A Ripple-Spreading Genetic Algorithm for the Airport Gate Assignment Problem

Xiao-Bing Hu and Ezequiel Di Paolo

Abstract—**Since the Gate Assignment Problem (GAP) at airport terminals is a combinatorial optimization problem, permutation representations based on aircraft dwelling orders are typically used in the implementation of Genetic Algorithms (GAs), The design of such GAs is often confronted with feasibility and memory-efficiency problems. This paper proposes a hybrid GA, which transforms the original order based GAP solutions into value based ones, so that the basic a binary representation and all classic evolutionary operations can be applied free of the above problems. In the hybrid GA scheme, aircraft queues to gates are projected as points into a parameterized space. A deterministic model inspired by the phenomenon of natural ripple-spreading on liquid surfaces is developed which uses relative spatial parameters as input to connect all aircraft points to construct aircraft queues to gates, and then a traditional binary GA compatible to all classic evolutionary operators is used to evolve these spatial parameters in order to find an optimal or near-optimal solution. The effectiveness of the new hybrid GA based on the ripple-spreading model for the GAP problem are illustrated by experiments.**

Keywords—**Genetic Algorithm, Representation, Combinatorial Optimization, Ripple-Spreading Model, Gate Assignment Problem.**

I. Introduction

The Gate Assignment Problem (GAP) at airport terminals is a major issue in daily air traffic control operations. Simply speaking, the 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 [1], [2]. In the past few decades, many optimization 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], have been reported to improve the gate assignment operation at airport terminals.

As a large-scale parallel stochastic searching and optimizing algorithm, GAs have a good potential for solving NP-hard combinatorial optimization problems such as the GAP. For instance, reference [11] developed a GA to minimize the delayed time during the gates reassignment process. Reference [12] 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. Recently, reference [13] reported an efficient GA with a matrix representation and uniform crossover for the multi-objective GAP, where passenger walking distance, baggage transport distance, and aircraft waiting time on the apron were considered simultaneously.

As is well known, GAs were originally developed based on the binary representation of variables, the value of which stands for candidate solutions to the problem [14]. They were soon extended to many combinatorial optimization problems, such as the GAP, whose solutions were represented as combination of elements [15]. In the design of GAs for combinatorial optimization problems, the basic binary representation is difficult to apply, and instead various problem-specific permutation representations have been proposed. Unfortunately, infeasible chromosomes and memory-inefficiency are common problems confronted by such permutation representations. As a result, in many applications of GAs to combinatorial optimization problems, evolutionary operators have to be modified mainly to serve feasibility purposes. Many classic evolutionary operators, such as uniform crossover, which were developed based on binary representation, can hardly be applied in GAs based on permutation representations without introducing computationally expensive measures, e.g., see [13]. Some applications with permutation representations even have opted for discarding crossover, because it appears more destructive than effective [16].

In contrast to the relevant existing literature, this paper attempts to use a basic binary representation friendly to all classic evolutionary operators to design a hybrid GA for the airport GAP. To this end, a deterministic model inspired by the phenomenon of the natural spread of ripples on a liquid surface is used to project aircraft waiting to dwell at gates as points into a especially parameterized space, and then a traditional GA with the basic binary representation

This work was supported by the EPSRC Grant EP/C51632X/1.

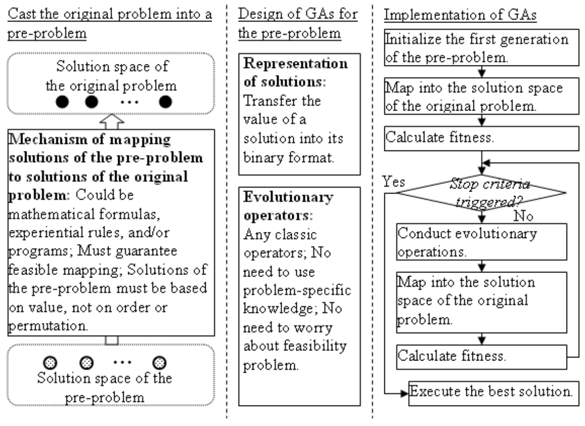
Xiao-Bing Hu is with the School of Engineering, University of Warwick, Coventry, CV4 7AL, UK (e-mail: Xiaobing.Hu@warwick.ac.uk).

Ezequiel. Di Paolo is with the Centre for Computational Neuroscience and Robotics, Department of Informatics, University of Sussex, Brighton, BN1 9QH, UK (e-mail: ezequiel@sussex.ac.uk).

compatible to classic evolutionary operators is designed to evolve those spatial parameters in order to find an optimal or near-optimal solution to the GAP. Actually, the idea of ripple-spreading model and the associated GA has already been used to handle complex network problems, and demonstrated to have good potential [17]. The remainder of this paper is organized as follows. The basic idea of the hybrid GA scheme is explained in Section 2. The mathematical description of the GAP is given in Section 3. The associated ripple-spreading model for the GAP is described in Section 4. Some important GA-related techniques are discussed in Section 5. The paper ends with some simulation results and plans for future research work in Sections 6 and 7.

