An Efficient Genetic Algorithm with Uniform Crossover for the Multi-Objective Airport Gate Assignment Problem

Xiao-Bing Hu[1] and Ezequiel Di Paolo[2]

[1] Centre for Computational Neuroscience and Robotics, University of Sussex
xiaobing.hu@sussex.ac.uk
[2] Centre for Computational Neuroscience and Robotics, University of Sussex
ezequiel@sussex.ac.uk

Genetic Algorithms (GAs) have a good potential of solving the Gate Assignment Problem (GAP) at airport terminals, and the design of feasible and efficient evolutionary operators, particularly, the crossover operator, is crucial to successful implementations. This paper reports an application of GAs to the multi-objective GAP. The relative positions between aircraft rather than their absolute positions in the queues to gates are used to construct chromosomes in a novel encoding scheme, and a new uniform crossover operator, free of feasibility problems, is then proposed, which is effective and efficient to identify, inherit and protect useful common sub-queues to gates during evolution. Extensive simulation studies illustrate the advantages of the proposed GA scheme with uniform crossover operator.

1 Introduction

As a major issue in Air Traffic Control (ATC) operations, the Gate Assignment Problem (GAP) at airport terminals aims to assign aircraft to terminal gates to meet operational requirements while minimizing both inconveniences to passengers and operating costs of airports and airlines. The term gate is used to designate not only the facility through which passengers pass to board or leave an aircraft but also the parking positions used for servicing a single aircraft. These station operations usually account for a smaller part of the overall cost of an airlines operations than the flight operations themselves. However, they can have a major impact on the efficiency with which the flight schedules are maintained and on the level of passenger satisfaction with the service [1], [2].

Most airline companies create monthly or quarterly Master Flight Schedules (MFSs) containing flight numbers and along with the corresponding arrival and departing times. The ground controllers use the MFSs to examine the capacity of gates to accommodate proposed schedules. There are several considerations that can bear on the decisions, such as aircraft size and servicing requirements, idle time of gates, flight crew and aircraft rotation, passenger walking distance, baggage transport distance, ramp congestion, aircraft waiting time, and use of remote parking stands. In the past few decades, many optimization methods have been reported to improve the gate assignment operation at airport terminals by focusing on one or two of the above considerations. For instance, passenger walking distance has been widely studied in the GAP research, and methods such as branch-and-bound algorithms [2], [3], integer programming [4], linear programming [5], expert systems [6], [7], heuristic methods [1], tabu search algorithms [8] and various hybrid methods [9], [10] were reported to minimize this distance. Baggage transport distance has been relatively less discussed in the GAP literature [1], [11]-[13], but the algorithms developed to solve the minimum passenger walking distance GAP can be easily extended to the case where baggage transport distance needs to be considered [1]. During the peak hours, it often happens that, particularly at hub airports, the number of aircraft waiting to dwell exceeds the number of available gates. In this case, aircraft waiting time on the apron should also be minimized [9], [14], [15]. The gate idle time is a criterion often used to assess the efficiency of using gate capacity [16]. However, the multi-objective GAP is relatively less discussed in literature. Reference [17] reported some interesting results, where passenger walking distance and passenger waiting time were both considered, the GAP was modelled as a zero-one integer program, and a hybrid method was developed based on the weighting method, the column approach, the simplex method and the branch-and-bound technique.

As large-scale parallel stochastic search and optimization algorithms, GAs have a good potential for solving NP-hard problems such as the GAP. For instance, reference [15] developed a GA to minimize the delayed time during the gates reassignment process, but the walking distance was not included. Reference [16] proposed a unified framework to specifically treat idle time of gates in the previous GAP models, and then developed a problem-specific knowledge-based GA. This paper aims to shed a little more light on how to design efficient GAs for the multi-objective GAP (MOGAP), where passenger walking distance, baggage transport distance, and aircraft waiting time on the apron need to be considered simultaneously. The design of highly efficient evolutionary operators, i.e., mutation and crossover, is crucial to successful applications of GAs to the GAP. Basically, mutation can increase the diversity of chromosomes in GAs to exploit the solution space, while crossover, in order to help GAs to converge to optima, needs to identify, inherit and protect good common genes shared by chromosomes, and at the same time to recombine non-common genes. Due to the stochastic nature of mutation and crossover, it is not an easy task to design efficient evolutionary operators free of the feasibility problem. For instance, infeasible solutions were particularly discussed in [16], where the mutation operator, rather than introducing diversity, was used to repair infeasible chromosomes generated by a conventional one-point split crossover operator.

This paper attempts to develop an infeasibility-free GA for the multi-objective GAP. Since crossover is often a main source of infeasible chromosomes, effort is particularly put on the design of a novel uniform crossover operator free of the feasibility problem. To this end, the relative positions between aircraft rather than the absolute positions of aircraft in the queues to gates is used to construct chromosomes in the new GA. As a result of the new uniform crossover operator, the design of mutation operator can concentrate on the original purpose of diversifying chromosomes.

2 Problem Formulation of the MOGAP

As mentioned before, there are quite a few different considerations in the GAP, and the MOGAP in this paper will focus on three of them: passenger walking distance, baggage transport distance, and aircraft waiting time on the apron. Passenger walking distance has a direct impact on the customer satisfaction. The typical walking distances in airports considered are: (I) the distance from check-in to gates for embarking or originating passengers, (II) the distance from gates to baggage claim areas (check-out) for disembarking or destination passengers, and (III) the distances from gate to gate for transfer or connecting passengers. Baggage transport distance occurs when baggage is transferred between aircraft and baggage claim areas. Basically, these distances can be reduced by improving the method by which scheduled flights are assigned to the airport terminal gates. Aircraft waiting time on the apron is the difference between the planned entering time to gates and the allocated entering time to gates. Due to the shortage of gates at peak hours, some scheduled aircraft have to wait extra time on the apron, which could end up with delayed departure and even cause passengers miss connection flights. Although this kind of ground delay is more tolerable than airborne delay in terms of safety and costs, it largely affects the customer satisfaction. Besides, aircraft waiting time can help address another big issue in the GAP: the efficiency of using gate capacity, which is often represented by how even the distribution of idle times is. In the minimum distance GAP, some special constraints have to be included in order to avoid most aircraft being assigned to a same single gate, which however can automatically be ensured by minimizing aircraft waiting time. Therefore, in the MOGAP, we will construct an objective function by combining the above three considerations. A simple way to conduct gate assignment is the first-come-first-served (FCFS) principle according to the planned entering time to gates, but the result is usually not optimal or even not near-optimal, because the FCFS principle does not take into account the layout of airport terminals. Even for a queue at a single gate, the FCFS principle is not the first option, mainly because different aircraft may have different ground time and different number of passengers. Obviously, putting ahead an aircraft with more passengers and less ground time could bring benefits, even if its planned entering time is later. Fig.1 gives a simple illustration of the MOGAP.

Suppose N_{AC} aircraft need to be assigned to N_G gates during a given time period $[T_S,T_E]$. Let P_i and G_i denote the planned entering time to gates and the ground time of the ith aircraft in the original set of aircraft under consideration, respectively. Assume P_i and G_i to be known in advance. In this paper, the planned entering time to gates for arrival aircraft is assumed to be the scheduled arrival time to the airport (A_i), and the planned entering time for departing aircraft is the scheduled departure time (D_i) minus the ground time, i.e., $P_i = D_i - G_i$. Let Q_g denote the queue at gate g, $Q_g(j)$ is the jth aircraft in Q_g, $g = 1,\dots,N_G, j = 1,\dots,H_g$, and H_g is the number of aircraft in Q_g satisfying

$$\sum_{g=1}^{N_G} H_g = N_{AC} \tag{1}$$

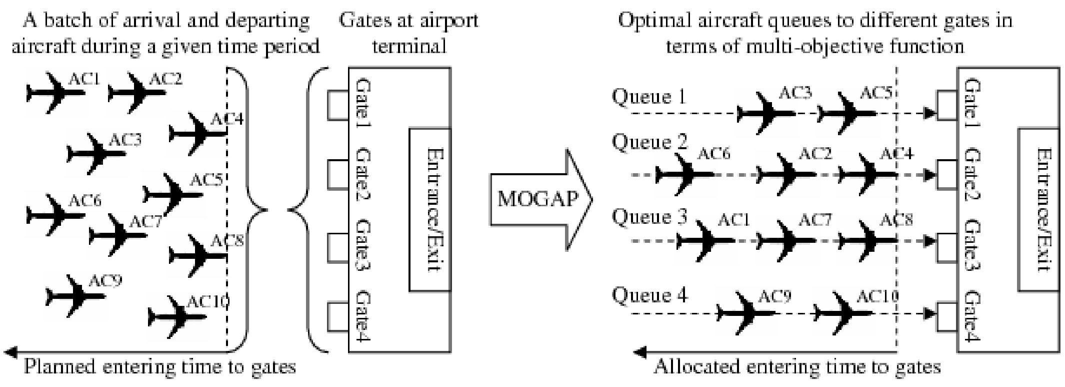

A batch of arrival and departing aircraft during a given time period
Gates at airport terminal
Optimal aircraft queues to different gates in terms of multi-objective function
AC1
AC2
AC3
AC4
AC5
AC6
AC7
AC8
AC9
AC10
Gate1
Gate2
Gate3
Gate4
Entrance/Exit
MOGAP
Queue 1
Queue 2
Queue 3
Queue 4
Planned entering time to gates
Allocated entering time to gates

