Steady State Evolutionary Algorithm and Operators for the Airport Gate Assignment Problem

Amadeo Asco*

**School of Computer Science, University of Nottingham, UK*

Received: May 28, 2019; Accepted: July 16, 2019; Published: August 12, 2019

***Corresponding author:** Amadeo Asco, School of Computer Science, University of Nottingham, UK, E-mail: a.asco@bocaditos.co.uk

Abstract

Correct assignment of airport resources can greatly affect the quality of service which airlines and airports provide to their customers. Good assignments can help airlines and airports keep to published schedules, by minimising changes in published schedules, reducing delays, and considering the different stakeholder interests. When taking into account the passenger's walking distance, the Airport Gate Assignment Problem (AGAP) can be modeled by analogy with the NP-hard quadratic assignment problem. Also given the expected increases in civil air traffic, the complexities of the resource assignment problems continue to increase. For these reasons, as well as the dynamic nature of the problems, scheduling has become more difficult.

The Steady State Evolutionary Algorithm (SSEA) used in the Airport Baggage Sorting Station Assignment Problem (ABSSAP) is potentially important for a wider variety of problems other than the ABSSAP. By way of illustration, here we look at applying the SSEA to the more widely studied AGAP.

Some of the metaheuristics and constructive algorithms previously used in the ABSSAP are modified, modifications which are required to be able to apply these metaheuristics to the AGAP, and the constructive algorithms are used as initial solutions in the SSEA presented here for the AGAP. The SSEA is studied, and compared with other metaheuristics approaches, which performance is studied in regard to the different objectives used at London Heathrow airport.

Keywords: Airport Gate Assignment Problem; Scheduling; Heuristics; Evolutionary Algorithms

1. Introduction

Aircraft depart from an airport and arrive at their destination airport, from which the aircraft may again de-part to yet another airport, and this may be repeated many times a day for each aircraft, see Figure 1. During the time between arrival and departure, while the aircraft is still at the airport, it needs to have a space allocated at a stand on the airport air-side, where the aircraft is parked and some operations may need to be performed before it is ready to continue its cycle of departure and arrival. Many parking positions are located beside the airport buildings, next to the gate from which passengers will board the aircraft. There are often other parking positions of the terminal (on the

apron) which are also called stands, where aircraft may be parked for longer periods, but when used to embark passengers then they must be transported by mobile lounge or bus, thus increasing the congestion on the ground. Gates, however, correspond to those stands which are located in the airport terminal buildings, as shown in Figure 2, where the gates are numbered in blue. Those stands not located immediately by the air-port buildings are called remote stands. The gates provide extra services to those that are provided at a re-mote stand. The stands next to the airport gates are scarce and expensive resources which must be used efficiently and be assigned to aircraft effectively. The gate assigned to an aircraft arrival may not be the same as that assigned to the same

Figure 1: Airport Arrival and Departure Overview for dual runway segregated parallel operations.

Figure 2: London Heathrow airport Terminal 1 with gate location.

aircraft for departure, and the intermediate parking operation if any is required, between arrival and departure assignments may also be at a different stand. This may either be a remote stand (not a gate) or another gate depending on the availability of these resources at the time. Aircraft only directly compete for resources if their stay at the airport over-laps in time.

The assignment of gates already scheduled to flights is known as the Airport Gate Assignment Problem (AGAP) and is one of the most important operations in an airport, having repercussions on many other resources, such as baggage sorting stations (BSSs) (Airport Baggage Sorting Station Assignment Problem (ABSSAP)). The gate assignment problem is normally presented as a multi-constraint and multi-objective problem where different objectives have been used in different papers but not always together, and in some cases not all of the objectives were used, to reduce the complexity of the problem. [1] used Branch and Bound (B&B) with the object of minimizing passenger walking distance, with some enhancements to accelerate the computation. [2] took account of the transfer of passengers using linear programming relaxation and greedy algorithms, and [3] used Binary Integer Linear Programming (BILP) to solve the minimum walking distance, whereas a Genetic Algorithm (GA) was used by [4] and [5], and an Evolutionary Algorithm (EA) was presented and used by [6]. Where traditional Ant Colony Optimisation (ACO) used pheromone trail information to construct complete solutions to the AGAP, [7] used a hybrid ant-local search system where pheromone trail information is used to perform modifications on AGAP solutions. [8] presents a Bee Colony Optimization (BCO) in the field of Swarm Intelligence to find an optimal flight gate assignment for a given schedule when considering two objectives: minimization of passenger total walking distance and remote gate usage. Whereas [9] uses gate stochastic constraints by incorporating the inherent stochastic flight delays together with a specified threshold for expected gate conflict probability of two flights assigned to the same gate at the same time, which threshold is used to regulate the robustness level, and historical data of flight movements at Amsterdam Airport Schiphol (AAS) were used to predict probability distributions of flight presence at the apron as previously historical data was used in [6] for the 'Maximise Airline Preferences' objective and different robustness approaches in the AGAP.

According to [10] whereas only 16% of the air traffic delays are attributable to the point at which the aircraft is airborne with 26% from taxi-out and 8% from taxi-in, the remaining delays may be derived from delays where the aircraft are at a gate (50%), which reveals the gates to be of considerable importance in reducing overall airport delays.

The objective most used in the current AGAP literature corresponds to the improvement in service satisfaction, assured to be achieved by reducing passenger walking distance inside the terminal building. When considering the passengers walking distance the AGAP can be modeled by analogy with the NP-hard quadratic assignment problem, as shown by [11], [12] and [13], which is a facility location problem where the cost of assigning a flight to a gate depends on the assignment of other resources

and the transport volume between two resources (see also [14]).

Also with the increase in passenger traffic volumes and number of flights, the complexity of this task and the number of factors to be considered have increased significantly, and efficient gate utilisation has received considerable attention in past years, e.g. [18], [6], [15] and [16], [17], [18], and [19]. Given that a gate may be required for up to three different operations, namely arrival, parking and departure, the number of assignments required may have increased significantly in comparison with those for the ABSSAP. These also increase the problem complexity and provide more reasons for investigating some metaheuristic approaches, such as the Steady State Evolutionary Algorithm (SSEA) initially presented in [20] and used in [21] and [22] for the ABSSAP.

The similarities between the AGAP and the AB-SSAP makes particularly interesting to determine whether the SSEA is also appropriate for the AGAP. However, the SSEA needs first to be adapted to the AGAP.

2. Model

The model used for the AGAP is based on that proposed in [23], which considers the problem as a resource constrained project scheduling problem, originally presented in [24]. Flights serviced by the same aircraft may not generally be assigned to the same stand, in which case they may need to be moved to another assigned stand by using tugs, and known as towing, but the use of towing should be kept to a minimum given the extra cost involved, such as hiring towing trucks and the increase of ground traffic. If the time for servicing an aircraft between its arrival and departure is sufficiently long then the aircraft may be assigned to a parking stand in order to release the stand originally assigned to its arrival. It is assumed that even if there are insufficient stands at the pier, there are always sufficient remote stands where the aircraft awaiting departure can be parked. The aircraft will then be towed to its assigned departure stand, which may not be the same as that assigned to it on arrival, Figure 3. The movement

Figure 3: Assignment of flights to stands when towing is required

of aircraft around the air side of the airport terminal potentially poses difficulties, e.g. increasing traffic on the airport ground side, and always incurs extra costs, such as hiring the towing machine. To consider these, an objective is introduced into the model with the aim of reducing the level of unnecessary remote parking, Section 5.3. This also penalizes the impact of towing an aircraft from the assigned arrival stand to the remote stand

and finally to the departure stand. The problem is an Activity Assignment Problem where the arrival, departure and parking periods of an aircraft at a stand are the activities and the stands are the resources.

It is anticipated that there will always be sufficient remote parking stands, so there is unlikely to be a problem in their use, nor as to which stand is used. Further-more, given that arrival and departure flights require the extra facilities provided by a gate, whereas the parking activity does not, it is preferable to assign the parking activity to a remote stand if this increases the assignment of more arrival and departure flights to gates. This may be modeled either by introducing a new objective which takes this preference into account, or by building it into the model as a hard constraint. Here the latter approach is used, where parking assignments are restricted to gates already assigned to arrival or departure flights of the same aircraft or a remote stand. This prompted modeling the use of a remote stand in a similar fashion to the dummy, which is equivalent to a stand with unlimited capacity, where assignments can overlap, but is restricted to intermediate operations such as parking. The service time must be of at least a specified minimal duration otherwise the remote stand will not be required, when arrival and departure flights will be treated as one activity. The current procedure in London Heathrow airport usually involves the assignment of arrival, parking and departure of the same aircraft to the same gate when the time between arrival and departure is less than 3 hours.

The model used in the ABSSAP, which it is presented in [25], is modified and extended to represent the remote stand, where $i = 0$ equates to the dummy stand as used in the ABSSAP, and $i = N + 1$ represents the remote stand, and where N represents the number of real stands at gates. Where the term remote stand is used, it refers to the dummy remote stand. The dummy remote stand may also be used solely for parking operations where arrivals and departures are not permitted by this resource. This means that when a solution is obtained and the dummy re-mote stand has been assigned to certain aircraft, then these aircraft must be assigned to real physical stands whether remote or otherwise.

The main reason for this procedure is to speed up the generation of solutions, as the algorithm should already be endeavoring to reduce the number of tows (objective 'Minimize Number of Towing Operations' presented in Section 5.3), which includes the reduction of the number of remote assignments. In this representation, those which are commonly regarded as two different flights, namely the arrival and departure flights, are considered here to be only one group composed of three operations: arrival, parking and departure, as shown in Figure 3. The relationship between these operations relates to the same aircraft, but may differ in the use of other resources such as the crew.

3. Problem Representations

The problem is presented as an Integer Linear Programming (ILP) with y_{ij}^k being a Boolean variable with a value of one if activity k ($k \in \{a, p, d\}$ where 'a' corresponds to the arrival flight, 'p' to the parking and 'd' to the departure flight activities) of

group j is assigned to gate i or zero otherwise. The degree of reduction in service time for the assignment of activity k of flight j is represented by r_j^k which is deemed to be calculated in seconds (as an integer). These constants and variables are listed in Tables 1 and 2, the full model being presented in the following sections.

In the following sections the only case under consideration is that where the parking of an arriving and/or departing flight may be assigned to the same stand as the arrival or departure activity, or to the remote dummy stand, Figure 4.