II. The basic idea of a hybrid GA

Permutation representations are typically used when applying GAs to combinatorial optimization problems such as the GAP, but they make it difficult to design effective and efficient evolutionary operators often due to feasibility problems. In this paper, we aim to test the idea of using very basic binary GAs to solve combinatorial optimization problems which usually require permutation representations.

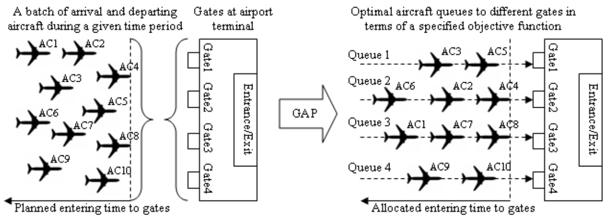


Cast the original problem into a pre-problem
Design of GAs for the pre-problem
Implementation of GAs
Solution space of the original problem
Mechanism of mapping solutions of the pre-problem to solutions of the original problem: Could be mathematical formulas, experiential rules, and/or programs; Must guarantee feasible mapping; Solutions of the pre-problem must be based on value, not on order or permutation.
Solution space of the pre-problem
Representation of solutions: Transfer the value of a solution into its binary format.
Evolutionary operators: Any classic operators; No need to use problem-specific knowledge; No need to worry about feasibility problem.
Initialize the first generation of the pre-problem.
Map into the solution space of the original problem.
Calculate fitness.
Stop criteria triggered?
Yes
No
Conduct evolutionary operations.
Map into the solution space of the original problem.
Calculate fitness.
Execute the best solution.

Fig. 1. Design of the hybrid GA.

In the absence of feasibility problems, the binary GAs used should be compatible with all classic evolutionary operators. To this end, we propose a hybrid GA, the basic idea of which is illustrated in Fig. 1. Basic binary GAs are easy to design for those problems where the solutions are based on value, and to such problems all classic evolutionary operations, such as mutation, one-point crossover and uniform crossover, are usually applicable. However, if the solutions are based on combination of elements, such as aircraft queues to gates in the GAP, it is very difficult, if not impossible, to design binary GAs, and crossover is often discarded because it is generally more destructive than effective. In our hybrid GA scheme, an original problem, whose solutions are based on combination of elements and therefore are unsuitable for binary representation, needs to be cast into a pre-problem whose solutions are based on value. Then we design a binary GA for this new pre-problem, and most classic GA techniques can apply straightforwardly. In the implementation of the hybrid GA, the only difference from a conventional GA is that, before the fitness of a chromosome is calculated, the represented solution to the pre-problem needs to be mapped onto the associated solution to the original problem. Obviously the most important, also the most difficult step in the hybrid GA scheme is to design a proper pre-problem, which depends largely on each individual original problem. In this paper, inspired by the natural ripple-spreading phenomenon, we will propose a pre-problem especially designed for the GAP, and then we report on a hybrid GA which has a good potential of resolving the GAP on all problem scales.

III. Problem Formulation of GAP



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 a specified objective function
AC1
AC2
AC3
AC4
AC5
AC6
AC7
AC8
AC9
AC10
Gate1
Gate2
Gate3
Gate4
Entrance/Exit
GAP
Queue 1
Queue 2
Queue 3
Queue 4
Planned entering time to gates
Allocated entering time to gates

Fig. 2. Illustration of GAP.

There are several considerations that can bear on the GAP decisions, such as aircraft size and servicing requirements, idle time of gates, flight crew and aircraft rotation, passenger walking distance, baggage transfer, ramp congestion, aircraft waiting time, and use of remote parking stands. In this paper we consider two GAP objectives: passenger walking distance (PWD), which has been widely studied in the GAP research, and aircraft waiting time (AWT), which results from the fact that the number of aircraft waiting to dwell often exceeds the number of available gates during the peak hours. PWD 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. Basically, these distances can be reduced by improving the method by which scheduled flights are assigned to the airport terminal gates. AWT 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. Besides, minimizing AWT can automatically ensure that aircraft are allocated to available gates as evenly as possible [13]. Therefore, in the GAP, we will construct an objective function by combining the above two considerations. A simple but not necessarily optimal way to conduct gate assignment is the first-come-first-served (FCFS) principle according to planned entering time to gates [1]. Fig.2 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 grounding time of the ith aircraft in the original set of aircraft under consideration, respectively. Suppose 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 grounding 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, $g=1,\ldots,N_G$, $j=1,\ldots,H_g$, and H_g is the number of aircraft in Q_g satisfying

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

$Q_g(j)=i$ 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

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

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

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

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 $N_{AC}+1$. Then we have two data matrices, $M_P \in R^{(N_{AC}+1)\times(N_{AC}+1)}$, and $M_{PWD} \in R^{(N_G+1)\times(N_G+1)}$, to record the number of passengers transferred between aircraft, passenger walking distances between gates, respectively [13]. Given $i \le N_{AC}$ and $j \le N_{AC}$, 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 $i=1,\ldots,N_{AC}+1$. $M_{PWD}(i,j)$ is the passenger walking 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. Besides these two matrices, we still need a data vector: $V_G=[v_1,\ldots, v_{N_{AC}+1}]$, where $1 \le v_i \le N_G+1$ 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.

