Math. Struct. in Comp. Science (2014), vol. 24, iss. 5, e240517, 14 pages. © Cambridge University Press 2014
doi:10.1017/S0960129513000765

Optimised assignment of airport gate configurations using an immune genetic algorithm†

LI WANG, QI-LIN ZHU and XIAO-FANG XU

College of Aeronautical Automation, Civil Aviation University of China, Tianjin, 300300, China
Email: lwang@cauc.edu.cn;xfxu_007@163.com

Received 20 July 2011; revised 18 August 2013

The function of airport gate assignment is to assign appropriate gates for arrival and departure flights and to ensure the flights are on schedule. A key task of airport ground operations is assigning the airport gate with high efficiency and a reasonable arrangement. In this paper we establish an optimised model based on the characteristics of both the flights (the flight type, flight down time and the number of passengers) and the airport gates (the ease of access of the airport gate). We give a representation of the solution space and a direct graph model of the airport gate configuration based on dynamic scheduling parallel machines. We design a method for solving the airport gate configuration based on an immune genetic algorithm. The simulation results show the effectiveness of this model and algorithm.

1. Introduction

The airport gate assignment task assigns an appropriate gate to every arrival and departure flight during some period of future time, ensures the flights are on schedule and provides gates for the passengers to get on or off the airplanes. A reasonable assignment of the gates is a key task of airport ground operations. Factors affecting it are the route type, route distance, aircraft type, flight number, flight density, flight down time and the walking distance for passengers transiting between inbound and outbound flights. However, civil aviation is currently developing rapidly and airport facilities have many shortcomings. These shortcomings limit the size of airport operations, so an optimised assignment of the airport gate configuration becomes an important problem.

To solve this problem, scientists at home and abroad have proposed several analytical methods. For example, Wen *et al.* (2004) proposes a sorting model for airport gate assignment and finds an assignment to make every flight have an airport gate. However, this model does not consider passenger requirements or any benefits to the airport. Gosling (2000) and Su and Srihari (2001) approach the problem through the use of expert systems. These systems combine the assignment principle with a knowledge system, and consider many non-quantifiable criteria. However, the results are not ideal because they are

† The work for this paper was supported by the Civil Aviation University of China fund 'based on infrared image processing for TCAS system depth testing system maintenance' (item number 2010kyE07) and by the National Natural Science Foundation of China (60472130).

Downloaded from https://www.cambridge.org/core. University of Rhode Island Library, on 23 Aug 2019 at 23:55:53, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129513000765

limited by the scope of the search and some critical factors are ignored. Wang *et al.* (2006), Wang (2007), Ding *et al.* (2005), Hu and Di Paolo (2007), Drexl and Nikulin (2008), Yan and Tang (2007) and Ju and Xu (2008) use the method of mathematical programming. This method analyses the feasibility of a configuration and how to deploy it by selecting an optimised function using 0-1 integer programming. The main difficulty with this method is choosing the objective function. Because there are so many factors influencing the airport gate, we propose an objective function addressing actual practical needs, and design a rapid and effective algorithm considering all the required factors. The work in the current literature optimises the airport gate configuration from the perspective of passengers, without considering the airport gate, flight type, flight number, flight density, downtime and so on.

In the current paper, we establish an optimised model based on the characteristics of both the flights (the flight type, flight downtime and the number of passengers) and the airport gates (the ease of access of the airport gate). We give a representation of the solution space and a direct graph model of the airport gate configuration based on dynamic scheduling parallel machines. We design a method for solving the airport gate configuration based on an immune genetic algorithm. The simulation results show the effectiveness of this model and algorithm.

2. Description of the airport gate assignment task

The airport gate assignment task is to produce a reasonable assignment of a limited number of parking bays to a large number of flights considering the optimisation goals of passenger transit time, flight type, flight density, flight number and downtime. That is to say, the aim is to achieve the greatest passenger satisfaction, highly efficient airport ground operations and effective benefits to the airport. The passenger transit time is the time taken for:

— departing passengers to arrive at the boarding gate from the security gate;
— arriving passengers to arrive at the baggage claim from the boarding gate; and
— transiting passengers to move between the arrival and departure gates.

The flight type measures the degree of importance of the flights to distinguish between important flights (for example, international and high volume flights) and non-critical flights. The flight downtime is the time spent at the airport gate by the flights. Depending on the actual needs, we need to know the number of airport gates, the number, type and downtime of the landing flights every day and the case of transiting passengers boarding.

Wang (2010) gives a description of airport gate assignment that provides a possible distributed assignment to maximise the sum of the flight and distributed airport gate characteristics. The objective function for this method is

$$\max \pi(A_{ss}) = \sum_{i \in l} \gamma_l * \sigma(A_{SS_i}).$$

The optimised assignment of the flight collection A is denoted by $A^*_{SS}(A)$.