Fig. 1. Illustration of the MOGAP

$Q_g(j) = i$ means the ith aircraft in the original set is assigned as the jth aircraft to dwell at gate g. The allocated entering time to gates (E_i) for the ith aircraft in the original set can then be calculated as

$$E_{Q_g(j)} = \begin{cases} P_{Q_g(j)}, & j = 1 \\ max(P_{Q_g(j)}, E_{Q_g(j-1)} + G_{Q_g(j-1)}), & j > 1 \end{cases} j = 1, \ldots, H_g, g = 1, \ldots, N_G \quad (2)$$

The waiting time on the apron for the ith aircraft in the original set is

$$W_i = E_i - P_i, i = 1, \ldots, N_{AC}. \quad (3)$$

For the sake of simplicity of modeling, besides the N_G real gates, the entrance/exit of the airport terminal is usually considered as a dummy gate (e.g., see [1]), and we call it gate $N_G + 1$ in this paper. Associated with this dummy gate $N_G + 1$, we introduce a dummy aircraft $N_{AC} + 1$. Of course there is no real aircraft queue for this dummy gate, except the dummy aircraft which dwells at the dummy gate all time.

Three data matrices, $M_p \in R^{(N_{AC}+1)\times(N_{AC}+1)}$, $M_{PWD} \in R^{(N_G+1)\times(N_G+1)}$, and $M_{BTD} \in R^{(N_G+1)\times(N_G+1)}$, are used to record the number of passengers transferred between aircraft, passenger walking distances between gates, and baggage transport distances between gates, respectively. Given $i \leq N_{AC}$ and $j \leq N_{AC}$, the value of $M_P(i, j)$ is the number of passengers transferred from aircraft i to aircraft j, $M_P(i, N_A + 1)$ records the number of arriving passengers from aircraft i to exit, i.e., the dummy aircraft $N_A + 1$, and $M_P(N_A + 1, j)$ the number of departing passengers from entrance to aircraft j. For those passengers who just pass by the airport with a long-haul aircraft, we assume they do not leave the aircraft when the aircraft stops at the airport. Therefore, we always have $M_P(i, i) = 0$ for $i = 1, \ldots, N_{AC} + 1$. $M_{PWD}(i, j)$ are the passengers walking distance from gate i to gate j, and $M_{BTD}(i, j)$ the baggage transport distance from gate i to gate j. Although $M_{PWD}(N_G + 1, N_G + 1) = 0$, we do not have $M_{PWD}(i, i) \neq 0$, $i = 1, \ldots, N_G$, because, even though passengers transfer between two aircraft which are successively assigned to the same gate, they still need to leave the first aircraft and wait in a certain terminal lounge before they can board the second aircraft. Similarly, for $M_{BTD}(i, i)$ one has $M_{BTD}(N_G + 1, N_G + 1) = 0$, but $M_{BTD}(i, i) \neq 0$. Besides these three matrices, we still need a data vector: $V_G = [v_1, \ldots, v_{N_{AC}+1}]$, where $1 \leq v_i \leq N_G + 1$ indicates that the

ith aircraft in the original set is assigned to gate v_i, and $v_{N_{AC}+1} \equiv N_G + 1$ means the dummy aircraft $N_{AC} + 1$ is always assigned to the dummy gate $N_G + 1$.

Now we can calculate the total passenger walking distance (TPWD), the total baggage transferring distance (TBTD), and the total passenger waiting time (TPWT) as

$$J_{TPWD} = \sum_{g=1}^{N_G+1} \sum_{j=1}^{H_g} \sum_{i=1}^{N_{AC}+1} M_P(Q_g(j), i) M_{PWD}(g, v_i), \tag{4}$$

$$J_{TBTD} = \sum_{g=1}^{N_G+1} \sum_{j=1}^{H_g} \sum_{i=1}^{N_{AC}+1} M_P(Q_g(j), i) M_{BTD}(g, v_i), \tag{5}$$

$$J_{TPWT} = \sum_{i=1}^{N_{AC}} W_i \sum_{j=1}^{N_{AC}+1} (M_P(i, j) + M_P(j, i)), \tag{6}$$

respectively. In the MOGAP of this paper, the following weighted objective function is used to cover these three aspects:

$$J_{MOGAP} = \alpha J_{TPWD} + \beta J_{TBTD} + (1 - \alpha - \beta)\phi J_{TPWT}, \tag{7}$$

where α and β are tuneable weights to adjust the contributions of TPWD, TBTD and TPWT,

$$\alpha + \beta \leq 1, 0 \leq \alpha \leq 1, 0 \leq \beta \leq 1, \tag{8}$$

and ϕ is a system parameter to make the waiting time comparable to the distances. In this paper, the distances are measured in meters, and the times measured in minutes. Assuming the average passenger walking speed is 3km/h, then 1 minute waiting time for a passenger can be considered as 50 meters extra walking distance for him/her. In this paper, we take the half, i.e., set $\phi = 25$ because we assume that for passengers walking is physically more uncomfortable than waiting.

The MOGAP can now be mathematically formulated as a minimization problem:

$$min_{Q_1,\ldots,Q_{N_G}} J_{MOGAP}, \tag{9}$$

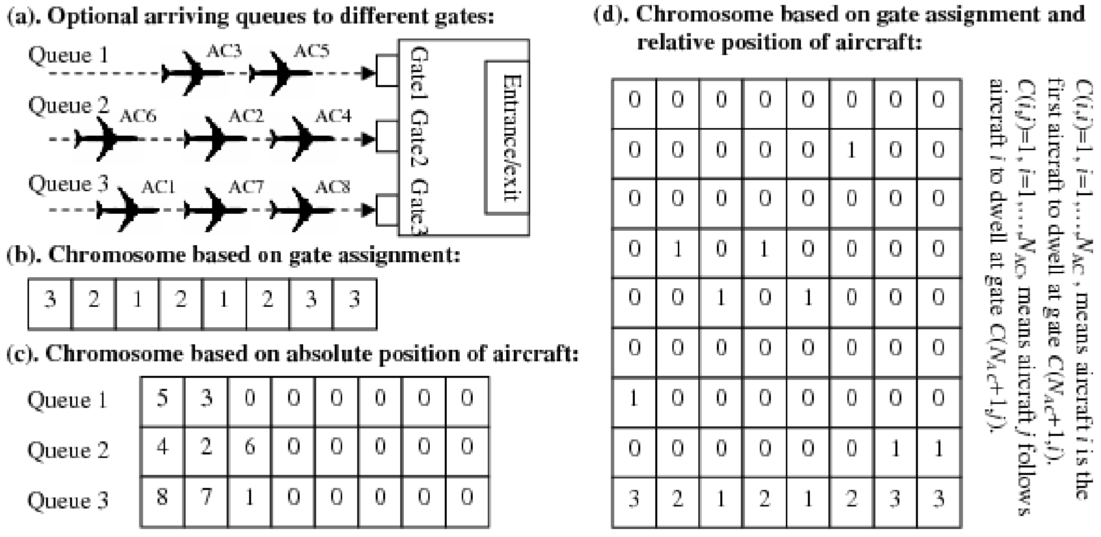
subject to (1) to (8). Clearly, how to assign aircraft to different gates to form N_G queues and how to organize the order of aircraft in each queue compose a solution, i.e., $Q_1, \ldots, Q_{N_G}$, to the minimization problem (9). Unlike other existing GAP models, the above formulation of the MOGAP needs no binary variables due to the usage of $Q_1, \ldots, Q_{N_G}$.