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

$$\min_{Q_1,\cdots,Q_{N_G}} J_{MOGAP} = \min_{Q_1,\cdots,Q_{N_G}} (\alpha J_{TPWD} + (1-\alpha)\varphi J_{TPWT}) \tag{4}$$

subject to (1) to (3), where $0 \le \alpha \le 1$ is tunable weight to adjust the contributions of the total passenger walking distance (TPWD) and the total passenger waiting time (TPWT), which are calculated as

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

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

respectively, and φ is a system parameter to make the waiting time comparable to the distances. Like in [13], we set $\varphi = 25$.

IV. Ripple-spreading model of GAP

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., $Q_1,\ldots,Q_{NG}$, to the minimization problem (4). To apply the hybrid GA scheme, we need to cast the GAP into a pre-problem whose solution is simply based on value. To this end, firstly we set up a 3-dimensional space where the x axis is a time axis, the y axis is an aircraft grounding time axis, and the z axis is a passenger number axis but measured in units of time. Then we project all aircraft as points into the space according to their P_i, G_i and $\sum M_P(i,.)+\sum M_P(.,i)$. Here we define 2 spatial parameters which can partially determine how aircraft points are distributed in the space: δ_{xy} and δ_{xz}, which are the ratio between x and y, and the ratio between x and z, respectively. With δ_{xy} and δ_{xz}, we can measure y and z in the same time units as those used in the x axis. We also need another N_G spatial parameters: N_G reference points in the space, denoted as RPs and defined by their coordinates (x_i,y_i,z_i), $i=1,\ldots,N_G$. Basically the RPs can be given randomly, but we assume their x's are different from all P_i, $i=1,\ldots,N_{AC}$. As will be discussed later, the RPs are crucial for the deterministic ripple-spreading method to connect all aircraft points to form aircraft queues to gates. Besides, we need another 2 more spatial parameters: r_1, which is the minimal radius of a sphere including all RPs, and r_2, which is a sampling step of the ripple-spreading process as will be discussed later. Fig.3.(a) and Fig.3.(b) illustrate how the original set of aircraft waiting to dwell at gates are projected into a space with the parameters discussed above, where for the sake of simplicity, we assume $\delta_{xy}=1$ and all aircraft have a same passenger number, so that the 3-dimensional space is reduced to a 2-dimensional space.

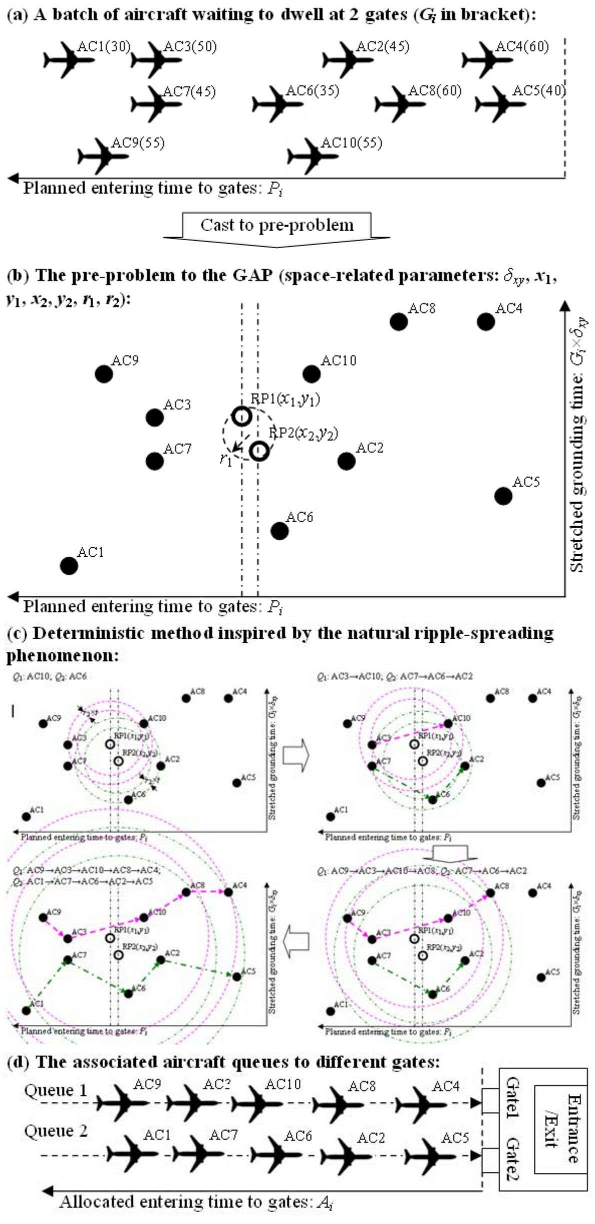
We then need to design a method that uses these spatial parameters as input and can output a unique solution to the original GAP by connecting all aircraft points to form aircraft queues to gates. Here by mimicking the natural ripple-spreading phenomenon, we propose a deterministic method for the pre-problem as following. The method can be roughly likened to throwing N_G stones into a pool of water where there stand N_{AC} stakes, i.e. aircraft points in the pre-problem. When the ripples spread out from the points where the stones hit the pool, i.e., the RPs, each ripple reaches every stake sooner or later according to the distance from each stake to the associated hit point. Base on the order in which each ripple reaches each stake, plus referring to the width of ripples, we can collect all N_{AC} aircraft to form some aircraft queues to gates.

```
Step 1: Calculate the distances between aircraft points to the given RPs. Let $M_{Dis}(i,j)$ be the distance between ACi and the RPj. Sort $M_{Dis}(i,j)$ in an ascending order. Let $S_{Dis}(j,k)=i$ mean ACi has the kth shortest distance to the RPj, in other words, $M_{Dis}(i,j)$ is the kth shortest distance to the RPj. Initialize U={1,…,$N_{AC}$}, i.e., set U as the original set of aircraft waiting to dwell at gates, let the current aircraft queues $Q_g$=[], i.e., set $Q_g$ as an empty vector, and set $t$=0.
Step 2: Do while U≠Ø.
  Step 2.1: $t=t+1$. For all aircraft points left in U, find the shortest distance to whichever RP. Suppose this shortest distance is $d_S$. Then with each RP as the center, a ring is generated whose radius is $d_S$, and whose width is $t \times r_2$.
  Step 2.2: For each RP, say RPg, choose up to 2 aircraft points covered by the associated ring. One aircraft point should have the shortest distance to the RPg among all those aircraft points that are covered by the ring and have a $P_i$ smaller than $x_g$. The other should have the shortest distance to the RPg among all those aircraft points that are covered by the ring and have a $P_i$ larger than $x_g$. Remove these aircraft from U.
  Step 2.3: Associate $Q_g$ with RPg. Then insert the aircraft with a $P_i$ smaller than $x_g$ to the front of the associated aircraft queue, and append the aircraft with a $P_i$ larger than $x_g$ to the end of the associated aircraft queue. Do this to every RP.
```