Downloaded from https://www.cambridge.org/core. University of Rhode Island Library, on 23 Aug 2019 at 23:55:53, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129513000765

3. Optimised assignment of airport gate configurations based on an immune genetic algorithm

Wang (2010) gives a model of optimised airport gate assignment based on dynamic scheduling parallel machines.

The model is $\bar{A} = \{\bar{A}_1, \bar{A}_2, \cdots, \bar{A}_k\}$, where the set is made up of all the schedulings of the set P of airport gates, and it is limited by the following ordered product:

$$ {}_C\Omega_l^n = {}_{C_1}\Omega_l^{n_1} \times {}_{C_2}\Omega_l^{n_2} \times \cdots \times {}_{C_k}\Omega_l^{n_k} = \prod_{i=1}^{k} {}_{C_1}\Omega_l^{n_i} $$

and every scheduling of the set P is defined so that

$$ \begin{aligned} \omega &= \omega_1 \times \omega_1 \times \cdots \times \omega_k \\ &= \left(\begin{pmatrix} w_{11} \\ w_{21} \vdots \\ w_{k1} \end{pmatrix} \begin{pmatrix} w_{12} \\ w_{22} \vdots \\ w_{k2} \end{pmatrix} \cdots \begin{pmatrix} w_{1l} \\ w_{2l} \vdots \\ w_{kl} \end{pmatrix} \right) \in {}_C\Omega_i^n \end{aligned} \tag{1} $$

where $\omega_i \in {}_{C_i^{\omega j-1}}\Omega_l^{n_i}$, $i \in \underline{k}$. Alternatively, we can use a matrix form: $\omega = (\omega_{ij})$, $i \in \underline{k}$, $j \in \underline{l}$.

Every possible distributed assignment ω of the solution space is treated as an individual. According to the characteristics of the space structure of the solution, every individual ω is made up of k genes, that is, $(w_1, w_2, \cdots, w_k)$. The following sections describe the immune genetic algorithm based on this model of the solution space.

3.1. *Code for individuals*

In accordance with the characteristics of the model of dynamic scheduling parallel machines for the optimised assignment of an airport gate configuration of the type (1), we identified the following two-dimensional array for the encoding:

$$ \begin{aligned} \omega &= \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_k \end{pmatrix} = \begin{pmatrix} \omega_{11} & \omega_{12} & \cdots & \omega_{1l} \\ \omega_{21} & \omega_{22} & \cdots & \omega_{2l} \\ \vdots & \vdots & \vdots & \vdots \\ \omega_{kl} & \omega_{kl} & \cdots & \omega_{kl} \end{pmatrix} \\ &= (\omega_{ij}), i \in \underline{k}, j \in \underline{l} \end{aligned} \tag{2} $$

where

$$ \omega_i \in {}_{C_i^{\omega_{i-1}}}\Omega_l^{n_i} \qquad (i \in \underline{k}). $$

So, an individual ω is a two-dimensional array made up of k rows and l columns. Each row of the array represents a gene, and an individual $(w_1, w_2, \cdots, w_k)$ is made up of k genes. Each gene ω_i includes l pieces of information $(w_{i1}, \cdots, w_{il})$, each with a range of values $\{0 \cdots, n\}$. We describe the fact that flight x is parking in the airport gate j by

$$ \omega_{ij} = x, \quad (x \in \underline{n}), $$

Downloaded from https://www.cambridge.org/core. University of Rhode Island Library, on 23 Aug 2019 at 23:55:53, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129513000765

and use $\omega_{ij} = 0$ to describe the fact that there is no flight in airport gate j. An individual can also be described by a one-dimensional array:

$$(\omega_{11}, \cdots, \omega_{1l}, \omega_{21}, \cdots, \omega_{2l}, \cdots, \omega_{k1}, \cdots, \omega_{kl}). \tag{3}$$

The length of a gene is thus $k \times l$.

Because the optimisation problem for airport gate assignment is subject to restrictions, we require the genetic composition of each individual to satisfy compatibility constraints:

$$\omega_i \in {}_{C_i^{\omega_{i-1}}}\Omega_l^{n_i}, i \in \underline{k}.$$

In other words, when we apply the crossover and mutation operators, we need to test whether the results satisfy this condition since only genetic changes that meet this condition can form new and effective individuals.