3 A GA with Uniform Crossover for the MOGAP

In this section we will report a new GA with uniform crossover for the MOGAP (The proposed new GA will be denoted as GAUC hereafter). For comparative purposes, we will also discuss two other GAs, particularly to compare their chromosome structures and crossover operators.

3.1 New Chromosome Structure

Basically, grouping aircraft according to gates, i.e., assigning aircraft to different gates, is one of the most important steps in the GAP, because it has direct influence on both walking distances and idle times of gates. If aircraft waiting time on the apron is not under consideration, then optimal grouping plus the FCFS principle can produce the best way of utilizing the gates at airport terminals. The GA proposed in [16] was designed based on aircraft grouping information. The structure of its chromosome is illustrated in Fig.2.(b), where a gene $C(i) = g$ means the ith aircraft in the original set of aircraft is assigned to dwell at gate g, in other words, gate g is assigned to aircraft i. Hereafter, we call aircraft grouping as gate assignment. Clearly, the GA based on gate assignment is not concerned about the order of aircraft in the queue to each gate, which is crucial to the minimization of aircraft waiting time on the apron.


(a). Optional arriving queues to different gates:
Queue 1
AC3
AC5
Queue 2
AC6
AC2
AC4
Queue 3
AC1
AC7
AC8
Gate1
Gate2
Gate3
Entrance/exit
(b). Chromosome based on gate assignment:
(c). Chromosome based on absolute position of aircraft:
(d). Chromosome based on gate assignment and relative position of aircraft:
$C(i,i)=1$, $i=1,\dots,N_{AC}$, means aircraft i is the first aircraft to dwell at gate $C(N_{AC}+1,i)$.
$C(i,j)=1$, $i\neq j=1,\dots,N_{AC}$, means aircraft j follows aircraft i to dwell at gate $C(N_{AC}+1,j)$.

(b)

3	2	1	2	1	2	3	3

(c)

Queue 1	5	3	0	0	0	0	0	0
Queue 2	4	2	6	0	0	0	0	0
Queue 3	8	7	1	0	0	0	0	0

(d)

0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0
0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1
3	2	1	2	1	2	3	3

Fig. 2. Chromosome structures (see text)

As discussed before, different aircraft may have different number of passengers and different ground time, and therefore, switching the positions of some aircraft in a FCFS-principle-based queue could reduce the total passenger waiting time, which is another criterion to assess the level of customer satisfaction with the service. The GA proposed for the arriving sequencing and scheduling problem in [18] can be modified and extended to handle the position switching in the GAP. The chromosome structure is illustrated in Fig.2.(c), where one can see the absolute positions of aircraft in queues to gates are used to construct chromosomes, i.e., a gene $C(g, j) = i$ means the ith aircraft in the original set of aircraft is assigned as the jth aircraft to dwell to gate g. Apparently, the underlying physical meaning of a chromosome, i.e., queues to gates, is expressed in a straightforward way by the absolute-position-based structure. However, it is difficult to carry out genetic operations on common genes, i.e., to identify, to inherit and to protect them, in these chromosomes based on absolute position of aircraft.

Basically, common genes should be defined as those sub-structures or sections which are shared by some chromosomes and play an important role in evolving the fitness of

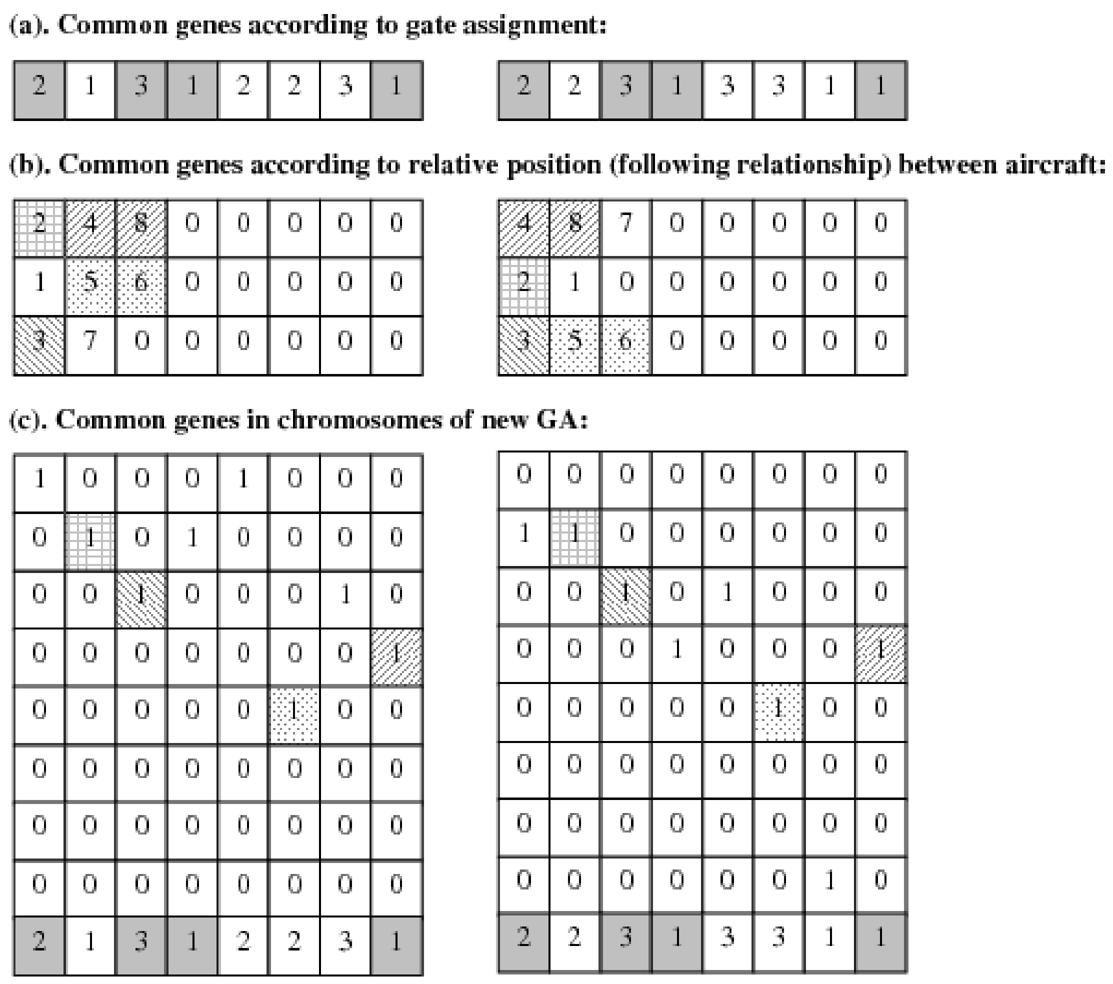

(a). Common genes according to gate assignment:
(b). Common genes according to relative position (following relationship) between aircraft:
(c). Common genes in chromosomes of new GA:

Fig. 3. Two definitions of common genes in the MOGAP

chromosomes. In the MOGAP, walking distances are sensitive to gate assignment, while relative position between aircraft in queues affects aircraft waiting time. In the evolutionary process, if many fit chromosomes assign a same gate to a same aircraft, and/or apply the same order to a same pair of successive aircraft in a queue, it is likely that this gate assignment and/or this relative position will also appear in the fittest chromosomes. Therefore, the same gate assignment and the same relative position are used to define common genes in the MOGAP. Fig.3.(a) and Fig.3.(b) illustrate these two definitions. From Fig.3.(a) one can see that the gate assignment based chromosome structure makes it very easy to identify common genes, i.e., if $C_1(i) = C_2(i)$, then these two genes are common genes. However, this structure has no information for identifying common relative position between aircraft in queues. An absolute position based chromosome structure has sufficient information for identifying both common gate assignment and common relative position, but unfortunately extra computationally expensive procedures are required.