Fig.3.(c) and Fig.3.(d) illustrates how the above ripple-spreading-likened procedure determines a solution to the GAP step by step.

Different values for these spatial parameters may result in different solutions to the GAP. For example, Fig.4.(a) and Fig.4.(b) use the same set of δ_{xy}, δ_{xz}, r_1 and r_2 as Fig.3.(b) does, but they have different RPs, and as a result, we have different aircraft queues to gates. Actually, changing δ_{xy}, δ_{xz}, r_1 and/or r_2 may also produce different aircraft queues. However, once the values for these spatial parameters are fixed, one can see that the output of the above ripple-spreading procedure is uniquely determined. This is why we call this procedure as a deterministic ripple-spreading model of the GAP.

One may argue that the output of the above deterministic model might not cover all possible combinations of the original set of aircraft. However, many possible combinations are actually bad solutions to the GAP in terms of TPWD and TPWT. For example, if the RPs are not extremely far away from all aircraft points, then it is normally impossible to generate a queue where the aircraft with the smallest P_i and the aircraft with the largest P_i become two successive aircraft to dwell at a same gate. Such a queue sequence will usually increase the idle time of the gate and potentially increase the waiting time of other aircraft. Another example is that, in reality, aircraft with similar P_i should be allocated to different gates in order to reduce aircraft waiting time. Basically, our deterministic method, given a reasonably small r_1 and large r_2, provides a high probability to separate those aircraft with similar P_i to different gates. Therefore, the ripple-spreading


(a) A batch of aircraft waiting to dwell at 2 gates (G_i in bracket):
AC1(30)
AC3(50)
AC2(45)
AC4(60)
AC7(45)
AC6(35)
AC8(60)
AC5(40)
AC9(55)
AC10(55)
Planned entering time to gates: P_i
Cast to pre-problem
(b) The pre-problem to the GAP (space-related parameters: δ_{xy}, x_1, y_1, x_2, y_2, r_1, r_2):
RP1(x_1,y_1)
RP2(x_2,y_2)
r_1
Stretched grounding time: $G_i \times \delta_{xy}$
(c) Deterministic method inspired by the natural ripple-spreading phenomenon:
Q_1: AC10; Q_2: AC6
Q_1: AC3→AC10; Q_2: AC7→AC6→AC2
Q_1: AC9→AC3→AC10→AC8→AC4; Q_2: AC1→AC7→AC6→AC2→AC5
Q_1: AC9→AC3→AC10→AC8; Q_2: AC7→AC6→AC2
(d) The associated aircraft queues to different gates:
Queue 1
Queue 2
Gate1
Gate2
Entrance /Exit
Allocated entering time to gates: A_i
Fig. 3. Illustration of pre-problem and deterministic method.

model helps to filter out many bad combinations of the original set of aircraft, which is what is expected for a good pre-problem. Actually, many GAs use problem-specific knowledge and heuristic rules to keep chromosomes away from bad solutions, which, in our hybrid GA, is largely achieved by the pre-problem. Further more, we can integrate into the pre-problem as many problem-specific knowledge and heuristic rules as we like, without causing any undesired implication to the design of evolutionary operators in the hybrid GA. This is however often difficult for those existing

GAs for the GAP problem which directly integrate problem-specific knowledge and/or heuristic rules into their evolutionary operators.

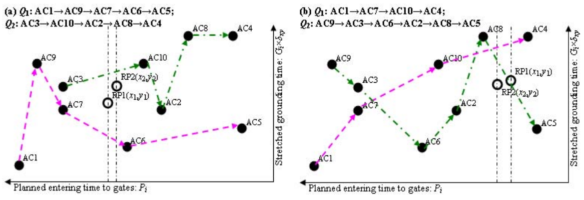


Fig. 4. Influence of spatial parameters.

