
Genetic Algorithms for the Airport Gate Assignment: Linkage, Representation and Uniform Crossover

Xiao-Bing Hu¹ and Ezequiel Di Paolo²

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

Abstract A successful implementation of Genetic Algorithms (GAs) largely relies on the degree of linkage of building blocks in chromosomes. This paper investigates a new matrix representation in the design of GAs to tackle the Gate Assignment Problem (GAP) at airport terminals. In the GAs for the GAP, a chromosome needs to record the absolute positions of aircraft in the queues to gates, and the relative positions between aircraft are the useful linkage information. The proposed representation is especially effective to handle these linkages in the case of GAP. As a result, a powerful uniform crossover operator, free of feasibility problems, can be designed to identify, inherit and protect good linkages. To resolve the memory inefficiency problem caused by the matrix representation, a special representation transforming procedure is introduced in order to better trade off between computational efficiency and memory efficiency. Extensive comparative simulation studies illustrate the advantages of the proposed GA scheme.

1 Introduction

Since Genetic Algorithms (GAs) were proposed by Holland [1], they have been broadly and successfully applied to solve problems in numerous domains. As the scale and complexity of problems handled by GAs increase, researchers begin to realize that for practical use, certain crucial mechanisms have to be integrated into the framework of GAs. Among these crucial mechanisms suggested by practitioners is the ability to learn linkage, referred to as the relationship between variables or elements represented by a chromosome. In the past few decades, there has been growing recognition that effective GAs demand understanding of linkage in order to tackle complicated, large scale problems [1]-[5]. Studies have shown that easy problems can be solved by any

ordinary GAs, but when harder problems are considered, scalability has been elusive. As indicated by the results reported in [6], even separable problems could be exponentially hard if the knowledge of the variable groups were not available. Therefore, a successful implementation of GAs largely depends on a good understanding of the relationship between variables or elements, i.e., the linkage of building blocks of a chromosome, an appropriate representation of linkage, and some effective methods to distinguish between good and bad linkage, and to store linkage information as well [5]. Linkage related techniques are so diverse, sophisticated, and highly integrated with each problem-specific implementation of GAs. This paper, bearing the linkage concept in mind, attempts to design an effective and efficient GA for tackling the Gate Assignment Problem (GAP) at airport terminals. To this end, we need to identify and analyze the useful linkage between the elements in a solution to the GAP, to choose a suitable chromosome structure to represent the useful linkage, and then to design effective evolutionary operators, particularly crossover, to take advantage of the useful linkage.

As the bottle-neck resource in the air transportation system, airports play an extremely important role in the campaign to meet the constantly increasing demands for air traffic services. The Gate Assignment Problem (GAP) at airport terminals is a major issue during the daily airport operations, which involves a set of aircraft with arrival and departure times specified in monthly or quarterly master flight schedules and a set of gates with considerations from airlines and the airport. In other words, the airport GAP 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 airline's 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 [7], [8]. Those considerations, or criteria in another term, adopted in the GAP are mainly based on the availability and the compatibility of gates, their sizes and capacities, operational rules, service and managerial directions. Basically, aircraft should be assigned to gates in an optimal way such that some primary criteria can be best satisfied, e.g., the distance a passenger is required to walk in an airport to reach his departure gate, the baggage claim area, or his connecting flight can be minimized, the distance baggage is transported between aircraft and terminal or another aircraft can be significantly shortened, and the time an aircraft needs to wait before dwelling on a gate can be made as short as possible.

Many optimization methods have been reported in the past few decades to address the GAP at airport terminals. For instance, passenger walking distance has been widely studied in the GAP research, and methods such as branch-and-bound algorithms [8], [9], integer programming [10], linear pro-

gramming [11], expert systems [12], [13], heuristic methods [7], tabu search algorithms [14] and various hybrid methods [15], [16] were reported to minimize this distance. Baggage transport distance has been relatively less discussed in the GAP literature [7], [17]-[19], 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 [7]. 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 [15], [20], [21]. The gate idle time is a criterion often used to assess the efficiency of using gate capacity [22]. The multi-objective GAP is discussed in [23], 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 [21] developed a GA to minimize the delayed time during the gates reassignment process. Reference [22] 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. More recently, Reference [24] reported an efficient GA with Uniform Crossover (UC) for the multi-objective airport GAP, where a novel matrix representation of the GAP queues plays a crucial role in the effectiveness and efficiency of the proposed GA. Based on the work reported in [24], this paper aims to investigate the importance of linkage information in the design of efficient GAs for the multi-objective airport GAP. As will be shown later, identifying the useful linkage information in the GAP, i.e., the relative positions of aircraft in the GAP queues, is a key step in the success of GA design. With this linkage information, matrix representation and uniform crossover can work efficiently in the evolutionary process of GA.

The remainder of this paper is organized as follows. In Section 2, a mathematical model of multi-objective airport GAP is provided. The useful linkage information in the GAP and the associated representations are discussed in Section 3. The design of GA is described in Section 4. Section 5 gives some simulation results, particularly analyzing the usefulness of the linkage information in the GA, and the paper ends with some conclusions in Section 6.

2 Problem formulation of the multi-objective GAP

The GAP in this paper will focus on three most popular considerations: 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 this paper, 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 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 GAP.

