ARTICLE IN PRESS

Available online at www.sciencedirect.com

ScienceDirect

Computers & Operations Research ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

computers & operations research

www.elsevier.com/locate/cor

An efficient genetic algorithm with uniform crossover for air traffic control☆

Xiao-Bing Hu*, Ezequiel Di Paolo

Department of Informatics, University of Sussex, Falmer, Brighton, BN1 9QH, UK

Abstract

Aircraft arrival sequencing and scheduling (ASS) is a major issue in the daily air traffic control (ATC) operations. This paper reports on the application of genetic algorithms (GAs) to tackle the ASS problem in multi-runway systems. Most existing GAs for ASS are confronted with feasibility and efficiency problems in the design of their evolutionary operators, particularly the crossover. The new GA reported in this paper uses the following relationship between aircraft to construct chromosomes. This makes it possible to design a highly efficient crossover operator–uniform crossover, which is hardly applicable to those GAs designed directly based on the order of aircraft in arrival queues. The main benefit from the proposed uniform crossover operator is the effectiveness and efficiency in identifying, inheriting and protecting common sub-traffic-sequences without sacrificing the capability of diversifying chromosomes, which is demonstrated in the extensive comparative simulation study. By adopting the strategy of receding horizon control, the reported GA exhibits a good potential of real-time implementation in the ASS problem.

Keywords: Air traffic control; Scheduling and sequencing; Multi-runway systems; Genetic algorithm; Uniform crossover; Following relationship

1. Introduction

Of particular concern in air traffic control (ATC) are increased air traffic volume and aircraft delays. According to [1], world-wide air traffic is expected to grow to unexpected levels over the coming decades: revenue passenger miles worldwide of 1.7 trillion in 1996 are anticipated to reach 4.5 trillion by 2016. Existing demands on the air traffic system routinely exceed the capacity of airports leading to air-traffic-imposed ground and airborne delays of aircraft. In USA alone, this has been estimated to cost domestic airlines as much as $3.5 billion per year. In the increasingly competitive airline industry, with its market-driven pricing and very thin profit margins, such economic operating penalties are magnified [2].

Of primary importance in the efficient operation and profitability of an airline is adherence to its flight schedule. For large air carriers operating in hub and spoke networks, the maintenance of the carrier's flight schedule at its hubs drives the business efficiency of the entire flight network. In a hub and spoke network, many aircraft arrive at a central hub airport in rapid succession and depart to other spoke cities with very little time at the airport gate. This allows an airline to offer service to more cities with fewer airplanes [3]. Hub control of arriving groups or "banks" of aircraft,

☆ This work is supported by EPSRC Grant EP/C51632X/1.
* Corresponding author. Tel.: +44 1273 872606.
E-mail addresses: Xiaobing.Hu@sussex.ac.uk (X.-B. Hu), ezequiel@sussex.ac.uk (E. Di Paolo).

0305-0548/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2007.09.005

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005

and their subsequent turn-around in departure banks of aircraft is termed "bank management". Reliable and efficient bank management is critical to the success of a hub and spoke airline. Closely spaced arrivals and departures in hub and spoke networks are very sensitive to timing miscues, such as those caused by bad weather or airport congestion. Arrival time miscues lead to missed passenger and crew connections, inefficient ground operations caused by incorrect gate assignments, and occasional aircraft diversions to alternate airports. These cause passenger inconvenience, flight delays and lost airline avenue [2].

This paper examines the ATC segment termed as traffic management adviser (TMA), which is concerned with the complex task of scheduling arriving aircraft to the available runways in a manner that minimizes delays and satisfies safety constraints. Much of the recent research on ATC has been conducted by the NASA Ames Research Center (e.g., see [4–6]). From this research has come the Center-TRACON automation system (CTAS) to assist traffic management coordination in real time, with the task of planning and scheduling arrival traffic. A key component of CTAS is the TMA, which assists controllers with the scheduling of arrival aircraft that are within approximately 35–200 nautical miles from the airport. To help optimize the arrival times, the TMA compute sequence, arrival times, and runway assignments to ensure a smooth flow of traffic into the terminal area. In doing this, the TMA must take into account scheduling constraints that restrict the traffic flow or affect the required separation between aircraft.

In this paper, we call the above task of the TMA as arrival sequencing and scheduling (ASS). As is well known, the ASS problem is an NP-hard problem, which has no known algorithm for finding global optimal solution within a polynomial-bounded amount of time [7]. In the past decades many methods, such as travelling salesman problem modelling, dynamical programming, expert system, and evolutionary computing, have been used for tackling the ASS problem [7,8,16,18–27]. This paper aims to shed a little more light on how to design efficient genetic algorithms (GAs) for the ASS problem. As a large-scale parallel stochastic searching and optimizing algorithm, GAs are effective to solve NP-hard problems such as the ASS problem, e.g., see [1,9–11,17]. As is well known, how to construct chromosomes and how to design highly efficient evolutionary operators, i.e., mutation and crossover, are crucial to a successful implementation of GAs. Although GAs were originally proposed with binary chromosomes [28], it is difficult to use binary format to represent arriving queues in the ASS problem, particularly in multi-runway systems. Therefore, permutation representation based on the arriving order of aircraft dominates the chromosome structures used in existing research work on ASS [1,9–11]. Based on such a permutation representation, unfortunately, it is difficult to design a highly efficient crossover operator to handle arriving order or time based traffic sequences without cause any feasibility problem. As a result, most existing GAs for the ASS problem mainly rely on the mutation operation, and some of them even discard the crossover operation [1,10,11]. Some reported so-called crossover operators for the ASS problem actually play the role of mutation, e.g, see [1], because they can hardly serve the original purpose of crossover, i.e., identifying, inheriting and protecting good common genes (common sub-arrival-queues of high quality in the ASS case). This paper attempts to design an efficient GA with real crossover for the ASS problem in multi-runway systems. Particularly, we aim at uniform crossover, because it is probably the most powerful crossover allowing the offspring chromosomes to search all possibilities of re-combining those different genes in parents [13–15]. To this end, the following relationship between aircraft rather than the arriving order of aircraft is used to construct chromosomes, which makes it possible to develop a highlyefficient uniform crossover operator to inherit useful gene sections and evolve others.

The remainder of this paper is organized as follows. The ASS problem in multi-runway systems, denoted as MRASS hereafter, is described in Section 2. For comparative purposes, an existing GA, designed based on the arriving order of aircraft, for ASS in single-runway systems is extended to multi-runway systems in Section 3. The new GA with uniform crossover based on following relationship between aircraft is proposed in Section 4. Extensive simulation study is reported in Section 5, and the paper ends with some conclusions in Section 6.

2. MRASS problem formulation

Simply speaking, MRASS is the function of assigning arriving aircraft to different runways at the airport and generating efficient landing sequences and landing times for them so that the safety separation between arriving aircraft is guaranteed, the available capacity at the airport is efficiently used and airborne delays are significantly reduced. A simple way to perform MRASS is to allocate runways and schedule arriving aircraft in a first-come-first-served (FCFS) order based on planned landing times (PLTs) at the airport. Fig. 1 gives a simple demonstrating example of MRASS based on FCFS.