V. Design of binary GA for the GAP

In this section we report on a binary GA for the GAP. For comparative purposes, we also discuss two other GAs based on permutation representations.

A. New Chromosome Structure

Reference [13] discussed two different permutation representations for the GAP. One is based on the absolute position of aircraft in queues to gates, 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, as illustrated in Fig.5.(b). The underlying physical meaning of a chromosome, i.e., queues to gates, is expressed in a straightforward way by the absolute-position-based chromosome structure. However, with this representation, it is difficult to operate on the relative position, i.e., the following relationship, between aircraft in queues, which is an important linkage in the GAP. Therefore, another permutation representation was reported in [13] specifically to handle the relative position between aircraft in queues to gates. As illustrated in Fig.5.(c), one can see that the associated chromosome 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,…,$N_{\rm AC}$, j=1,…,$N_{\rm AC}$, record relative position between aircraft in queues, and the last $N_{\rm AC}$ genes, i.e., $C(N_{\rm AC}+1,j)$, j=1,…,$N_{\rm AC}$, record gate assignments. If $C(i,i)$=1 and $C(N_{\rm AC}+1,i)=g$, 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 $C(N_{\rm AC}+1,j)=g$, this means aircraft j is assigned to follow aircraft i to dwell at gate g. Like most other permutation representations, some special constraints for the sake of feasibility have to be introduced in both of the above chromosome structures.

Thanks to the ripple-spreading model, in our new hybrid GA a chromosome needs to record the information of neither aircraft queues nor gate assignments. Instead, a chromosome is just a binary string of the values of those spatial parameters, as depicted in Fig.5.(d), where

$$L_{XYZ} = ceil(\log_2((\overline{X} - \underline{X})/X_S)), \tag{7}$$

$$L_{\delta} = ceil(\log_2((\overline{\delta} - \underline{\delta})/\delta_S)), \tag{8}$$

$$L_r = ceil(\log_2((\overline{r} - \underline{r})/r_S)), \tag{9}$$

and "ceil" is a function which rounds a number to the nearest integer towards infinite, $\overline{X}/\overline{\delta}/\overline{r}$, $\underline{X}/\underline{\delta}/\underline{r}$ and $X_S/\delta_S/r_S$ are the upper bound, the lower bound and the searching step of $(x,y,z)/\delta/r$, respectively.

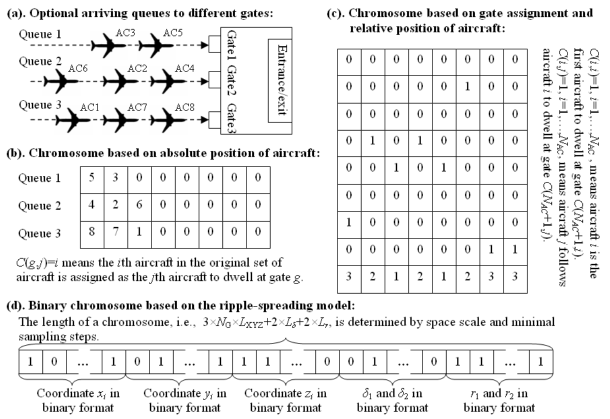

(a). Optional arriving queues to different gates:
Queue 1
AC3
AC5
Queue 2
AC6
AC2
AC4
Queue 3
AC1
AC7
AC8
Gate1
Gate2
Gate3
Entrance/exit
(b). Chromosome based on absolute position of aircraft:
Queue 1: 5 3 0 0 0 0 0 0
Queue 2: 4 2 6 0 0 0 0 0
Queue 3: 8 7 1 0 0 0 0 0
$C(g,j)=i$ means the ith aircraft in the original set of aircraft is assigned as the jth aircraft to dwell at gate g.
(c). Chromosome based on gate assignment and relative position of aircraft:
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
$C(i,i)$=1, i=1,…,N_{AC}, means aircraft i is the first aircraft to dwell at gate $C(N_{AC}+1,i)$.
$C(i,j)$=1, i=1,…,N_{AC}, means aircraft j follows aircraft i to dwell at gate $C(N_{AC}+1,j)$.
(d). Binary chromosome based on the ripple-spreading model:
The length of a chromosome, i.e., $3\times N_G\times L_{XYZ}+2\times L_\delta+2\times L_r$, is determined by space scale and minimal sampling steps.
1 0 … 1 0 1 … 1 1 1 … 0 0 1 … 0 1 1 … 1
Coordinate x_i in binary format
Coordinate y_i in binary format
Coordinate z_i in binary format
δ_1 and δ_2 in binary format
r_1 and r_2 in binary format

Fig. 5. Chromosome structures.