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 i th aircraft in the original set of arrival and departure aircraft, 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., $P_i = D_i - G_i$. Here we consider aircraft rather than flights, because, normally each individual aircraft can relate to two different flights (with different flight numbers): one is an arrival flight associated with an A_i , and the other is a departure flight, whose D_i is ideally determined as the A_i plus the G_i . Since the two flights associated with the same aircraft dwell at the same gate, for the sake of simplicity, we use aircraft rather than flights for the modelling. Sometimes, more than 2 commuter flights are actually related to the same physical aircraft. In this case, each physical aircraft will be considered as one or several shadow aircraft, each of which relate to two certain successive

flights. Therefore, N_{AC} is actually the number of all shadow aircraft. In the real gate assignment operation, the P_i of an arrival aircraft could be different from the A_i , normally later. It often happens that some arrival aircraft have to wait on the apron due to the shortage of gates. Therefore there could be a waiting time before an aircraft can dwell at a gate. The G_i of an aircraft is assumed to be optimal and fixed, i.e., the airport and airlines have predetermined the minimum time span for each individual aircraft to dwell at gates according to various considerations, such as operation efficiency and redundancy. For example, for the sake of redundancy, Reference [14] used a buffer time between the aircraft's departure time and the next aircraft's arrival time at the same gate. In this paper, the G_i is assumed to include this buffer time, which means the G_i is longer than the actual time span the aircraft physically dwells at a gate. Different aircraft may have different predetermined G_i , but, for the sake of simplicity, we assume that the same G_i always applies to a given aircraft no matter which gate it dwells at.

Most existing methods use binary variables in the modelling of the GAP, e.g., see [7], [9], and [22]. These binary variables are used for each possible assignment: If, at a certain time instant, aircraft i is assigned to gate g , then the associated binary variable is set to 1; otherwise to 0. The usage of binary variables is a key technique for those IP, LP or QAP formulation based methods. Otherwise their modelling of the GAP is impossible. However, from the programming point of view, the usage of binary variables could cause memory inefficiency problems, because for each aircraft at each time instant, besides a binary variable that indicates which gate it is assigned to, other $N_G - 1$ more binary variables are required to show which gates it is not assigned to. These $N_G - 1$ extra binary variables are obviously not necessary in the common sense, but they are crucial to formulate the GAP as an IP, LP or QAP. Reference [7] even introduced more binary variables in order to transform QAP to LP. As mentioned by [14], billions of binary variables will be required before those methods can be applied to a real GAP, which means there could be a scalability problem. GAs do really not need those binary variables for the modelling of the GAP. Actually, GAs can be designed based on almost any kind of formulation of the GAP. This paper attempts to model the GAP in a more straightforward way according to the physical process of gate assignment.

Let Q_g denote the queue at gate g , $Q_g(j)$ is the j th 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

$$\sum_{g=1}^{N_G} H_g = N_{AC} \quad (1)$$

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

Fig. 1. Illustration of airport GAP.

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

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

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

For the sake of simplicity of modelling, besides the N_G real gates, the entrance/exit of the airport terminal is usually considered as a dummy gate (e.g., see [7]), 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 transferring 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 $i = 1, \dots, N_{AC} + 1$. $M_{PWD}(i, j)$ are the passengers walking distance from gate i to gate j , and $M_{BTD}(i, j)$ the baggage transferring distance from gate i to gate j . Although $M_{PWD}(N_G + 1, N_G + 1) = 0$, we do not have $M_{PWD}(i, i) \neq 0$, $i = 1, \dots, N_G$, 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. For $M_{BTD}(i, i)$ is the same case, i.e., $M_{BTD}(N_G + 1, N_G + 1) = 0$, but $M_{BTD}(i, i) \neq 0$. Besides these three matrices, we still need a data vector: $V_G = [\nu_1, \dots, \nu_{N_{AC}+1}]$, where $i \leq \nu_i \leq N_G + 1$ indicates that the i th aircraft in the original set is assigned to gate ν_i , and $\nu_{N_{AC}+1} = N_G + 1$ 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

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

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

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

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

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

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

$$\alpha + \beta \leq 1, 0 \leq \alpha \leq 1, 0 \leq \beta \leq 1, \quad (8)$$

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:

$$\min_{Q_1, \dots, Q_{N_G}} J_{MOGAP}, \quad (9)$$

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, \dots, 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, \dots, Q_{N_G} .

3 Linkage information and expressions in GAP

In this section we will discuss the useful linkage information to the design of GAs for the airport GAP. The general definition of linkage is given first, the GAP-specific linkage information is then identified, and some possible expressions of the linkage information in the GAP are discussed.

3.1 General definition of linkage

As is well known, the GA is a powerful search methodology by imitating the procreation process and operating on the principle of the survival of the fittest.

Fig. 2. Linkage in biological systems.

Fig. 3. An example of linkage in GA.

Understanding the bond and resemblance between the (natural) biology system and the (artificial) genetic and evolutionary algorithm may be helpful to realize the role and importance of learning linkage.