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005

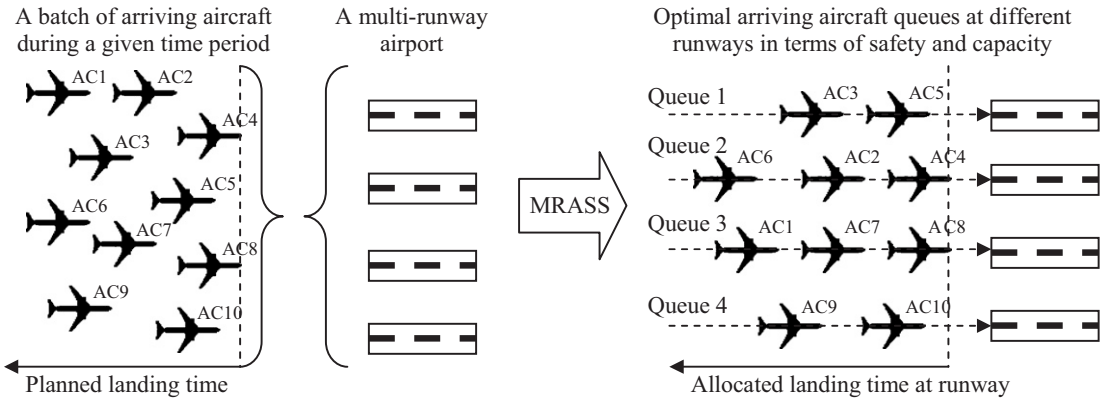

Fig. 1. Illustration of MRASS.

Table 1
Minimum landing time intervals (LTIs) [7]

S_{ij} (s)		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 = Boeing 747; 2 = Boeing 727; 3 = Boeing 707; 4 = Mc Donnel Douglas DC9.

Landing time interval (LTI), which is a minimum permissible time interval between two successive landings at a same runway, must be observed in the MRASS operation. Basically, LTI is a variable quantity, because (I) safety regulations state that any two co-altitudinal aircraft must maintain a "minimum horizontal separation", which is a function of the types and of the relative positions of the two aircraft, (II) the "landing speed" of a type of aircraft is generally different from that of another type of aircraft, and (III) different runway facilities and operational conditions may result in different LTIs even for a same pair of successive landings. Table 1 gives a simplified example of minimum LTIs related to some main categories of commercial aircraft.

It is evident that the LTIs in Table 1 are asymmetric. For example, a minimum LTI of 200 s is required for a Boeing 727 to follow a Boeing 747, while a minimum LTI of only 72 s needs to be satisfied for the same pair of aircraft in reverse order. Although FCFS strategy establishes a fair order based on PLTs, it is well known that, by taking advantage of the asymmetries of the LTIs, in other words, by shifting positions of aircraft in an FCFS landing sequence of each runway and swapping aircraft between runways, it is possible to reduce delays and to improve the capacity of the multi-runway airport.

Suppose N_{AC} aircraft are planned to land at an airport with N_R runways during a given time period T_H. Let C_i, R_i, P_i and A_i denote the category, the allocated runway, the PLT and the ALT (allocated landing time) of the ith aircraft in the original predicted arrival sequence, respectively. Q_r denotes the landing queue at runway r, $Q_r(j)$ is the jth aircraft in Q_r, $r = 1, \ldots, N_R$, $j = 1, \ldots, H_r$, and H_r is the number of aircraft in Q_r. $Q_r(j) = i$ means the ith aircraft in the original predicted arrival sequence turns out to be the jth aircraft in the optimized arrival sequence at runway r. With Q the ALTs of aircraft in the MRASS can be calculated as following:

$$A_{Q_r(j)} = \begin{cases} \max(P_{Q_r(j)}, T_O(r)), & j = 1, \\ \max(P_{Q_r(j)}, A_{Q_r(j-1)} & j > 1, \\ \quad +S(C_{Q_r(j-1)}, C_{Q_r(j)}, r), & \\ & j = 1, \ldots, H_r, \quad r = 1, \ldots, N_R, \end{cases} \tag{1}$$

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005

where $T_O(r)$ is the time when runway r starts to service, $S(c_1, c_2, r)$ is the LTI required when an aircraft of category c_2 follows an aircraft of category c_1 to land at the same runway r, and $S(., c_2, r) = \infty$ means aircraft of category c_2 cannot land at runway r. From Eq. (1), one can see that: the first aircraft assigned to a runway will land at its PLT if that runway is already open at that time; for a following aircraft, it can land at its PLT only if there is sufficient time separation between itself and its leading aircraft, i.e., the relevant LTI must be observed. The airborne delay of the ith aircraft in the original predicted arrival sequence is

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

In the real world, for each individual aircraft i, there is a maximum allowable airborne delay (MAAD) which needs to be observed in the ASS operation. For the sake of simplicity, we assume all aircraft have the same MAAD, denoted as $\overline{D}$, which should be treated as a hard constraint

$$D_i \leqslant \overline{D}, \quad i = 1, \ldots, N_{\mathrm{AC}}. \tag{3}$$

As discussed before, the MRASS aims to deliver safe, efficient, and punctual arrival traffic at airports. Since safety is guaranteed as long as LTIs are satisfied according to (1), the objective functions for MRASS need to reflect the requirements on the efficiency of airport operations and the punctuality in flight service. Basically, we can have two different objective functions defined as

$$J_1 = \sum_{i=1}^{N_{\mathrm{AC}}} D_i, \tag{4}$$

$$J_2 = \max(A_{Q_r(H_r)}), \quad r = 1, \ldots, N_{\mathrm{R}}, \tag{5}$$

where J_1 is the total airborne delay of all aircraft, and J_2 records the maximum length of all arrival queues. The total airborne delay J_1 emphasizes the operating cost of airlines, while the maximum length of all queues J_2 focuses more on the efficiency of using airport capacity. In many cases, a minimum total airborne delay occurs simultaneously along with a minimal maximum length of all queues, but this does not mean they are equivalent to each other. The relationship between J_1 and J_2 will be further investigated in the simulation section.

With the above mathematical preparation, the MRASS can be formulated as the following minimization problem:

$$\min_{Q_1,\ldots,Q_{N_{\mathrm{R}}}} J_1 \tag{6}$$

or

$$\min_{Q_1,\ldots,Q_{N_{\mathrm{R}}}} J_2 \tag{7}$$

subject to Eqs. (1)–(5). Clearly, how to assign arriving aircraft to different runways to form N_{R} arriving queues and how to organize the order of aircraft in each queue compose a solution, i.e., $Q_1, \ldots, Q_{N_{\mathrm{R}}}$, to this minimization problem. Different $Q_1, \ldots, Q_{N_{\mathrm{R}}}$ result in different values of J_1 or J_2. This paper attempts to develop an effective and efficient method to find the best or nearly best $Q_1, \ldots, Q_{N_{\mathrm{R}}}$ for the above MRASS problem.

3. GA for MRASS based on arriving order of aircraft

For comparative purposes, we discuss in this section a GA whose searching power mainly relies on mutation. This method actually originates from the GA proposed for the ASS problem in single-runway systems in [11], where chromosomes are constructed based on the arriving order of aircraft in queues, and mutation is the only way to evolve them. To extend the GA in [11] to multi-runway systems, the following modifications are required.

Firstly, the structure of chromosomes in multi-runway systems becomes a matrix with a dimension of $N_{\mathrm{R}} \times N_{\mathrm{AC}}$. A gene of the chromosome, denoted as $g(r, j)$, is associated with an entry of the matrix. $g(r, j) = i$ means aircraft i in the original arrival traffic is the jth aircraft to land at runway r, as illustrated in Fig. 2(b). This structure based on the arriving order of aircraft reveals the underlying physical meaning of chromosomes in a very straightforward way. The feasibility of chromosomes is defined by two constraints: (I) each aircraft appears once and only once in a chromosome, and (II) if $g(r, j) > 0$, then $g(r, h) > 0$ for all $1 \leqslant h < j$.