TABLE I
FEATURES OF DIFFERENT REPRESENTATIONS

	Meaning of a gene	Meaning of a chromo.	Size of a chromo	Feasibility constraints
Chromo. Based on absolute position	$C(g,j)=i>0$ means ACi is the the jth to gate g	Absolute position between aircraft	$N_G\times N_{AC}$	2 constraints [13]
Chromo. Based on relative position	$C(i,j)$=1 means ACj follows ACi to dwell to gate $C(N_{\rm AC}+1,i)$	Absolute position and gate assignments	$N_{AC}\times(N_{AC}+1)$	5 constraints [13]
Binary Chromo.	A digit bit in the binary string	Spatial parameters	Relying on sampling steps	None

The above three chromosome structures are further compared in Table I, from which one can easily see two advantages of our binary representation. The first advantage is that the binary representation has no feasibility problems because no constraints are imposed on chromosomes. The second advantage of the binary representation is its memory-efficiency. The matrix representation based on the relative position of aircraft has a potential O(n×(n+1)) memory problem. That is, if $N_{\rm AC}$ is too large, the memory demand may exceed the available resources of a computer system. In contrast, the binary GA can easily apply to all problem scales, because the length of its binary chromosomes has no direct link to $N_{\rm AC}$. The hybrid GA uses the deterministic ripple-spreading model in Section 4 to reconstruct online the full information of aircraft queues to gates from the simple data stored in a chromosome. In other words, the deterministic model helps the hybrid GA to save the memory allocated to chromosomes. The cost is the computational burden caused by the deterministic model. Fortunately, this added computational burden is largely offset by the reduced computational burden in the evolutionary operations due to the use of binary representation.

As discussed in [13], the relative position between aircraft is an important linkage in solutions to the GAP, and should be used to define common genes in chromosome, i.e., a common relative position represents a common sub-queue. The binary representation based on the ripple-spreading model can handle this kind of common sub-queues quite effectively because if two sets of RPs have similar coordinates in the space, it is very likely they will generate two solutions which have some common sub-queues.

B. Evolutionary Operators

With a problem-specific permutation representation, problem-specific evolutionary operators usually need to be designed and employed. Due to feasibility problems, sometimes crossover has to be discarded. For instance, it is difficult to design an effective crossover, free of feasibility problems, for the chromosome based on the absolute position of aircraft to operate on common sub-queues. Although a so-called crossover operator was proposed in [13], it is actually a complex combination of simple mutations. The chromosome based on the relative position between aircraft makes it possible to design a uniform crossover to identify, inherit and protect common sub-queues, but the operation is computationally expensive [13].

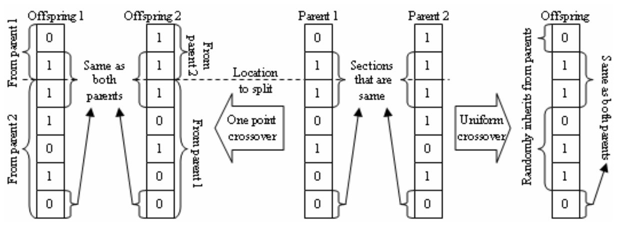


Fig. 6. Uniform crossover vs one point crossover.

In the ripple-spreading GA, since a binary chromosome is based on the value of parameters rather than on the position of aircraft, there is no need for any problem-specific evolutionary operators, and all classic techniques can apply straightforwardly. This full freedom of choosing and designing evolutionary operators mainly comes from no constraint to the binary representation. The mutation operator used in the hybrid GA is to randomly reverse a gene in the chromosome: if a gene C(i) is chosen to mutate at a given probability, then we have

$$C(i)=1-C(i). \tag{10}$$

Crossover is used to exchange sections of chromosomes. In the hybrid GA, we choose the uniform crossover operator, which is probably the most widely used crossover [18]. Thanks to the binary representation, the original uniform crossover can be applied directly to the hybrid GA. Uniform crossover uses two parents to produce only one offspring, and the principle is simple: the *i*th gene in the offspring inherits the *i*th gene of either parent at a half-and-half chance. Fig.6 compares one-point crossover with uniform crossover, from which one can see that one-point crossover is actually a special case of uniform crossover.

C. Heuristic Rules

In permutation representation based GAs for the GAP, many problem-specific heuristic rules are integrated into their evolutionary operations. For example, in [13], the FCFS sequence is used to initialize a proportion of chromosomes, otherwise it is very likely that almost all chromosomes at the beginning of the evolution will be very unfit; the probability of applying the mutation operation largely depends on the P_i of the chosen genes, in order to avoid allocating two aircraft that are far away from each other as successive dwellings at a same gate. Besides, those feasibility constraints have to be checked and satisfied whenever necessary during the evolutionary process.

In comparison, in the evolutionary operations we focus on pure GA-related heuristic rules. For example, evenly distributing a proportion of the chromosomes within the parameterized space when initializing a generation, and adjusting mutation and crossover probability throughout the optimization according to the fitness of each individual chromosome as well as the overall fitness level of the current generation of chromosomes [19]. Besides, some GA-related parameters may be included in the chromosome structure and then get evolved along with those spatial parameters. For instance, we can introduce a GA-related parameter to restrict the area where mutation may apply in a chromosome, and then adjust this GA-related parameter online according to the fitness of a chromosome [20]. Since this GA-related parameter is based on value, and the spatial parameters are also based on value, there is no difference from the viewpoint of chromosome structure. As a result, the GA-related parameter can be directly appended to a chromosome such as illustrated in Fig.5.(d), with no need to modify the evolutionary operators. For those GAs based on permutation representations, to evolve GA-related parameters simultaneously almost means the design of an additional GA totally separated from the original .