3.2. *Fitness function*

The fitness function is used to evaluate the individuals and is the basis for the development of the optimisation process. For simple optimisation problems, we can usually directly convert the objective function into the adaptation value function: for example, we could define the adaptation value $f(X)$ of an individual X to be

$$f(X) = M - c(X)$$

or

$$f(X) = e^{-\alpha c(X)}.$$

Where M is a large enough positive number, $c(X)$ is the objective function value of the individual and $\alpha > 0$. However, for the optimisation of complex problems, we often need to construct an appropriate evaluation function for optimisation in the GA. For the current paper, we will use the objective function directly as the fitness function, so

$$\pi(\omega) = \sum_{i=1}^{k} \sum_{j=1}^{l} c(\Omega_{ij}) * \gamma_j$$

where:

— $c(\omega_{ij})$ describes the characteristic of the flight ω_{ij}, and $c(0) = 0$; and
— γ_j describes the characteristic of airport gate j.

The optimal airport gate assignment corresponds to finding the distribution plan giving the maximum value of the objective function. Hence, the degree of adaptation of individuals measured by the corresponding objective function value is also large, and maximal for the best individual.

From the objective function, we define the function

$$\pi(\omega_{ij}) = \sum_{j-1}^{l} c(\omega_{ij}) * \gamma_j$$

reflecting the quality of the gene ω_i, so a gene with a large function value is a good gene.

Downloaded from https://www.cambridge.org/core. University of Rhode Island Library, on 23 Aug 2019 at 23:55:53, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129513000765

Definition 3.1. If

$$\pi(\omega_i^*) = \max_{\omega_i \in C_i^{\omega_{i-1}} \Omega_l^{n_i}} \pi(\omega_i) \qquad (i \in \underline{k}),$$

we consider ω_i^* to be a *good gene* of

$$\omega_i \in C_i^{\omega_{i-1}} \Omega_l^{n_i}.$$

3.3. *Population initialisation*

The initial population is the starting point for iteration of the genetic algorithm and consists of a certain number of individuals. Here we assume the initial population is composed of s individuals, which includes a satisfactory solution s_1 of the problem obtained by heuristic algorithms and s_2 randomly selected individuals from the solution space, that is $s = s_1 + s_2$ individuals in total.

We will now describe how we randomly select the s_2 individuals in the solution space. We begin by identifying the first gene by random selection for the s_2 individuals formed by

$$\omega_i \in C_i \Omega_l^{n_i} \qquad (i \in \underline{s_2}).$$

For the conflict flight set

$$\overline{A_1} = \{{}_1\overline{a_1} \cdots, {}_1\overline{a_{m_1}}\}$$

containing a set of m_1 extended flights, we randomly order parking bays for each flight. So, for flight ${}_1\overline{a_1}$, we randomly select a parking bay $P({}_1\overline{a_1})$ from the airport gate set P, then distribute flight ${}_1\overline{a_1}$ to the parking bays $P({}_1\overline{a_1})$. For flight ${}_1\overline{a_2}$, we randomly select a parking bay $P({}_1\overline{a_2})$ from the rest of the airport gate set P after removing $P({}_1\overline{a_1})$, then distribute flight ${}_1a_2$ to the parking bay $P({}_1\overline{a_2})$. In this way, we randomly identify a distribution

$$\{P({}_1\overline{a_2}), P({}_1\overline{a_2}), \cdot, P({}_1\overline{a_{m_1}})\}$$

for the flight set

$$\overline{A_1} = \{{}_1\overline{a_1}, \cdot, {}_1\overline{a_{m_1}}\}.$$

For any flights not assigned parking bays, we assume that the flight number is 0, that is, we have

$$\{P({}_1\overline{a_2}), P({}_1\overline{a_2}), \cdots, P({}_1\overline{a_{m_1}}), P(0), \cdots, P(0)\}$$

where the number of $P(0)$ elements is $l - m_1$. Consequently, all flights are allocated parking bays. Now, taking the airport gate numbers in ascending order, we will regard the flight number being allocated in the first parking bay as the first element, the flight in the second parking bay as the second element, and so on until all the parking bays are filled. We describe this assignment using the set

$$\{{}_1\widetilde{a}_1, {}_1\widetilde{a}_2, \cdots, {}_1\widetilde{a}_l\},$$

where

$$P({}_1\widetilde{a}_i) = i \qquad (i \in \underline{l}).$$

Downloaded from https://www.cambridge.org/core. University of Rhode Island Library, on 23 Aug 2019 at 23:55:53, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129513000765

