within the receding horizon but only relies on the number of those aircraft whose $A_i(\cdot|k)$ are beyond the receding horizon. This can effectively prevent the assignment of few aircraft without imposing delays but leaving many aircraft not covered by the receding horizon. The maximum allowable airborne delay applicable to each individual aircraft can be used β.

With the aforementioned mathematical preparation, the dynamic ASS problem can be formulated as

$$\min_{Q(1,\cdot|k),\ldots,Q(N_{\mathrm{AC}}(\cdot|k),\cdot|k)} J_2(\cdot|k) \tag{5}$$

subject to (1) and (2). Supposing $Q^*(1,\cdot|k),\ldots,Q^*(N_{\mathrm{AC}}(\cdot|k),\cdot|k)$ is the optimal solution for the current receding horizon, then only those aircraft whose $A_i(\cdot|k)$ are within the kth operational interval will be cleared to land, i.e.,

$$Q(n,k)=Q^*(n,\cdot|k)$$
$$\text{if } kT_{\mathrm{OI}} < A_{Q^*(n,\cdot|k)} \le (k+1)T_{\mathrm{OI}};\quad n=1,\ldots,N_{\mathrm{AC}}(\cdot|k). \tag{6}$$

Then, following the RHC strategy, we set $k=k+1$ and go on to use updated information to calculate the optimal arrival sequence for the new receding horizon.

B. Permutation-Representation-Based GA

The introduction of the RHC strategy not only can soften the influence of uncertainties and disturbances in the dynamic ASS problem but can also reduce the computational burden in each online optimization, which makes it more realistic to apply GAs to solve this NP-hard problem in real time. With this motivation, [22] developed a GA based on the aforementioned dynamic ASS model in the RHC format (except the modification to J_2). Like other papers that applied GAs to the ASS problem, [22] also used a permutation representation of an optional arriving sequence. As illustrated in Fig. 2(c), the permutation representation describes the underlying physical meaning of chromosomes in a straightforward way. It is not difficult to design a mutation operator for the permutation-representation-based chromosomes, that is, to merely swap the positions of two genes. However, the crossover operation is encountered with both feasibility and efficiency problems. The feasibility issue could be addressed by introducing extra constraints and feasibility-checking measures, but the efficiency problem is really hard to solve without largely increasing the

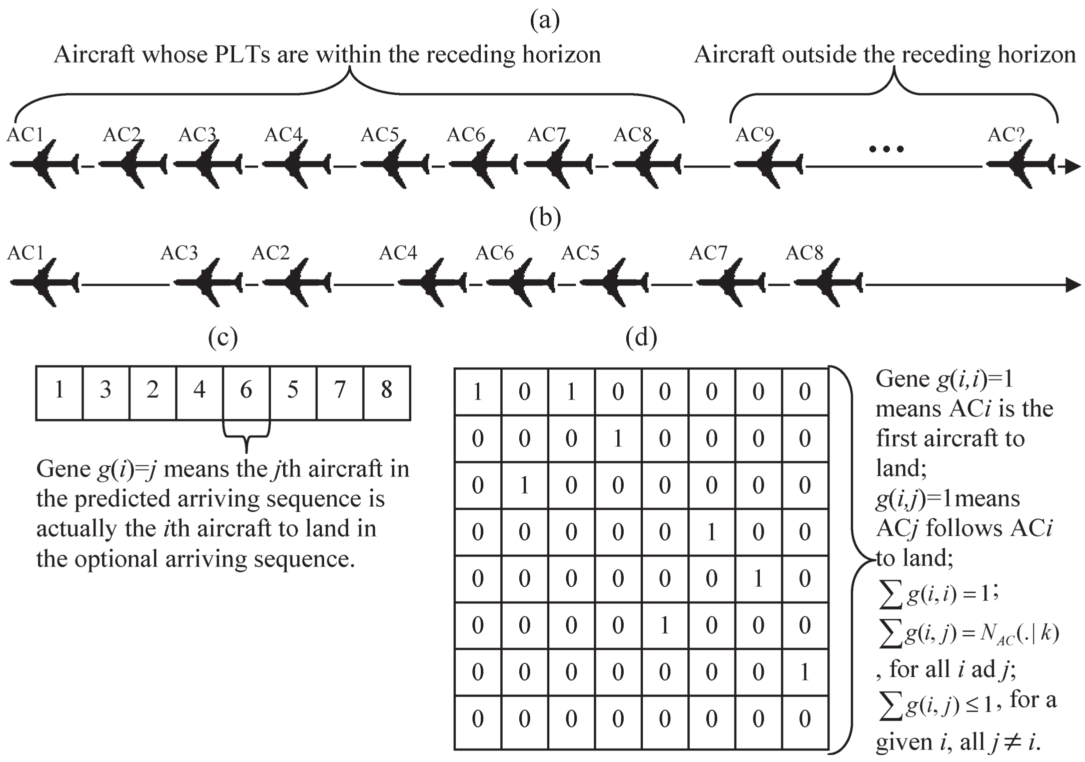
(a)
Aircraft whose PLTs are within the receding horizon
Aircraft outside the receding horizon
AC1 AC2 AC3 AC4 AC5 AC6 AC7 AC8 AC9 ... AC?
(b)
AC1 AC3 AC2 AC4 AC6 AC5 AC7 AC8
(c)
1 3 2 4 6 5 7 8
Gene g(i)=j means the jth aircraft in the predicted arriving sequence is actually the ith aircraft to land in the optional arriving sequence.
(d)
Gene g(i,i)=1 means ACi is the first aircraft to land;
g(i,j)=1means ACj follows ACi to land;
∑g(i,i)=1;
∑g(i,j)=N_AC(.|k), for all i ad j;
∑g(i,j)≤1, for a given i, all j≠i.