The GAUC introduced in this paper will use the information of both gate assignment and relative position, not absolute position, to construct chromosomes. As illustrated in Fig.2.(d), a chromosome in the GAUC is a matrix with a dimension of $(N_{AC}+1) \times N_{AC}$, where the first $N_{AC} \times N_{AC}$ genes, i.e., $C(i,j), i = 1,\ldots,N_{AC}, j = 1,\ldots,N_{AC}$, record relative positions between aircraft in queues, and the last N_{AC} genes, i.e., $C(N_{AC}+1,j), j = 1,\ldots,N_{AC}$, record gate assignments. If $C(i,i) = 1$ and $C(N_{AC}+1,i) = g$, this means the ith aircraft in the original set of aircraft is assigned as the first aircraft to dwell at gate g; If $C(i,j) = 1$ and $C(N_{AC}+1,j) = g$, this means aircraft j is assigned to follow aircraft i to dwell at gate g. As illustrated in Fig.3.(c), with this new structure, common genes under both definitions can be easily identified: If $C_1(i,j)\&C_2(i,j) = 1$, then they are common relative position; If $C_1(N_{AC}+1,j) = C_2(N_{AC}+1,j)$, then they are common gate assignment.

Feasibility is a crucial issue in the design of a chromosome structure. For the structure based on gate assignment, if aircraft waiting time is allowed, which is the case in the MOGAP, then there is no feasibility problem as long as $1 \le C(i) \le N_G$ for any $i = 1,\ldots,N_{AC}$. For another two structures, some special constraints must be satisfied. The feasibility of chromosomes based on absolute position of aircraft is defined by two constraints: (I) each aircraft appears once and only once in a chromosome, and (II) if $C(g,j) > 0$, then $C(g,h) > 0$ for all $1 \le h < j$. For the GAUC proposed in this paper, a feasible chromosome must satisfy the following constraints according to the underlying physical meaning in the MOGAP:

$$\sum_{i=1}^{N_{AC}} \sum_{j=1}^{N_{AC}} C(i,j) = N_{AC}, \tag{10}$$

$$\sum_{j=1}^{N_{AC}} C(i,j) \begin{cases} \le 2, & C(i,i) > 0 \\ \le 1, & C(i,i) = 0 \end{cases} \tag{11}$$

$$\sum_{i=1}^{N_{AC}} C(i,j) = 1, \tag{12}$$

$$1 \le \sum_{i=1}^{N_{AC}} C(i,i) = \bar{N_G} \le N_G, \tag{13}$$

$$\sum_{C(N_{AC}+1,j)=g, j=1,\ldots,N_{AC}} C(j,j) = 1 \text{for any} g \in \bar{\Phi_G}, \tag{14}$$

where, without losing the generality, it is assumed that only $\bar{N_G}$ gates in all N_G gates are assigned to aircraft and $\bar{\Phi_G}$ denotes the set of assigned gates. Constraints (10) to (14) are actually a new version of the two feasibility constraints for chromosomes based on absolute position. From constraints (10) to (11), one can derive that there may often be some empty rows, no more than $\bar{N_G}$ empty rows, in the matrix. If the ith row is empty, then it means aircraft i is the last aircraft to dwell to gate $C(N_{AC}+1,i)$.

Actually, constraints (10) to (14) will rarely be used in the GAUC. In the initialization of a chromosome, the following procedure can efficiently generate a feasible chromosome only with a need to check against constraint (13):

Step 1: Create a $(N_{AC}+1) \times N_{AC}$ matrix with all entries set as 0. Let $U = \{1,\ldots,N_{AC}\}$ represent the original set of aircraft, and let $\Phi_G = \{1,\ldots,N_G\}$ be the set of gates.

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

Step 2.1: For $1 \leq i \leq N_{AC}$, if one more new $C(i,i) = 1$ will violate Constraint (13), then randomly choose an existing $C(i,i) = 1$, let $g = C(N_{AC} + 1, i)$, and go to Step 2.2. Otherwise, choose $i \in U$ and $g \in \Phi_G$ randomly, set $C(i,i) = 1$, $C(N_{AC} + 1, i) = g$, and remove i from U and g from Φ_G, i.e., let $U = U - \{i\}$ and $\Phi_G = \Phi_G - \{g\}$.

Step 2.2: Regardless of $C(i,i)$, if there is no non-zero entry in row i, then randomly choose $j \in U$, set $C(i,j) = 1$, $C(N_{AC} + 1, j) = g$, and remove j from U, i.e., let $U = U - \{j\}$. Otherwise, find the $C(i,j) = 1$, let $i = j$, and repeat Step 2.2.

3.2 Mutation Operator

Mutation is used by GAs to diversify chromosomes in order to exploit solution space as widely as possible. In the case of MOGAP, the mutation operation should be able to reassign an aircraft to any gate at any order. Therefore, we need two mutation operators: (I) one to shift randomly the positions of two successive aircraft in a same queue, and (II) the other 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. The chromosome structure based on gate assignment only supports the second mutation. The structure based on absolute position supports both as denoted as follows:

Mutation I: $C(g,j) \leftrightarrow C(g,j+1), j = 1,\ldots,H_g - 1, g = 1,\ldots,N_G$.
Mutation II: $C(g_1,j) \leftrightarrow C(g_2,k), j = 1,\ldots,H_{g_1}$ and $k = 1,\ldots,H_{g_2} + 1, g_1 \neq g_2, g_1 = 1,\ldots,N_G, g_2 = 1,\ldots,N_G$.

In the GAUC, the above two mutation operators need to be re-designed as the following in order to fit in the chromosome structure based on both gate assignment and relative position between aircraft in queues:

Mutation III: Randomly choose a non-zero gene, say, $C(i,j) = 1$. If there exist a $C(j,m) = 1$ (and maybe further a $C(m,h) = 1$), then change the values of some genes by following the instructions given in Fig.4.(a).

Mutation IV: Randomly choose two non-zero genes, say, $C(i,j) = 1$ and $C(h,x) = 1$ (there may be $C(j,m) = 1$ and/or $C(x,y) = 1$), which have different assigned gates, i.e., $C(N_{AC} + 1, j) \neq C(N_{AC} + 1, x)$. Then reset the values of some genes by following the instructions given in Fig.4.(b).

The mutation operations given in Fig.4 automatically guarantee the feasibility of resulting chromosomes as long as the original chromosomes are feasible.

3.3 Crossover Operator

There has long been a strong debate about the usefulness of crossover, and some people consider crossover as a special case of mutation, which is true in many designs of GAs [22]. However, we believe the fundamental role of crossover is different from that of mutation. Mutation aims to diversify chromosomes, while crossover can converge them by identifying, inheriting and protecting their common genes. As it is well known, it