It is assumed that aircraft j may be used for three assignments, such that the model may be expressed as follows: the assignment of aircraft j to stands i , l , and q is expressed as $y_{ij}^a = 1$, $y_{ij}^p = 1$ and $y_{ij}^d = 1$ respectively. There are now two new base service duration constants, one for the flight arriving, T_j^a , and the other which corresponds to the parking base service duration, T_j^p , if the corresponding aircraft were to be assigned to a re-mote stand. The commencement and completion times of a parking operation are fixed, based on the values of the departure time e_j^d and arrival time τ_{ja} , given that $e_j^a = \tau_{ja} + T_j^a$ and $\tau_j^d = e_j^d - T_j^d$ then $\tau_j^p = e_j^a$, $e_j^p = \tau_j^d$ and $T_j^p = \tau_j^d - e_j^a$. The list of constants for this model is shown in Table 1 and the list of decision variables is shown in Table 2.

In the AGAP the activity k of group j may take the values a , p or d ($P_j = 3$) whereas in the ABSSAP activity p was represented as an integer ($1 \leq p \leq P_j$). However in the AGAP y refers to aircraft, and there are extra constraints which do not allow parking activities to be assigned to stands other than the dummy parking stand (also called the dummy remote stand), or their corresponding arrival and departure stands. The operations represented by 'a', 'p' and 'd' refer to the arrival, parking and departure operations respectively of the same aircraft.

Table 1: List of the constants and input values for the AGAP model

Name	Description
N	The total number of gates under consideration.
M	The total number of aircraft to which gates should be allocated.
k	The type of operation, arrival, parking or departure, $k \in \{a, p, d\}$.
P_j	The total number of activities associated with aircraft j in a full cycle, $1 \leq P_j \leq 3$.
T_j^k	The base service duration for aircraft j and activity k .
B_j^k	The desired buffer time for aircraft j and activity k . The parking operation does not have any buffer time associated with it, i.e. $B_j^p = 0$.
e_j^k	The end service time for aircraft j and activity k .
τ_j^k	The base starting service time for aircraft j and activity k , $\tau_j^k = e_j^k - T_j^k$.
t_j^k	The target starting service time for aircraft j and activity k , $t_j^k = e_j^k - T_j^k - B_j^k$, assuming the full buffer time is available. Whereas $T_j^p = t_j^d - e_j^a$ and $e_j^p = t_j^d$.
x_{ij}	Expresses to which stand (i) aircraft j can be assigned, $i \in \{1 \dots N\}$. $x_{ij} = 1$ if aircraft j can be assigned to stand i , otherwise $x_{ij} = 0$.

Table 2: List of the decision variables which are used in this AGAP model

Name	Description
y_{ij}^k	Specifies the assignment of aircraft to stands. $y_{ij}^k = 1$, if gate $i \in [1 \dots N]$ is allocated to aircraft $j \in [1 \dots M]$ for activity $k \in \{a, p, b\}$, and 0 otherwise.
r_j^k	Specifies the necessary reduction in service time for activity k of aircraft $j \in [1 \dots M]$, given the allocated starting service time, s_j^k .
s_j^k	The allocated starting service time for activity $k \in \{a, p, d\}$ of aircraft $j \in [1 \dots M]$ and given that a gate can only service one aircraft at a time then s_j^k can be determined from r_j^k since $s_j^k = t_j^k - r_j^k$ and $t_j^p = e_j^a$.

4. Constraints

4.1. Assignment Limits

Each stand can only be used by one aircraft at a time, with the exception of the dummy stand and the remote dummy stand, Equation 1.

$$\sum_{i=1}^N \sum_{k \in \{a, p, d\}} y_{ij}^k = P_j \quad \forall j \in [1 \dots M] \quad (1)$$

4.2. Assignment Restrictions

Each activity may be only assigned to one stand, Equation 2.

$$\sum_{i=0}^{N+1} y_{ij}^k = 1 \quad \forall j \in [1 \dots M] \text{ and } k \in \{a, p, d\} \quad (2)$$

Figure 4: Different stand assignments for group j

The remote dummy stand ($i = N + 1$) is only suitable for parking and not for any other activity, i.e. arrivals or departures, Equation 3.

$$y_{(N+1)j}^a = y_{(N+1)j}^d = 0 \quad \forall j \in [1 \dots M] \quad (3)$$

The dummy stand ($i = 0$) cannot be assigned to parking activities. The remote dummy ($i = N + 1$) allows overlapping activities and always has the capacity to be assigned to a parking activity, Equation 4.

$$y_{0j}^p = 0 \quad \forall j \in (1 \dots M) \quad (4)$$

The parking activity may be assigned to the same stand as its associated arrival or departure activities, Inequality 5.

$$y_{ij}^p \leq (y_{ij}^a + y_{ij}^d) \quad \forall i \in [1 \dots N], j \in [1 \dots M] \quad (5)$$

4.3. Stand and Aircraft Size Restriction

Each stand has a size code assigned to it which identifies the range of aircraft assignable to it. The aircraft sizes match those specified by the International Civil Aviation Organization (ICAO). This constraint is modeled by the constant x_{ij} shown in Table 1 (example in Figure 5), Inequality 6.

$$y_{ij}^k \leq x_{ij} \quad \forall i \in [1 \dots N], j \in [1 \dots M] \text{ and } k \in \{a, p, d\} \quad (6)$$

4.4. Combining Stands

Certain gate assignments may cause blocking of neighboring gates, sometimes described as 'Shadowing Assignments'. For example, in some cases two adjacent stands can jointly host a larger aircraft where they would be unable to do it individually. To represent the combined stand a fictitious stand is postulated which can only be assigned to aircraft larger than the largest capable of assignment to any of its component stands, Figure 6. It does not make any sense for the combined stand to be assigned to

an aircraft which could also be assigned to any of its component stands. In this case the smaller stands can accommodate an aircraft each, or the combined fictitious stand may host a larger aircraft. So for a stand i composed of stands l and r , and an aircraft j if $x_{lj} = 1$ or $x_{rj} = 1$ then $x_{ij} = 0$, whereas if $x_{lj} = 1$ then $x_{ij} = x_{rj} = 0$. If O_j contains all of the flights overlapping this flight j then all assignments to stands r , l and i ($r \cup l$), which overlap, must comply with Inequalities 6 and 7.

$$\sum_{q,u,v \in O_j \cup j} \sum_{k \in \{a,p,d\}} (y_{rq}^k + y_{lu}^k + (2 * y_{iv}^k)) \leq 2 \quad (7)$$

Based on Inequality 6, example in Figure 6 provides the following restrictions. This means that stand 3 occupies the combined space of stands 1 and 2, and aircraft 3 cannot be assigned to either stands 1 or 2 ($x_{13} = x_{23} = 0$), whereas aircraft 1 and 2 cannot be assigned to stand 3 ($x_{31} = x_{32} = 0$), as shown in Table 3.

Figure 5: Example of two stands ($i \in \{1, 2\}$) of different sizes and two aircraft ($j \in \{1, 2\}$) where aircraft 1 is too large to fit in stand 2 ($x_{21} = 0$)

Table 3: Valid values for x_{ij} and y_{ij}^k for the example in Figure 6

$y_{11}^k \leq x_{11} = 1$	$y_{12}^k \leq x_{12} = 1$	$y_{13}^k \leq x_{13} = 0$
$y_{21}^k \leq x_{21} = 1$	$y_{22}^k \leq x_{22} = 1$	$y_{23}^k \leq x_{23} = 0$
$y_{31}^k \leq x_{31} = 0$	$y_{32}^k \leq x_{32} = 0$	$y_{33}^k \leq x_{33} = 1$
$k \in \{a, p, d\}$		

5. Objectives

5.1. Maximise Number of Assignments

This objective aims to maximize the number of assignments, Formula 8, and it is equivalent to the same objective as defined in the ABSSAP.

$$\max \sum_{i=1}^N \sum_{j=1}^M \sum_{k \in \{a,p,d\}} y_{ij}^k \quad (8)$$

Given that parking is always assured since the parking activity can always be assigned to the dummy remote stand, then Formula 8 is equivalent to Formula 9.

$$\max \sum_{i=1}^N \sum_{j=1}^M \sum_{k \in \{a,d\}} y_{ij}^k \quad (9)$$

The preference for assigning the parking activities to the same stand as one of the associated arrival or departure flights, is taken account of in the objective 'Minimize Number of Towing Operations' presented in Section 5.3.

5.2. Maximise Airline Preferences

Airlines may have some preference as to the gates for

assignment to their flight. These could be based on their position in relation to some of the airline facilities such as offices or other resources used, for example buses.

To take account of the different airline preferences a list of gates and a weight, representing the level of preference for the gate, is used. This list could be compiled based on past historical data of gates assigned to the airline, such that the constant θ_{aj} is 1 when aircraft j belongs to airline α and zero otherwise and there is a set of historical flights represented by H . The preference of airline α for stand i may then be expressed by Equation 10, and the objective by Formula 11.

$$\delta_{\alpha i} = \frac{\sum_j^H \sum_{k \in \{a,d\}} \theta_{\alpha j} * y_{ij}^k}{\sum_{i=1}^N \sum_j^H \sum_{k \in \{a,d\}} \theta_{\alpha j} * y_{ij}^k} \quad (10)$$

$$\max \sum_{i=1}^N \sum_{j=1}^M \sum_{k \in \{a,d\}} \sum_{\alpha} \delta_{\alpha i} * y_{ij}^k \quad (11)$$

5.3. Minimise Number of Towing Operations

This objective aims to minimize the number of towing operations required from Formula 12. A towing operation is required every time an aircraft changes its location. The number of towing operations is shown in Figure 4.

$$\min \sum_{i=1}^N \sum_{j=1}^M (|y_{ij}^a - y_{ij}^p| + |y_{ij}^d - y_{ij}^p|) \quad (12)$$

5.4. Maximise Handler Preferences

Handlers normally provide their services to multiple customers, so one of their preferences may be to concentrate their operations within the minimum number of piers, considering gates within the same pier to be closer to each other than those in other piers.

(a) Stand size restriction matrix.

(b) View of potential stands assignment.

Figure 6: Example of two stands ($i \in \{1, 2\}$) and one combined fictitious stand ($i = 3$) for three aircraft ($j \in \{1, 2, 3\}$) where aircraft 3 is too large to fit in either stands 1 and 2 but both aircraft 1 and 2 fit any of the stands.

To take account of the preferences of the different handling agents, it is assumed that fitness increases as the number of assignments to a handler at the same pier increases.