Fig. 2. Permutation representation versus binary representation. (a) Predicted arriving sequence at the operational interval k. (b) Optional arriving sequence. (c) Permutation-representation-based chromosome to represent the aforementioned optional arriving sequence (suppose the serial number of aircraft ACi is i). (d) Associated binary-representation-based chromosome.

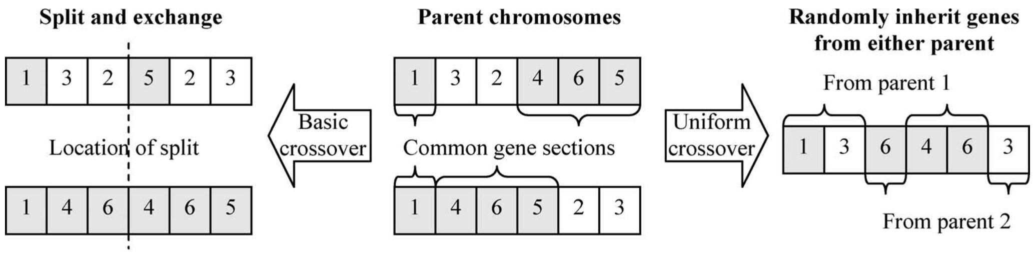

Fig. 3. Crossover operations based on a permutation representation.

computational burden, which is not desirable in the design of GAs for real-time implementations. Here, efficiency refers to efficiently identifying, inheriting, and protecting common gene sections. As shown in Fig. 3, two permutation-representation-based chromosomes share a common gene section, i.e., [AC4, AC6, AC5], which is obvious for a human to see but difficult for a machine to find out. If a crossover operator is applied, i.e., either basic crossover or uniform crossover, the common gene section will very likely be destroyed rather than be inherited. In this case, the effect of crossover is actually equal to the effect of mutation. What is worse, applying crossover to permutation-representation-based chromosomes may cause serious feasibility problems. As illustrated in Fig. 3, the offspring chromosomes of either basic crossover or uniform crossover contain some aircraft twice, which is impossible in a real arrival sequence, because each aircraft must be assigned with a unique order to land. Therefore, these offspring chromosomes generated by crossover are infeasible, although their parents are feasible. The aforementioned reasons explain why [22] did not use crossover in the reported GA.

III. Binary-Representation-Based GA

In this section, we follow the common practices of designing GAs to develop the new GA for the dynamic ASS problem. Chromosome structure, mutation operator, and crossover operator are the key steps when adopting a binary representation of arrival sequences, and heuristic rules are some problem-specific knowledge.

A. Structure of Chromosomes

A chromosome in the new GA is an $N_{\mathrm{AC}}(\cdot|k) \times N_{\mathrm{AC}}(\cdot|k)$ 0-1-valued matrix based on the neighboring relationship between each pair of aircraft in the represented arrival sequence: If the ith aircraft in the predicted arrival sequence is the first aircraft to land in the represented arrival sequence, then the entry $g(i,i)$ of the matrix is set to 1. If, in the represented arrival sequence, the ith aircraft in the predicted arrival sequence is followed by the jth aircraft in the predicted arrival sequence, then the entry $g(i,j)$ of the matrix is set to 1. Otherwise, the left entries of the matrix are set to 0. The entries of the matrix are called the genes of the chromosome. According to the underlying physical meaning of an optional arrival sequence, a feasible chromosome must satisfy the following constraints:

$$\sum_{i=1}^{N_{\mathrm{AC}}(\cdot|k)} \sum_{j=1}^{N_{\mathrm{AC}}(\cdot|k)} g(i,j) = N_{\mathrm{AC}}(\cdot|k) \tag{7}$$

$$\sum_{i=1}^{N_{\mathrm{AC}}(\cdot|k)} g(i,i) = 1 \tag{8}$$

$$\sum_{j=1}^{N_{\mathrm{AC}}(\cdot|k)} g(i,j) \begin{cases} = 1, & g(i,i) = 1; \quad N_{\mathrm{AC}}(\cdot|k) = 1 \\ = 2, & g(i,i) = 1; \quad N_{\mathrm{AC}}(\cdot|k) > 1 \\ \leq 1, & g(i,i) = 0 \end{cases} \tag{9}$$

$$\sum_{i=1}^{N_{\mathrm{AC}}(\cdot|k)} g(i,j) = 1 \tag{10}$$

which guarantee that each of the $N_{\mathrm{AC}}(\cdot|k)$ aircraft appears once and only once in an optional arrival sequence, and each aircraft is assigned a unique order in the optional arrival sequence. From (7)–(9), one can derive that there is always an empty row in the matrix. Suppose the ith row is empty, i.e., $g(i,j) = 0$ for all $j = 1, \ldots, N_{\mathrm{AC}}$. Then, it means that the ith aircraft in the predicted arrival sequence is the last aircraft in the optional arrival sequence. Fig. 2(d) gives an example of the proposed binary-representation-based chromosomes.

The aforementioned constraints will actually never be used in the proposed GA. In the initialization of a chromosome, the following procedure can effectively generate a feasible chromosome without needing to check against the constraints given by (7)–(10).

Step 1) Create an $N_{\mathrm{AC}}(\cdot|k) \times N_{\mathrm{AC}}(\cdot|k)$ matrix with all entries set to 0. Let $U = \{1, \ldots, N_{\mathrm{AC}}(\cdot|k)\}$ represent the predicted arrival sequence.