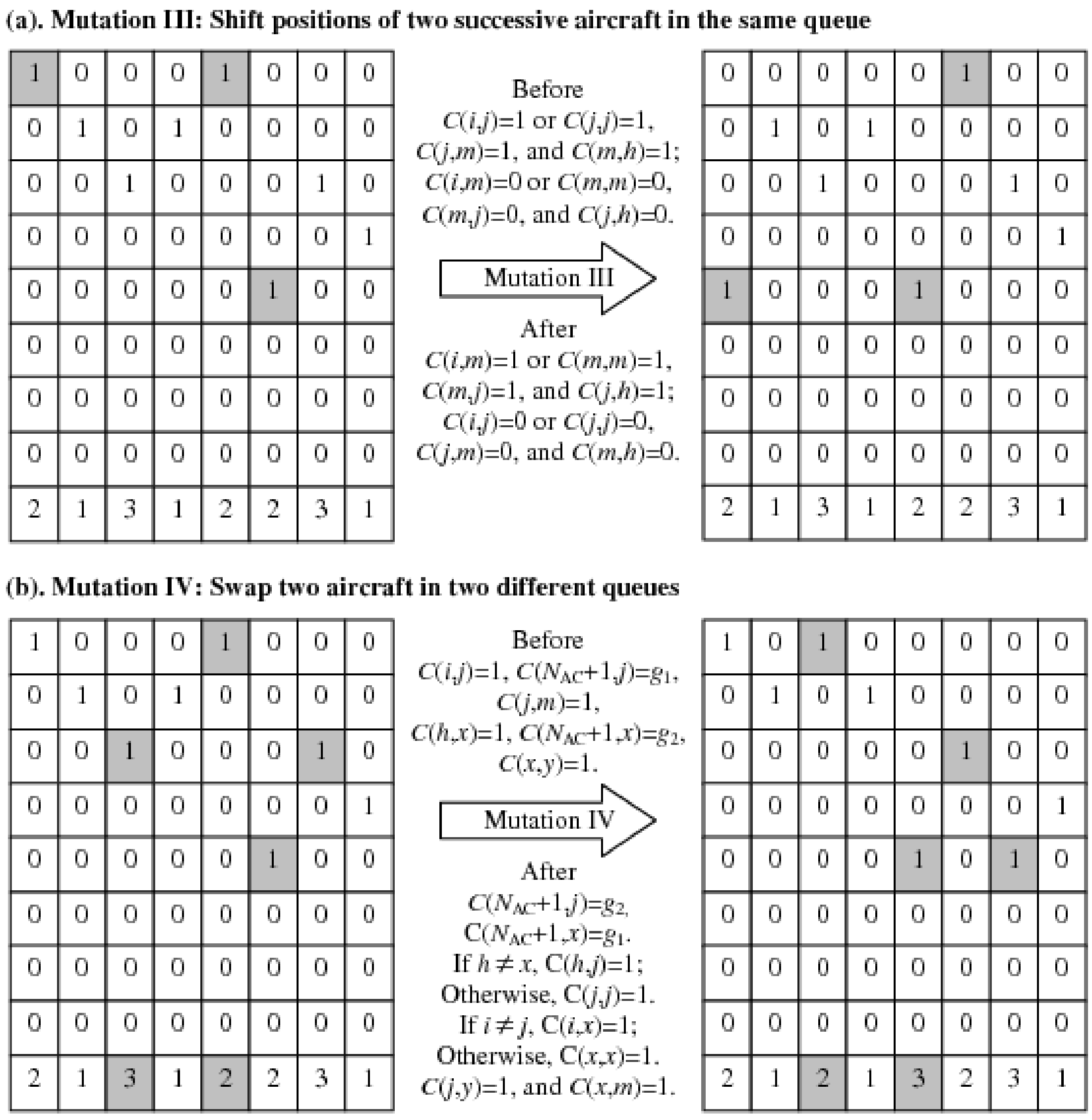

(a). Mutation III: Shift positions of two successive aircraft in the same queue
Before
C(i,j)=1 or C(j,j)=1, C(j,m)=1, and C(m,h)=1; C(i,m)=0 or C(m,m)=0, C(m,j)=0, and C(j,h)=0.
Mutation III
After
C(i,m)=1 or C(m,m)=1, C(m,j)=1, and C(j,h)=1; C(i,j)=0 or C(j,j)=0, C(j,m)=0, and C(m,h)=0.
(b). Mutation IV: Swap two aircraft in two different queues
Before
C(i,j)=1, C(N_AC+1,j)=g_1, C(j,m)=1, C(h,x)=1, C(N_AC+1,x)=g_2, C(x,y)=1.
Mutation IV
After
C(N_AC+1,j)=g_2, C(N_AC+1,x)=g_1. If h ≠ x, C(h,j)=1; Otherwise, C(j,j)=1. If i ≠ j, C(i,x)=1; Otherwise, C(x,x)=1. C(j,y)=1, and C(x,m)=1.

Fig. 4. Mutation operators in GAUC

is crucial for GAs to keep a good balance between diversity and convergence in the evolutionary process, which is really a challenging task in the design of GAs. The difficulties in the design of highly efficient crossover operators do not and also should not mean that crossover is useless. Otherwise, we should expect to see a natural world dominated by single gender species, which however we all know is not true. Therefore, as a nature-inspired algorithm, GAs should have some primary tasks, to identify, to inherit and to protect good/useful common genes in chromosomes, only or at least mainly for crossover.

Uniform crossover is probably the most wildly used crossover operator because of its efficiency in not only identifying, inheriting and protecting common genes, but also re-combining non-common genes [19]-[21]. Fig.5, using the chromosome structure based on gate assignment, compares uniform crossover with another also wildly used crossover operator: one position split crossover. From Fig.5 one can see that the

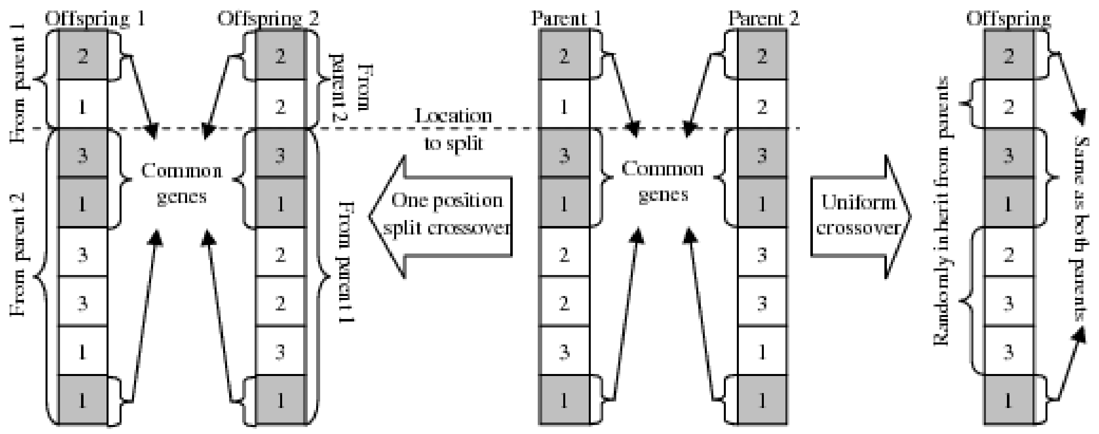


Fig. 5. Uniform crossover vs. on position split crossover

uniform crossover is actually a $(N_{AC} - 1)$-position-split crossover, which is obviously much more powerful than the one position split crossover in terms of exploiting all possibilities of recombining non-common genes.

To design an effective and efficient uniform crossover operator to handle common relative position between aircraft in queues is a major objective of this paper. Although the chromosome structure based on absolute position of aircraft contains the information of relative position, due to the feasibility issue in chromosomes, it is very difficult to design an effective crossover operator to identify, inherit and protect common relative positions. However, for comparative purposes, here we still manage to design a crossover operator for the absolute position based chromosome structure:

$$\begin{array}{ll} \text{If} & \{C_1(1,j),\ldots,C_1(N_G,j)\} = \{C_2(1,k),\ldots,C_2(N_G,k)\} \neq \{0,\ldots,0\} \\ \text{then} & C_1(.,j) \leftrightarrow C_2(.,k) \end{array} \tag{15}$$

Equation (15) requires the set of all jth aircraft in C_1 to be the same as the set of all kth aircraft in C_2. This crossover does not often cause feasibility problems. Actually, it has no feasibility problem if the following constraint is added

$$C_1(g,j) > 0, C_2(g,k) > 0 \quad \text{for all} \quad g = 1,\ldots,N_G \tag{16}$$

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 either gate assignment or relative position between aircraft. Actually the above crossover operator can be considered as a combination of Mutation I and II.