In biological systems, linkage refers to the level of association in inheritance of two or more non-allelic genes that is higher than to be expected from independent assortment [31]. During meiosis, crossover events might occur between strands of the chromosome that genetic materials are recombined. Therefore, if two genes are closer to each other on a chromosome, there is a higher probability that they will be inherited by the offspring together. Genes are said to be linked when they reside on the same chromosome, and the distance between each other determines the level of their linkage. Fig. 2 gives an illustrative example of different genetic linkage between two genes. The upper part shows that if the genes are closer, they are likely to maintain the allele configuration. The lower part shows that if the genes are far away from each other, it is likely for a crossover event to separate them and to change the configuration. In summary, The closer together a set of genes is on a chromosome; the more probable it will not be split by chromosomal crossover during meiosis.

When applying genetic algorithms, we usually use strings of characters drawn from a finite alphabets as chromosomes and genetic operators to manipulate these artificial chromosomes. Reference [1] suggested that genetic operators which can learn linkage information for recombining alleles might be necessary for genetic and evolutionary algorithms to succeed. Many well known and widely employed crossover operators, including one-point crossover and two-point crossover, work under the similar situation subject to the linkage embedded in the chromosome representation as their biological counterparts do. For example, if we have a 6-bit function consisting of two independent 3-bit subfunctions, three possible coding schemes for the 6-bit chromosome are where $C_n(A)$ is the coding scheme n for an individual A , and a_i^j is the i th gene of A and belongs to the j th subfunction, as shown in Fig. 3.

Taking one-point crossover as an example, it is easy to see that genes belonging to the same subfunction of individuals encoded with C_1 are unlikely to be separated by crossover events. However, if the individuals are encoded with C_2 , genes of the same subfunction are split almost in every crossover event. For C_3 , genes of subfunction 1 are easily to be disconnected, while genes of subfunction 2 are likely to stay or to be transferred together.

From the viewpoint of genetic algorithms, linkage is used to describe and measure how close those genes that belong to a building block are on a chromosome. In addition to pointing out the linkage phenomenon, Reference [1]

Fig. 4. Encoding schemes in GAP (see text).

also suggested that the chromosome representation should adapt during the evolutionary process to avoid the potential difficulty directly caused by the coding scheme, which was identified as coding traps, the combination of loose linkage and deception among lower order schemata [32].

Because encoding the solutions as fixed strings of characters is common in genetic algorithm practice, it is easy to see that linkage can be identified as the ordering of the loci of genes as the examples given above. It is clear that for simple genetic algorithms with fixed genetic operators and chromosome representations, one of the essential keys to success is a good coding scheme that puts genes belonging to the same building blocks together on the chromosome to provide tight linkage of building blocks. The linkage of building blocks dominates all kinds of building-block processing, including creation, identification, separation, preservation, and mixing. However, in the real world, it is usually difficult to know such information a priori. As a consequence, handling linkage for genetic algorithms to succeed is very important. For a good survey on linkage in GAs, readers may refer to [5].

3.2 Useful linkage information in GAP

Linkage information, or the relationship between building blocks of a chromosome, is completely problem-dependent. Here we will analyze and identify the useful linkage information in the airport GAP.

Naturally, one may think the basic elements in a GAP solution are the mathematical characters which represent all aircraft needed to be assigned to gates. Basically, grouping aircraft according to gates, i.e., assigning aircraft to different gates, is the most important step 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 [22] was designed based on aircraft grouping information. The encoding scheme adopted in [22] is illustrated in Fig. 4.(b), where a gene $C(i) = g$ means the i th aircraft in the original set of aircraft is assigned to dwell at gate g . Hereafter, we call aircraft grouping as gate assignment. However, it is difficult to find any useful linkage information here, because, according to the encoding scheme based on gate assignment, each gene in Fig. 4.(b) can be considered as a subfunction, which means, no matter whatever crossover is applied, no subfunction will be split. Therefore, the concept of linkage is of little use to the encoding scheme in Fig. 4.(b).

Clearly, the encoding scheme 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. As discussed before, different aircraft may have different number of passengers and differ-

ent 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 [25] can be modified and extended to handle the position switching in the GAP. The encoding scheme is illustrated in Fig. 4.(c), where one can see that absolute positions of aircraft in queues to gates are used to construct chromosomes, i.e., a gene $C(g, j) = i$ means the i th aircraft in the original set of aircraft is assigned as the j th 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.

Based on the position of aircraft in queues, one can easily define the linkage information as the distance between two aircraft in a queue, but is this linkage information useful to a GA for the GAP? Actually, it is not the absolute position of aircraft in queues, but the relative position of aircraft in queues that affects passenger waiting times. Basically, if two aircraft are assigned to a gate successively, then a desirable order for this pair of aircraft is often that the aircraft with more passengers and less ground time should dwell first, no matter what is the absolute position of the leading aircraft in the pair. Therefore, the useful linkage information in the GAP is the pairwise of aircraft in queues, i.e., as illustrated in Fig. 4.(a), every two successive aircraft in queues composes a unit of useful linkage information in the GAP. An efficient GA for the GAP should be able to take advantage of this useful linkage information. It should be efficient to identify good pairwise of aircraft in queues, then inherit and protect them.