Step 2) Randomly choose $i \in U$, and set $g(i,i) = 1$. Remove i from U, i.e., let $U = U - \{i\}$.

Step 3) While $U \neq \varnothing$, do

Step 3.1) Randomly choose $j \in U$, and set $g(i,j) = 1$.

Step 3.2) Remove j from U, i.e., let $U = U - \{j\}$. Let $i = j$.

B. Mutation Operator

The basic idea of mutation is to randomly swap two successive aircraft in an arriving sequence. With the binary representation, this mutation operator is mathematically described in the following.

Step 1) Randomly choose a nonzero gene in the chromosome under mutation, e.g., $g(i,j) = 1$, $i = 1, \ldots, N_{\mathrm{AC}}(\cdot|k)$, and $j = 1, \ldots, N_{\mathrm{AC}}(\cdot|k)$. If there exist $g(j,m) = 1$ and/or $g(m,h) = 1$, $m = 1, \ldots, N_{\mathrm{AC}}(\cdot|k)$, and $h = 1, \ldots, N_{\mathrm{AC}}(\cdot|k)$, go to Step 2.

Step 2) If $i \neq j$, let

$$g(i,j) = 0 \quad g(j,m) = 0 \quad g(m,h) = 0 \tag{11}$$

$$g(i,m) = 1 \quad g(m,j) = 1 \quad g(j,h) = 1. \tag{12}$$

Otherwise

$$g(i,i) = 0 \quad g(i,m) = 0 \quad g(m,h) = 0 \tag{13}$$

$$g(m,m) = 1 \quad g(m,i) = 1 \quad g(i,h) = 1. \tag{14}$$

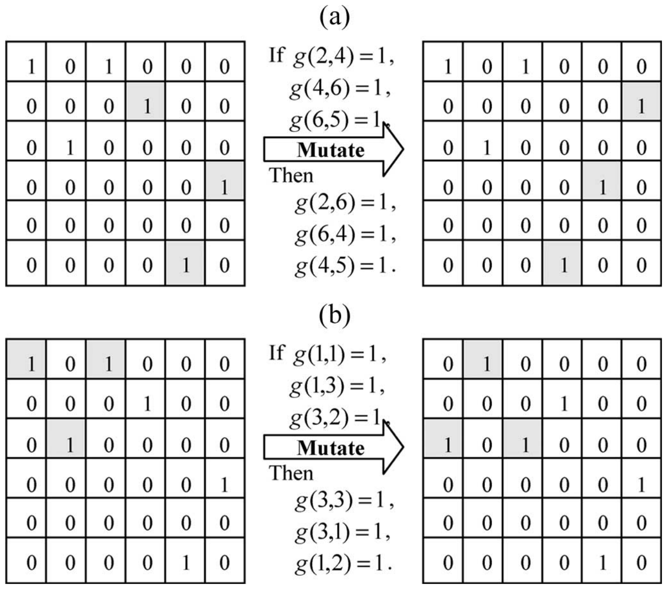
(a)
If g(2,4) = 1, g(4,6) = 1, g(6,5) = 1,
Mutate
Then g(2,6) = 1, g(6,4) = 1, g(4,5) = 1.
(b)
If g(1,1) = 1, g(1,3) = 1, g(3,2) = 1,
Mutate
Then g(3,3) = 1, g(3,1) = 1, g(1,2) = 1.

Fig. 4. Mutation based on a binary representation. (a) Case of $i \neq j$. (b) Case of $i = j$.

Clearly, the aforementioned procedure automatically guarantees the feasibility of the resulted chromosomes as long as the original chromosomes are feasible. Fig. 4 gives two examples of the mutation operation based on a binary representation. The aforementioned mutation operator is slightly different from that used in [22], which can swap not just two successive aircraft but any two aircraft in an arriving sequence. It seems more powerful but not necessarily efficient, because, first, repeating the mutation operation of swapping two successive aircraft can achieve the same result as the mutation operator in [22] does; second, there is a much higher probability in the ASS problem to swap two successive aircraft, which is actually a heuristic rule used in [22]. As will be illustrated in the simulation study, if the total number of mutation operations is fixed, then swapping two successive aircraft is more efficient than the mutation operator in [22].

C. Uniform Crossover Operator

As previously discussed in this paper, the main motivation of adopting a binary representation of solutions to the ASS problem is to design a highly efficient crossover operator, which is expected to be able to effectively and efficiently identify, inherit, and protect common gene sections in two parent chromosomes without causing feasibility problems. Basically, as illustrated in Fig. 3, there are two categories of common gene sections: One is related to those aircraft that are assigned the same orders in two optional arrival sequences, and the other is related to those identical subsequences of aircraft in two optional arrival sequences. With the permutation representation, it is computationally expensive to identify the second category of common gene sections, let along inherit or protect them. In the binary representation, common gene sections are defined as those rows (except the diagonal elements, i.e., $g(i,i)$, in the rows) and those columns in two parent chromosomes that have the same $g(i,j) = 1$, $i \neq j$, as well as the column and the diagonal line in two parent chromosomes that have the same