A focus of this paper is to design an effective and efficient crossover operator which can identify, inherit and protect both common gate assignment and common relative position between aircraft in queues to gates, and at the same time which can exploit all possibilities of re-combining non-common genes. The feasibility issue of the new crossover operator should be addressed in a computationally cheap way. With the new chromosome structure illustrated in Fig.2.(d) and the definitions of common genes shown in Fig.3.(c), we propose a novel uniform crossover operator described as the following procedure, which is further illustrated by Fig.6:

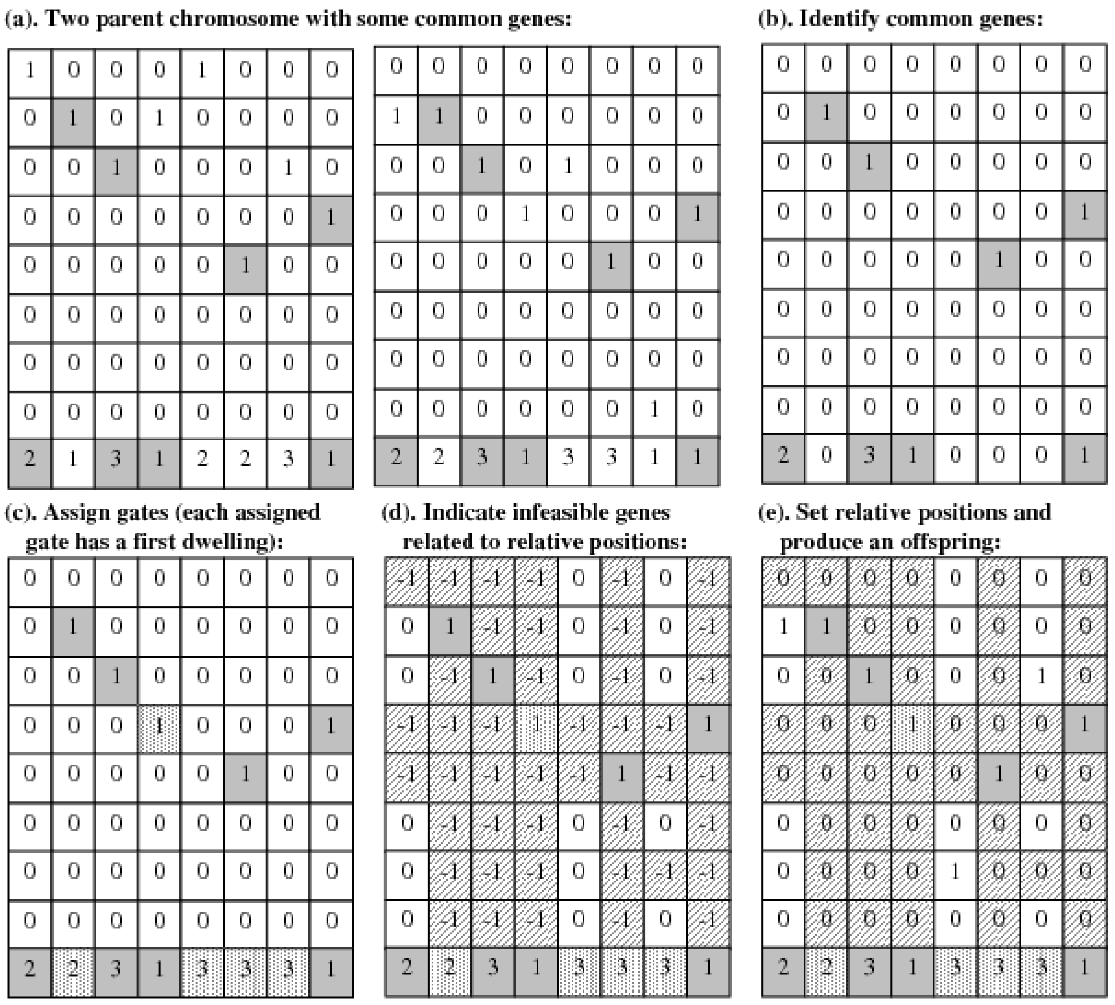

(a). Two parent chromosome with some common genes:
(b). Identify common genes:
(c). Assign gates (each assigned gate has a first dwelling):
(d). Indicate infeasible genes related to relative positions:
(e). Set relative positions and produce an offspring:

Fig. 6. Uniform crossover operation in GAUC

Step 1: Given two parent chromosomes C_1 and C_2, calculate C_3 to locate common genes

$$C_3(i,j) = C_1(i,j) \& C_2(i,j),$$

$$C_3(N_{AC}+1,j) = \begin{cases} C_1(N_{AC}+1,j), & C_1(N_{AC}+1,j) = C_2(N_{AC}+1,j) \\ 0 & C_1(N_{AC}+1,j) \neq C_2(N_{AC}+1,j) \end{cases} \tag{17}$$

$$i = 1,\ldots,N_{AC}, j = 1,\ldots,N_{AC},$$

i.e., $C_3(i,j) = 1$ or $C_3(N_{AC}+1,j) > 0$ means this location has a common gene shared by C_1 and C_2.

Step 2: Assign gates to C_3 by referring to C_1 and C_2. Basically, $C_3(N_{AC}+1,j)$ is set as $C_1(N_{AC}+1,j)$ or $C_2(N_{AC}+1,j)$, and $C_3(j,j)$ is set as $C_1(j,j)$ or $C_2(j,j)$, $j = 1,,N_{AC}$, at a half-and-half chance, subject to Constraint (14). Let $C_4 = C_3$.

Step 3: Indicate infeasible genes related to relative positions in C_4: Set $C_4(i,i) = -1$ for $i = 1,\ldots,N_{AC}$; If $C_3(i,j) = 1$ for $i \neq j$, then set $C_4(m,j) = -1$ and $C_4(i,m) = -1$

for $m = 1, \ldots, N_{AC}$; if $C_3(i,i) = 1$, then set $C_4(m,i) = -1$ for $m = 1, \ldots, N_{AC}$. $C_4(i,j) \neq 0$ means this location will not be considered when a new relative position between aircraft needs to be set up.

Step 4: While $\sum C_3(i,j) < N_{AC}, i = 1, \ldots, N_{AC}$, and $j = 1, \ldots, N_{AC}$, do

Step 4.1: Randomly choose j, such that $\sum C_3(i,j) = 0, i = 1, \ldots, N_{AC}$.

Step 4.2: Suppose $C_1(i_1,j) = 1$ and $C_2(i_2,j) = 1$, $i_1 = 1, \ldots, N_{AC}$, and $i_2 = 1, \ldots, N_{AC}$. If $C_3(N_{AC}+1,i_1) = C_3(N_{AC}+1,i_2) = C_3(N_{AC}+1,j)$ and $C_4(i_1,j) = C_4(i_2,j) = 0$, then set $i_3 = i_1$ or $i_3 = i_2$ at a half-and-half chance; Else if $C_3(N_{AC}+1,i_n) = C_3(N_{AC}+1,j)$ and $C_4(i_n,j) = 0$, $n = 1$ or 2, then set $i_3 = i_n$; Otherwise, randomly choose i_3 such that $C_3(N_{AC}+1,i_3) = C_3(N_{AC}+1,j)$ and $C_4(i_3,j) = 0$.

Step 4.3: Set $C_3(i_3,j) = 1$, $C_4(i_3,j) = 1$, $C_4(m,i_3) = -1$ and $C_4(i_3,m) = -1$ for $m = 1, \ldots, N_{AC}$.

Clearly, with the above crossover procedure, all common genes are efficiently identified, inherited and protected, and all possibilities of feasibly re-combining non-common genes can be exploited. As will be proved later, this uniform crossover is a very powerful search operator in the proposed GA.

3.4 Heuristic Rules

To further improve the performance of GA, e.g., to stimulate necessary and/or potentially useful local search, the following problem-specific heuristic rules are introduced:

- To help the algorithm to converge fast, not all of the new chromosomes are initialized randomly, but some are generated according to the FCFS principle. This is because, according to the real-world GAP operation, an FCFS-based solution is fairly reasonable, and an optimal or sub-optimal solution is often not far away from such an FCFS-based solution. Therefore, initializing some chromosomes according to the FCFS principle can effectively stimulate the local search around the FCFS-based solution.
- When initializing a chromosome randomly, we still follow the FCFS principle but in a loose way, i.e., an aircraft with an earlier P_i is more likely to be assigned to the front of a queue. This rule is also used to increase the chance of local search around the FCFS-based solution.
- If two aircraft have a same P_i, or their P_is are within a specified narrow time window, then the one with more passengers stands a better chance to be allowed to dwell first. This rule may help to reduce the total passenger waiting time significantly.
- For the sake of diversity, in each generation, a certain proportion of worst chromosomes are replaced by totally new ones.
- Like in [18], the population in a generation, $N_{Population}$, and the maximum number of generations in the evolutionary process, $N_{Generation}$, are adjusted according to N_{AC} in order to roughly keep the level of solution quality