If ng_j is the number of assignments to stands in pier $j \in [1 \dots J]$ for agent $g \in [1 \dots G]$, and n_g corresponds to the total number of flights serviced by agent g the fitness may be calculated by Formula 13.

$$\max \sum_{g=1}^G \sum_{j=1}^J \frac{n_{gj}}{n_g} \quad (13)$$

5.5. Robustness

Different methods for increasing robustness exist, similar to the ABSSAP in [22] and [6], where the existence of a gap is taken into account. Given that the model always takes account of the possibility of assigning the parking operation of an aircraft to a remote stand, which is discouraged, then no buffer time is ever associated with a remote operation. Moreover, if the parking activity is assigned to the same stand as the departure activity, then the departure activity will have no reduction in service where the duration of the parking activity is at least as long as the buffer time associated with the departure activity.

If an aircraft j does not have an arrival activity ($T_j^a = B_j^a = 0$) then no parking is considered in the model, which means the base and target service times for the parking operation are both zero: $T_j^p = 0$, (note that B_j^p is already zero). The approach already considers the case where a flight arriving does not have a departing flight associated with it, i.e. after the aircraft has completed the arrival procedure the aircraft is required to follow maintenance procedures for which it will be taken to the appropriate installation to perform any of these necessary maintenance operations.

Some of the robustness approaches are: 'Distribute Idle Time', 'Reduce Reassignment on Disruption', 'Area Reduction in Service', 'Sub-area Reduction in Service', 'Unsupervised Estimated Stochastic Reduction in Service', 'Reduction in the Number of Conflicts' and 'Probability of Conflicts Based on the Gap' which are similar to those presented in [22] for the ABSSAP. The 'Minimize Reduction in Service' is used here and requires some changes in order to be applicable to the AGAP, which is described below.

5.5.1. Minimize Reduction in Service

This objective may be expressed by Formula 14.

$$\min \sum_{j=1}^M \sum_{k \in \{a,d\}} r_j^k \quad (14)$$

The unassigned flights may also be penalized by a cost proportionally higher than the buffer time. In order to treat all unassigned flights equally, and not discriminate between them, this extra cost may involve a multiple of the maximum buffer time for all flights, Formula 15. In this case, given that the number

$$\min \left(\sum_{j=1}^M \sum_{k \in \{a,d\}} r_j^k + \beta * \max_{j=1, k \in \{a,d\}}^M (B_j^k) * \underbrace{\sum_{j=1}^M \sum_{k \in \{a,d\}} \left(1 - \sum_{i=1}^N y_{ij}^k \right)}_{\text{number of unassigned flights}} \right) \quad (15)$$

of assignments is the most important objective, the factor used to multiply the maximum buffer time should be higher than two, given that this would represent the buffer time taken by the reduced service incurred when a new assignment is placed between two consecutive flights. In this case, the buffer times at each end are removed from Figure 7.

6. Steady State Evolutionary Algorithm

The operators and selectors used are those introduced in [21], and [22] for the ABSSAP. The SSEA is based on [26] theory of natural selection and Mendelian genetics ([27]), which are recognized as the foundation of evolutionary biology, and it was originally used for the ABSSAP, but in this problem the resources are now gates instead of BSSs and only the corresponding constraints and objectives presented in Section 2 are applicable. Some modifications are necessary before the SSEA can be applied to the AGAP, and these are the only ones described. The main intention is to establish the suitability of the SSEA for the AGAP. Some experiments were conducted using metaheuristics, the results of which are compared and studied in Section 9, showing that the SSEA also provides good results for the AGAP.

New operators are required to allow the reassignment of parking activities from the dummy remote stand to a gate. This is not required if the Multi Exchange Mutation Operators (MEMOs) described in Section 7 for the ABSSAP are modified, such that their recovery stage also considers the parking activities assigned to the dummy remote stand. This removes the need to use tailored mutation operators to assign parking activities from the dummy remote stand to gates. These tailored operators are presented in Section 7.

7. Operators

Two main groups of operators are reviewed in the following sections: Mutation 7.1 and Crossover 7.2, and their combination 7.3. All of which were original described for the ABSSAP at [21] and [20].

7.1. Mutation

The operators introduced here are local search (guided mutation) operators which generate feasible solutions.

All flights which have not been assigned to a stand are assigned to the 'dummy' stand. Some operators can switch flights between the real and dummy stands.

When a stand is to be selected, the roulette wheel se-lection

method is used where every stand has the same probability of being selected.

When a time has to be determined (for instance for the start or end of a time range) a uniform random variable is used so that any time within the time range of the flights under consideration has an equal probability of being chosen.

7.1.1. Dummy Single Exchange Mutation Operator

The Dummy Single Exchange Mutation Operator (DSEMO) is equivalent to the 'Apron Exchange Move' used by [28] and [29]. A solution is selected from the population by the member selector (S_m) then a new solution is built by moving a flight from the 'dummy' stand from this solution to a randomly selected stand, potentially moving another flight back onto the 'dummy' stand when it can no longer be fitted in.

This operator may increase the number of assignments where the operation does not move a flight back onto the 'dummy' stand. It is necessary that some flights be unassigned in the parent solution. So when full assignment has been attained for the given number of stands this operator clearly will not provide a new solution.

7.1.2. Dummy Single Move Mutation Operator

In the Dummy Single Move Mutation Operator (DSMMO) a random unallocated flight and initial target gate are chosen and an attempt is made to assign the flight to the selected stand. If the assignment cannot be achieved then the next gate is selected and the process is repeated until the flight is assigned or no more gates are available, in which case the flight is returned to the 'dummy' stand. When full assignments have been attained for the given number of stands, then this opera-tor obviously will not provide a new solution.

The DSMMO described in 7.1.2 originally presented for the ABSSAPs at [21, 20] and [21], which here moves assignments from the dummy stand to a gate, is preferred to the DSEMO, which exchanges assignments between the dummy stand and a gate, where N is greater than or equal to the Lower Maximum Assignment Point (LMAP), given that full assignment is achievable, and this operator helps to achieve it. The DSEMO may also need to execute an exchange, which will not improve the number of assignments. Nevertheless, in the early iterations, where there may be many solutions lacking full assignment, these may be forced into assignment by the use of the DSMMO ($N \geq \text{LMAP}$). When $N < \text{LMAP}$ it is impossible to reach full assignment, so the DSEMO is preferable to the DSMMO.

Figure 7: Example: two solutions for the same problem with different assignments.

7.1.3. Multi Exchange Mutation Operators

A set of stands is randomly selected within a random time period, t_{rs} to t_{re} . All assignments where the base service durations are entirely within the time period are then moved to the next stand in the set, as shown in Figure 8, provided they fit. This operation is repeated from one stand in the set to the next, until they have all been covered. Flights which cannot be moved are added to the set of flights which will be considered for assignment at the end, potentially reducing the number of flights which otherwise would not be assigned. These operators generalize the 'Interval Exchange Move' which was presented by [28], and cannot increase the number of assignments.

Figure 8: Example of multi exchange between 3 stands.

Three variants have been developed

1. Multi Exchange between a Fixed Number of Re-sources (MEFNR n): The number of stands between which flights are exchanged is fixed at n , where $2 \leq n \leq N$.
2. Multi Exchange between a Random Number of Re-sources (MERNR n): The number of stands between which flights are exchanged is randomly chosen each time, between 2 and n , where $2 < n \leq N$.
3. Multi Exchange between a Range Random Number of Resources (MERRNR xy): The number of stands between which flights are exchanged is randomly chosen each time, between x and y , where $2 \leq x < y \leq N$.

7.1.4. Multi Exchange by Pier Mutation Operators

These operators are a specialized case of the Multi Exchange Mutation Operators, where the stand selection element ensures that no two consecutive stands in the set are on the same pier. The idea is to improve the distance objective by encouraging the movement of assignments between piers. Once again, this operator cannot increase the number of assignments. As for the Multi Exchange Mutation Operators, three variants have been created:

1. Multi Exchange by Pier between a Fixed Number of Resources (MEBPFNR n): The number of stands to exchange flights between is fixed at n , where $2 \leq n \leq N$.
2. Multi Exchange by Pier between a Random Number of Resources (MEBPRNR n): The number of stands between which the flights are exchanged is randomly chosen each time, between 2 and n , where $2 < n \leq N$.

3. Multi Exchange by Pier between a Range Random Number of Resources (MEBPRNR xy): The number of stands between which the flights are exchanged is randomly chosen each time, between x and y , where $2 \leq x < y \leq N$.

7.1.5. Range Multi Exchange Mutation Operators

These are the same as the Multi Exchange Mutation Operators, however they add an additional feasibility recovery step when flights cannot be moved. Flights which cannot be moved are added to the set of flights which will be considered for assignment to the next stand, potentially reducing the number of flights which will not be assigned in the end. Finally, flights which have still not been moved are again considered for assignment to the other stands in the set, except the last one, once again potentially reducing the number of flights which otherwise would not be assigned, in the same way as the Multi Exchange Mutation Operators, Figure 9.

Figure 9: Example of range multi exchange between 3 Stands.

Once again, this operator cannot increase the number of assignments. Three variants have been developed:

1. Range Multi Exchange between Fixed Number of Resources (RMEFNR n): The number of stands between which to exchange flights is fixed at n , where $2 \leq n \leq N$.
2. Range Multi Exchange between Random Number of Resources (RMERNR n): The number of stands between which to exchange flights is randomly chosen each time, between 2 and n , where $2 < n \leq N$.
3. Range Multi Exchange between Range Random Number of Resources (RMERRNR xy): The number of stands between which to exchange flights is randomly chosen each time, between x and y , where $2 \leq x < y \leq N$.

7.1.6. Range Multi Exchange by Pier Mutation Operators

These are a specialized version of the Range Multi Exchange Mutation Operators, which ensure that consecutive stands in the set are not on the same pier, to encourage the movement of flights between piers, so potentially improving the distance objective. These operators cannot increase the number of assignments. As for the Multi Exchange Mutation Operators, three variants have been created: Range Multi Exchange by Pier between Fixed Number of Resources (RMEBPFNR n) and with Random Number of Resources Range Multi Exchange By Pier between Random Number of Re-sources (RMEBPRNR n) and Range Multi Exchange By Pier between Range Random Number of Resources (RMEBPRNR xy).

The Multi Exchange Mutation Operators may also be extended by using multiple points in time instead of two points in time (a time range). However, this will also increase the complexity and time required to execute the operations, and equates to several executions of the current implementation and was not therefore investigated.