Hence, we get the first gene

$$\omega_1^1 = (\omega_{11}^1, \cdots, \omega_{1l}^1)$$

for the first individual, and we then get the first gene of all the individuals using the same method. We then need to identify the second gene for each individual. For

$$\omega_1^i \in {}_1\Omega_l^{n_1} \qquad (i \in \underline{s_2}),$$

we randomly select

$$\omega_2^i \in {}_{C_2^{\omega_1^j}}\Omega_l^{n_2},$$

as the second gene of the individual i. We continue in this way until all

$$\omega_k^i \in {}_{C_k^{\omega_{k-1}^j}}\Omega_l^{n_2} \qquad (i \in \underline{s_2})$$

have been identified, and thus s_2 individuals will also have been identified. Combining these with the s_1 satisfactory solution to the problem given by the heuristic, we will get $s = s_1 + s_2$ individuals as the initial population. The initial population is denoted by

$$\widetilde{\Omega_0} = \{{}_1\omega_0, {}_2\omega_0, \cdots, {}_s\omega_0\}, \tag{4}$$

and the population of the jth generation is denoted by

$$\widetilde{\Omega_j} = \{{}_1\omega_j, {}_2\omega_j, \cdots, {}_s\omega_j\} \tag{5}$$

where

$${}_i\omega_j = \{{}_i\omega_j^1, {}_i\omega_j^1, \cdots, {}_i\omega_j^k\} \qquad (i \in \underline{s}),$$

and ${}_i\omega_q^j$ represents the q gene of the i individual in the initial population.

3.4. *The genetic operators*

The basic idea of the genetic algorithm design is survival of the fittest, and this should be reflected in the selection, crossover and mutation operators, which should also take account of the impact on the algorithm's efficiency and performance.

3.4.1. *The selection operator.* The copy operation avoids the loss of effective genes so that good individual performance has a greater probability of survival, thereby enhancing the global convergence and computational efficiency. The most common methods are scale replication and rank-based replication. Scale replication selects the appropriate value for the individual, which is proportional to the probability of individual adaptation. Rank-based replication selects the appropriate individual according to the ranking of the individual in the population. As for population replacement, the adoption of the program can replace some individuals or the entire group.

The size s of the population is given by Equation (5), where each individual is represented as a two-dimensional $k \times l$ array. We will now give some definitions:

Downloaded from https://www.cambridge.org/core. University of Rhode Island Library, on 23 Aug 2019 at 23:55:53, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129513000765

— **Similarity**

Consider any two individuals,

$$
\begin{aligned}
{}_{i_1}\omega_{j_1} &= \{{}_{i_1}\omega_{j_1}^1, {}_{i_1}\omega_{j_1}^2, \cdots, {}_{i_1}\omega_{j_1}^k\} \\
{}_{i_2}\omega_{j_1} &= \{{}_{i_2}\omega_{j_1}^1, {}_{i_2}\omega_{j_1}^2, \cdots, {}_{i_2}\omega_{j_1}^k\}
\end{aligned}
$$

where $i_1, i_2 \in \underline{s}$ with $i_1 \neq i_2$, and j_1 represents the generation number of the evolution. We assume that the adaptation of the individuals ${}_{i_1}\omega_{j_1}$ and ${}_{i_2}\omega_{j_1}$ are $\pi({}_{i_1}\omega_{j_1})$ and $\pi({}_{i_2}\omega_{j_1})$, respectively. Then, assuming ε is an appropriate small number ($\varepsilon > 0$), if

$$
1 - \varepsilon \leqslant Q({}_{i_1}\omega_{j_1}, {}_{i_2}\omega_{j_1}) = \frac{\pi({}_{i_1}\omega_{j_1})}{\pi({}_{i_2}\omega_{j_1})} \leqslant 1 + \varepsilon \tag{6}
$$

holds, we say the two individuals ${}_{i_1}\omega_{j_1}$ and ${}_{i_2}\omega_{j_1}$ are similar.

$Q({}_{i_1}\omega_{j_1}, {}_{i_2}\omega_{j_1})$ measures the degree of similarity of the quality of the individuals ${}_{i_1}\omega_{j_1}$ and ${}_{i_2}\omega_{j_1}$, and ε is the defined similarity threshold. Here, the quality is the quality of individual fitness, so the greater the individual fitness the greater the quality. For example, if $\varepsilon = 0.02$ and the condition holds, we say the two individuals (that is, ${}_{i_1}\omega_{j_1}$ and ${}_{i_2}\omega_{j_1}$) have 98% similarity in their quality.

This definition is a new method for defining the similarity of individuals and has two advantages compared with methods based on information entropy and Euclidean distance:

(a) it is more intuitive;
(b) it requires less computation.

— **Concentration**

As for particular individuals, the scale of the individuals is s, and the number of individuals being similar to the individual $q \in \underline{s}$ (including individual q itself) is called the concentration of individual q, and denoted c_q.

— **Expected rate of reproduction**

As for particular individuals, the scale of the individuals is s, and the expected reproduction rate e_q of the individual $q \in \underline{s}$ is defined to be

$$
e_q = \frac{\pi(q\omega_j)}{(c_q)^\beta} \tag{7}
$$

where:

- $\pi(q\omega_j)$ is the fitness of the individual q;
- c_q is the concentration of the individual q;
- β is a parameter reflecting the relative importance of the concentration and fitness for the reproduction rates.