$$N_{Population} = 30 + 10(round(max(0, N_{AC} - 10)/5)), \tag{18}$$

$$N_{Generation} = 40 + 15(round(max(0, N_{AC} - 10)/5)). \tag{19}$$

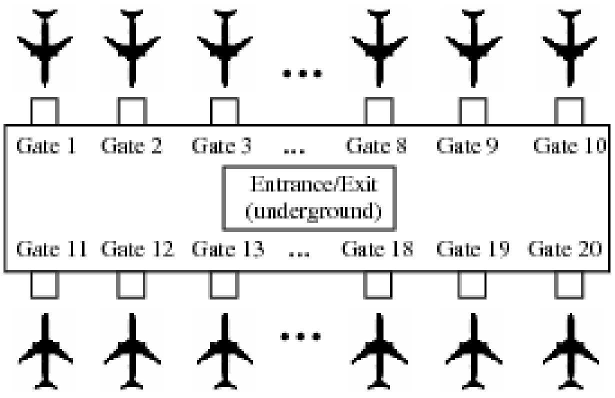


Fig. 7. Two-sided parking terminal layout

4 Simulation Results

The terminal layout has a big influence on the cost-efficiency of daily airport operations [23]. In our study, a typical terminal layout, two-sided parking terminal, is used, as illustrated in Fig.7. The terminal is assumed to have 20 gates. The data matrix M_{PWD} and M_{BTD}, i.e., distances for passenger walking and baggage transporting, are generated according to (20) to (23)

$$M_{PWD}(n,m) = M_{PWD}(m,n) = d_1 + d_2|nrem(n) - nrem(m)| \tag{20}$$

$$M_{PWD}(n,N_G+1) = M_{PWD}(N_G+1,n) = d_3 + d_2|nrem(n) - 5.5| \tag{21}$$

$$M_{BTD}(n,m) = M_{BTD}(m,n) = d_4 + d_5|nrem(n) - nrem(m)| \tag{22}$$

$$M_{BTD}(n,N_G+1) = M_{BTD}(N_G+1,n) = d_6 + d_5|nrem(n) - 5.5| \tag{23}$$

where $n = 1,\ldots,N_G$, $m = 1,\ldots,N_G$, and d_1 to d_6 are constant coefficients which can roughly determine the terminal size and the gate locations,

$$nrem(n) = \begin{cases} rem(n,11), & n < 11 \\ rem(n-10,11), & n \geq 11 \end{cases} \tag{24}$$

and *rem* is a function that calculate the remainder after division.

Traffic and passenger data are generated randomly under the assumption that the capacity of an aircraft varies between 50 and 300, the ground time span at a gate is between 30 and 60 minutes, and all aircraft are planned to arrive or depart within a period of one-hour time window. The congestion condition is indicated by N_{AC}. For comparative purposes, the GA reported in [18] is extended to solve the MOGAP. As discussed in Section 3, this extended GA employs the chromosome structure based on absolute position, and its crossover is actually a more complex mutation operator. Therefore, it is denoted as GACMO, in order to distinguish from the proposed GAUC

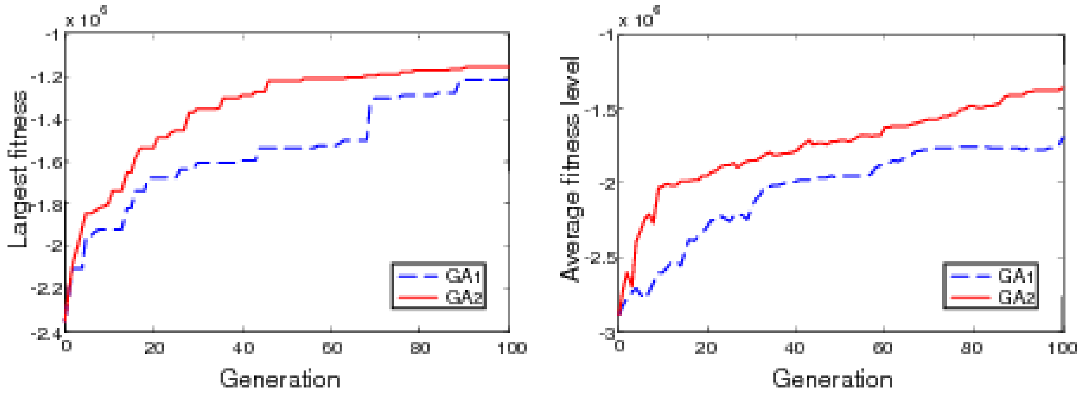


Fig. 8. Fitness levels in a test

with uniform crossover. Due to limited space, here we only give in Table 1 the results of a relatively simple case study, in order to illustrate how different GAs optimize gate assignment. In this test, J_{MOGAP} under GAUC is about 5% smaller than that of GACMO, and Fig.8 shows how the fitness, i.e. $-J_{MOGAP}$, changes in the evolutionary processes of GACMO and GAUC. From Fig.8.(a), one can see that the largest fitness of a generation in GAUC increases more quickly than that in GACMO, which means GAUC, which uses the new chromosome structure and uniform crossover, has a faster convergence speed than GACMO, which is designed based on absolute position of aircraft and has a crossover only equivalent to a combination of Mutation I and II. Actually, on average in this test, it takes GAUC 2.7778 generations to make a breakthrough in the largest fitness, while for GACMO, it takes 4.7619 generations. From Fig.8.(b) one can see that the average fitness of a generation in GAUC increases faster and stays larger than that in GACMO. This implies GAUC can effectively improve the overall fitness level, which is probably because the new uniform crossover proposed in this paper really works well in identifying, inheriting and protecting good common genes.

However, to get general conclusions about different GAs, we need to conduct extensive simulation tests, where N_{AC} is set as 30, 60 or 90 to simulate the situation of under-congestion, congestion, or over-congestion, and one of the single-objective functions in (4) to (6) or the multi-objective function in (7) is used. For each N_{AC} and objective function, 100 simulation runs are conducted under each GA, and the average results are given in Table 2 to Table 5, from which we have the following observations:

- Overall, GAUC is about $3\% \sim 10\%$ better than GACMO in terms of the specific objective function, which illustrates the advantages of the new chromosome structure based on relative position of aircraft in queues to gates and of the proposed uniform crossover operator based on the new structure.
- In the cases of single-objective GAP, as given in Table 2 to Table 4, GAUC achieves a better performance at the cost of other non-objective criteria. For instance, in Table 2, GAUC gets a smaller TPWD by sacrificing TAWT (total aircraft waiting time). TPWD and TBTD share a similar trend of change, i.e., if GAUC has a smaller/larger TPWD than GACMO, then it also has a smaller/larger TBTD. This is probably because both TPWD and TBTD, in a similar way, are determined largely by the terminal layout.

Table 1. Result of gate assignment in a single test

AC Code	$P_i(min)$	$G_i(min)$	GACMO $E_i(min)$	GACMO Gate	GAUC $E_i(min)$	GAUC Gate
1	28	40	28	3	28	3
2	26	50	26	16	26	16
3	5	40	5	9	5	9
4	12	45	12	7	12	7
5	43	50	43	19	43	19
6	27	45	27	4	27	4
7	34	40	34	2	34	2
8	10	35	10	12	10	12
9	48	30	48	12	48	12
10	56	35	56	14	56	14
11	25	35	25	15	25	15
12	52	35	53	13	53	13
13	7	50	7	11	7	8
14	56	40	63	8	57	8
15	56	60	60	15	60	15
16	49	35	49	20	49	5
17	39	30	39	1	39	1
18	47	40	47	9	47	9
19	13	40	13	13	13	13
20	52	35	57	11	63	11
21	25	50	25	5	25	20
22	35	40	35	18	35	18
23	16	50	16	6	16	6
24	20	35	20	14	20	14
25	56	45	66	6	66	6
26	4	40	4	10	4	10
27	8	55	8	8	8	11
28	54	50	57	7	57	7
29	45	35	45	10	45	10
30	28	45	28	17	28	17

Table 2. J_{TPWD} is used as objective function

$(\times 10^5)$		J_{TPWD}	TPWD(m)	TBTD(m)	TAWT(min)	MaxQL	MinQL
$N_{AC} = 30$	GACMO	7.2656	7.2656	18.9600	43.4807	29.5	0.2
	GAUC	7.0330	7.0330	18.4091	44.7622	29.6	0.2
$N_{AC} = 60$	GACMO	14.0606	14.0606	38.5475	201.1071	59.1	0.3
	GAUC	13.2538	13.2538	37.2206	209.5881	59.3	0.3
$N_{AC} = 90$	GACMO	19.7178	19.7178	56.4425	442.9681	88.6	0.6
	GAUC	18.8373	18.8373	55.0299	455.4340	88.5	0.5