7.1.7. Remote Mutation Operators

A new fictitious dummy stand, namely the remote dummy, was introduced in this problem to explore the parking activities between arrival and departure flights by the same aircraft, as it is presented in Section 2.

In order to allow these parking activities to be unassigned from the remote dummy stand, it is necessary to add another operator. The Remote Dummy Single Exchange Mutation Operator (RDSEMO) selects only one of the parking activities assigned to the remote dummy for exchange, namely the RDSEMO, described in Algorithm 1. The parking activity for reassignment may be randomly selected when there is more than one assignment to the remote dummy stand. The RDSEMO will be seen in Section 9 to perform poorly given that it is restricted to solutions with parking activities assigned to the remote dummy, together with the extra constraint of assigning them to the gate already assigned to either the arrival or departure flight of the same aircraft as the parking activity.

Algorithm 1: Remote Dummy Single Exchange Mutation Operator

```

Randomly select a parking activity between all assigned
to the remote dummy;
if arrival or departure activity associated with this
parking activity is assigned to a gate then
    Select randomly one of the assigned gates to either
    the arrival or departure activity;
    if is possible to assigned the parking activity to this
    gate then
        Assign the parking activity to the gate;
    end
    if is possible to assigned the parking activity to this
    gate once the appropriate activity is unassigned
    from this gate then
        Assign the parking activity to the gate;
        Assign the unassigned activity to the
        appropriate dummy stand;
    end
end

```

The Remote Dummy Exchange All Mutation Operator (RDEAMO) removes each parking activity from the remote dummy and assigns it to an appropriate gate, perhaps by removing one of the activities assigned to that gate, as described in Algorithm 2, so repeating the process followed by the RDSEMO for each of the parking activities assigned to the remote dummy stand.

Algorithm 2: Remote Dummy Exchange All Mutation Operator

```

forall parking activities in assigned to the remote
dummy stand do
    if arrival or departure activity associated with this
    parking activity is assigned to a gate then
        Select randomly one of the assigned gates to
        either the arrival or departure activity;
        if is possible to assigned the parking activity to
        this gate then
            Assign the parking activity to the gate;
        end
        if is possible to assigned the parking activity to
        this gate once the appropriate activity is
        unassigned from this gate then
            Assign the parking activity to the gate;
            Assign the unassigned activity to the
            appropriate dummy stand;
        end
    end
end

```

An example of the RDSEMO operator is shown in Figure 10(a), where the problem is composed of three gates and five groups. A parking activity is selected randomly from those assigned to the remote dummy, for example the parking activity of group 3. A gate is next randomly selected, e.g. gate 1, from which the search to assign the parking activity commences. As a parking activity must be assigned to the same gate as either its arrival or departure flight, then this remote activity cannot be assigned to gate 1. So the search moves to the next gate, gate 3, but the same applies to this gate so the parking activity cannot be assigned to this gate either. Finally, the next gate, gate 2, which has not yet been looked at, is now checked and the parking activity for group 3 can be assigned to it given that the flight arrival at this parking activity is also assigned to this gate, but group 4 must first be unassigned. The process is repeated in turn for each of the other parking activities assigned to the remote dummy which has not yet been considered, e.g. parking activity 2. This parking activity can only be assigned to gate 3, but it would overlap with group 5, so firstly group 5 is unassigned and then parking activity 4 is assigned to gate 3.

Figure 10: Examples of the process for the Remote Dummy Exchange All Mutation Operators (RDEAMO).

Another operator moves one or multiple parking activity assignments from the remote dummy stand to appropriate gates with a sufficient gap to accommodate them all, namely Remote Dummy Move All Mutation Operator (RDMAMO). Therefore only those parking activities will be assigned where there is a gate with adjacent assignments which have a sufficient gap to accommodate the parking activity, and where one of those activities is an arrival or departure for the parking activity.

An example of the RDMAMO operator is shown in Figure 11, where the problem is composed of three gates and four groups. A parking activity is randomly selected from those assigned to the remote dummy, e.g. a parking activity of group 3. A gate is next randomly selected, e.g. gate 1, from which the search to assign the parking activity begins. As the parking activity must be assigned to the same gate as either its arrival or departure flight this remote activity cannot be assigned to gate 1. So the search moves to the next gate, gate 3, but this assignment is not possible either, as otherwise it would overlap with the activity for group 4. Finally, the next gate, 2, which has not yet been looked at, is now checked and the parking activity for group 3 can be assigned to it, given that this parking activity does not overlap with any of the activities already assigned to that gate, and the arrival flight for this parking activity is also assigned to the gate. The process is repeated for each of the other parking activities assigned to the remote dummy which has not yet been considered, for example parking activity 2. However this parking activity cannot be assigned to any of the gates as it would overlap other assigned activities.

Figure 11: Example of the process for the Remote Dummy Move All Mutation Operator (RDMAMO).

When the remote dummy stand has no remote activity assigned to it then obviously none of the remote dummy mutation operators presented here provide a new solution, as there are no remote activities available to be unassigned from the remote dummy and assigned to a gate. These operators may therefore only be used when there is parking activities assigned to the remote dummy, and once this is no longer the case, they should not be used.

When only one or many of the mutation operators introduced in Section 7 are used it may be advantageous to include at least one of the remote dummy mutation operators, as the other mutation operators do not have the capability of reassigning parking activities to gates. The crossover operators may be able to reassign parking activities to gates, but only where at least one of both parents have not assigned the same parking activity to the remote dummy gate. This applies similarly to the dummy operators DSEMO and DSMMO in respect of unassigned flight arrivals and departures.

7.2. Crossover

The crossover operators involve the generation of new solutions from multiple parents. Each parent will be chosen using the Parent Selectors (S_m) and multiple child solutions may be generated in each case.

7.2.1. 2-point Crossover

In the 2-point crossover (C2P), two points in time are randomly selected within the time range of the flights, to generate a time window. All flight assignments which lie within this time period, for all of the stands in each solution, are exchanged between the parent solutions, as shown in Figure 12. The flight timings are identical across all solutions, except that the flights in the exchanged region may overlap flights which are not exchanged in the case of some stands. Such overlapping flights in the exchange region are reassigned to other stands where possible, otherwise they are assigned to the dummy stand (i.e. are unassigned).

Whereas in the classic crossover a chromosome is divided into 3 sections, here the chromosome is divided into $3 * N$ sections which correspond to 3 sections per stand.

7.2.2. 1-point Crossover

The 1-point crossover (C1P) is a specific case of the above 2-point crossover, where the window extends to the end time of the solution, Figure 13. In my representation, 1-point crossover is a special case of 2-point crossover ($n = 2$, number of points), where the second point corresponds to the end of the chromosome. This can be better understood if the chromosome is represented as a loop, Figure 14.

7.3. Combination of Operators

Based on how the operator is selected, the types which are of interest are described in the following subsections. It is noted that the operators could be used in complex ways by combining these different types with different parameters.

7.3.1. Probability Single Multi Operator

The Probability Single Multi Operator (PSMO) is composed of several sub-operators (which are described in Section 7), each one of which has a specified probability of being used for the creation of new population members, Algorithm 3. The combined probabilities across all operators must add up to 1.

Figure 12: 2-point crossover.

Figure 13: 1-point crossover.

Figure 14: Crossover representations where 's' refers to the start and 'e' to the end.

As an example, consider a PSMO operator which uses the operators C1P (with a 0.1 probability of being selected) and Multi Exchange between a Fixed Number of 3 Resources (MEFNR3) (with a 0.90 probability of being selected), which may be represented as PSMO(C1P:10+MEFNR3:90). Given that the total probability must amount to 1, it is not necessary to specify the probability for the last sub-operator, so the representation may also be PSMO(C1P:10+MEFNR3).

Algorithm 3: Probability Single Multi Operator.

```

Input: Member Selector  $S_m$ 
Input: Population of solutions  $P$ 
Input: Operators;  $O_k \forall k \in [1 \dots R]$ 
Input: Probability for operators  $p_k$ ,
 $0 < p_k \leq 1 \forall k \in [1 \dots R]$  and  $\sum_{k=1}^R p_k = 1$ 
begin
    // Initialise
     $P_0 \leftarrow \emptyset$ ; // empty list of children
     $r = \text{rnd}[0 \dots 1]$ ;
     $k = 1$ ; // initialise sub-operator index to
    first operation
     $p = p_1$ ;

    // Select operator
    while  $k < R$  and  $r > p$  do
         $k = k + 1$ ; // next operator
         $p = p + p_k$ ;
    end
     $Q \leftarrow S_m(P, O_k)$ ; // get parent solutions for
    operator  $O_k$ 
     $P_0 \leftarrow O_k(Q)$ ; // build children by applying
    operator to parents

    return  $P_0$ ; // return the obtained children
end

```

Algorithm 4: Sequential Operator.

```

Input: Member Selector  $S_m$ 
Input: Population of solutions  $P$ 
Input: Operators;  $O_k \forall k \in [1 \dots R]$ 
Input: Probability for each operator  $p_k$ ,
 $0 < p_k \leq 1 \forall k \in [1 \dots R]$ 
begin
    // Initialise
     $P_0 \leftarrow S_m(P, O)$ ; // select parents based on
    operators

    // Build children
    for  $k = 1 \rightarrow R$  do
         $r = \text{rnd}[0 \dots 1]$ ;
        if  $r < p_k$  then
             $Q \leftarrow P_0$ ; // previous children as
            parent solutions
             $P_0 \leftarrow O_k(Q)$ ; // applying operator to
            the parent solutions
        end
         $i \leftarrow k + 1$ ; // next sub-operator
    end

    return  $P_0$ ; // return the obtained children
end

```

7.3.2. Sequential Operator

Considering the way the Canonical Genetic Algorithm (CGA) operates, where a crossover operator may be applied to the parents with a high probability and its children may be further modified by applying a mutation operator, the operators may be extended by defining a new operator composed of multiple sub-operators, which are applied sequentially with a given probability ($0 < p \leq 1$), Algorithm 4. This new operator is called the Sequential Operator (SO) herein.

As an example, consider the operators C1P with a selection probability of 1 and the MEFNR3 with a probability of selection of 0.01, which may be represented as SO(C1P:100,MEFNR3:1), where a 1-point crossover is always applied to generate the intermediate children for which there is a small probability of 0.01 for application of the MEFNR3 operator in order to generate the final children solutions.

8. Problem Data

Long-haul flights have a buffer time of 15 min. and a service time of 40 min., whereas the other flights have 10 min. and a service time of 25 min. The buffer time may only be reduced for pre-scheduled flights, i.e. no buffer time is considered for the parking activities.