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005

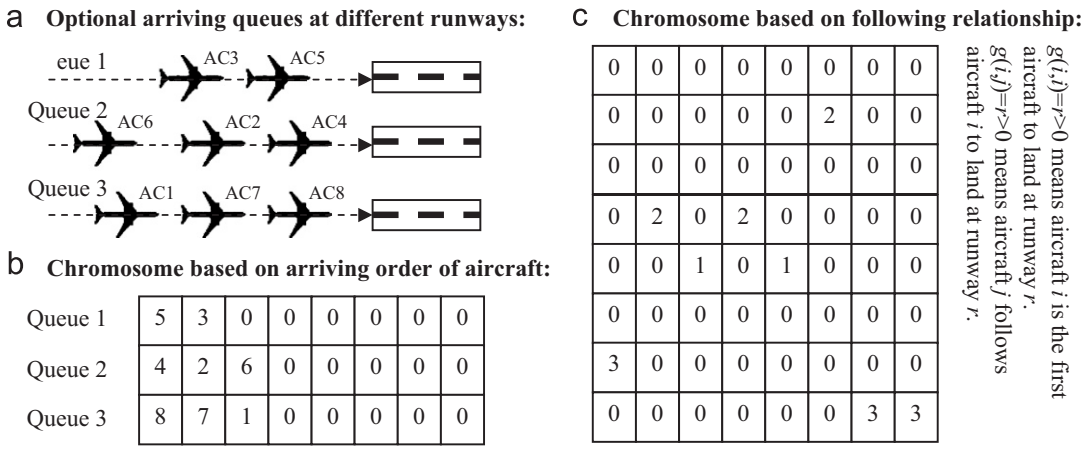

Fig. 2. Chromosome structures. (a) Optional arriving queues at different runways, (b) Chromosome based on arriving order of aircraft, (c) Chromosome based on following relationship.

With this chromosome structure, we have the following two mutation operators:

- Mutation I: Denoted as $g(r, j) \leftrightarrow g(r, j+1)$, $j = 1, \ldots, H_r - 1$, $r = 1, \ldots, N_R$, which means to shift randomly the positions of two successive aircraft in a same arriving queue.
- Mutation II: Denoted as $g(r_1, j) \leftrightarrow g(r_2, k)$, $j = 1, \ldots, H_{r_1}$ and $k = 1, \ldots, H_{r_2} + 1$, $r_1 \neq r_2$, $r_1 = 1, \ldots, N_R$, $r_2 = 1, \ldots, N_R$, which means to swap randomly aircraft in two different queues, or to remove an aircraft from one queue, and then append it to the end of another queue.

These two mutation operators together enable the GA to search the whole solution space of the MRASS problem, and have no feasibility problem, i.e., as long as the feasibility constraints are satisfied before the mutation, so are they after the mutation.

However, the chromosome structure based on the arriving order of aircraft makes it difficult to design a highly efficient crossover operator to identify, to inherit and to protect common genes shared by chromosomes. First of all, it is obviously very restrictive to define common genes as those aircraft which are assigned to the same runway with the same arriving order, i.e., to define common genes according to the absolute position of aircraft in arriving queues. As discussed in Section 2, the key factor in the ASS problem is the LTIs, which actually rely on the relative position between aircraft. Therefore it will make more sense if we define common genes according to the relative position between aircraft, such as the following relationship between aircraft, rather than to the absolute position of aircraft in arriving queues. Fig. 3 gives an illustration about these two definitions of common genes. Unfortunately, with either definition of common genes, we find it difficult to design an effective crossover operator for the arriving order based chromosome structure. The main problem during crossover is frequently losing the feasibility of chromosomes. Another problem for the arriving order based chromosome structure is: it is a computationally expensive task for computer to identify those common genes of the second definition. These may explain why most existing GAs for the ASS problem use no crossover operator.

For comparative purposes, here we still manage to design a crossover operator for the arriving order based chromosome structure:

$$\text{If} \quad \{g_1(1, j), \ldots, g_1(N_R, j)\} = \{g_2(1, k), \ldots, g_2(N_R, k)\} \neq \{0, \ldots, 0\}$$
$$\text{Then} \quad g_1(r, j) \leftrightarrow g_2(r, k) \quad \text{for all } r = 1, \ldots, N_R. \tag{8}$$

Eq. (8) requires the set of all jth aircraft in chromosome 1 is the same as the set of all kth aircraft in Chromosome 2. This crossover does not often causes feasibility problem. Actually, it has no feasibility problem if the following

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005

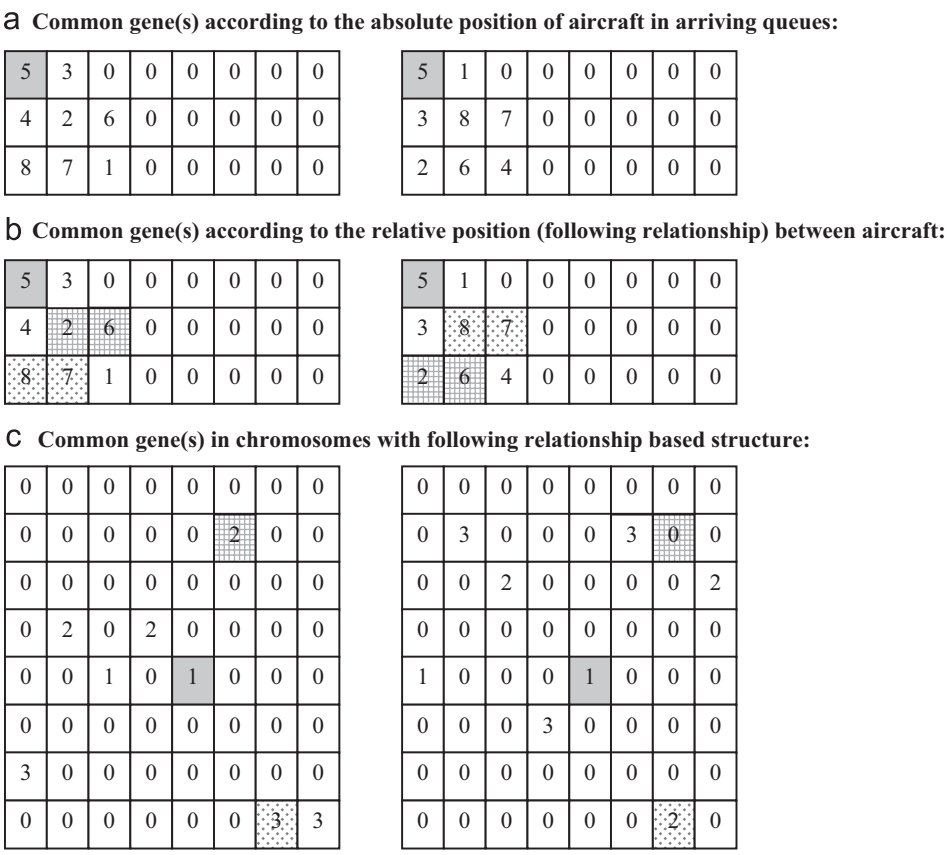

Fig. 3. Two definitions of common genes in MRASS problem. (a) Common gene(s) according to the absolute position of aircraft in arriving queues, (b) Common gene(s) according to the relative position (following relationship) between aircraft, (c) Common gene(s) in chromosomes with following relationship based structure.

constraint is added

$$g_1(r, j) > 0, \quad g_2(r, k) > 0 \quad \text{for all } r = 1, \ldots, N_R. \tag{9}$$

However, what this crossover operator does is not what a crossover operator is expected to do, supposing we want to identify, inherit and protect those common genes defined by the following relationship between aircraft. Actually the above crossover operator can be considered as a combination of Mutation I and II.