There are two encoding schemes which can express the linkage information associated with pairwise of aircraft. One is already discussed above, i.e., the absolute-position-based encoding scheme in Fig. 4.(c). This scheme can be considered using physical linkage, as the linkage information emerges from physical locations of two or more genes on the chromosome. However, under this encoding scheme, it is difficult to carry out genetic operations on common linkage information, i.e., to identify, to inherit and to protect it, in parent chromosomes, as shown in Fig. 5.(a). The other encoding scheme adopts virtual linkage, as illustrated in Fig. 4.(d), where a matrix with a dimension of $(N_{AC} + 1) \times N_{AC}$ is used to record directly the linkage information in the GAP queues. The first $N_{AC} \times N_{AC}$ genes, i.e., $C(i, j), i = 1, \dots, N_{AC}, j = 1, \dots, N_{AC}$, record relative positions between aircraft in queues, i.e., the pairwise of aircraft in queues, and the last N_{AC} genes, i.e., $C(N_{AC} + 1, j), j = 1, \dots, N_{AC}$, record gate assignment. If $C(i, i) = 1$ and $C(N_{AC} + 1, i) = g$, this means the i th 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. 5.(b), under this encoding scheme, common linkage information can be easily identified: If $C_1(i, j) \& C_2(i, j) = 1$, then they are common linkage information, i.e., common pairwise of aircraft.

Fig. 5. Common linkage information in GAP.

4 Design of GA for GAP

This section will discuss how to design an efficient GA which is particularly capable of taking advantage of the useful linkage information in the GAP.

4.1 Choose chromosome structure

In the airport GAP, the total passenger waiting time is sensitive to the relative position between aircraft in queues, i.e., the linkage information in the GAP. Therefore, a good chromosome structure should be able to capture the linkage information. Both the absolute-position-based encoding scheme and the relative-position-based encoding scheme discussed in Section 3.2 can meet this demand. Here we will further analyze the merits and demerits of these two encoding schemes from the following aspects:

- A chromosome structure efficient in term of handling common linkage information is preferable. As analyzed in Section 3.2, the relative-position-based encoding scheme has an obvious advantage because the linkage information is directly recorded in each gene, while the absolute-position-based encoding scheme is much less friendly to efficient crossover operators unless some additional measures are taken in order to abstract the useful knowledge on pairwises of aircraft in queues.
- Feasibility is a crucial issue in the design of an efficient chromosome structure. Due to the underlying physical meaning, some special constraints usually must be satisfied by chromosomes. For chromosomes based on absolute positions of aircraft, the feasibility 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 $1 \leq h < j$. Under the relative-position-based scheme, a feasible chromosome must satisfy the following 5 constraints according to the underlying physical meaning in the airport GAP:

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

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

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

$$1 \leq \sum_{i=1}^{N_{AC}} C(i, i) = \bar{N}_G \leq N_G, \quad (13)$$

$$\sum_{C(N_{AC}+1, j)=g, j=1, \dots, N_{AC}}^{N_{AC}} C(j, j) = 1 \text{ for any } g \in \bar{\Phi}_G, \quad (14)$$

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 i th row is empty, then it means aircraft i is the last aircraft to dwell to gate $C(N_{AC} + 1, i)$.

- Memory efficiency is another concern when choosing chromosome structure. According to Fig. 4, an absolute-position-based chromosome is composed of $N_G \times N_{AC}$ genes, while a relative-position-based one has $(N_{AC} + 1) \times N_{AC}$ genes. Clearly, the latter could confront an $O((n+1) \times n)$ memory problem when too many aircraft need to be considered at one time.

In order to obtain the merits of both encoding schemes and avoid their demerits, in other words, in order to be efficient to handle linkage information, simple regarding feasibility, and cheap in terms of memory demand, here we combine the above two encoding schemes to construct chromosomes by introducing a special representation transforming procedure in our GA for the airport GAP. As shown in Fig. 6, the generation of chromosomes are constructed under the absolute-position-based encoding scheme, which is more memory-efficient. The mutation operator is also designed to operate on the absolute-position-based structure, as it has less feasibility constraints. However, the relative-position-based scheme needs to be used to design an efficient crossover operator to identify, inherit and protect common linkage information in parent chromosomes. To this end, before two absolute-position-based parent chromosomes can crossover, they need to be transformed into their corresponding relative-position-based peers, and after the crossover operation, the relative-position-based offspring chromosomes need to be transformed back to the absolute-position-based format in order to be consistent with the generation's format. Fortunately, transforming from the absolute-position-based format to the relative-position-based format or in the reverse direction is a rather straightforward process. In an absolute-position-based chromosome, if $C(g, 1) = i$, then in its relative-position-based peer, one has $C(i, i) = 1$ and $C(N_{AC} + 1, i) = g$; if $C(g, h) = i$ and $C(g, h + 1) = j$, then in its relative-position-based peer, one has $C(i, j) = 1$ and $C(N_{AC} + 1, j) = g$. Vice versa.

Fig. 6. Combination of two encoding schemes and representation transforming procedure.

Fig. 7. Uniform crossover vs. one point crossover.

4.2 Mutation Operator

Mutation is used by GAs to diversify chromosomes in order to exploit solution space as widely as possible. In the case of airport GAP, 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 positions supports both well as denoted as follows:

Mutation I: $C(g, j) \leftrightarrow C(g, j + 1), j = 1, \dots, H_g - 1, g = 1, \dots, N_G$.