— **Copy (clone) the probability**

As for particular individuals, the scale of the individuals is s, and we assume the expected rate of reproduction of individual $q \in \underline{s}$ is e_q. The probability P_q that individual q is chosen to be copied (cloned) is then

$$
P_q = \frac{e_q}{\sum_{i=1}^{s} e_i}. \tag{8}
$$

Downloaded from https://www.cambridge.org/core. University of Rhode Island Library, on 23 Aug 2019 at 23:55:53, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129513000765

Formula (8) shows that the probability of selection in the immune genetic algorithm does not just depend on the individual fitness but also depends on the concentration of the individual. This is the biggest difference between the immune genetic algorithm and the genetic algorithm.

We adopt the proportional selection method in the current paper. The basic idea is that the probability of each individual being selected is proportional to its fitness and inversely proportional to the concentration: in other words, the individual breeds in the population to the next generation in accordance with the probability P_q of expected reproduction, which is proportional to the individual. To facilitate the crossover operation, an even number of individuals needs to be selected.

For the selected individuals, we now need to implement the crossover and mutation operations.

3.4.2. *The crossover operator*. The role of the crossover operation is to produce a new individual in the solution space for the effective search, while reducing the failure probability of effective models. The word crossover refers to the fact that part of the structure of the parent individuals is replaced and reorganised to generate a new individual. Common variants include single-point crossovers, multi-point crossovers, arithmetic crossovers, partially mapped crossovers, order crossovers and cycle crossovers. The following describes the implementation of the single-point crossover and multi-point crossover with cross probability we use here. During the crossover operation, the individuals are divided into several groups, with each part composed of a group of two individuals to complete the single-point and multi-point crossover operations.

The single-point crossover randomly selects two individuals:

$$\begin{aligned} {}_{i_1}\omega_{j_1} &= \{{}_{i_1}\omega_{j_1}^1, {}_{i_1}\omega_{j_1}^2, \cdot, {}_{i_1}\omega_{j_1}^k\} \\ {}_{i_2}\omega_{j_1} &= \{{}_{i_2}\omega_{j_1}^1, {}_{i_2}\omega_{j_1}^2, \cdot, {}_{i_2}\omega_{j_1}^k\} \end{aligned}$$

where

$$i_1, i_2 \in \underline{s}, \qquad i_1 \neq i_2.$$

We then randomly select a point $r \in \underline{k}$, with $r \geqslant 2$. If

$$\begin{aligned} {}_{i_1}\omega_{j_1}^{\gamma} &\in {}_{C_r^{i_2}\omega_{j_1}^{r-1}}\Omega_l^{n_r} \\ {}_{i_2}\omega_{j_1}^{\gamma} &\in {}_{C_r^{i_2}\omega_{j_1}^{r-1}}\Omega_l^{n_r} \end{aligned} \tag{9}$$

(that is, if the constraints on the problem of optimal allocation of parking bays and the compatibility condition between genes constituting the same individual are satisfied), then performing the crossover operation to generate two offspring individuals gives

$$\begin{aligned} {}_{i_1}\omega_{j_1} &= \{{}_{i_1}\omega_{j_1}^1, {}_{i_1}\omega_{j_1}^2, \cdots, {}_{i_1}\omega_{j_1}^{\gamma-1}, {}_{i_1}\omega_{j_1}^{\gamma}, {}_{i_1}\omega_{j_1}^{\gamma+1}, \cdots, {}_{i_1}\omega_{j_1}^k\} \\ {}_{i_2}\omega_{j_1} &= \{{}_{i_2}\omega_{j_1}^1, {}_{i_2}\omega_{j_1}^2, \cdots, {}_{i_2}\omega_{j_1}^{\gamma-1}, {}_{i_2}\omega_{j_1}^{\gamma}, {}_{i_2}\omega_{j_1}^{\gamma+1}, \cdots, {}_{i_2}\omega_{j_1}^k\}. \end{aligned} \tag{10}$$

If the compatibility conditions are not met, the crossover operation is not performed.

Downloaded from https://www.cambridge.org/core. University of Rhode Island Library, on 23 Aug 2019 at 23:55:53, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129513000765

The multi-point crossover is treated similarly. After randomly selecting a multi-point, we first need to determine if the compatibility conditions of the intersection point for genes are satisfied: if they are, the crossover operation is carried out; if not, the crossover is not performed.

3.4.3. *The mutation operator.* Mutation refers to changing part of the parent structure of an individual and the operation of generating new individuals. Mutation can help increase the diversity of the population. There are various types of mutation operator. A commonly used operator is replacement, which replaces the original gene in a location with another gene. Another operator is a disturbance operator, which is attached to the original individual to achieve a certain variation of the disturbance mechanism. Other operators include: interchangeable variation; reverse-type variation; and plug-in mutation.