Not every aircraft can be parked at all stands, and a stand code identifies the type of aircraft which can be parked at a stand. Each code identifies the largest aircraft which may be parked at the stand. The stand codes at London Heathrow airport are shown in Table 4 and stand codes per gate are presented in Table 5.

Table 4: Stand codes for London Heathrow airport (provided by BAA)

Code	Comment
F	
E3	744 & 773/346
E2	744 but not 773/A346
E1	772
D (767-300)	
D (757)	
C (A321)	
C (A319)	

The representation presented here provides two values for each LMAP and Upper Maximum Assignment Point (UMAP) for those occasions when the parking activity is not taken into account and when it is taken into account respectively. They are named according to occasions when parking is not taken into account LMAP, UMAP, and when parking is taken into account Lower Maximum Assignment Point with Parking (LMAP_p) and Upper Maximum Assignment Point with Parking (UMAP_p). These maximum assignment points comply with the Inequalities LMAP $\leq$ LMAP_p and UMAP $\leq$ UMAP_p, given that the LMAP and UMAP only consider flight arrivals and departures, whereas LMAP_p and UMAP_p cover those activities already considered by the LMAP and

Table 5: Stand codes for London Heathrow airport (LHR) Terminal 4

Stand	Code	Pier	Full gate ID
1	D(767-300)	1	4101
2	E2	1	4102
3	E2	1	4103
5	F	1	4105
6	F	1	4106
7	E3	2	4207
8	E2	2	4208
9	E2	2	4209
10	E3	2	4210
11	E3	2	4211
12	E2	2	4212
14	E1	2	4214
15	C(A321)	2	4215
16	D(767-300)	2	4216
17	C(A319)	2	4217
19	C(A321)	2	4219
20	C(A321)	2	4220
21	D(767-300)	2	4221
22	E2	3	4322
23	E2	3	4323
24	E2	3	4324
25	E2	3	4325
29	E2	3	4029

UMAP together with the parking activities, potentially requiring extra resources to service them all.

A week record of flight assignments to stands was provided by London Heathrow airport for terminal four, composed of schedules from the 6th to the 12th September 2010 (H4T1009dd). Some details are shown in Table 6 which were generated from the data supplied. Using this data summarized in Table 6 for Terminal

4, tables were generated showing the preferences of each airline, and these were used in the Maximise Airline Preferences, objective, which is described in Section 5.2. Also a table was generated showing the preferences of each handler, which is used in the 'Maximise Handler Preferences' objective described in Section 5.4, and shown in Figure 15.

The topology used for Terminal 4 at London Heathrow airport is shown in Figure 16, which is composed of 23 gates and three piers.

The consecutive arrival and departure flights which make use of the same aircraft are considered a group, and if the times between the consecutive flight arrival and flight departure for the same aircraft are less than 3 hours then they are considered as one activity, stretching from the arrival to the departure times, otherwise the service of each individual flight is considered to be an activity, and a parking activity links them both (for the time between the completion of the arrival activity and the commencement of the departure activity).

8.1. Generate New Base Schedules

Based on the London Heathrow airport Terminal 4 schedules for 6th to 12th September 2010 and Algorithm 5 new schedules were generated with 37 extra groups, a summary of which is shown in Table 7.

Real schedules for London Heathrow airport Terminal 4 were used, which were obtained from BAA. The numbers of flights are lower than for Terminal 1, how-ever, data for terminals 1, 3 and 5, was not available. Schedules were generated based on the density of the schedules provided for Terminal 4, to investigate the effect on busier terminals as described below.

The consecutive flights serviced by the same aircraft are herein called a group, such that any flight always belongs to a group, although it may be the only flight in its group. For an original schedule of G_o groups, ordered in ascending base starting time (τ_i), a set of group lists is generated, each containing all of the groups where the base starting time is within the same time range (n_i). Algorithm 5 was then used to generate the new schedule.

Table 6: Data set information provided by British Airports Authority (BAA) for London Heathrow airport Terminal 4

ID	Date	LMAP	UMAP	LMAP _p	UMAP _p	No. of Activities	No. of Parking Activities
H4T100906	6-Sep-10	8	10	17	19	118	15
H4T100907	7-Sep-10	11	14	18	20	120	15
H4T100908	8-Sep-10	7	10	16	18	119	16
H4T100909	9-Sep-10	8	10	18	20	119	15
H4T100910	10-Sep-10	9	12	15	18	120	15
H4T100911	11-Sep-10	9	10	16	16	110	11
H4T100912	12-Sep-10	11	11	18	19	117	15

	4101	4102	4103	4105	4106	4207	4208	4209	4210	4211	4212	4214	4215	4216	4217	4219	4220	4221	4322	4323	4324	4325	4104
AZ	36	24	24	13	33	34	25	7	5	6	3	8	20	4	0	30	26	30	0	0	0	0	0
PH	0	10	0	16	16	2	2	4	30	32	0	0	0	0	0	0	0	0	0	0	0	0	0
CGS	16	20	23	18	2	17	22	39	18	27	53	51	38	15	17	30	43	43	16	9	2	2	1
SRV	2	10	11	6	6	1	4	2	6	12	11	16	10	4	0	4	4	0	2	6	9	4	0

Figure 15: Overall number of flights assigned to each gate, and each handler at London Heathrow airport Terminal 4 for period of 6th to 12th September 2009

Table 7: Generated data sets information with an extra 37 groups

ID	Date	LMAP	UMAP	LMAP _p	UMAP _p	No. of Activities	No. of Parking Activities
N4T100906	6-Sep-10	17	20	23	26	164	21
N4T100907	7-Sep-10	21	23	25	28	160	19
N4T100908	8-Sep-10	18	20	23	25	169	24
N4T100909	9-Sep-10	21	21	28	28	168	22
N4T100910	10-Sep-10	19	20	20	21	164	21
N4T100911	11-Sep-10	19	21	21	21	154	15
N4T100912	12-Sep-10	19	21	23	24	167	22

Figure 16: General view of London Heathrow airport Terminal 4 composed of three piers.

Algorithm 5: Generation of new scheduled from distribution of groups.

```

Input:  $G_o$  groups in the base schedule
Input:  $n_i$  list of groups with a given base starting time for the original schedule
Input:  $N$  number of extra groups,  $N > 0$ 
Input:  $rnd$  random number generator
begin
    // Initialise
     $G \leftarrow \emptyset$ ; // empty new group
     $j = \frac{N}{|G_o|}$ ; // number of full copies of original
     $n = 0$ ;

    // Copy all groups from original schedule
    while  $j \geq 0$  do
        // Copy groups from original schedule
        forall  $g_o \in G_o$  do
             $g = clone(g_o)$ ; // build copy of original group
             $G \leftarrow g \cup G$ ; // add new group to new schedule
        end
         $j = j - 1$ ;
         $n = n + 1$ ;
    end

     $N = N - (n - 1) * |G_o|$ ; // subtract the number of full copies

    // Partial copies
    while  $N > 0$  do
        // Use roulette wheel to select new group
         $n = 0$ ; // no copies yet
         $p_o = rnd$ ; // probability of being selected
        forall  $n_i$  do
             $n = n + |n_i|$ ; // add number of groups in  $n_i$ 
             $p = \frac{n_i}{\sum |n_i|}$ ; // probability of selecting a group from  $n_i$ 
            if  $p_o < p$  then
                // Select randomly a group
                 $j = rnd(|n_i|)$ ; // select randomly a group from  $n_i$ 
                 $g_o \leftarrow G_o(n_i, j)$ ; // get group in  $n_i$  at  $j$ 

                // Build and add group to new schedule
                 $g = new(g_o)$ ; // build a new group with different aircraft
                 $G \leftarrow g \cup G$ ; // add new group to new solution

                 $n_i \leftarrow g_o \setminus n_i$ ; // remove used group from  $n_i$ 
                break;
            end
        end
         $N = N - 1$ ; // decrease number of groups left to generate
    end

    return  $G$ ; // returns the new schedule
end

```

9. Results for the Steady State Evolutionary Algorithm

Several experiments were conducted to evaluate the performance of the different operators for the problem data presented in Section 8. Many results obtained from the different experiments cannot be said to follow a normal distribution for a significance level of 0.05, so the t-test for statistical significance cannot be used and the Mann-Whitney statistical significance test was used instead.

The data sets used in the experiments conducted are presented in Section 8, and a summary is presented in Table 8 for convenience. Multiple combinations of the parameters used in both the SSEA and the CGA were used, the values for which are summarized in Table 9.

The following section looks at the performance of the SSEA for all combinations of the parameters presented in Tables 8 and 9, when only one operator is used in each execution. Experiments were next conducted with various combinations of the same operators and parameters, which are compared and studied, and a single operator is used to establish their performance, Section 9.2. The results from the experiments conducted for the SSEA are compared to those obtained from the application of other metaheuristics, which are then studied in Section 9.3. Some experiments were next conducted to determine the effects of the parking restriction, where a parking activity may only be assigned to a gate if such a gate is already assigned to either the flight arrival or departure associated with it, the results of which are presented in Section 9.4. Finally in Section 9.5 experiments are conducted to study the effect on the performance of the different operators when the number of flights increases.

9.1. Single Operators

In this section the results obtained from the experiments conducted when only one operator is used per run are presented and studied. The parameters are those summarized in Tables 8 and 9.

Given that the MEMOs lack the ability to assign activities from the dummies, thus keeping fitness low, removal of this disadvantage improves the Multi Ex-change Mutation Operator by introducing a recovery stage after the child solution has been generated whereby unassigned activities (those assigned to one of either dummy) are randomly assigned to a gate where possible. This is very important as it is a constraint on the static problem achieving full assignment, $N \geq \text{LMAP}$, as presented in Section 5.1. As a solution reaches full assignment this extra step may not be needed, therefore having no detrimental effect on the speed of the operator. The word 'Improved' is added at the beginning of the name of the original base operators to identify the new operators. This explains the poor fitness results obtained when the MEMO were used when compared with those obtained by their 'Improved' version, as shown in Table 10. The Improved MEMOs do not need to be combined with any of the Dummy operators to allow them to increase the number of assignments to gates.