4. New GA with uniform crossover

There has long been a debate about the usefulness of crossover in the area of evolutionary computing [12]. The negative opinion in the debate is accidentally backed by those existing implementations of GAs to the ASS problem. However, the underlying idea of crossover is still sound from evolutionary point of view: pick out two parent chromosomes and exchange their gene sections randomly, so that the offspring chromosomes can inherit those common genes in the parents and at the same time search new possibilities of re-combining those different genes in the parents. Normally, the 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 is probably the most powerful crossover because it allows the offspring chromosomes to search all possibilities of re-combining those different genes in parents [13–15]. Since its popularization by [13], uniform crossover has become perhaps the most widely used crossover operator [15]. It is particularly effective and efficient to apply uniform crossover

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005

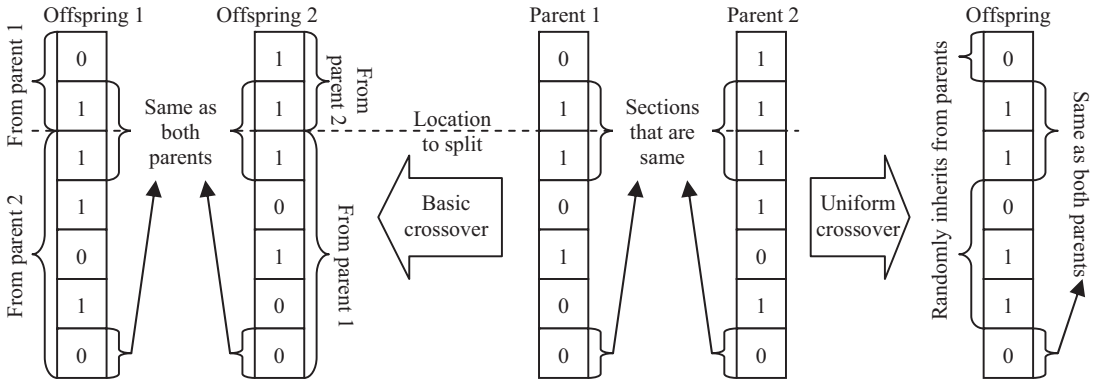

Fig. 4. Uniform crossover vs basic crossover (one-point crossover).

in those GAs based on binary representation, as illustrated by Fig. 4. However, it is difficult to use binary format to represent arriving queues in the ASS problem, particularly in multi-runway systems, which largely restricts the implementation of uniform crossover in the ASS problem. This paper is probably the first attempt to design an efficient GA with uniform crossover for the MRASS problem. In this section, the structure of those chromosomes is based on the following relationship between aircraft rather than the arriving order of aircraft in queues, and with the new chromosome structure, a real uniform crossover operator is designed, whose effect to evolve chromosomes is not achievable by mutation.

4.1. New chromosome structure

Our new constructed chromosome is a matrix with the dimension of $N_{AC} \times N_{AC}$. Each entry of the matrix, i.e., a gene in the chromosome, records the following relationship between aircraft and the runway assignment information. Gene $g(i,i) = r > 0$ means aircraft i is the first aircraft assigned to land at runway r, while $g(i,j) = r > 0$, $i \neq j$, means aircraft j follows aircraft i to land at runway r. Otherwise, $g(i,i) = 0$ means aircraft i follows other aircraft, and $g(i,j) = 0$ means aircraft j does not follow aircraft i. According to the underlying physical meaning of chromosomes in the MRASS problem, a feasible chromosome must satisfy the following constraints:

$$\sum_{i=1}^{N_{AC}} \sum_{j=1}^{N_{AC}} ceil(g(i,j)/N_R) = N_{AC}, \tag{10}$$

$$1 < \sum_{i=1}^{N_{AC}} ceil(g(i,i)/N_R) \leqslant N_R \quad \text{and} \quad g(i,i) \neq g(j,j) \quad \text{for } g(i,i) > 0,\ g(j,j) > 0, \tag{11}$$

$$\sum_{j=1}^{N_{AC}} ceil(g(i,j)/N_R) \begin{cases} \leqslant 2, & g(i,i) > 0 \\ \leqslant 1, & g(i,i) = 0 \end{cases}, \tag{12}$$

$$\sum_{i=1}^{N_{AC}} ceil(g(i,j)/N_R) = 1, \tag{13}$$

where "ceil" is a function to round a number to the nearest integer towards infinite. The above constraints are actually a new version of the two feasibility constrains discussed in Section 3, but this time in the format of the following relationship between aircraft. Eq. (10) guarantees there are N_{AC} following relationships (including those first landings) recorded in a chromosome, Eq. (11) means each runway should have no more than one first landing, Eq. (12) means

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005

each aircraft has no more than one following aircraft, and Eq. (13) requires each aircraft must follow a certain aircraft or land as the first at a runway. From Eqs. (10)–(12), one can derive that there may often be some empty rows, no more than N_R empty rows, in the matrix. If the ith row is empty, then it means aircraft i is the last aircraft in the associated arriving queue. If $g(i, i)$ is the only entry larger than 0 in row i, then aircraft i is the only one assigned to the associated runway. Fig. 2(c) gives an example of the new chromosome structure.

Actually the constraints given by Eqs. (10)–(13) will rarely be used in the new GA with uniform crossover. In the initialization of a chromosome, the following procedure can efficiently generate a feasible chromosome only with a need to check against the constraint given by Eq. (11):

Step 1: Create a $N_{AC} \times N_{AC}$ matrix with all entries set as 0. Let $U = \{1, \ldots, N_{AC}\}$ represent the original arrival traffic, and let $R = \{1, \ldots, N_R\}$ be the set of runways with no aircraft assigned to land.

Step 2: While $U \neq \emptyset$, do

Step 2.1: If one more new $g(i, i) > 0$ will violate Constraint (11), then randomly choose an existing $g(i, i) > 0$, let $r = g(i, i)$, and go to Step 2.2. Otherwise, choose $i \in U$ and $r \in R$ randomly, set $g(i, i) = r$, and remove i from U and r from R, i.e., let $U = U - \{i\}$ and $R = R - \{r\}$.

Step 2.2: Regardless of $g(i, i)$, if there is no entry in row i larger than 0, then randomly choose $j \in U$, set $g(i, j) = r$, and remove j from U, i.e., let $U = U - \{j\}$. Otherwise, find the $g(i, j) > 0$, let $i = j$, and repeat Step 2.2.

4.2. Mutation operator

The two mutation operators given in Section 3 are effective and efficient to GAs for the MRASS problem. In this section, we need to re-design them as the following in order to fit in the new chromosome structure:

- Mutation III: Randomly choose a non-zero gene, say, $g(i,j) = r > 0$. If there exist a $g(j, m) = r$ (and maybe further a $g(m, h) = r$), then change the values of some genes by following the instructions given in Fig. 5(a).
- Mutation IV: Randomly choose two non-zero genes, say, $g(i, j) = r_1 > 0$ and $g(h, x) = r_2 > 0$ (there may be $g(j, m) = r_1 > 0$ and/or $g(x, y) = r_2 > 0$). Then change the values of some genes by following the instructions given in Fig. 5(b).

The mutation procedures given in Fig. 5 automatically guarantee the feasibility of resulted chromosomes as long as the original chromosomes are feasible.