For the parent individual ${}_i\omega_j = \{{}_i\omega_j^1, {}_i\omega_j^2, \cdot, {}_i\omega_j^k\}$, we implemented single-point and multi-point mutation for the current paper, according to the rate of the variation. Single-point mutation varies a single point in an individual. We first randomly select a point

$$q \in \underline{k}$$

and check if

$${}_i\omega_j^q \in {}_{C_q^{i\,\omega_j^{q-1}}}\Omega_l^{n_q}$$
$${}_i\omega_j^{q+1} \in {}_{C_{q+1}^{i}\ {}^{\omega_j^q}}\Omega_l^{n_q+1},$$

that is, whether the constraints on the problem for optimal allocation of parking bays and the compatibility condition between genes constituting the same individual are satisfied. If they are, we replace ${}_i\omega_j^q$ with ${}_i\omega_j^{q'}$, and get the offspring individual

$${}_i\omega_j = \{{}_i\omega_j^1, {}_i\omega_j^2, \cdots, {}_i\omega_j^{q-1}, {}_i\omega_j^q, {}_i\omega_j^{q+1}, \cdots, {}_i\omega_j^k\}. \tag{11}$$

Multi-point mutation is treated similarly by first randomly selecting multiple points and then checking whether the compatibility conditions of the variation points for the genes are met: if they are, the mutation operation is carried out; if not, the mutation is not performed.

3.4.4. *The immune operator.* For an individual

$${}_i\omega_j = \{{}_i\omega_j^1, {}_i\omega_j^2, \cdots, {}_i\omega_j^k\},$$

vaccination is used to modify some locus on the gene in accordance with prior knowledge so that individuals with higher value have a higher probability of fitness.

For a population

$$\widetilde{\Omega_j} = \{{}_1\omega_j, {}_2\omega_j, \cdots, {}_s\omega_j\}, \tag{12}$$

vaccinating the population refers to an operation in which we first randomly select a proportion α ($0 < \alpha \leqslant 1$) of members of the population (so $s_\alpha = \alpha s$) and then replace individual genes in each of the selected members with good genes (as defined in Definition 3.1).

Downloaded from https://www.cambridge.org/core. University of Rhode Island Library, on 23 Aug 2019 at 23:55:53, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129513000765

For an extracted individual

$$ {}_i\omega_j = \{{}_i\omega_j^1, {}_i\omega_j^2, \cdots, {}_i\omega_j^k\}, $$

we need to vaccinate each locus. For the $q \in \underline{k}$ gene locus, the vaccination operation is as follows.

We assume that the good gene according to Definition 3.1 at the gene locus is ${}_i\omega_j^*$. If

$$ {}_i\omega_j^* \in {}_{C_q^{i\,\omega_j^{q-1}}}\Omega_l^{n_q} $$
$$ {}_i\omega_j^{q+1} \in {}_{C_{q+1}^{i\,\omega_j^*}}\Omega_l^{n_q+1}, $$

then we replace the $q \in \underline{k}$ gene of the individual

$$ {}_i\Omega_j = \{{}_i\omega_j^1, {}_i\omega_j^2, \cdots, {}_i\omega_j^k\}, \tag{13} $$

with the good gene ${}_i\omega_j^*$ to generate the new individual

$$ {}_i\omega_j = \{{}_i\omega_j^1, {}_i\omega_j^2, \cdots, {}_i\omega_j^{q-1}, {}_i\omega_j^*, {}_i\omega_j^{q+1}, \cdots, {}_i\omega_j^k\} \tag{14} $$

However, if the compatibility conditions are not met, the vaccination is not carried out. Obviously, the fitness function value for the new individual is greater than for the original individual, so it is a better individual.

3.4.5. *Normalising the offspring population.* Once the selection, crossover, mutation and vaccination operations have been carried out on the current parent population

$$ \widetilde{\Omega}_j = \{{}_1\omega_j, {}_2\omega_j, \cdots, {}_s\omega_j\}, $$

the result will be a a new group of individual offspring denoted by $\widetilde{\Omega}_{j+1}$. We then apply the parent selection operator to the selected group of individual offspring, which is still denoted by $\widetilde{\Omega}_{j+1}$. To maintain the population size at s:

— If the number of the individuals in the offspring population $\widetilde{\Omega}_{j+1}$ is greater than s, we need to select s individuals of the parent population as a collection $\widetilde{\Omega}_{j+1}$.