Table 8: Summary of the general data used in the experiments conducted

Name	Values
Terminal	4
Topology	3-pier
Data sets	London Heathrow airport schedules for Terminal 4 shown in Table 6
Fitness weights	$W1 = 0.75193$, $W2 = 0.6$, $W3 = 0.00025$, $W4 = 0.11$ and $W5 = 0.25$

Table 9: Summary of the algorithm parameters used in the experiments conducted

Name	Values
Total number of iterations	800,000
Population sizes	5, 10, 15, 30, 50, 100, 200, 500, 1000 and 2000
Iterations in a generation (ℓ only for SSEA)	1, 5, 10 and 20
Replacement Strategies	ES, SUMS, IS1ES and IS1SUMS (later IS1fES and IS1fSUMS are used too)
Member Selectors	Tournament Selection
Operators	C1P, C2P, MEFNR2, MEFNR3, MEFNR4, MEFNR5, IMEFNR2, IMEFNR3, IMEFNR4, IMEFNR5, RMEFNR2, RMEFNR3, RMEFNR4, RMEFNR5, IRMEFNR2, IRMEFNR3, IRMEFNR4, IRMEFNR5

When studying all of the operators for the SSEA it was found that the population sizes which provide the best overall performance correspond to small values of between 5 to 15 for the 'Improved Multiple Exchange Operators' (Table 10) similar as was seen in the AB-SSAP for the 'Multiple Exchange Operators' at [21]. Furthermore, this also applies to all of the MEMOs examined, as shown in Tables 10 and 12. Tables 10 and 12 only show those replacement strategies and population sizes that cannot be said to be statistically significantly less fit (Mann-Whitney test) than any of the other re-placement strategies and population sizes for each of the operators and data sets studied. The population sizes in bold text are those which can be said to provide significantly statistically fitter solutions in many more cases than the other combinations considered.

A summary of all of the experiments conducted for the SSEA using different combinations of ℓ operators, selectors and population sizes (Section 8) is presented in Table 10, which only shows those combinations which best solution obtained cannot be said to be significantly statistically less fit than the best solutions obtained for any of the other combinations. Table 10 clearly shows that the Index Selection with Elitist Selection and a group size of 1 (IS1ES) replacement strategy provides significantly statistically better solutions overall, which is similar to the result obtained for the ABSSAP at [21]. The only operators covering all of the data sets studied, where a fitness solution cannot be said to be significantly statistically worse than the solutions obtained

when using any of the other combinations, are:

1. $\ell = 1$, IRMEFNR2 and IS1ES and population size of 5
2. $\ell = 5$, IRMEFNR2 and IS1ES

Similarly, the good performance of the operator IRME-FRN2 was also seen for the ABSSAP at [21] for operator RMEFNR2, particularly where the number of BSSs is greater than the UMAP, which also equates here to the data set cases considered for Terminal 4 at London Heathrow airport.

The 'Index Selector' operators remove all solutions having the same fitness, but which do not have interesting parts useful in future generations. Further-more, this also reduces the fitness pressure, since there are fewer solutions with duplicates. This approach may be too strong so a new version of the Index Selection with Elitist Selection (ISxES) and Index Selection with Stochastic Universal Modified Sampling (ISxSUMS) is proposed, whereby duplicates are removed only if the population is greater than expected. The results, which include the new 'Index Selector', are shown in Table 11, where an 'f' is inserted in the selector's name to represent the new 'Index Selector'. The performance of both versions of the 'Index Selector', with and without full removal of duplicates, generally appears to be close. The overall approaches achieving solutions where fitness is no worse than in the other operators in all the data sets considered are:

1. $\ell = 1$, IMEFNR2 and IS1fES
2. $\ell = 5$, IRMEFNR2, IS1ES and population size of 5

When the new 'Index Selector' is used these empirical results show an improvement by the IMEFNR2 over previous results using a lower number of iterations per generation (ℓ), which also corresponds to a small increase in search pressure as less fit solutions in the population have less chance of being selected (the diversity is retained for longer at a higher ℓ). Whereas, the removal of duplicate solutions by the replacement strategy equates to an increase in diversity. The new 'Index Selectors' were not applied to the ABSSAP. These results show that the solutions obtained may potentially be improved when using this selection enhancement in other resource assignment problems, such as the ABSSAP, especially for data sets where a slight extra increase in the search pressure may be advantageous. Furthermore, the ability to activate or deactivate this characteristic may be beneficial as the search advances, based on the particular circumstances at each time, such as switching it off later in the search when diversity has sufficiently decreased in order to slightly reduce the search pressure.

9.2. Multiple Operators

In order to establish which operator combinations provide solutions which are statistically significantly fitter than the solutions obtained by single operators, or which cannot at least be said to be worse, two operator approaches, composed of a crossover and mutation, are analyzed and compared with the single operators for the AGAP. It emerges that multiple operators do not perform well when compared to single operators. This may be because the crossover operators cannot provide different

Table 10: SSEA` single operators which provide statistically significantly fitter solutions for the data sets from 6th to 12th September 2010, where the 'Index Selectors' remove all duplicates

ℓ	Operator	Selector	Population sizes			
			H4T100906	H4T100907	H4T100908	H4T100909
1	IMEFNR2	IS1ES	5, 10, 15, 30	5	5	5, 10, 15, 30
		IS1SUMS	5			
		SUMS	5, 10, 15, 50, 500		15, 50, 200, 1000	
	IRMEFNR2	IS1ES	5, 15	5	5	5, 10
	RMEFNR2	IS1ES				10
5	IMEFNR2	IS1ES	10	5	5, 15	5, 10
	IRMEFNR2	IS1ES	5, 10, 15	5	5, 15	5, 10
	MEFNR2	IS1ES				5
		SUMS			200	
10	IMEFNR2	IS1ES	5, 10, 15	5	5	5, 10
	IRMEFNR2	IS1ES	5, 10, 30	10	5, 10	5, 10
	MEFNR2	IS1ES	5			
20	IMEFNR2	IS1ES	5, 10, 15		10	5, 10
	IRMEFNR2	IS1ES	5, 10	10	5, 10	5
	MEFNR2	IS1ES	15			10

ℓ	Operator	Selector	Population sizes		
			H4T100910	H4T100911	H4T100912
1	IMEFNR2	IS1ES	5	5, 10	5, 10, 15, 30
		SUMS	15, 30, 100, 2000	10, 50, 100, 500, 2000	
	IRMEFNR2	IS1ES	5, 15	5, 15	5, 10
5	IMEFNR2	IS1ES		5	5
	IRMEFNR2	IS1ES	15	5	10
10	IRMEFNR2	IS1ES	5	5, 10	
	MEFNR2	IS1ES		5	
20	IMEFNR2	IS1ES	15		
	IRMEFNR2	IS1ES	5	5	
	MEFNR2	IS1ES		10	

Table 11: SSEA` single operators which provide statistically significantly fitter solutions for data sets from 6th to 12th September 2010

ℓ	Operator	Selector	Population sizes			
			H4T100906	H4T100907	H4T100908	H4T100909
1	IMEFNR2	IS1ES	5	5, 10, 15	5	5, 10, 15
		IS1fES	10	10	10	10
		SUMS		15, 50	15, 50, 200, 1000	
	IRMEFNR2	IS1ES	5	5, 10	5	5, 15
		IS1fES			10	10, 15

5	IMEFNR2	IS1ES	5		5, 15	
		IS1fES	5	5, 10	5	5
	IRMEFNR2	IS1ES	5	5	5, 15	5
		IS1fES	5	5		5
	MEFNR2	SUMS			200	
	IMEFNR2	IS1ES	5	10	5	
10	IMEFNR2	IS1fES	5	10	5, 10	5
		IS1ES			5, 10	5, 10
	IRMEFNR2	IS1fES	5		5, 10	5
		IS1ES		10, 15	10	5, 10
	IMEFNR2	IS1fES	5		5, 10	5, 10, 15
		IS1ES	10		5, 10	5
20	IRMEFNR2	IS1fES		5		
		IS1ES		15		
	MEFNR2	IS1ES				
		IS1fES				

solutions, where the parent solutions selected are identical, which may occur more often at a later time in the search when the population has lost diversity, such that many more duplicates and solutions with less differences may exist. To alleviate or remove this disadvantage, the solution selector should be modified to take account of the operator characteristics which will be used when selecting solutions to generate a new solution. Alternatively the crossover operators may be used only early in the search, where more diversity exists and there are therefore fewer duplicates. Furthermore, cases may exist where even when the parent solutions are different, the new solutions are the same as some of the solutions already present in the population. This may, however, be reduced by increasing the population size and using a replacement strategy which is able to maintain the population diversity for longer, namely Stochastic Universal Modified Sampling (SUMS) and ISxSUMS.

Furthermore, given the good performance of the mutation operators when used in the SSEA on their own, some experiments were executed using a 'Probability Single Multi Operator' with both a crossover operator and a mutation operator with a probability from 0.1 to 0.9. The solutions obtained by the 0.1 crossover + 0.9 mutation were statistically significantly fitter than those obtained by the other combinations of operators and either of the crossover operators on their own. The 1-point crossover did perform better than the 2-point crossover, which may be attributed in part to the fact that some service periods are very long and the extra hard constraints, i.e. parking, can only be assigned to a gate which has previously been assigned to the arrival and/or departure flights of the same aircraft. This may in turn reduce efficiency when using time regions, as it is more probable that the time limits of the region fall within those long time services, so reducing the number of activities with which to exchange. It also appears that IS1ES and SUMS assist in reaching fitter solutions.

9.3. Other Metaheuristics

Following the results for the SSEA, two other meta-heuristics are studied and compared with the proposed SSEA. Firstly, the Tabu Search (TS) is considered, which adds the best solution in each local walk to the tabu list. Table 12 shows a summary of the statistical significance with a significance level of 0.05 for the TS, where operators provide solutions which cannot be said to be less fit than any of the solutions obtained by the other operators, and which also cover all of the data sets for different local walk sizes and tabu list sizes. These empirical results show that the TS performs better for multi exchange mutation operators with a higher number of gates between which to exchange assignments, i.e. $n = 3$, than those seen for the SSEA. Higher n in the Multi Exchange Mutation operators corresponds to children with a potentially greater number of differences than their parents, which should mean more diversity over a longer period.

Several experiments were designed and executed to examine the performance of the CGA when using the same operators as previously described, so this implementation of the CGA does not correspond to the standard definition of the CGA since it does not make use of a binary representation, and does not use binary or random mutation operators as classically presented in [30]. Both 1-point and 2-point crossover operators together with one of the previously described mutation operators were used, with a probability of 0.99 for the crossover operators and 0.01 for the mutation operators with population sizes of 500, 1000 and 2000. Also based on the good performance of the mutation operators, some experiments with a probability of 0.9 crossovers and 0.1 mutations were executed to establish whether a higher percentage of mutation operators were desirable for the data sets used. These were later extended to include a population size of 400, given that a number of the experiments provided good results for a population size of 500.