4.3. Uniform crossover operator

As discussed in Section 3, the relative position between aircraft, i.e., the following relationship between aircraft is the most important in the ASS problem. A same following relationship means a same sub-queue in arrival traffic, no matter when to land at which runway. How to design a highly efficient uniform crossover operator, free of feasibility problem, to effectively identify, inherit and protect valuable common sub-queues, and at the same time to fully exploit the possibility of recombining other non-common sub-queues is the emphasis of this sub-section. First of all, we need to know how those common sub-queues are represented in the new chromosome structure. Fig. 3(c) gives an illustration. Thanks to directly using the following relationship between aircraft to construct chromosomes, common sub-queues appear as non-zero entries having same indexes in the matrix, i.e., if $g_1(i, j) > 0$ and $g_2(i, j) > 0$, then we know aircraft i followed by aircraft j is a common sub-queue in the two MRASS solutions represented by the chromosome g_1 and g_2. According to the constraints given by Eqs. (10)–(13), if $g_1(i, j) > 0$, those entries in the appropriate row, column and/or the diagonal line must be zero. Therefore, in the new chromosome structure, the common genes related to a common sub-queue actually include both the specific non-zero $g(i, j)$ and the associated 0-valued entries. Bearing this new definition of common genes in mind, the new crossover operation is described as the following procedure, which is further illustrated by Fig. 6:

Step 1: Identify common sub-queues. Given two parent chromosomes g_1 and g_2, calculate g_3 by applying the '&' operation

$$g_3(i, j) = g_1(i, j) \& g_2(i, j), \quad i = 1, \ldots, N_{AC}, \quad j = 1, \ldots, N_{AC}. \tag{14}$$

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005

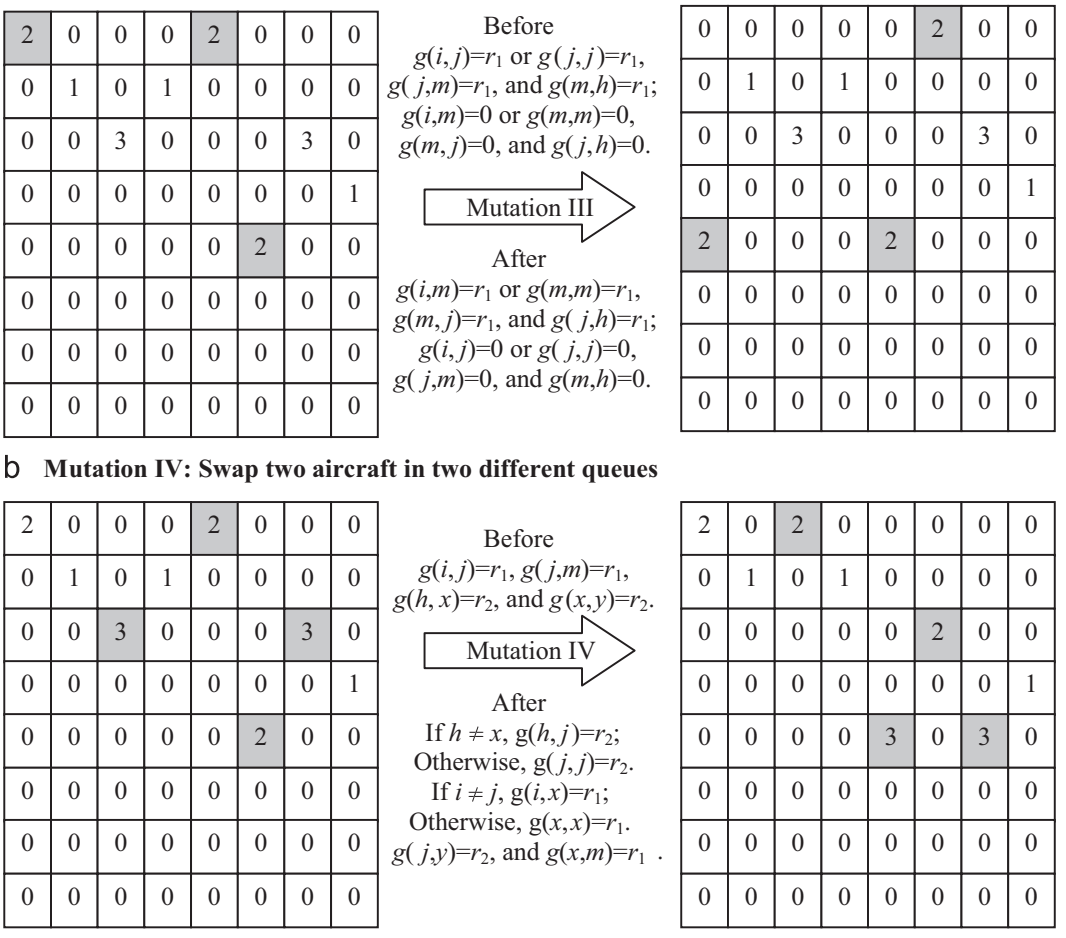

Fig. 5. Mutation in new GA for ASS in multi-runway systems, (a) Mutation III: Shift two successive aircraft in the same queue, (b) Mutation IV: Swap two aircraft in two different queues.

Step 2: Locate common genes based on g_3, i.e., if $g_3(i, j) = 1$ for $i \neq j$, then set $g_4(m, j) = 1$ for $m = 1, \ldots, N_{AC}$, and set $g_4(i, m) = 1$ for $m = 1, \ldots, N_{AC}$, $m \neq i$; if $g_3(i, i) = 1$, then set $g_4(m, i) = 1$ for $m = 1, \ldots, N_{AC}$; if $\Sigma g_3(i, i) = N_R$, then set $g_4(i, i) = 1$ for $i = 1, \ldots, N_{AC}$. $g_4(i, j) = 1$ means this location has a common gene.

Step 3: Create an embryo of offspring chromosome. Firstly, let $g_5 = g_3$, i.e., inherit common genes. Secondly, recombine non-common genes as the following:

Step 3.1: If $\Sigma g_5(i, i) = N_R$, then set $g_4(i, i) = 1$ for $i = 1, \ldots, N_{AC}$, and go to Step 3.2. Otherwise, randomly choose an i with $g_4(i, i) = 0$, let $g_5(i, i) = 1$, set $g_4(m, i) = 1$ for $m = 1, \ldots, N_{AC}$, and repeat Step 3.1.

Step 3.2: If g_4 is a unit matrix, go to Step 4. Otherwise, randomly choose an i with $g_5(i, i) = 1$.

Step 3.3: If $g_4(i, j) = 0$ for some values of j, then randomly choose such a j, let $g_5(i, j) = 1$, set $g_4(m, j) = 1$ for $m = 1, \ldots, N_{AC}$ and $g_4(i, m) = 1$ for $m = 1, \ldots, N_{AC}$, $m \neq i$, and go to Step 3.2. Otherwise, find out the j with $g_5(i, j) = 1$ and $j \neq i$, let $i = j$, and repeat Step 3.3.

Step 4: Let $R = \{1, \ldots, N_R\}$ be a set of runways, then finalize the offspring chromosome:

Step 4.1: If $g_3(i, i) = 1$ for some values of i, then randomly choose such an i, set $g_5(i, i)$ as $g_1(i, i)$ or $g_2(i, i)$ at an half-and-half chance, if, supposing now $g_5(i, i) = r$, $r \notin R$, then choose randomly a new $r \in R$, let $g_5(i, i) = r$, and set $g_3(i, i) = -1$. Otherwise, choose randomly an i with $g_3(i, i) = 0$ and $g_5(i, i) = 1$, let $g_5(i, i) = r$, $r \in R$, and set $g_3(i, i) = -1$.