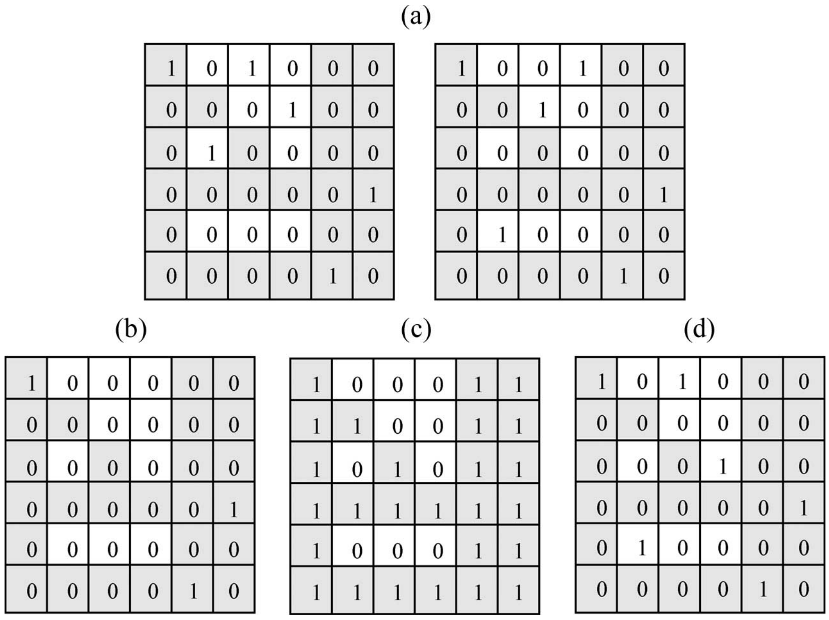

Fig. 5. Uniform crossover based on a binary representation. (a) Parent chromosomes. (b) "&" operator to identify common gene sections. (c) Indicate common gene sections. (d) Recombine noncommon genes.

$g(i,i) = 1$. In other words, if two parent chromosomes have the same $g(i,j) = 1$, $i \neq j$, then $g(m,j)$ and $g(i,m)$, except $g(i,i)$, for all $m = 1,\ldots,N_{\mathrm{AC}}(\cdot|k)$ are common genes; if two parent chromosomes have the same $g(i,i) = 1$, then $g(m,i)$ and $g(m,m)$ for all $m = 1,\ldots,N_{\mathrm{AC}}(\cdot|k)$ are common genes. Fig. 5(a) illustrates the common gene sections of two binary-representation-based chromosomes, where the common genes are grayed. To identify these common genes, all we need to do is to apply the "&" operation to two parents. Suppose

$$g_3(i,j) = g_1(i,j) \,\&\, g_2(i,j) \qquad i = 1,\ldots,N_{\mathrm{AC}}(\cdot|k); \quad j = 1,\ldots,N_{\mathrm{AC}}(\cdot|k) \tag{15}$$

where g_1 and g_2 are parent chromosomes, and g_3 is the result of the "&" operation. If $g_3(i,j) = 1$ for $i \neq j$, then set $g_4(m,j) = 1$ for $m = 1,\ldots,N_{\mathrm{AC}}(\cdot|k)$, and set $g_4(i,m) = 1$ for $m = 1,\ldots,N_{\mathrm{AC}}(\cdot|k)$, $m \neq i$. If $g_3(i,i) = 1$, then set $g_4(m,i) = 1$ and $g_4(m,m) = 1$ for $m = 1,\ldots,N_{\mathrm{AC}}(\cdot|k)$. $g_4(i,j) = 1$ means that this is a common gene. Fig. 5(b) is an example of the "&" operation. Based on g_3, we create g_4, where all common genes are set to 1 while all noncommon genes are set to 0, as illustrated in Fig. 5(c). With g_3 and g_4, we can efficiently recombine noncommon genes, as follows:

```
Step 1)   Let $g_5 = g_3$. If there is an $i$ with $g_3(i,i) = 1$, then go to Step 2. Otherwise, randomly choose an $i$ with $g_4(i,i) = 0$, let $g_5(i,i) = 1$, and set $g_4(m,i) = 1$ and $g_4(m,m) = 1$ for $m = 1,\ldots,N_{\mathrm{AC}}(\cdot|k)$.
Step 2)   While $g_4$ is not a unit matrix, do
  Step 2.1)   If $g_4(i,j) = 0$ for some values of $j$, then randomly choose such a $j$, let $g_5(i,j) = 1$, and set $g_4(m,j) = 1$ for $m = 1,\ldots,N_{\mathrm{AC}}(\cdot|k)$ and $g_4(i,m) = 1$ for $m = 1,\ldots,N_{\mathrm{AC}}(\cdot|k)$, $m \neq i$. Otherwise, find $j$ with $g_5(i,j) = 1$ and $j \neq i$.
  Step 2.2)   Let $i = j$.
```

Clearly, the aforementioned crossover procedure needs no constraints, such as given by (7)–(10), to guarantee the feasibility of the resulted chromosomes. All common gene sections are effectively and efficiently identified, inherited, and protected, and all possibilities of feasibly recombining noncommon genes can be exploited. Therefore, the aforementioned crossover can be considered as a uniform crossover operation. As will be proved in the case study, this uniform crossover is the most powerful searching technique in the proposed GA based on the binary representation.

D. Necessary Heuristic Rules

To further improve the performance of GA, four problem-specific heuristic rules are introduced.