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

4.3 Crossover Operator

Crossover is an effective evolutionary process, mainly by operating on common genes, to speed up a GA to converge to optimas or sub-optimas. Uniform crossover is probably the most widely used crossover operator because of its efficiency in not only identifying, inheriting and protecting common genes, but also re-combining non-common genes [26]-[28]. Fig. 7, using the chromosome structure based on gate assignment in Fig. 4.(b), compares uniform crossover with another also widely used crossover operator: one point crossover. From Fig. 7 one can see that the uniform crossover is actually an ultimate multi-point crossover, which is obviously much more powerful than the one point crossover in terms of exploiting all possibilities of recombining non-common genes.

To design an effective and efficient uniform crossover operator to handle common linkage information in GAP queues is a major objective of this paper. Although the chromosome structure based on absolute positions of aircraft contains the information of relative positions of aircraft, due to the feasibility issue in chromosomes, it is very difficult to design an effective crossover operator in favor of the absolute-position-based format to identify, inherit and protect common relative position.

Fortunately, under the relative-position-based encoding scheme, both common gate assignments and common relative positions between aircraft in queues can easily be handled by uniform crossover, which can exploit all possibilities of re-combining non-common genes at the same time. The feasibility

Fig. 8. Uniform crossover using relative-position-based chromosome structure.

issue of the uniform crossover operator can also be addressed in a computationally cheap way. This novel uniform crossover operator is described as the following procedure, which is further illustrated by Fig. 8:

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

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

$$C_3(N_{AC} + 1, j) = \begin{cases} C_1(N_{AC} + 1, j), & C_1(N_{AC} + 1, j) = C_2(N_{AC} + 1, j) \\ 0 & C_1(N_{AC} + 1, j) \neq C_2(N_{AC} + 1, j) \end{cases} \quad (15)$$

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

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 $C_2(j, j)$, $j = 1, \dots, N_{AC}$, 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 $i = 1, \dots, N_{AC}$; If $C_3(i, j) = 1$ for $i \neq j$, then set $C_4(m, j) = -1$ and $C_4(i, m) = -1$ for $m = 1, \dots, N_{AC}$; if $C_3(i, i) = 1$, then set $C_4(m, i) = -1$ for $m = 1, \dots, N_{AC}$. $C_4(i, j) \neq 0$ means this location will not be considered when a new relative position between aircraft needs to be set up.

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

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

Step 4.2: Suppose $C_1(i_1, j) = 1$ and $C_2(i_2, j) = 1$, $i_1 = 1, \dots, N_{AC}$, and $i_2 = 1, \dots, N_{AC}$. 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 $m = 1, \dots, N_{AC}$.

Clearly, with the above crossover procedure, all common genes, i.e., both common linkage information and common gate assignments, 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 searching operator in the proposed GA.

4.4 Heuristic Rules

To further improve the performance of GA, 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.
- 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.
- If two aircraft have a same P_i , or their P_i s are within a specified narrow time window, then the one with more passengers stands a better chance to be allowed to dwell first.
- For the sake of diversity, in each generation, a certain proportion of worst chromosomes are replaced by totally new ones.
- Like in [25], 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

$$N_{Population} = 30 + 10(\text{round}(\max(0, N_{AC} - 10)/5)), \quad (16)$$

$$N_{Generation} = 40 + 15(\text{round}(\max(0, N_{AC} - 10)/5)). \quad (17)$$

5 Simulation Results

5.1 Simulation setup

The terminal layout has a big influence on the cost-efficiency of daily airport operations [30]. In our study, a typical terminal layout, two-sided parking terminal, is used, as illustrated in Fig. 9. 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 (18) to (21)

$$M_{PWD}(n, m) = M_{PWD}(n, m) = d_1 + d_2|nrem(n) - nrem(m)| \quad (18)$$

$$M_{PWD}(n, N_G + 1) = M_{PWD}(N_G + 1, n) = d_3 + d_2|nrem(n) - 5.5| \quad (19)$$

$$M_{BTD}(n, m) = M_{BTD}(n, m) = d_4 + d_5|nrem(n) - nrem(m)| \quad (20)$$

Fig. 9. Two-sided parking terminal layout.

$$M_{BTD}(n, N_G + 1) = M_{BTD}(N_G + 1, n) = d_6 + d_5 |nrem(n) - 5.5| \quad (21)$$

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

$$nrem(n) = \begin{cases} rem(n, 11), & n < 11 \\ rem(n - 10, 11), & n \geq 11 \end{cases} \quad (22)$$