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005

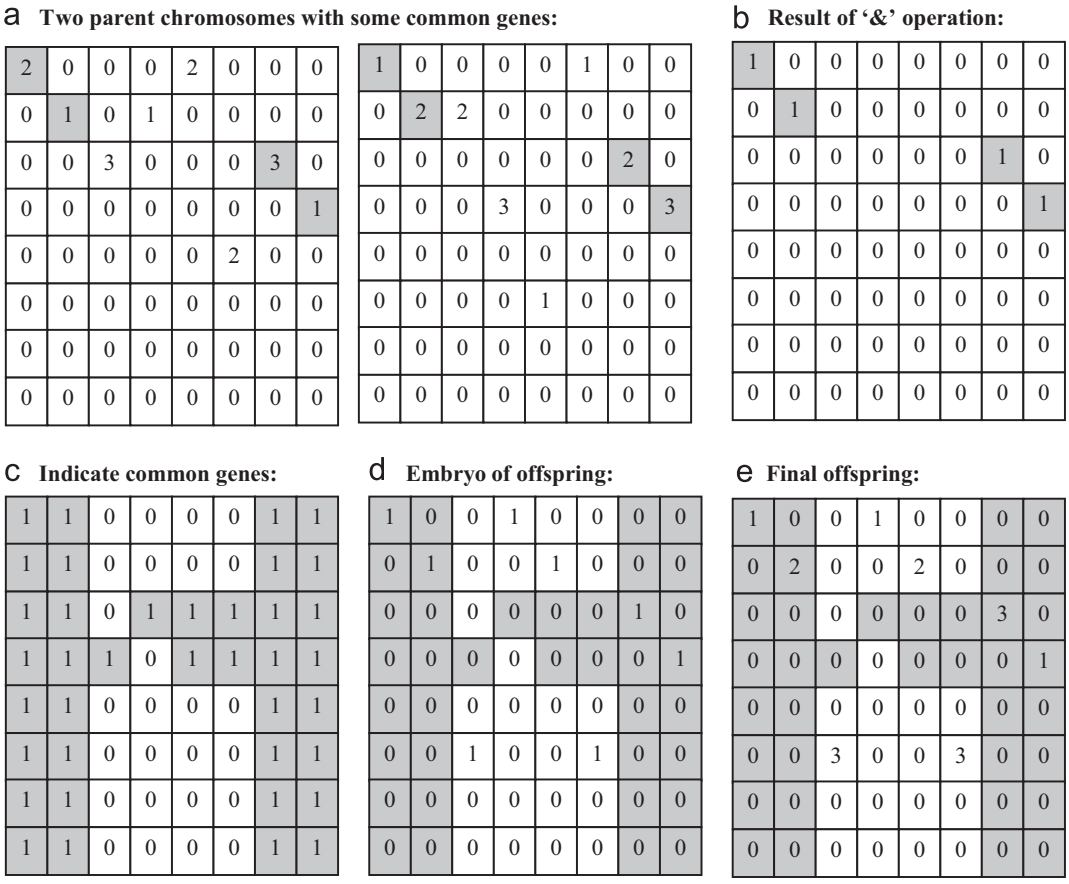

Fig. 6. Uniform crossover in GA for ASS in multi-runway systems. (a) Two parent chromosomes with some common genes, (b) Result of '&' operation, (c) Indicate common genes, (d) Embryo of offspring, (e) Final offspring.

Step 4.2: If there exists a j such that $g_5(i, j) = 1$ and $j \neq i$, then set $g_5(i, j) = r$, let $i = j$, and repeat Step 4.2. Otherwise, let $R = R - \{r\}$, i.e., remove the current runway from the set R.

Step 4.3: If R$\neq \emptyset$, go to Step 4.1.

Regarding the feasibility of resulted chromosomes, the above crossover procedure needs to check no constraints such as given by Eqs. (10)–(13), mainly because of the usage of g_4 to track those 0-valued entries which can not be re-valued in g_3 and g_5. For instance, if $g_5(i, j) = 1$ and $j \neq i$, then constraints (12) and (13) require all 0-valued entries in row i and column j remain zero. g_4 recodes all these unchangeable 0-valued entries in g_5 and therefore guarantees constraints (12) and (13) will never be invaded.

In the above crossover procedure, all common genes are effectively and efficiently identified, inherited and protected, and all possibilities of feasibly re-combining non-common genes can be exploited. Therefore, this crossover can be considered as a uniform crossover operation. As will be proved in the simulation study, this uniform crossover is a most powerful searching technique in the proposed GA.

5. Simulation results

The simulation study in this section is used to test the new GA with uniform crossover by comparing with the Method 4 in [1] and the GA given in Section 3, which is an extended version of the GA for single-runway systems in [11]. For distinguishing purposes, the GA without and with crossover in Section 3 are denoted as GA1 and GA2, and the new GA without and with uniform crossover in Section 4 as GA3 and GA4, respectively. The emphasis is put on the

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005

Table 2
Scenario 1 in [1] (delays in time unit)

Initial traffic data			Method 4 in [1]			GA4		
AC No.	Cat.	PLT	Runway No.	ALT	Delay	Runway No.	ALT	Delay
DL130	Heavy	10	3	10	0	3	10	0
AA335	Small	15	1	15	0	1	15	0
UA123	Heavy	7	1	7	0	1	7	0
DL1920	Heavy	6	1	6	0	3	6	0
UA1133	Large	10	1	10.5	0.5	2	10	0
NW2123	Heavy	5	3	5	0	3	5	0
AA205	Large	15	1	16	1	2	15	0
DL3319	Heavy	7	1	8	1	3	7	0
SW200	Small	6	2	7	1	2	7	1
DL510	Heavy	9	1	9	0	1	9	0
UA410	Heavy	4	3	4	0	3	4	0
SW185	Large	6	3	6.5	0.5	1	6	0
	TAD			4			1	

Table 3
Summary of test cases in [1] (delays in time unit)

Case	N_{AC}	N_R	Runway restriction	TAD under Method 4 in [1]	TAD under GA4
Scenario 1 in [1]	12	3	–	3.5	1.00
Scenario 2 in [1]	15	3	–	9	5.50
Scenario 3 in [1]	20	5	–	12	7.65
Scenario 4 in [1]	12	3	No heavy aircraft to Runway 3	8.5	4.30

role of the new uniform crossover operator, the relationship between J_1 and J_2 is also investigated, and the strategy of Receding Horizon Control (RHC) is used to improve the real-time properties of the new GA.

5.1. Case study

The test cases in [1] are adopted and modified for comparative purposes. In those test cases, an aircraft has a specific PLT to each runway, in other words, each aircraft has totally N_R PLTs in a system with N_R runways. However, the delay is calculated by comparing the ALT with the earliest PLT, no matter which runway is assigned. This is because of the fact that airline companies only publish one PLT for each flight. Therefore, in our study, an aircraft has only one PLT to the airport, i.e., the PLT expected by airline companies and passengers. To be able to adopt the test cases in [1], we assume the earliest PLT in [1] is the PLT published by airline companies. We also assume a runway is not opened until a specific time instant. For instance, Runway 2 is not opened until the 7th time unit, which can explain why the first aircraft landing at Runway 2 still has 1 time unit delay in [1]. Since few details in the design of GAs are given in [1], we cannot repeat exactly the methods reported in [1]. Therefore, we quote directly the best results achieved by Method 4 in [1], i.e. the best method in [1], and then compare them with the results under our GA4. Readers should be aware that the LTIs used in [1] are totally different from those given in Table 1, which is adopted in the following Monte Carlo simulations. Actually, the LTIs used in [1] are based on three categories of aircraft, and therefore its complexity is relatively less than that defined by Table 1.

To illustrate how Method 4 in [1] and our GA4 optimize arrival traffic, Table 2 gives the test results of Scenario 1 in [1], where 12 aircraft arrive at a 3-runway system. In this case study, our GA4 outperforms Method 4 in [1] in terms of total airborne delay (TAD). Due to the limited space, here we only summarize the test results of all four scenarios in [1]. Table 3 lists all shortest delays under Method 4 in [1] as well as the average results of 10 runs of GA4 in each scenario. The advantage of our new algorithm with uniform crossover against the Method 4 in [1] is clearly demonstrated.