1) To help the algorithm converge fast, not all of the new chromosomes are randomly initialized, but some are generated by the FCFS procedure. On the other hand, for the sake of diversity, in each generation, a certain proportion of worst chromosomes are replaced by totally new ones.
2) When randomly initializing a chromosome, we still follow the FCFS principle but in a loose way, i.e., the aircraft with earlier PLTs are more likely to be assigned to the front of an arrival sequence.
3) Similar to [22], in the mutation operation, if two successive aircraft are of the same category, then the one with an earlier PLT should land first. Otherwise, the order that leads to a smaller LTI should stand a better chance.
4) Like [22], the population in a generation, i.e., N_P, and the maximum number of generations in the evolutionary process, i.e., N_G, are adjusted according to $N_{\mathrm{AC}}(\cdot|k)$. Thus

$$N_P = 30 + 10\,(round\,(\max\,(0, N_{\mathrm{AC}}(\cdot|k) - 10)\,/5)) \tag{16}$$

$$N_G = 20 + 10\,(round\,(\max\,(0, N_{\mathrm{AC}}(\cdot|k) - 10)\,/5))\,. \tag{17}$$

TABLE II
RESULTS OF A SINGLE TEST

Initial traffic data			Scheduled by DTSPM				Scheduled by PRGA				Scheduled by BRGA2			
AC No.	Cat.	PLT (s)	AC No.	Cat.	ALT (s)	Delay (s)	AC No	Cat.	ALT (s)	Delay (s)	AC No.	Cat.	ALT (s)	Delay (s)
1	1	1935	9	4	35	0	9	4	35	0	9	4	35	0
2	3	400	5	3	142	0	5	3	142	0	5	3	142	0
3	4	879	10	1	307	0	10	1	307	0	10	1	307	0
4	1	328	4	1	403	75	4	1	403	75	4	1	403	75
5	3	142	19	1	499	5	19	1	499	5	19	1	499	5
6	2	1980	17	1	595	30	17	1	595	30	17	1	595	30
7	2	915	2	3	776	376	2	3	776	376	12	2	795	433
8	2	1814	18	3	846	181	18	3	846	181	18	3	865	200
9	4	35	12	2	946	584	12	2	946	584	2	3	935	535
10	1	307	3	4	1056	177	3	4	1056	177	7	2	1035	120
11	3	1414	15	4	1146	193	15	4	1146	193	3	4	1145	266
12	2	362	7	2	1226	311	7	2	1226	311	15	4	1235	282
13	4	1279	14	1	1298	318	13	4	1336	57	13	4	1325	46
14	1	980	13	4	1526	247	20	2	1416	8	20	2	1408	0
15	4	953	20	2	1606	198	14	1	1488	508	11	3	1478	64
16	3	1726	11	3	1676	262	11	3	1669	255	14	1	1550	570
17	1	565	16	3	1746	20	16	3	1739	13	16	3	1731	5
18	3	665	8	2	1846	32	8	2	1839	25	8	2	1831	17
19	1	494	6	2	1980	0	6	2	1980	0	6	2	1980	0
20	2	1408	1	1	2052	117	1	1	2052	117	1	1	2052	117
Total delay (s)					**3126**				**2915**				**2765**	

IV. Simulation Results

In this section, the proposed binary-representation-based GA is compared with the permutation-representation-based GA reported in [22], the importance of the uniform crossover operation is also studied, and some RHC-related parameters adopted in [22] are further investigated. For distinguishing purposes, the permutation-representation-based GA in [22] is denoted as PRGA hereafter, whereas the proposed binary-representation-based GA is denoted as BRGA. To study the uniform crossover operation, there are two versions of BRGA: 1) BRGA1, which has only the mutation operator; and 2) BRGA2, which has both the mutation and crossover operators. In the simulation, unless specifically indicated, the RHC-related parameters are set as follows: $N_H = 4$; $T_{\text{OI}} = 5$ min. Moreover, the method reported in [15], which is denoted as DTSPM since it is a deterministic method developed based on traveling salesman problem (TSP) modeling, is also used for comparative purposes.

Like [22] and [26], the initial traffic data used in the simulation are randomly generated by referring to the data used in [15]. Assuming that the total number of aircraft coming to land is 60, the operating day is 150 min long in the undercongested case, 100 min long in the normal case, and 50 min long in the congested case. In each case, we randomly generate 20 sets of initial traffic data. For each set of initial traffic data, we then introduce 20% uncertainties, i.e., at each time interval, 20% of aircraft will have new PLTs different from their old PLTs predicted in the last time interval. A new PLT may be up to 5 min earlier or later than the associated old PLT. For each set of original PLTs, 20 sets of uncertainty histories of each original PLT are randomly generated and then saved as traffic data files. In a single comparative test, the same traffic data file with a set of original PLTs and a set of uncertainty history is used to test all algorithms. In each test, one simulation run is conducted under the DTSPM, whereas 40 simulation runs are conducted under each GA, and then, the averages are taken with regard to the average airborne delay (AAD) and online computational time (OCT) consumed by each optimization routine in an operational interval. All tests are conducted in a MATLAB environment on a personal computer whose central processing unit is of 2.0 GHz. Table II gives an example of how the optimized arrival sequences look like under different methods, where due to limited space, the initial traffic data have only 20 aircraft arriving in 35 min, and the results under BRGA1 are not included.

TABLE III
SIMULATION RESULTS IN THE STATIC CASE

	Under-congested case		Normal case		Congested case	
(in second)	AAD	OCT	AAD	OCT	AAD	OCT
DTSPM	98.9	7.03	389.5	7.06	1326.6	7.05
PRGA	97.4	14.88	371.4	14.88	1319.2	14.94
BRGA1	95.3	16.52	348.4	16.45	1304.8	16.39
BRGA2	93.8	24.08	331.3	24.25	1294.1	24.61

A. Static Case

Many papers mainly tested their methods in the static case, where no uncertainties are present, and no iteration of optimization or implementation in each optional interval is required. Instead, based on the initial traffic data, the arrival sequence for the entire operating day is optimized once for all. Here, we also compare different methods in the static case, and the key results are given in Table III, which is based on 20 sets of initial traffic data and 40 simulation runs for each set under each GA with J_1 in (3) and one run for each set under the DTSPM.