Table 12: TS summary of statistical significance of fitness with a significance level of 0.005 and different 'tabu list' sizes for the data sets of September 2010

Algorithm	Walk Size	Operator	Tabu List Sizes			
			H4T1009dd			
			Mon 6	Tue 7	Wed 8	Thur 9
TS	10	IMEFNR3	5	5, 10	5, 15	15
		IRMEFNR3	5, 30	15	30	15
	30	IRMEFNR3	5, 15	10	5, 15	5, 10, 15, 30
Algorithm	Walk Size	Operator	Tabu List Sizes			
			H4T1009dd			
			Fri 10	Sat 11	Sun 12	
TS	10	IMEFNR3	10, 30	10	5, 30	
		IRMEFNR3	5, 10, 15, 30	10, 15	5, 10, 15, 30	
	30	IRMEFNR3	5, 10, 15, 30	30	5, 10, 15	

Table 13 provides a summary of the results, only showing those operators which provide solutions which cannot be said to be statistically significantly less fit than any of the other operators studied for the CGA. A reduction in the preferred population size can be seen for the combined operators when compared with the single operators used, as was also previously seen when using SSEA. The solutions obtained by the probability of 0.9 crossover + 0.1 mutation were significantly statistically fitter than those obtained by the other combination of operators for the CGA, and cannot be said to be less fit than those solutions obtained by the alternative operators evaluated. These results show that a higher participation by mutation operators is advantageous in this problem.

The 1-point crossover did perform better than the 2-point crossover, which may partly be attributable to the fact that some service periods are very long and the extra hard constraints, i.e. assignment of parking activities, may reduce the efficiency when using time regions. In these cases, it is more probable that the time limits of the region fall within those long services, thus reducing the number of assigned activities for use in the children. Table 13 only shows combinations of operators with the replacement strategy Elitist Selection (ES) which indicates that the tendency of the ES in reducing the diversity, which is normally maintained for longer by the larger population sizes typically used by crossover operators when used alone, is a beneficial one for the data sets considered.

Table 13: CGA summary of statistically significant fitness with a significance level of 0.005, 1-point (C1P) and 2-point (C2P) crossover, mutation operators and different population sizes

Operator		Selector	Population Sizes						
			H4T1009xx						
Crossover 0.9	Mutation 0.1		6	7	8	9	10	11	12
C1P	IMEFNR3	ES	400 and 500	400, 500 and 2000	400 and 500	400, 1000 and 2000	400 and 500	400, 500 and 1000	400, 500, 1000 and 2000
	IRMEFNR2	ES	400 and 500	500 and 1000	400	400, 500 and 1000	400, 500 and 2000	400 and 2000	2000
	MEFNR2	ES	500	400, 500, 1000 and 2000	400, 500 and 2000	400, 500, 1000 and 2000	400, 1000 and 2000	500 and 1000	2000
	RMEFNR2	ES	400	400 and 500	400	1000 and 2000	400, 1000 and 2000	1000	400 and 2000

9.4. No Restrictions on Assigning Parking Activities

Some experiments were also executed when no hard constraint was applied to the assignment of parking activities, such that they could be assigned to any gate. The algorithms have been identified by appending an extra character, '+', to the name. The summary Tables 14 and 15 only show those combinations providing solutions no less fit than any other solution considered

using all of the data sets. Given that both models with and without the extra parking constraint appear in Table 14, then both models cannot be said to provide statistically significantly worse solutions than the other. Nevertheless the preferred population sizes for the SSEA and the IRMEFNR2 operator is slightly lower than when the parking hard constraint is in place, whereas for the crossover the preferred population sizes are also slightly lower.

Table 14: Summary of algorithms which provide statistically significantly fitness solutions for the data sets from 6th to 12th September 2010 and both models

Algorithm	Operator		Selector	Population Sizes						
				H4T1009xx						
				6	7	8	9	10	11	12
CGA	C1P 0.9	IRMEFNR2 0.1	ES	500, 1000	400, 500, 1000	400	400, 500	400, 500, 2000	400, 2000	400, 500, 1000, 2000
		MEFNR2 0.1		400, 500, 1000, 2000	400, 500, 1000, 2000	400, 500, 2000	500	1000, 2000	500, 1000	1000, 2000
		RMEFNR2 0.1		400, 500	400, 1000, 2000	400	400	400, 1000, 2000	1000	400, 500, 2000
CGA+	C1P 0.9	IMEFNR3	SUMS	400, 500	400	400	400, 500	400, 500	400	400, 500
		IMEFNR3 0.1		400	400	400	400	400, 500	400	400, 500
SSEA	1	IMEFNR2	IS1fES	10	10	10	10	5	5, 15, 30	5, 10
	5	IRMEFNR2	IS1ES	5	5	5, 15	5	15	5	10
SSEA+	1	IMEFNR2	IS1ES	5, 10, 15	5, 10, 15, 30	5, 10, 15, 30	10, 15	10, 15	5, 10	5, 10, 15, 30
		IRMEFNR2		5, 10, 15, 30	5, 10, 15, 30	5, 10, 15, 30	30	30	5, 10	5, 10, 15, 30
	5	IMEFNR2		5, 10, 15	5, 10, 15, 30	5, 10, 15, 30	30	30	5, 10, 30	5, 10, 15, 30
		IRMEFNR2		5, 10, 15, 30	5, 10, 15	5, 10, 15, 30	5, 10, 15, 30	10, 15, 30	30	5, 10, 15, 30
	10	IMEFNR2		10, 15, 30	5, 10, 15, 30	5, 10, 15, 30	15, 30	15, 30	5, 10, 15, 30	5, 10, 15, 30
		IRMEFNR2		5, 10, 15, 30	5, 10, 15	5, 10, 15	10, 15, 30	10, 15, 30	15	5, 10, 15, 30
	20	IMEFNR2		5, 10, 15, 30	5, 10, 15, 30	5, 10, 15, 30	10, 15	10, 15	5, 15	5, 10, 15, 30
		IRMEFNR2		5, 10, 15	5, 10, 15	5, 10, 15	30	30	5, 10	5, 10, 15, 30

Table 15: TS with a single operator which provides statistically significantly fitness solutions for the data sets from 6th to 12th September 2010 and both models

Algorithm	Walk Size	Operator	Tabu List Size						
			H4T1009xx						
			6	7	8	9	10	11	12
TS	10	IMEFNR3	5, 10	15	5, 15	5	10, 30	10	15
		IRMEFNR3	15	15	30	5, 30	5, 10, 15, 30	10, 15	5
	30	IRMEFNR3	10	5, 10, 15, 30	5, 15	5, 15	5, 10, 15, 30	30	5, 15
	50	IRMEFNR3	5, 30	5, 15, 30	10	5, 30	5, 10, 15, 30	10, 15	5, 10, 30
TS+	10	IRMEFNR3	5, 10, 15, 30	5, 10	5, 30	5, 10, 15	10, 30	5, 10, 15, 30	5, 10, 15, 30
		IRMEFNR4	5, 10, 15, 30	5, 30	10, 15	5, 15	5, 10, 15	10	5, 15, 30
	30	IMEFNR3	5, 10, 15, 30	5, 10, 15, 30	10, 15	30	15, 30	5, 10, 15, 30	5, 10, 15, 30
		IRMEFNR4	10, 15, 30	10, 15	5, 30	5, 10, 15, 30	5, 15, 30	10, 15	5, 15, 30
	50	IMEFNR3	10, 15, 30	10, 15, 30	5, 10, 15, 30	10, 15	15	10, 15	5, 10, 15, 30
		IRMEFNR4	5, 30	5, 10, 15, 30	10, 30	5, 15	5, 15	5, 15, 30	5, 10, 15, 30

9.5. Generate New Base Schedules

In this section some experiments were designed and executed to evaluate the performance of the operators when used in the SSEA for a lower number of available gates for the number of flights. A set of new schedules was generated based on those already available from London Heathrow airport Terminal 4 from 6th to 12th September 2010 as described in Section 8.1 and summarized in Table 7.

The new data sets all have a number of gates ($N=23$) larger or equal to the UMAP, so it should be possible to assign all the flights to gates. However on reviewing the parking activities there was found to be an insufficient number of gates to which

to assign all of the activities ($N < LMAP_p$), with the exception of the first and last data sets. The most difficult data sets to solve were for H4T100907 with UMAP equal to 23 BSSs, which is also the number of gates, followed by the H4T100909, which has the highest $LMAP_p$ and $UMAP_p$ as well. Thus all flights from the sets generated can be assigned to a gate as $N \leq UMAP$, which meets one of the conditions for a real static airport problem.

Experiments using the SSEA were executed for the new base schedules, and a summary of the results is shown in Table 16. The results show an improvement on the solution obtained by the IMEFNR2, IMEFNR3 and IRMEFNR3 as it is similarly seen at [21] for MEFRN2, MEFRN3 and RMEFRN3 when fewer resources (BSSs and gates) are available.

Table 16: SSEA` single operator with significantly statistically fitter solutions for data sets generated from original date sets from 6th to 12th Sept 2010 and 23 gates

ℓ	Operator	Population Sizes			
		N4T100906	N4T100907	N4T100908	N4T100909
1	IMEFNR2	5		10, 15, 30	5, 10
	IRMEFNR2			3	5, 10
	IMEFNR3		5, 10, 30		
	IRMEFNR3		10, 30		
5	IMEFNR2	5, 10, 15		15, 30	5
	IRMEFNR2	10, 15		5, 10, 30	5, 10
	IMEFNR3		15		
	IRMEFNR3	5	5, 10		
10	IMEFNR2	10, 15		5, 10, 30	
	IRMEFNR2			10, 15, 30	
	IMEFNR3		5, 15		
	IRMEFNR3	5			
20	IMEFNR2	5, 10, 15		5, 10	5, 10
	IRMEFNR2	5, 10		15	
	IMEFNR3		5, 15		
	IRMEFNR3		5		

ℓ	Operator	Population Sizes		
		N4T100910	N4T100911	N4T100912
1	IMEFNR2	5	5, 10	5, 15
	IRMEFNR2	5	5	5
5	IMEFNR2			10
	IRMEFNR2	5	10	5
10	IMEFNR2		5, 10	10, 15
	IRMEFNR2		5	
20	IMEFNR2	5		
	IRMEFNR2		5, 10	5, 15

Results of the experiments executed for the CGA, different operators and the data sets generated are summarized in Table 17, which shows a preference for lower population sizes than when used alone in the SSEA, which may be taken as an indication of the mutation operator's influence. This also reveals a preference for a lower use of the crossover operators and a higher use of the mutation operators, indicating a departure from the general view as to what extent mutation operators should be used in a CGA, as may also be noted in the results of the real data sets for Lon-don Heathrow airport Terminal 4. The deterioration

in the performance of the 2-point crossover operator may in part be a consequence of the need to identify two cutting points in time which delimit the time region within which the assignments are copied from each parent. This, plus the long service time postulated, may excessively reduce the effective range of assignments from which to copy, thus reducing the operator's efficiency. The crossover operator does not consider assignments where the base service duration lies between two different time sections from which to copy, and this situation is more likely to occur in cases of longer base service duration.