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 [25] is extended to solve the GAP. This extended GA, denoted as GA1 hereafter (the proposed new GA with uniform crossover is denoted as GA2), employs the chromosome structure based on absolute position, and its crossover is actually a more complex mutation operator, as explained in [24]. The crossover probability and mutation probability are 0.5 and 0.3, respectively. 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 GA2 is about 5% smaller than that of GA1, and Fig. 10 shows how the fitness, i.e. $-J_{MOGAP}$, changes in the evolutionary processes of GA1 and GA2. From Table 1, one may easily find out that, under GA1, aircraft 14 is assigned to follow aircraft 27 to dwell at gate 9, and aircraft 20 to follow aircraft 13 to dwell at gate 11, while under GA2, aircraft 14 is assigned to follow aircraft 13 to dwell at gate 9, and aircraft 20 to follow aircraft 27 to dwell at gate 11. These different gate assignments cause no difference in total aircraft waiting time, i.e., aircraft 14 has to wait 6 more minutes under GA1, while aircraft 20 has to wait 6 more minutes under GA2. However, the passenger data show that aircraft 14 has more passengers than aircraft 20, and, more importantly, there are more transfer passengers within aircraft pairs (13,14) and (20,27) than within aircraft pairs (13,20) and (14,27). This means the above gate assignments under GA2, i.e., aircraft 20 following aircraft 27, and aircraft 14 following aircraft 13, stand a good chance to have smaller total passenger waiting time and shorter passenger walking distance and baggage transferring distance. Therefore, they are useful linkage information in this GAP. GA1 fails to identify and protect this useful linkage information, while GA2 succeeds. From Fig. 10.(a), one can see that the largest fitness of a generation in GA2 increases more quickly than that in GA1, which means GA2, owing to taking advantage of useful linkage information in the GAP and employing uniform crossover, has a faster convergence

Fig. 10. Fitness levels in a test.

speed than GA1. Actually, on average in this test, it takes GA2 2.7778 generations to make a breakthrough in the largest fitness, while for GA1, it takes 4.7619 generations. From Fig. 10.(b) one can see that the average fitness of a generation in GA2 increases faster and stays larger than that in GA1. This implies GA2 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 pairwise of aircraft in queues.

5.2 General performance

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, GA2 is about 3% to 10% better than GA1 in terms of the specific objective function, which illustrates the advantages of the proposed uniform crossover operator good at handling the linkage information in the GAP.
- In the cases of single-objective GAP, as given in Table 2 to Table 4, GA2 achieves a better performance at the cost of other non-objective criteria. For instance, in Table 2, GA2 gets a smaller TPWD by sacrificing TAWT (total aircraft waiting time). TPWD and TBTD share a similar trend of change, i.e., if GA2 has a smaller/larger TPWD than GA1, 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.
- In the case of multi-objective GAP, if the weights in the objective function are properly tuned ($\alpha = 0.5$ and $\beta = 0.1$ in the associated tests), GA2 is better than GA1 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

Table 1. Result of gate assignment in a single test

AC Code	$P_i(\min)$	$G_i(\min)$	GA1 $E_i(\min)$	GA2 $E_i(\min)$	Gate	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

of waiting time. Therefore, since waiting time is considered in the multi-objective GAP, 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.

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

$(\times 10^5)$	J_{TPWD}	TPWD(m)	TBTD(m)	TAWT(min)	MaxQL	MinQL	
$N_{AC} = 30$	GA1	7.2656	7.2656	18.9600	43.4807	29.5	0.2
	GA2	7.0330	7.0330	18.4091	44.7622	29.6	0.2
$N_{AC} = 60$	GA1	14.0606	14.0606	38.5475	201.1071	59.1	0.3
	GA2	13.2538	13.2538	37.2206	209.5881	59.3	0.3
$N_{AC} = 90$	GA1	19.7178	19.7178	56.4425	442.9681	88.6	0.6
	GA2	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	
$N_{AC} = 30$	GA1	18.5846	7.2739	18.5846	43.6277	29.5	0.3
	GA2	17.8939	7.1005	17.8939	45.1086	29.6	0.2
$N_{AC} = 60$	GA1	38.1412	14.2805	38.1412	202.0039	59.3	0.3
	GA2	36.9374	13.1288	36.9374	210.1956	59.5	0.3
$N_{AC} = 90$	GA1	55.8907	20.1136	55.8907	440.7336	88.8	0.7
	GA2	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	
$N_{AC} = 30$	GA1	1.5273	18.8120	24.8046	0.0611	2.2	0.9
	GA2	1.4595	19.0023	24.9367	0.0583	2.2	0.9
$N_{AC} = 60$	GA1	71.2180	36.9188	50.4188	2.8487	3.9	2.3
	GA2	64.2053	37.3578	51.9046	2.5774	3.8	2.3
$N_{AC} = 90$	GA1	219.8557	51.6150	73.1843	8.7942	5.3	3.9
	GA2	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	
$N_{AC} = 30$	GA1	11.9457	16.1300	23.5272	0.1528	2.0	0.9
	GA2	11.4672	15.5086	23.0442	0.1477	2.1	1.0
$N_{AC} = 60$	GA1	53.0853	35.9684	49.8836	3.0112	3.9	2.2
	GA2	49.5900	34.1606	48.7724	2.8031	4.0	2.1
$N_{AC} = 90$	GA1	120.2156	49.7772	72.3854	8.8088	5.3	4.0
	GA2	115.7206	47.8941	72.2129	8.4692	5.2	4.0