— If the number of individuals in the offspring population $\widetilde{\Omega}_{j+1}$ is less than s, we go back to the individuals in the original parent population

$$ \widetilde{\Omega}_j = \{{}_1\omega_j, {}_2\omega_j, \cdots, {}_s\omega_j\} $$

that were not selected by the selection operator, and add the individual with the maximum value of the fitness function, and so on until the size of the offspring population is again s.

3.5. *The elite retention policy*

We use the 'elitist' strategy for the genetic algorithm. The ability to converge to the global optimal solution is the main objective of genetic algorithms. G. Rudolph used Markov chain theory to prove that if we use a canonical genetic algorithm, CGA (which only uses crossover, mutation and selection as the three standard genetic operators), and if the

Downloaded from https://www.cambridge.org/core. University of Rhode Island Library, on 23 Aug 2019 at 23:55:53, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129513000765

crossover rate is $P_c \in (0, 1)$, the mutation rate is $P_m \in (0, 1)$ and we adopt the proportional selection method, then it cannot converge to the global optimal solution.

To prevent the best individual in the current group being lost in the next generation, which would mean the genetic algorithm did not converge to the global optimal solution, K. A. de Jong proposed the 'elitist selection strategy' (which is also referred to as the 'elitist strategy'). The idea is that the best individual (called the elite individual) appearing so far in the evolution of the group is always copied to the next generation without any changes – this selection operation is also known as replication.

Suppose the best individual in generation t of the evolution of the genetic algorithm is ω_t^*, and suppose the next generation is $\widetilde{\Omega}_{t+1}$. Then, if ω_t^* does not already appear in $\widetilde{\Omega}_{t+1}$, we add it. If we do this, it will make the size of the new generation $s + 1$, so to maintain a constant population size, we remove the individual with the smallest fitness from the new generation. The elite individual is the highest fitness individual that can be selected by the genetic algorithm in the evolution of the population to the current date.

The advantage of using the elitist strategy is that the best individuals so far cannot be lost or damaged by the selection, crossover and mutation operations in the evolutionary process of the genetic algorithm. The elitist strategy has a significant role in improving the global convergence of the standard genetic algorithm – the standard genetic algorithm with the elitist strategy is globally convergent.

3.6. *The algorithm – parameter selection, termination condition and steps*

The convergence theory for genetic algorithms describes the limit properties of genetic algorithms with probability 1 convergence, so our goal is to improve the convergence rate, which affects the parameter selection and the design of the algorithm.

3.6.1. *Parameter selection.* The size of the population is one of the factors determining the performance and efficiency of the algorithm. Typically, if the population is too small to provide sufficient sampling points, the result is a poor performance of the algorithm, which may not even give a feasible solution of the problem. So the population must be large enough to prevent the occurrence of premature convergence, but larger populations will undoubtedly increase the computation time, so the convergence time may become too long. Of course, a certain number of species in the optimisation process are required to allow changes.

The crossover probability is used to control the frequency of crossover operations. If the probability is too large and the population updated too often, high-fitness value individuals may be destroyed too quickly. However, if the probability is too small, the crossover operation is rarely implemented, which will lead to search stagnation.

The mutation probability is an important factor in increasing population diversity. In the genetic algorithm based on binary encoding, a low mutation rate is enough to prevent the entire group of genes in any position from changing. So the probability rate may be too small to produce new individuals, but if it is too large, it makes the genetic algorithm a random search algorithm.

Downloaded from https://www.cambridge.org/core. University of Rhode Island Library, on 23 Aug 2019 at 23:55:53, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129513000765

Table 1. *An example – the error rate is calculated with reference to the objective function upper bound (6,529,041).*

Evolution generation	0	20	40	60	80	90
Objective function value	5,930,492	6,030,222	6,096,818	6,163,414	6,220,870	6,220,870
Error rate (%)	9.17	7.64	6.62	5.60	4.72	4.72
Computing time (minutes)	0	11	20	27	34	40

3.6.2. *Termination condition.* The most common termination conditions are to reach the biggest evolutionary step that has been set by the people, or when the cadres with the best fitness value do not change significantly and continuously.

3.6.3. *Algorithm steps.* The algorithm consists of the following steps:

(1) Create the initial parent population

$$\widetilde{\Omega}_J = \{{}_1\omega_j, {}_2\omega_j, \cdots, {}_s\omega_j\},$$

and determine the fitness of each individual.
Then store the individual with the largest fitness (the elite individual) in a dedicated variable.

(2) If this is the first pass, go to step 5, otherwise, continue to the next step.

(3) Determine the fitness of each individual.
If this generation of individuals does not contain any individuals matching the fitness of the current elite group of individuals, then copy the elite individual stored in the special variable to this generation and remove the individual with the smallest fitness.

(4) If the largest fitness value of the individuals in this generation is greater than the fitness value of the current elite individual, copy the individual with the largest fitness value to the special variable as the new elite individual.