Table 3. J_{TBTD} is used as objective function

($\times 10^5$)		J_{TBTD}	TPWD(m)	TBTD(m)	TAWT(min)	MaxQL	MinQL
$N_{AC} = 30$	GACMO	18.5846	7.2739	18.5846	43.6277	29.5	0.3
	GAUC	17.8939	7.1005	17.8939	45.1086	29.6	0.2
$N_{AC} = 60$	GACMO	38.1412	14.2805	38.1412	202.0039	59.3	0.3
	GAUC	36.9374	13.1288	36.9374	210.1956	59.5	0.3
$N_{AC} = 90$	GACMO	55.8907	20.1136	55.8907	440.7336	88.8	0.7
	GAUC	54.0407	18.9287	54.0407	451.1360	89.0	0.6

Table 4. J_{TPWT} is used as objective function

($\times 10^5$)		J_{TPWT}	TPWD(m)	TBTD(m)	TAWT(min)	MaxQL	MinQL
$N_{AC} = 30$	GACMO	1.5273	18.8120	24.8046	0.0611	2.2	0.9
	GAUC	1.4595	19.0023	24.9367	0.0583	2.2	0.9
$N_{AC} = 60$	GACMO	71.2180	36.9188	50.4188	2.8487	3.9	2.3
	GAUC	64.2053	37.3578	51.9046	2.5774	3.8	2.3
$N_{AC} = 90$	GACMO	219.8557	51.6150	73.1843	8.7942	5.3	3.9
	GAUC	208.5154	53.0487	75.5549	8.3508	5.3	4.0

Table 5. J_{MOGAP} is used as objective function

($\times 10^5$)		J_{MOGAP}	TPWD(m)	TBTD(m)	TAWT(min)	MaxQL	MinQL
$N_{AC} = 30$	GACMO	11.9457	16.1300	23.5272	0.1528	2.0	0.9
	GAUC	11.4672	15.5086	23.0442	0.1477	2.1	1.0
$N_{AC} = 60$	GACMO	53.0853	35.9684	49.8836	3.0112	3.9	2.2
	GAUC	49.5900	34.1606	48.7724	2.8031	4.0	2.1
$N_{AC} = 90$	GACMO	120.2156	49.7772	72.3854	8.8088	5.3	4.0
	GAUC	115.7206	47.8941	72.2129	8.4692	5.2	4.0

- In the case of MOGAP, if the weights in the objective function are properly tuned ($\alpha = 0.5$ and $\beta = 0.1$ in the associated tests), GAUC is better than GACMO not only in terms of the multi-objective function adopted, but also in terms of each single-objective function not adopted.
- In the minimum distance (passenger walking distance or baggage transporting distance) GAP, as shown in Table 2 and Table 3, we use no extra constraints to enforce assigning gates evenly to aircraft. As a result, the gap between the maximum queue length (MaxQL) and the minimum queue length (MinQL) is huge, which implies many aircraft are assigned to a certain gate. While in the minimum waiting time GAP, as given in Table 4, the gap between MaxQL and MinQL is very small, which means evenly using gates is automatically guaranteed during the minimization of

waiting time. Therefore, since waiting time is considered in the MOGAP, the gap between MaxQL and MinQL is also very small, as shown in Table 5.

- Basically, in a more congested case, i.e., with a larger N_{AC}, the operation of gate assignment is more expensive. Roughly speaking, the distances increase linearly in terms of N_{AC}, while the waiting time goes up exponentially, mainly because of the heavy delay applied to aircraft during a congested period. This might suggest, in a more congested case, waiting time should be given a larger weight.

5 Conclusion

Uniform crossover is usually efficient in identifying, inheriting and protecting common genes in GAs, but it could be difficult to design or apply when chromosomes are not properly constructed. This paper aims to design an efficient GA with uniform crossover to tackle the multi-objective gate assignment problem (MOGAP) at airport terminals. Instead of the absolute position of aircraft in queues to gates, which is widely used in existing GAs for the GAP, the relative position 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. The advantages of the new GA are demonstrated in extensive simulation tests. 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 which need to be collected and analyzed.

Acknowledgements

This work was supported by the EPSRC Grant EP/C51632X/1. A previous version of this paper was presented at The 2007 IEEE Congress on Evolutionary Computation (CEC2007), 25-28 Sep 2007, Singapore.

References

1. Haghani, A., Chen, M.C.: Optimizing gate assignments at airport terminals. Transportation Research A 32, 437–454 (1998)
2. Bolat, A.: Procedures for providing robust gate assignments for arriving aircraft. European Journal of Operations Research 120, 63–80 (2000)
3. Babic, O., Teodorovic, D., Tosic, V.: Aircraft stand assignment to minimize walking distance. Journal of Transportation Engineering 110, 55–66 (1984)
4. Mangoubi, R.S., Mathaisel, D.F.X.: Optimizing gate assignments at airport terminals. Transportation Science 19, 173–188 (1985)
5. Bihr, R.: A conceptual solution to the aircraft gate assignment problem using 0,1 linear programming. Computers & Industrial Engineering 19, 280–284 (1990)
6. Gosling, G.D.: Design of an expert system for aircraft gate assignment. Transportation Research A 24, 59–69 (1990)
7. Srihari, K., Muthukrishnan, R.: An expert system methodology for an aircraft-gate assignment. Computers & Industrial Engineering 21, 101–105 (1991)

8. Xu, J., Bailey, G.: Optimizing gate assignments problem: Mathematical model and a tabu search algorithm. In: Proceedings of the 34th Hawaii International Conference on System Sciences. Island of Maui, Hawaii, USA (2001)
9. Ding, H., Lim, A., Rodrigues, B., Zhu, Y.: The over-constrained airport gate assignment problem. Computers & Operations Research 32, 1867–1880 (2005)
10. Ding, H., Lim, A., Rodrigues, B., Zhu, Y.: New heuristics for the over-constrained flight to gate assignments. Journal of the Operational Research Society 55, 760–768 (2004)
11. Robuste, F.: Analysis of baggage handling operations at airports. PhD thesis, University of California, Berkeley, USA (1988)
12. Chang, C.: Flight sequencing and gate assignment in airport hubs. PhD thesis, University of Maryland at College Park, USA (1994)
13. Robuste, F., Daganzo, C.F.: Analysis of baggage sorting schemes for containerized aircraft. Transportation Research A 26, 75–92 (1992)
14. Wirasinghe, S.C., Bandara, S.: Airport gate position estimation for minimum total costs-approximate closed form solution. Transportation Research B 24, 287–297 (1990)
15. Gu, Y., Chung, C.A.: Genetic algorithm approach to aircraft gate reassignment problem. Journal of Transportation Engineering 125, 384–389 (1999)
16. Bolat, A.: Models and a genetic algorithm for static aircraft-gate assignment problem. Journal of the Operational Research Society 52, 1107–1120 (2001)
17. Yan, S., Huo, C.M.: Optimization of multiple objective gate assignments. Transportation Research A 35, 413–432 (2001)
18. Hu, X.B., Chen, W.H.: Genetic Algorithm Based on Receding Horizon Control for Arrival Sequencing and Scheduling. Engineering Applications of Artificial Intelligence 18, 633–642 (2005)
19. Sywerda, G.: Uniform crossover in genetic algorithms. In: Proceedings of the 3rd International Conference on Genetic Algorithms, USA (1989)
20. Page, J., Poli, P., Langdon, W.B.: Smooth uniform crossover with smooth point mutation in genetic programming: A preliminary study, Genetic Programming. In: Langdon, W.B., Fogarty, T.C., Nordin, P., Poli, R. (eds.) EuroGP 1999. LNCS, vol. 1598, p. 39. Springer, Heidelberg (1999)
21. Falkenauer, E.: The worth of uniform crossover. In: Proceedings of the 1999 Congress on Evolutionary Computation, USA (1999)
22. Eiben, A.E., Schoenauer, M.: Evolutionary computing. Information Processing Letters 82, 1–6 (2002)
23. Bandara, S., Wirasinghe, S.C.: Walking distance minimization for airport terminal configurations. Transportation Research A 26, 59–74 (1992)