VI. Simulation Results

The setup for the experiments in this section is the same as that in [13]. Basically, a typical two-sided parking terminal layout with 20 gates is adopted, the data matrix M_{PWD} is generated according to some simplified formulas, 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. For more details about the setup of the experiments, readers may refer to [13]. The congestion condition is indicated by N_{AC}. For comparative purposes, the GA based on the relative position between aircraft in [13] is also used in the simulation, and denoted as GA1 hereafter (the hybrid GA is denoted as GA2). Due to limited space here we only give in Table II the results of a relatively simple case study, in order to illustrate how different GAs optimize gate assignment. From Table II, one can see that GA2 achieves a reasonably good performance when compared with GA1, which has been proved to be very effective in [13] (the final objective values achieved are given in the brackets following the associated GAs). However, to get general conclusions about the 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 Eq.(5) and Eq.(6) or the multi-objective function in Eq.(4) 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 III to

Table V, where MaxQL and MinQL stands for maximal queue length and minimal queue length, respectively. From Table III to Table V we have the following observations:

TABLE II
RESULT OF GATE ASSIGNMENT IN CASE 1 (N_{AC}=25)

AC Code	P_i (min)	G_i (min)	GA1 (6.65×10^5) E_i (min)	GA1 Gate	GA2 (6.68×10^5) E_i (min)	GA2 Gate
1	28	40	28	3	28	12
2	26	50	26	1	26	14
3	5	40	5	12	5	3
4	12	45	12	15	12	6
5	43	50	43	9	43	18
6	27	45	27	2	27	13
7	34	40	34	20	34	15
8	10	35	10	14	10	5
9	48	30	48	7	48	2
10	56	35	56	14	57	4
11	25	35	25	19	25	10
12	52	35	52	11	53	7
13	7	50	7	17	7	4
14	56	40	57	15	56	5
15	56	60	56	16	57	6
16	49	35	49	10	49	1
17	39	30	39	6	39	17
18	47	40	47	8	47	19
19	13	40	13	16	13	7
20	52	35	52	12	52	3
21	25	50	25	4	25	11
22	35	40	35	5	35	16
23	16	50	16	13	16	8
24	20	35	20	18	20	9
25	56	45	57	17	56	20

- According to Table III to Table V, the overall performance of GA2 is quite similar to that of GA1, which means the new hybrid GA proposed in this paper is at least as good as the well-tuned GA1. In other words, the introduction of the ripple-spreading model is effective and useful in the GAP, and therefore, is worth further investigation.
- In the case of under-congestion (N_{AC}=30), the performance of GA1 is slightly better than that of GA2. This may imply that the chromosome structure based on the relative position of aircraft in queues to gates in [13] work more effectively in relatively simpler environments. This is probably because the associated evolutionary operators, particularly the special uniform crossover proposed in [13] is quite powerful in searching a relatively smaller solution space.
- In the case of congestion (N_{AC}=60), GA2 outperforms GA1 slightly. This may suggest that the ripple-spreading model proposed in this paper is suitable for dealing with more complicated situations, because the queues generated by the ripple-spreading process in the parameterized space may naturally exhibit some good features of quality queues.
- However, in the case of over-congestion, the performance of GA1 is a little better than that GA2 when J_{TPWT} is used as the objective function, while GA2 slightly outperforms GA1 when J_{TPWD} or J_{MOGAP} is used as the objective function. This probably means that, in too complicated situations, neither GA1 nor GA2 works very stably or effectively, and therefore they both need further modifications and improvements. For GA2, a direction for possible improvements is to try other ripple-spreading models, e.g., to redefine the parameterized space and the ripple-spreading process.

Table III J_{TPWD} is used as objective function

($x10^5$)		J_{TPWD}	TPWD(m)	TAWT(min)	MaxQL	MinQL
N_{AC}=30	GA1	7.0330	7.0330	44.7622	29.6	0.2
	GA2	7.0450	7.0450	43.9814	29.6	0.2
N_{AC}=60	GA1	13.2538	13.1538	209.5881	59.3	0.3
	GA2	13.1165	13.1165	211.9472	59.2	0.3
N_{AC}=90	GA1	18.8373	18.8373	455.4340	88.5	0.5
	GA2	18.0661	18.0661	459.3244	88.4	0.5

Table IV J_{TPWT} is used as objective function

($x10^5$)		J_{TPWT}	TPWD(m)	TAWT(min)	MaxQL	MinQL
N_{AC}=30	GA1	1.4595	19.0023	0.0583	2.2	0.9
	GA2	1.4714	18.8923	0.0602	2.2	0.9
N_{AC}=60	GA1	64.2053	37.3578	2.5774	3.8	2.3
	GA2	62.8903	37.9203	2.4846	3.7	2.3
N_{AC}=90	GA1	208.5154	53.0487	8.3508	5.3	4.0
	GA2	211.6872	52.1145	8.5102	5.3	3.9