In the static case, the following can be observed from Table III: 1) The DTSPM in [15] is a time-efficient algorithm to organize aircraft arrival sequences; 2) compared with DTSPM, properly designed GAs can achieve a smaller airborne delay through stochastic searching; 3) since considering 60 aircraft at one time means the GAs need to search a solution space with

TABLE IV
SIMULATION RESULTS IN THE DYNAMIC CASE (NO UNCERTAINTIES)

	Under-congested case		Normal case		Congested case	
(in second)	AAD	OCT	AAD	OCT	AAD	OCT
DTSPM	95.9	0.02	347.6	0.10	1260.1	2.00
PRGA	92.7	0.35	307.8	0.78	1260.0	5.32
BRGA1	92.4	0.39	306.3	0.84	1256.4	5.64
BRGA2	90.9	0.60	302.7	1.30	1253.0	8.23

TABLE V
SIMULATION RESULTS IN THE DYNAMIC CASE
(WITH 20% UNCERTAINTIES)

	Under-congested case		Normal case		Congested case	
(in second)	AAD	OCT	AAD	OCT	AAD	OCT
DTSPM	95.7	0.02	339.3	0.09	1257.9	1.91
PRGA	92.3	0.35	316.5	0.78	1247.5	5.23
BRGA1	92.8	0.39	306.7	0.83	1243.5	5.57
BRGA2	91.4	0.60	304.4	1.29	1239.4	8.16

a size of 60!, without an efficient crossover operator, PRGA only offers limited improvement in terms of airborne delay; 4) since all mutation operations in BRGA1 focus on successive aircraft, it achieves better performance than PRGA; 5) with an efficient uniform crossover operator, the performance of BRGA2 is further improved (on average, about 7.5% better than DTSPM, 5.4% better than PRGA, and 2.4% better than BRGA1); 6) roughly speaking, BRGA2 most significantly reduces the airborne delay in the normal case, which is probably because there is not much room to improve arriving sequences in the undercongested case, whereas the congested case is too complicated; and 7) the cost to pay for BRGA2 is the longest computational time, which is, however, tolerable for offline planning purposes.

B. Dynamic Case

The dynamic ASS problem is the focus of this paper. Here, all methods are integrated with the RHC strategy. More details on how to integrate DTSPM and PRGA with RHC can be found in [22] and [26]. The simulation in this section is designed to test the idea that the RHC strategy can automatically improve the performance of GAs for the ASS problem due to the following: 1) Using the receding horizon reduces the size of the solution space that the GAs need to search in each operational interval; and 2) as mentioned in [22] and [26], good subsequences for the beginning of the operating day usually benefit the later scheduling and sequencing procedure in the operating day. First, we test all RHC-based methods with the 20 sets of initial traffic data without applying uncertainties, and then, we introduce uncertainties to the initial traffic data. The simulation results are given in Tables IV and V.

In the dynamic case, the following can be observed from Tables IV and V: 1) Compared with the static case, using the RHC strategy improves the performance and largely reduces the computational time of all methods; 2) GAs outperform DTSPM; 3) as GAs without crossover, BRGA1 outperforms PRGA, which again shows that swapping two successive aircraft is more effective than the mutation operator in [22]; 4) BRGA2 is still the best method in terms of airborne delay (it is about 4.9%, 11.6%, and 1% better than the worst method (DTSPM) in the undercongested, normal, and congested cases, respectively); 5) however, the performance advantage of BRGA2 against other methods, particularly against other GAs, is not as significant as in the static case, which is probably because the RHC strategy largely reduces the solution space in each online optimization, therefore making it relatively easier for all methods to find good solutions; 6) for each method, the OCT increases as traffic becomes more congested (this is because, in a more congested case, more aircraft need to be considered in each operational interval); 7) even the OCT of BRGA2, which is the most time-consuming method, is ignorable when compared with the 5-min-long operational interval; and 8) PRGA and BRGA1 have the most significant improvement in performance when compared with their performance in the static case. The reason why the RHC strategy does not improve BRGA2 as much as it does to PRGA or BRGA1 is mainly because, due to the uniform crossover operator, BRGA2 is already a highly efficient algorithm, and therefore, the reduction in the size of the solution space does not have a significant influence on BRGA2.

C. RHC-Related Parameters

In this section, we investigate two RHC-related parameters in BRGA2 for the dynamic ASS problem with uncertainties: the steps of receding horizon N_H and the operational interval T_{OI}. The associated simulation results are given in Fig. 6. Basically, one can see from Fig. 6 that N_H should neither be too small, such as $N_H = 1$, to prevent shortsighted performance, nor should be too large, such as $N_H = 6$, to avoid an inefficient GA, which is too sensitive to uncertainties. When T_{OI} is reduced from 5 min (solid lines) to 2.5 min (dot-and-dashed lines), with a smaller T_{OI}, the performance of BRGA2 more smoothly changes. In the case of $T_{\mathrm{OI}} = 5$, the only good choice for N_H is $N_H = 2$; however, in the case of $T_{\mathrm{OI}} = 2.5$, we may set N_H between 2 and 5, which means a smaller T_{OI} offers more system flexibility. Due to the change of T_{OI}, the weight $\alpha(j)$ that is applicable to the same aircraft may become different according to (4), which means that different T_{OI} may result in different $J_2(\cdot|k)$ for the online optimization routine. Therefore, even for the same receding horizon length, i.e., the same value of $N_H \times T_{\mathrm{OI}}$, the performance measured by J_1 is different when T_{OI} changes. In the undercongested and normal cases, the smallest ADD is achieved when $T_{\mathrm{OI}} = 5$ and $N_H = 2$, whereas in the congested case, the smallest ADD comes when $T_{\mathrm{OI}} = 2.5$ and $N_H = 3$. This implies that at different congestion levels, we need to accordingly change T_{OI} and N_H to achieve the best traffic performance. However, how to adjust T_{OI} and N_H according to different traffic scenarios still demands more efforts and further study based on real traffic data. In Section IV-B, we set $T_{\mathrm{OI}} = 5$ and $N_H = 4$, because in the current air traffic control system, the ASS problem is normally concerned with arrival aircraft that are within a time window of approximately 20 min from the airport, and the frozen window for landing is about 5 min. Unfortunately, it turns out they are not the best choice, as Fig. 6 shows that a