5.3 Further analysis

As discussed before, the proposed uniform crossover is particularly good at identifying, inheriting and protecting common linkage information in parent chromosomes. Here we will check whether or not this is the case in the simulation. To this end, before each crossover is carried out in either GA1 or GA2, we need to count how many common pairwise of aircraft are shared by parent chromosomes, and then record these common pairwise. After the crossover operation, we need to check out how many common pairwise shared by parent chromosomes still exist in the resulting offspring chromosome(s) (in GA1, two parent chromosomes reproduce two offspring chromosomes, while in GA2, two parent chromosomes reproduce one offspring chromosome through uniform crossover). Then, we can calculate the survival rate of common linkage information in chromosomes during the two different crossover operations. As explained in [24], the crossover operator in GA1 is designed based on the absolute-position-based encoding scheme, and it is actually equivalent to a complex combination of Mutation I and Mutation II given in Section 4.2. As is well known, mutation aims to diversify chromosomes, and therefore pays no attention to any common linkage information. After conducting 50 random runs of both GA1 and GA2 (here random run means choosing N_{AC} and objective function randomly), we found the average survival rate of common linkage information in GA1 is 0.8127, while in GA2, this rate is 1, which means common linkage information is fully protected during the uniform crossover operation in GA2. One may argue that (I) not all common linkage information in parent chromosomes is good in terms of fitness, and (II) even good common linkage information should be possible to be destroyed according to the stochastic nature of biological evolution. Actually, in GA2, unfit common linkage information is more likely to be eliminated during the competitive selection operation, and good common linkage information may be destroyed by mutation. Here the point is: our proposed uniform crossover is doing exactly and perfectly what a crossover operator is expected to do: providing one hundred percent protection to common linkage information.

Unlike in [24], this paper adopts a combined encoding scheme, i.e., GA2 mainly uses the absolute-position-based chromosome structure, but the uniform crossover operator requires the relative-position-based one, as discussed in Section 4.1. Therefore, before and after each crossover operation in GA2, a representation transforming procedure needs to be executed to parents and offspring. This representation transforming procedure saves memory at a cost of computational time. Is this really a good trade-off between memory efficiency and computational burden? Here we compare GA2 with the GA reported in [24], which is denoted as GA3 hereafter. All evolutionary operators in GA3 are actually equivalent to those used in GA2, but GA3 is designed purely on the ground of the relative-position-based encoding scheme. In other words, GA3 needs no representation transforming procedure for uniform crossover. The memory demands (MD) and computational times (CT) of GA2 and GA3

in different congestion conditions are compared in Table 6, from which one can see clearly:

- Thanks to the combined encoding scheme, GA2 is much more memory-efficient than GA3, particularly when many aircraft need to be considered at one time. Basically, as N_{AC} goes up, the population of a generation and the total generations to evolve have to increase correspondingly in order for GAs to maintain the level of solution quality. Assuming $N_{AC} = 1000$ and the population of a generation, N_{popu} , is 1000, then the memory demand by GA3 to store a generation of chromosomes is $1001 \times 1000 \times 1000$ bytes, which already exceeds the memory capacity of most standard personal computers. In the case of GA2, the memory demand for a generation is $20 \times 1000 \times 1000$ bytes, which is still tolerable and manageable.
- Regarding computational time, there is no obvious difference between GA2 and GA3, this is probably because, as discussed in Section 4.1, the representation transforming procedure in GA2 is rather straightforward, and therefore its time cost is almost unrecognizable compared with the computational time consumed by evolutionary operations in GAs.
- In summary, one can see the combined encoding scheme in GA2 provides good trade-off between the memory efficiency and computational burden of the algorithm.

Table 6. Memory efficiency and computational time

	$(N_{AC} = 30, N_{popu} = 70)$		$(N_{AC} = 60, N_{popu} = 130)$		$(N_{AC} = 90, N_{popu} = 190)$	
	MD (b)	CT (s)	MD (b)	CT (s)	MD (b)	CT (s)
GA2	42000	11.9	156000	52.7	342000	123.3
GA3	65100	11.4	475800	53.1	1556100	121.9

6 Conclusion

The Airport Gate Assignment Problem (GAP) is a major issue in air traffic control operations, and GAs have a good potential of resolving this problem. To make a successful implementation of GAs to the GAP, choosing a good representation of aircraft queues to gates is a crucial step in the design of GAs, and the relative positions between aircraft, i.e., the useful linkage information in chromosomes for the GAP, should be represented in a computationally efficient way. This paper studies a new matrix representation for the GAP, which adopts the relative positions between aircraft, rather than the widely used absolute positions of aircraft in queues to gates, to construct chromosomes. Based on this new matrix representation, an effective uniform crossover

operator is then designed in order to identify, inherit and protect those good linkages in chromosomes for the GAP. To better trade off between computational efficiency and memory efficiency, a special representation transforming procedure is introduced to resolve the $O((n+1) \times n)$ memory problem caused by the matrix representation. The advantages of the new GA are demonstrated in extensive simulation tests. Further research will be conducted, such as to introduce more complex factors and investigate the weights in order to make the GAP model more realistic, and to extend the report work from static air traffic situations to dynamical environments 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. Holland JH (1975) *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI
2. Goldberg DE (2002) *The design of innovation: Lessons from and for competent genetic algorithms*, Volume 7 of *Genetic Algorithms and Evolutionary Computation*. Kluwer Academic Publishers
3. Bosman PA, Thierens D (1999) Linkage information processing in distribution estimation algorithms. *Proceedings of Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*, 6067
4. Chen YP, Goldberg DE (2002) Introducing start expression genes to the linkage learning genetic algorithm. *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature (PPSN VII)*, 351360
5. Chen YP, Yu TL, Sastry K, Goldberg DE (2007) A Survey of Linkage Learning Techniques in Genetic and Evolutionary Algorithms. IlliGAL Report No. 2007014
6. Goldberg DE, Deb K, Thierens D (1993) Toward a better understanding of mixing in genetic algorithms. *Journal of the Society of Instrument and Control Engineers* 32:10-16
7. Haghani A, Chen MC (1998) Optimizing gate assignments at airport terminals. *Transportation Research A* 32:437-454
8. Bolat A (2000) Procedures for providing robust gate assignments for arriving aircraft. *European Journal of Operations Research* 120:63-80
9. Babic O, Teodorovic D, Tosic V (1984) Aircraft stand assignment to minimize walking distance. *Journal of Transportation Engineering* 110:55-66
10. Mangoubi RS, Mathaisel DFX (1985) Optimizing gate assignments at airport terminals. *Transportation Science* 19:173-188