(5) Using the similarity of the individuals, calculate the concentration of each individual.

(6) Calculate the expected reproductive rate and selection probability of each individual, according to the selection probability, and use the probability proportional selection strategy to implement the selection and copy operations.
To aid the description of the remaining steps, let the group of selected individuals be $\widetilde{\Omega}'_j$.

(7) Apply the crossover operation to the individuals in the population $\widetilde{\Omega}'_j$ to obtain population $\widetilde{\Omega}''_j$.

(8) Apply the mutation operation to the population $\widetilde{\Omega}''_j$ to obtain the population $\widetilde{\Omega}'''_j$.

(9) Vaccinate the population $\widetilde{\Omega}'''_j$ to get the new population $\widetilde{\Omega}_{j+1}$, and ensure that its size is still s.

(10) Determine whether the termination conditions are met: if they are, the algorithm stops with the final population as the output; otherwise, if the conditions are not met, return to Step 3 and continue.

Downloaded from https://www.cambridge.org/core. University of Rhode Island Library, on 23 Aug 2019 at 23:55:53, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129513000765

4. An example and analysis of results

As a test example, we chose a busy domestic airport with 20 stands and 120 flights in the peak hours (a period of 6 hours) in one day. We wrote the program using MATLAB software, and ran it on an ordinary desktop computer (Lenovo brand with 2G memory and 2.98GHZ CPU running Windows XP) taking one hour 10 minutes. We set the crossover rate to be $P_c = 0.6$ and the mutation rate to be $P_m = 0.2$, and set $\beta = 1.5$ and $m = 30$. The termination condition was satisfied when there was no change in the best individual fitness for 10 consecutive generations. Table 1 show the results and the evolution of the convergence process.

5. Conclusion

In this paper, we have established an optimised model for airport gate assignment to aid airport operation and management based on the characteristics of the flight (the flight type, flight down time and the number of passengers) and airport gate (ease of access). We have given a representation of the solution space and a direct graph model of the airport gate configuration based on dynamic scheduling parallel machines. We then designed a solving method for the airport gate configuration based on an immune genetic algorithm. The simulation results show the effectiveness of this model and algorithm.

It should be noted that the assignment of gates is a complex issue. In this paper, we have considered an optimisation method of gate assignment based on the immune genetic algorithm, but there are some unresolved practical issues. In particular, we have assumed that there are always enough gates, so the problem when there is a shortage of free gates and flights need to wait before landing needs further study – the dynamic allocation of flight delays and other problems also requires further study.

References

Ding, H., Lim, A., Rodrigues, B. and Zhu, Y. (2005) The over-constrained airport gate assignment problem. In: Meltzer, B. and Michie, D. (eds.) *Computers & Operations Research*, Edinburgh University Press **32** 1867–1880.

Drexl, A. and Nikulin, Y. (2008) Multicriteria airport gate assignment and Pareto simulated annealing. *IIE Transactions (Institute of Industrial Engineers)* **40** (4) 385–397.

Gosling, G. (2000) Design of an expert system for aircraft gate assignment. *Transportation Research A* **24** (1) 59–69.

Hu, X. B. and Di Paolo, E. (2007) An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. *IEEE Congress on Evolutionary Computation* 55–62.

Ju, S.-M. and Xu, L. (2008) Airport gate assignment problems based on GSAA. *Journal of Transportation Systems Engineering and Information Technology* **8** (1) 138–143.

Su, Y. Y and Srihari, K. (2001) A knowledge-based aircraft-gate assignment advisor. *Computers and Industrial Engineering* **25** (2) 123–126.

Wang, L. (2007) Optimization of Civil Airport Gate Computer Simulation. *The 26th Chinese Control Conference* 351–355.

Downloaded from https://www.cambridge.org/core. University of Rhode Island Library, on 23 Aug 2019 at 23:55:53, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129513000765

Wang, L. (2010) Optimized Assignment of Civil Airport Gate. *2010 International Conference on Intelligent System Design and Engineering Application* 33–38.

Wang, L., Liu, C. Y. and Tu, F. S. (2006) Optimized Assignment of Civil Airport Gate. *Journal of Nanjing University of Aeronautics and Astronautics* **38** (4) 433–437.

Wen, J., Sun, H., Xu, J. and Liang, Z. J. (2004) The distribution of airport gate based on sorting algorithm. *Systems Engineering* **22** (7) 102–106.

Yan, S. and Tang, C.-H. (2007) A heuristic approach for airport gate assignments for stochastic flight delays. *European Journal of Operational Research* **180** (2) 547–567.

Downloaded from https://www.cambridge.org/core. University of Rhode Island Library, on 23 Aug 2019 at 23:55:53, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129513000765