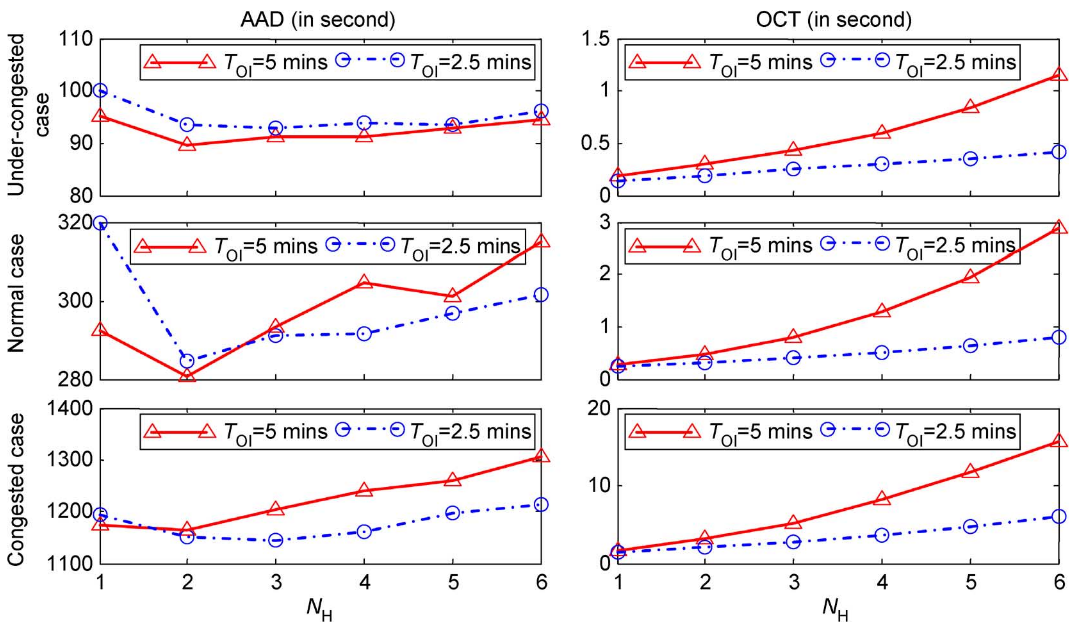

Fig. 6. Influence of N_H and T_{OI} on BRGA2.

relatively short horizon may work well. The potential reasons for this include the following: 1) A small delay for the leading aircraft often means a small delay for the following aircraft; in other words, as long as the landing sequence for the near future has small delays, it is very likely that the landing sequence for the far future will also have small delays. 2) Including too many aircraft that are in the far future usually has a small influence on the sequencing of those aircraft that are in the near future, because based on real-world experience, a good landing sequence only needs to shift the positions of those aircraft whose PLTs are similar, which means that those aircraft in the far future seldom affect the position shifting between aircraft in the near future. 3) Finally, a small horizon means fewer uncertainties and a smaller solution space for the GA to search. As a result, a small horizon may achieve quite good performance.

V. Conclusion

The arriving order and/or arriving time of each aircraft are often used to construct chromosomes in the design of GAs to tackle the aircraft ASS problem. These permutation representations make it difficult to design highly efficient evolutionary operators for GAs. This paper has used the neighboring relationship between each pair of aircraft to construct a chromosome as a 0-1-valued matrix, which makes it possible to easily adopt a uniform crossover operator. As a result, low-cost subqueues can be easily identified, effectively protected, and efficiently evolved. The strategy of RHC is also integrated into the proposed binary-representation-based GA. Comparative experiments prove the advantages of the new GA against permutation-representation-based GAs. Further research may include the following: 1) considering more realistic factors and constraints, such as deviations from PLTs and maximum allowable delay, to improve the model; 2) collecting and using real traffic data to assess the proposed method; and 3) extending the current work to the ASS problem in multirunway systems, as well as en route air traffic control.

References

[1] M. Pelegrin, *Towards global optimization for air traffic management*, 1994. AGARD-AG-321.

[2] J. K. Kuchar and L. C. Yang, “A review of conflict detection and resolution modeling methods,” *IEEE Trans. Intell. Transp. Syst.*, vol. 1, no. 4, pp. 179–189, Dec. 2000.

[3] B. Sridhar, G. Broto Chatterji, and S. Randall Grabbe, “Benefits of direct-to tool in national airspace system,” *IEEE Trans. Intell. Transp. Syst.*, vol. 1, no. 4, pp. 190–198, Dec. 2000.

[4] V. H. L. Cheng, V. Sharma, and D. C. Foyle, “A study of aircraft taxi performance for enhancing airport surface traffic control,” *IEEE Trans. Intell. Transp. Syst.*, vol. 2, no. 2, pp. 39–54, Jun. 2001.