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005

Table 4
Simulation scenarios

Scenario	S1	S2	S3	S4	S5
N_R	1	2	3	4	4
Runway restriction	–	–	–	–	Category 1 restricted to Runway 4

Table 5
Results of Monte Carlo simulations

(Average results)	GA1		GA2		GA3		GA4	
	AAD (s)	CT (s)	AAD (s)	CT (s)	AAD (s)	CT (s)	AAD (s)	CT (s)
S1	1319.3	29.5	1317.0	38.3	1308.1	31.3	1300.7	67.5
S2	177.6	22.0	169.4	30.1	163.9	22.9	160.2	49.6
S3	29.1	20.2	26.4	28.1	26.2	21.8	26.0	47.4
S4	7.9	19.4	7.2	27.3	7.3	21.5	6.9	46.5
S5	7.1	19.4	6.9	27.5	6.8	21.5	6.6	46.6

5.2. Monte Carlo simulations

To get general conclusions about GA4, we further conduct Monte Carlo simulations to compare it with GA1, GA2 and GA3. In the Monte Carlo simulations, the LTIs defined by Table 1 are used. Like in [11], the initial traffic data used in the simulation are randomly generated by referring to the data used in [7]. Suppose 60 aircraft arrive in 50 min. Table 4 defines five simulation scenarios by changing the total number of runways and restrictions. Obviously, the less runways, the more congested. For each scenario, we randomly generate 100 sets of initial traffic data. In each test, only one set of initial traffic data is used, 40 simulation runs are conducted under each GA, and then the averages are taken regarding average airborne delay (AAD) and computational time (CT) consumed by each GA. The results of Monte Carlo simulations are given in Table 5, from which one can make the following observations:

- In terms of airborne delay, GA4 achieves the best performance, which is about 10% better than the performance of GA1. The computational burden of GA4 is the heaviest, but it is within a reasonable and acceptable range when compared with the 50 min long time period under consideration. Using special software and hardware can further improve the computational efficiency of the new proposed GA for the MRASS problem.
- GAs with crossover operators, i.e., GA4 and GA2, outperform those without, i.e., GA3 and GA1. This illustrates the important role of crossover in the design of GAs for the MRASS problem.
- Basically, GA3 is better than GA2, which proves the crossover operator in Section 3 is actually no more than a special combination of the mutation operators. Therefore, with a more efficient representation based on following relationship between aircraft, GA3 can achieve better performance by only using the mutation operators.
- In less congested situation, i.e., with more runways, the improvement in performance achieved by GA4 is more significant than that in more congested situation. For instance, in S4, GA4 is averagely 7.4% better than the other three GAs, while in S1, GA4 is just about 1.1% better. This is probably because more congested situation means a higher degree of complexity. Since all GAs have a maximum number of generations allowed to evolve in the optimization, in an extremely complicated MRASS problem, all GAs are probably terminated before they can reach any near-optimal solutions. Therefore, more efforts are still required to further improve the GA reported in this paper. Fortunately, an extremely congested situation like in S1 rarely happens [7].
- Theoretically, for the same original traffic data and the same N_R, the optimal arrival queues without runway restrictions should have less airborne delays than those with certain runway restrictions. However, the results of S4 and S5 are against this expectation. Further analysis reveals that, due to the runway restriction in S5, it is more likely that all GAs assign aircraft of category 1 to a same runway, which turns out accidentally to be a useful rule to improve arrival queues in the MRASS problem. In S4, where there is no runway restriction, it is relatively less likely for GAs to

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005

Table 6
Relationship between J_1 and J_2

	GA4 with J_1			GA4 with J_2		
	J_1 (s)	J_2 (s)	CT (s)	J_1 (s)	J_2 (s)	CT (s)
S1	78041	5507.1	67.5	72654	5281.7	67.6
S2	9609.6	3144.1	49.6	11475	3120.2	49.9
S3	1559.2	2947.4	47.4	2682.9	2937.9	47.3
S4	412.8	2926.8	46.5	511.8	2925.7	46.0
S5	398.1	2925.7	46.6	455.1	2925.7	46.2

assign aircraft of category 1 to a same runway, because GAs belong to the family of stochastic searching algorithms. This implies that good problem-specific heuristic rules are important to the design of GAs, and therefore deserve further investigation.

5.3. Relationship between J_1 and J_2

As discussed in Section 2, the MRASS problem may have two different objective functions: J_1 in Eq. (4) and J_2 in Eq. (5). J_1 is the total airborne delay of all aircraft, which emphasizes the operating cost of airlines and is widely used in existing literature on ASS. However, can J_1 reflect the efficiency of using airport capacity? This sub-section tries to answer this question by investigating the relationship between J_1 and J_2, which records the minimal maximum length of all arrival queues. Firstly, we apply GA4 with J_1 to all 5 simulation scenarios, and then calculate J_2 based on the results. After that, we repeat all tests by applying GA4 with J_2, and then calculate J_1 accordingly. The average results are given in Table 6, where one can see clearly:

- Basically, when J_1 (or J_2) is used as the objective function of GA4, the values of J_1 (or J_2) are smaller than those with J_2 (or J_1) as the objective function. The minimal values of J_1 and J_2 are rarely achieved simultaneously, which means J_1 and J_2 are not equal to each other.
- When J_1 is used as the objective function, the values of J_2 are quite close to those achieved by using J_2 as the objective function, while considering the values of J_1, the choice of objective function makes relatively bigger difference. This may somehow support the fact that J_1 is widely used in the study on the ASS problem. However, as discussed before, minimizing J_2 means to accept a given number of aircraft to land within a shortest time period, which is directly related to the capacity of airport. Therefore, J_2 should not be completely ignored in the study on the ASS problem. Further research should be conducted to investigate how to properly combine J_1 and J_2.
- In S1, GA4 with J_1 as the objective function gives unexpectedly worse performance in terms of either J_1 or J_2. This is probably, again, because GAs are stochastic searching algorithms and therefore can never guarantee optimal or even near-optimal solutions.

5.4. Use receding horizon control strategy

In the real dynamical ATC environment, the presence of uncertainties and disturbances (such as flight speed and runway accidents) requires on-line adjustments to the arrival sequence. This is often a big challenge to GAs because they are always criticized for the time-consuming evolutionary process. Fortunately, as reported in [11], the strategy of receding horizon control (RHC) can make it realistic to extend GAs to real-time implementations. Basically, 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, e.g., delays in the ATC case, 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 into account of updated information and looking ahead for another N steps. Ref. [11] fully investigated how to integrate the RHC strategy into GAs for the ASS problem in a single runway system. Those RHC related techniques discussed in [11] can easily apply to GA4, which is specially designed for the MRASS problem. Here, due to the limited space, we skip the details on

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005

Table 7
GA4 with RHC strategy

N	S1		S2		S3		S4		S5	
	AAD (s)	CT (s)	AAD (s)	CT (s)	AAD (s)	CT (s)	AAD (s)	CT (s)	AAD (s)	CT (s)
1	1308.6	4.5	158.9	3.9	25.7	4.0	6.7	3.8	6.6	3.9
2	1261.1	6.8	153.2	5.0	25.3	5.1	6.6	4.9	6.5	5.0
3	1259.5	9.3	153.5	6.4	25.2	6.7	6.6	6.7	6.6	6.5
4	1263.9	12.5	154.1	8.3	25.4	8.1	6.7	7.9	6.6	8.2
6	1285.2	22.9	155.8	15.5	25.7	16.7	6.9	15.8	6.7	15.4
10	1300.7	67.5	160.2	49.6	26.0	47.4	6.9	46.5	6.6	46.6

how to integrate RHC into GA4, and simply give some simulation results studying the influence of horizon length on the performance and computational time of GA4. Table 7 summarizes the Monte Carlo simulation results of GA4 with the RHC strategy, where N is the horizon length, and $N = n$ means the receding horizon is composed of n time intervals (each time interval is assumed to be 5 min long). From Table 7 one can see that:

- Using the RHC strategy ($N < 10$) can often achieve smaller delays than the static case ($N = 10$). Basically, a shorter horizon is likely to give a better performance, but if the horizon is too short, e.g., $N = 1$, the performance of GA4 could degrade significantly. This is because, a shorter horizon covers less aircraft to be considered in a single run of optimization, which means a smaller solution space for GA4 to search; however, when N is too small, GA4 could end up with short-sighted performance. Table 7 shows $N = 2$ or 3 yields the best performance in all 5 ATC scenarios.
- The computational burden of GA4 decreases dramatically when N goes down. Table 7 shows the computational time at $N = 2$, where arrival sequences of very high quality are achieved, is just about one tenth of that in the static case ($N = 10$). This implies the RHC strategy is very helpful in extending GA4 to real-time solutions to the MRASS problem.

More detailed discussions on integrating the RHC strategy into GAs in order to tackle the real-time ASS problem can be found in [11].

6. Conclusions

This paper aims to design an efficient GA with uniform crossover to tackle the aircraft arrival sequencing and scheduling (ASS) problem in multi-runway systems. Uniform crossover is usually effective and efficient to identify, to inherit and to protect common genes in genetic algorithms (GAs), but it could be difficult to design or apply when chromosomes are not properly constructed. Instead of the order of aircraft in arrival queues, which is widely used in existing GAs for the ASS problem, the following relationship between aircraft is used to construct chromosomes in the new GA. A highly efficient uniform crossover operator is then designed, which is effective to keep a good balance between diversity and convergence in the evolutionary process. In the experiments, the new GA with uniform crossover is extensively compared with three other GAs, and the results illustrate clearly the advantages exhibited by the new GA in performance. The strategy of receding horizon control can also be integrated into the new GA, which increases largely the potential of real-time implementations of the reported GA to the ASS problem.

Further research will be conducted in order to extend the reported work from static air traffic situation to dynamical environment based on real traffic data. We are currently contacting and interviewing airport traffic controllers, in order to collect and analyze real traffic data in various patterns (e.g., distributions of congestion, uncertainties, and different whether conditions), to modify and improve our model and algorithm in terms of realistic factors, such as on-line adjustments to the arrival sequence, and particularly to conduct experiments with unique characteristics of certain major airports (e.g., optimize the arrival sequence by referring to the gate assignment at the associated airport).

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005

References

[1] Hansen JV. Genetic search methods for air traffic control. Computers and Operations Research 2004;31:445–59.
[2] Heere K, Zelenka R. A comparison of Center/TRACON automation system and airline time of arrival predictions. NASA/TM-2000-209584; 2000.
[3] Heimerman K. Air Traffic Control Modelling ⟨www.mitre.org/support/papers/atcmodel⟩.
[4] Wong G. The dynamic planner: sequencer, scheduler, and runway allocator for air traffic control automation. NASA/TM-2000-209586; 2000.
[5] Erzberger H, Davis TT, Green S. Design of Centre-TRACON automation system. Proceedings of the 56th symposium on machine intelligence in air traffic management; 1993. p. 11-1-12.
[6] Farley T, Foster J, Hoang T, Lee K. A time-based approach to metering arrival traffic to Philadelphia. Proceedings of AIAA 2001-5241, 2001.
[7] Bianco L, Dell'Olmo P, Giordani S. Scheduling models and algorithms for TMA traffic management. In: Bianco L, Dell'Olmo P, Odoni AR, editors. Modelling and simulation in air traffic management. Berlin: Springer; 1997. p. 139–67.
[8] Pelegrin M. Towards global optimization for air traffic management. AGARD-AG-321; 1994.
[9] Beasley JE, Sonander J, Havelock P. Scheduling aircraft landings at London Heathrow using a population heuristic. Journal of the Operational Research Society 2001;52:483–93.
[10] Cheng V, Crawford, L, Menon P. Air traffic control using genetic search techniques. Proceedings of the IEEE international conference on control applications; 1999.
[11] Hu XB, Chen WH. Genetic algorithm based on receding horizon control for arrival sequencing and scheduling. Engineering Applications of Artificial Intelligence 2005;18(5):633–42.
[12] Eiben AE, Schoenauer M. Evolutionary computing. Information Processing Letters 2002;82:1–6.
[13] Sywerda G. Uniform crossover in genetic algorithms. Proceedings of the third international conference on genetic algorithms, USA; 1989. p. 2–9
[14] Page J, Poli P, Langdon WB. Smooth uniform crossover with smooth point mutation in genetic programming: a preliminary study. Genetic Programming, Proceedings of EuroGP'99, Sweden; 1999.
[15] Falkenauer E. The worth of uniform crossover. Proceedings of the 1999 congress on evolutionary computation, vol. 1, CEC99, USA; 1999. p. 782.
[16] Hu XB, Chen WH. Receding horizon control for aircraft arrival sequencing and scheduling. IEEE Transaction on Intelligent Transportation System 2005;6(2):189–97.
[17] Delahaye D, Durand N, Alliot J, Schoenauer M. Genetic algorithms for air traffic control systems. Proceedings of the 14th IFORS triennial conference; 1996.
[18] Gregory CC, Erzberger H, Neuman F. Delay exchanges in arrival sequencing and scheduling. Journal of Aircraft 1999;36(5):785–91.
[19] Gregory CC, Erzberger H, Neuman F. Fast-time study of aircraft-influenced arrival sequencing and scheduling. Journal of Guidance, Control and Dynamics 2000;23(3):526–31.
[20] Andreatta G, Romanin-Jacur G. Aircraft flow management under congestion. Transactions in Science 1987;21:249–53.
[21] Bianco L, Bielli MM. System aspects and optimization models in ATC planning. In: Bianco L, Odoni AR, editors. Large scale computation and information processing in air traffic control. Berlin: Springer; 1993. p. 47–99.
[22] Dear RG. The dynamic scheduling of aircraft in the near terminal area, FLT R76.9, Flight Transportation Laboratory, MIT, Cambridge, 1976.
[23] Psaraftis HN. A dynamic programming approach to the aircraft sequencing problem, FTL R78-4, Flight Transportation Laboratory, MIT, Cambridge, 1978.
[24] Psaraftis HN. A dynamic programming approach for sequencing identical groups of jobs. Operations Research 1980;28:1347–59.
[25] Venkatakrishnan CS, Barnett A, Odoni AM. Landings at Logan Airport: Describing and increasing airport capacity. Transportation Science 1993;27:211–27.
[26] Bianco L, Rinaldi G, Sassano A. Scheduling task with sequence-dependent processing times. Naval Research Logistics 1988;35:177–84.
[27] Robinson III JE, Davis TJ, Issacson DR. Fuzzy reasoning-based sequencing of arrival aircraft in the terminal area. AIAA guidance, navigation and control conference, New Orleans, LA, August 1997.
[28] Holland JH. Adaptation in natural and artificial systems. Ann Arbor, MI: University of Michigan Press; 1975.

Please cite this article as: Hu X-B. Di Paolo E. An efficient genetic algorithm with uniform crossover for air traffic control. Computers and Operations Research (2007), doi: 10.1016/j.cor.2007.09.005