11. Bihr R (1990) A conceptual solution to the aircraft gate assignment problem using 0,1 linear programming. *Computers & Industrial Engineering* 19:280-284
12. Gosling GD (1990) Design of an expert system for aircraft gate assignment. *Transportation Research A* 24:59-69
13. Srihari K, Muthukrishnan R (1991) An expert system methodology for an aircraft-gate assignment. *Computers & Industrial Engineering* 21:101-105
14. Xu J, Bailey G (2001) 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
15. Ding H, Lim A, Rodrigues B, Zhu Y (2004) New heuristics for the over-constrained flight to gate assignments. *Journal of the Operational Research Society* 55:760-768
16. Ding H, Lim A, Rodrigues B, Zhu Y (2005) The over-constrained airport gate assignment problem. *Computers & Operations Research* 32:1867-1880
17. Robuste F (1988) Analysis of baggage handling operations at airports. PhD thesis, University of California, Berkeley, USA
18. Chang C (1994) Flight sequencing and gate assignment in airport hubs. PhD thesis, University of Maryland at College Park, USA
19. Robuste F, Daganzo CF (1992) Analysis of baggage sorting schemes for containerized aircraft. *Transportation Research A* 26:75-92
20. Wirasinghe SC, Bandara S (1990) Airport gate position estimation for minimum total costs-approximate closed form solution. *Transportation Research B* 24:287-297
21. Gu Y, Chung CA (1999) Genetic algorithm approach to aircraft gate reassignment problem. *Journal of Transportation Engineering* 125:384-389
22. Bolat A (2001) Models and a genetic algorithm for static aircraft-gate assignment problem. *Journal of the Operational Research Society* 52:1107-1120
23. Yan S, Huo CM (2001) Optimization of multiple objective gate assignments. *Transportation Research A* 35:413-432
24. Hu XB, Di Paolo E (2007) An Efficient Genetic Algorithm with Uniform Crossover for the Multi-Objective Airport Gate Assignment Problem. In: *Proceedings of 2007 IEEE Congress on Evolutionary Computation*. Singapore
25. Hu XB, Chen WH (2005) Genetic Algorithm Based on Receding Horizon Control for Arrival Sequencing and Scheduling. *Engineering Applications of Artificial Intelligence* 18:633-642
26. Sywerda G (1989) Uniform crossover in genetic algorithms. In: *Proceedings of the 3rd International Conference on Genetic Algorithms*. USA
27. Page J, Poli P, Langdon WB (1999) Smooth uniform crossover with smooth point mutation in genetic programming: A preliminary study, *Genetic Programming*. In: *Proceedings of EuroGP99*. Sweden
28. Falkenauer E (1999) The worth of uniform crossover. In: *Proceedings of the 1999 Congress on Evolutionary Computation*. USA
29. Eiben AE, Schoenauer M (2002) Evolutionary computing. *Information Processing Letters* 82:1-6
30. Bandara S, Wirasinghe SC (1992) Walking distance minimization for airport terminal configurations. *Transportation Research A* 26:59-74
31. Hartl DL, Jones EW (1998) *Genetics: principles and analysis* (4th ed.). Sudbury, MA: Jones and Bartlett Publishers.

32. Goldberg DE (1987) Simple genetic algorithms and the minimal, deceptive problem. In LD (Ed.): Genetic Algorithms and Simulated Annealing (Chapter 6, pp. 7488). Los Altos, CA: Morgan Kaufmann Publishers.

Appendix: Notation

A_i	Scheduled arrival time to the airport
D_i	Scheduled departure time
E_i	Allocated entering time to a gate
G_i	Ground time
H_g	Number of aircraft in Q_g
J_{MOGAP}	Performance index for the multi-objective GAP
J_{TBTD}	Total baggage transferring distance
J_{TPWD}	Total passenger walking distance
J_{TPWT}	Total passenger waiting time
N_{AC}	Number of aircraft which need to be assigned to gates
N_G	Number of gates at the airport
$N_{Generation}$	The maximum number of generations in the evolutionary process
$N_{Population}$	The population in a generation
M_{BTD}	Baggage transferring distance matrix
M_P	Passenger data matrix
M_{PWD}	Passenger walking distance matrix
P_i	Planned entering time to a gate
Q_g	The queue at gate g
T_E	End point of the time period for the GAP
T_S	Starting point of the time period for the GAP
W_i	Aircraft waiting time