[5] R. L. Hoffman and M. O. Ball, “The rate control index for traffic flow,” *IEEE Trans. Intell. Transp. Syst.*, vol. 2, no. 2, pp. 55–62, Jun. 2001.

[6] X.-B. Hu, W.-H. Chen, and E. Di Paolo, “Multi-airport capacity management: Genetic algorithm with receding horizon,” *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 2, pp. 254–263, Jun. 2007.

[7] C. C. Gregory, H. Erzberger, and F. Neuman, “Delay exchanges in arrival sequencing and scheduling,” *J. Aircr.*, vol. 36, no. 5, pp. 785–791, 1999.

[8] C. C. Gregory, H. Erzberger, and F. Neuman, “Fast-time study of aircraft-influenced arrival sequencing and scheduling,” *J. Guid. Control Dyn.*, vol. 23, no. 3, pp. 526–531, 2000.

[9] G. Andreatta and G. Romanin-Jacur, “Aircraft flow management under congestion,” *Transp. Sci.*, vol. 21, no. 4, pp. 249–253, Nov. 1987.

[10] L. Bianco and M. M. Bielli, “System aspects and optimization models in ATC planning,” in *Large Scale Computation and Information Processing in Air Traffic Control*, L. Bianco and A. R. Odoni, Eds. New York: Springer-Verlag, 1993, pp. 47–99.

[11] R. G. Dear, “The dynamic scheduling of aircraft in the near terminal area,” Mass. Inst. Technol., Cambridge, MA, Flight Transp. Lab. Rep. R76.9, 1976.

[12] H. N. Psaraftis, *A Dynamic Programming Approach to the Aircraft Sequencing Problem*. Cambridge, MA: MIT Press, 1978.

[13] H. N. Psaraftis, “A dynamic programming approach for sequencing groups of identical jobs,” *Oper. Res.*, vol. 28, no. 6, pp. 1347–1359, Nov./Dec. 1980.

[14] C. S. Venkatakrishnan, A. Barnett, and A. M. Odoni, “Landings at Logan Airport: Describing and increasing airport capacity,” *Transp. Sci.*, vol. 27, no. 3, pp. 211–227, 1993.

[15] L. Bianco, P. Dell’Olmo, and S. Giordani, “Scheduling models and algorithms for TMA traffic management,” in *Modelling and Simulation in Air Traffic Management*, L. Bianco, P. Dell’Olmo, and A. R. Odoni, Eds. New York: Springer-Verlag, 1997, pp. 139–167.

[16] L. Bianco, G. Rinaldi, and A. Sassano, "Scheduling tasks with sequence-dependent processing times," *Nav. Res. Logist.*, vol. 35, no. 2, pp. 177–184, 1988.

[17] J. E. Beasley, J. Sonander, and P. Havelock, "Scheduling aircraft landings at London Heathrow using a population heuristic," *J. Oper. Res. Soc.*, vol. 52, no. 5, pp. 483–493, May 2001.

[18] J. E. Robinson, III, T. J. Davis, and D. R. Issacson, "Fuzzy reasoning-based sequencing of arrival aircraft in the terminal area," in *Proc. AIAA Guid., Navig. Control Conf.*, New Orleans, LA, Aug. 1997.

[19] J. V. Hansen, "Genetic search methods in air traffic control," *Comput. Oper. Res.*, vol. 31, no. 3, pp. 445–459, Mar. 2004.

[20] A. Lecchini Visintini, W. Glover, J. Lygeros, and J. Maciejowski, "Monte Carlo optimization for conflict resolution in air traffic control," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 4, pp. 470–482, Dec. 2006.

[21] V. Cheng, L. Crawford, and P. Menon, "Air traffic control using genetic search techniques," in *Proc. IEEE Int. Conf. Control Appl.*, 1999, pp. 249–254.

[22] X. B. Hu and W. H. Chen, "Genetic algorithm based on receding horizon control for arrival sequencing and scheduling," *Eng. Appl. Artif. Intell.*, vol. 18, no. 5, pp. 633–642, Aug. 2005.

[23] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. of Michigan Press, 1975.

[24] A. E. Eiben and M. Schoenauer, "Evolutionary computing," *Inf. Process. Lett.*, vol. 82, no. 1, pp. 1–6, Apr. 2002.

[25] G. Sywerda, "Uniform crossover in genetic algorithms," in *Proc. 3rd Int. Conf. Genet. Algorithms*, 1989, pp. 2–9.

[26] X. B. Hu and W. H. Chen, "Receding horizon control for aircraft arrival sequencing and scheduling," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 2, pp. 189–197, Jun. 2005.

[27] D. W. Clarke, *Advances in Model-based Predictive Control*. Oxford, U.K.: Oxford Univ. Press, 1994.

Xiao-Bing Hu received the B.S. degree in aviation electronic engineering from the Civil Aviation Institute of China, Tianjin, China, in 1998, the M.S. degree in automatic control engineering from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2001, and the Ph.D. degree in aeronautical and automotive engineering from Loughborough University, Leicestershire, U.K., in 2005.

He is currently a Research Fellow with the Department of Informatics, University of Sussex, Brighton, U.K. His major field of research includes predictive control, artificial intelligence, air traffic management, and flight control.

Ezequiel Di Paolo received the M.S. degree in nuclear engineering from the Institute Balseiro, Bariloche, Argentina, and the Ph.D. degree from the University of Sussex, Brighton, U.K.

He is currently a Reader with the Evolutionary and Adaptive Systems Group, University of Sussex. He is also with the Centre for Research in Cognitive Science, University of Sussex. His research interests include adaptive behavior in natural and artificial systems, biological modeling, evolutionary robotics, and enactive cognitive science.