Table V J_{MOGAP} is used as objective function

($x10^5$)		J_{MOGAP}	TPWD(m)	TAWT(min)	MaxQL	MinQL
N_{AC}=30	GA1	10.8315	15.5086	0.1477	2.1	1.0
	GA2	11.2243	15.6320	0.1502	2.1	1.0
N_{AC}=60	GA1	44.2411	34.1606	2.8031	4.0	2.1
	GA2	42.8094	33.3741	2.7749	3.9	2.1
N_{AC}=90	GA1	102.4846	47.8941	8.4692	5.2	4.0
	GA2	98.0427	46.3172	8.2098	5.1	3.9

VII. CONCLUSION

This paper reports a hybrid GA to tackle the airport Gate Assignment Problem. In the new hybrid GA scheme, the original GAP is cast into a pre-problem, where aircraft waiting to dwell at gates are projected as points in a parameterized space, and a deterministic model inspired by the natural ripple-spreading phenomenon is developed to use relative spatial parameters as input to connect all aircraft points to form some aircraft queues to gates. A very traditional binary GA is then used to evolve these spatial parameters in order to find an optimal or near-optimal solution to the GAP. Since all feasibility problems can be addressed by the deterministic ripple-spreading model in a straightforward way, all classic evolutionary operations can easily apply in the hybrid GA. Because a chromosome only stores the values of a few spatial parameters, the hybrid GA has no memory-inefficiency problems regardless of the problem scale. The effectiveness of the proposed ripple-spreading GA for the GAP problem is illustrated by some simulation results. Further research will include a study and analysis of the problem casting process in more depth and conducting extensive comparative experiments against other GAs and methods for the GAP. It will also be important to consider the multi-objective GAP and collect and use real traffic data to assess the proposed method.

REFERENCES

[1] A. Haghani, and M.C. Chen, "Optimizing gate assignments at airport terminals," *Transportation Research* A, vol. 32, no. 6, pp. 437-454, 1998.

[2] A. Bolat, “Procedures for providing robust gate assignments for arriving aircraft,” *European Journal of Operations Research*, vol. 120, pp. 63–80, 2000.

[3] O. Babic, D. Teodorovic, and V. Tosic, “Aircraft stand assignment to minimize walking,” *Journal of Transportation Engineering*, vol. 110, no. 1, pp. 55–66, 1984.

[4] R.S. Mangoubi, and D.F.X. Mathaisel, “Optimizing gate assignments at airport terminals,” *Transportation Science*, vol. 19, no. 2, pp. 173-188, 1985.

[5] R. Bihr, “A conceptual solution to the aircraft gate assignment problem using 0,1 linear programming,” *Computers & Industrial Engineering*, vol. 19, no. 1-4, pp. 280–284, 1990.

[6] G.D. Gosling, “Design of an expert system for aircraft gate assignment,” *Transportation Research-A*, vol. 24, no. 1, pp. 59–69, 1990.

[7] K. Srihari, and R. Muthukrishnan, “An expert system methodology for an aircraft-gate assignment,” *Computers & Industrial Engineering*, vol. 21, no. 1-4, pp. 101–105, 1991.

[8] J. Xu, and G. Bailey, “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] H. Ding, A. Lim, B. Rodrigues, and Y. Zhu, “The over-constrained airport gate assignment problem,” *Computers & Operations Research*, vol. 32, pp. 1867-1880, 2005.

[10] H. Ding, A. Lim, B. Rodrigues, and Y. Zhu, “New heuristics for the over-constrained airport gate assignment problem,” *Journal of the Operational Research Society*, vol. 32, no. 7, pp. 1867-1880, 2005.

[11] Y. Gu, and C.A. Chung, “Genetic algorithm approach to aircraft gate reassignment problem,” *Journal of Transportation Engineering*, vol. 125, no.5, pp. 384-389, 1999.

[12] A. Bolat, “Models and a genetic algorithm for static aircraft-gate assignment problem,” *Journal of the Operational Research Society*, vol. 52, no. 10, pp. 1107-1120, 2001.

[13] X.B. Hu, E. Di Paolo, “An Efficient Genetic Algorithm with Uniform Crossover for the Multi-Objective Airport Gate Assignment Problem”, *The Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC2007)*, 25-28 Sep 2007, Singapore.

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

[15] Eiben, A.E. and J. E. Smith, *Introduction to Evolutionary Computing*, Berlin, Germany: Springer-Verlag, 2003.

[16] X.B. Hu, and W.H. Chen, “Genetic Algorithm Based on Receding Horizon Control for Arrival Sequencing and Scheduling,” *Engineering Applications of Artificial Intelligence*, vol. 18, no. 5, pp. 633-642, 2005.

[17] X. B. Hu, E. Di Paolo and L. Barnett, “Ripple-Spreading Model and Genetic Algorithm for Random Complex Networks: Preliminary Study”, *The World Congress on Computer Intelligence (WCCI2008)*, Hong Kong, China, 01-06 June 2008.

[18] E. Falkenauer, “The worth of uniform crossover,” *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 1, pp. 782, CEC99, USA, 1999.

[19] J. Zhang, H.S.H. Chung and W.L. Lo, “Clustering-Based Adaptive Crossover and Mutation Probabilities for Genetic Algorithms”, *IEEE Transaction on Evolutionary Computation*, vol.11, pp. 326-335, 2007.

[20] X.B. Hu and S.F. Wu, “Self-Adaptive Genetic Algorithm Based on Fuzzy Mechanism”, *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, 25-28 Sep 2007, Singapore.