Table 17: CGA significantly statistically fitter solutions for data sets generated from original date sets for 6th to 12th September 2010 and 23 gates

Operator		Selector	N4T1009xx						
Crossover 0.9	Mutation 0.1		6	7	8	9	10	11	12
C1P	IMEFNR2	IS1ES	400, 500, 1000	400, 500	400, 500	400, 500	400, 500	400, 500	400, 500
		SUMS		400					
	IRMEFNR2	IS1ES	400, 500	400, 500, 1000	400		400, 500	400, 500	400, 500
		SUMS		400, 500				400, 500	
	IMEFNR3	IS1ES			400		400	400, 500	
		SUMS						500	
	IRMEFNR3	IS1ES	400		500	400, 500		400, 500	
C2P	IMEFNR2	IS1ES			400, 500	400			
	IMEFNR3	IS1ES			500				
	IRMEFNR3	IS1ES			400				

10. Conclusions

This study presented the SSEA adapted for the AGAP, which it has been obtained by adapting the SSEA operators originally used for the ABSSAP in [21], [22] and [20].

Different algorithms and their parameters were studied to find characteristics which could be used to identify the algorithm and parameters most appropriate to the AGAP. Both the model and algorithms are derived by modifying those presented in previous work, and are based on the specific characteristics of the problem. These approaches were tested on real data from London Heathrow airport, using a fitness function composed of the weighted sum of the different real world objectives currently used in London Heathrow airport. When there were plentiful gates to which the flights could be assigned, there was little difference between the algorithms studied (SSEA, CGA and TS) when the same operators were used. Nevertheless, the mutation operators used are potentially faster than the crossover operators and have been able to cover more search space. There is also potential for combining these algorithms to generate new ones, which may improve the solutions still further. This potential may also be better fulfilled by combining algorithms with significant differences in their underlying approach, e.g. SSEA which is a population based approach, and TS which is an individual approach (local search).

The SSEA has been shown to provide fitter solutions for the Improved Range Multi Exchange with Fixed Number of Resources with two gates between which to exchange assignments (IRMEFNR2), and a sufficient number of gates to which all flights and parking activities can be assigned ($N \geq \text{UMAP}_p$, Upper Maximum Assignment Point with Parking) with a preference for the IS1ES, and in some cases also its modified version (Index Selection with Elitist Selection and a group size of 1 (IS1fES)). As the number of gates in the problem decreases, both the original and improved Multi Exchange between Fixed Number of Resources (MEFNRn and Improved Multi Exchange with Fixed Number of Resources (IMEFNRn)) with a higher 'n' (number of gates between which to exchange assignments), are preferable, as it was also seen for the ABSSAP.

The TS with Multiple Exchange Mutation operators has been seen to perform better for a higher number of gates between which to exchange assignments than the same operators for the SSEA. This is believed to be a consequence of the higher number of gates between which to exchange assignments, extending the search to a wider area of the search space. This would help to find a better solution, but only where the extra search space covered is not too wide, since this may also re-duce the effectiveness of the iterations as there is more danger of straying into disinterested areas of the search space. The SSEA achieves the same, however,

partly as a result of the differences within its population of solutions, so it does not necessarily need to extend the search further as this may well increase the number of iterations used to investigate uninteresting areas of the search space. This effect depends on both the problem under study and the model used in the two algorithms, which have a direct impact on the shape of the search space, as can be seen when comparing these results with those obtained for the ABSSAP ([22]). The good performance of the Multiple Exchange Mutation operators was also seen to extend to the CGA, where a higher probability of mutation was preferred, which I attribute to the good results provided by these mutation operators. Nevertheless too high an intervention by the mutation operators has been seen to be detrimental, perhaps in part due to the potential disruptive effect of the mutation operators.

It is envisaged that use of the serial crossover presented might be better suited to this problem than the 1-point and 2-point crossover operators studied here, given the longer service time required for the combined arrival/departure flights and parking. In the case of the serial crossover, since cut(s) in the time delimiting the areas of assignments for copying only affects gates, they are less likely to have activities between the two intersection sites than in the case of the other crossover operators used here, which clearly affects activities which are copied from each parent. The time an aircraft spends parked at a gate has a considerable effect on the operations which take place upstream in the overall air-port operation, especially when some of the resources required, such as gates, are limited. Delays in starting the departure sequencing may have important effects on the departure itself, which in turn may also require other aircraft to extend the time during which they are held at the gates. This could well affect other flights arriving which have had the same gates assigned to them. It would be therefore advisable to account for the effect of potential disturbances in the assignment plan.

To establish the validity of the model (original model) a different model was also considered (new model) where the extra constraint for the parking activity does not exist. The empirical results for the new model when compared with the original model show that both models find good solutions and it cannot be said that either is better. Nevertheless the new model performs well in a wider range of parameter values, making it preferable.

However, as the number of parking activities increases, so the search space also increases. This will increase the time required by the new model to find good solutions, so the use of the extra constraint (original model) should reduce the time taken to find good solutions, making it preferable.

Acknowledgement

This work was supported by the Engineering and Physical Sciences Research Council through a Doctoral Training Award, and NATS Ltd. I am grateful to the University of Nottingham for giving me the opportunity to take part in this project.

References

1. O Babic, D Teodorovic, V Tosic. Aircraft stand assignment to minimize walking. *Journal of Transportation Engineering*. 1984;110(1):55-66.
2. RS Mangoubi, DFX Mathaisel. Optimizing gate assignments at airport terminals. *Transportation Science*. 1985;19(2):173-188.
3. RA Bihr. A conceptual solution to the aircraft gate assignment problem using 0, 1 linear program-ming. *Computers& Industrial Engineering*. 1990;19(1-4):280-284.
4. A Lim, B Rodrigues, Y Zhu. Airport gate scheduling with time windows. *Artificial Intelligence Review*. 2005;24(1):5-31.
5. H Ghazouani, M Hammami, O Korbaa. Solving airport gate assignment problem using genetic algorithms approach. 2015 4th International Conference on Advanced Logistics and Transport (ICALT). 2015;175-180.
6. A Asco. Constructive and Evolutionary Algorithms for Airport Baggage Sorting Station and Gate Assignment Problems. PhD thesis, School of Computer Science. 2013.
7. CM Pintea, PC Pop, C Chira, D Du-mitrescu. A hybrid ant-based system for gate as-signment problem. HAIS (AAE Corchado and W Pedrycz, Eds.). *Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg, 2008;5271:273-280.
8. M Marinelli, M DellOrco, D Sassanelli. A metaheuristic approach to solve the flight gate as-signment problem. *Transportation Research Pro-cedia*. SIDT Scientific Seminar. 2015;5:211-220.
9. ORP van Schaijk, HG Visser. Robust flight-to-gate assignment using flight presence probabilities. *Transportation Planning and Technology*. 2017;40(8):928-945.
10. ER Mueller, GB Chatterji. Analysis of air-craft arrival and departure delay characteristics. AIAA's Aircraft Technology, Integration and Operations (ATIO). (Los Angeles, California). American Institute of Aeronautics and Astronautics. 2002;5866:1-14.
11. T Obata. The quadratic assignment problem: Evaluation of exact and heuristic algorithms. Rensselaer Polytechnic Institute. 1979.
12. PM Pardalos, F Rendl, H Wolkowicz. The quadratic assignment problem: A survey and re-cent developments. In *Proceedings of the DI-MACS Workshop on Quadratic Assignment Problems*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society. 1994;16:1-42.
13. E Cela. *The Quadratic Assignment Problem: Theory and Algorithms (Combinatorial Optimization)*. Springer. 1998.
14. EL Lawler. The qadratic assignment problem. *JSTOR*. 1963;9(4):586-599.
15. M Seker, N Noyan. Stochastic optimization models for the airport gate assignment problem. *Transportation Research Part E: Logistics and Transportation Review*. 2012;48(2):438-459.
16. SH Kim, E Feron. Impact of gate assignment on gate-holding departure control strategies. *Digital Avionics Systems Conference (DASC)*. IEEE/ AIAA. 2012;4E3-1-4E3-8.
17. F Jaehn. Solving the flight gate assignment problem using dynamic programming. *Z Betriebswirtsch*. 2010;80(10):1027-1039.
18. C Li. Airport gate assignment- a hybrid model and implementation. *CoRR*. 2009;abs/0903.2528:1-5.

19. X Hu, E Di Paolo. An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. *Evolutionary Computation*. 2007 IEEE Congress. 2007;55-62.
20. A Asco, JAD Atkin, EK Burke. An evolutionary algorithm for the over-constrained air-port baggage sorting station assignment problem. 9th Intl Conf on Simulated Evolution and Learning. SEAL2012 (L Bui, Y Ong, N Hoai, H Ishibuchi, P Suganthan, Eds.). Lecture Notes in Computer Science. Springer Berlin Heidelberg. 2012;7673:32-41.
21. A Asco. An evolutionary algorithm and operators for the airport baggage sorting station problem. *Soft Computing*. 2018;1-29.
22. A Asco. An analysis of robustness approaches for the airport baggage sorting station assignment problem. *Journal of Optimization*. 2016.
23. U Dorndorf. Project Scheduling with Time Windows: From Theory to Applications. Contributions to Management Science. Physica-Verlag. 2002.
24. U Dorndorf, E Pesch, T Phan-Huy. Constraint propagation techniques for the disjunctive scheduling problem. *Artificial Intelligence*. 2000;122(1-2):189-240.
25. A Asco, J Atkin, E Burke. An analysis of constructive algorithms for the airport baggage sorting station assignment problem. *Journal of Scheduling*. 2014;17(6):601-619.
26. C Darwin, AR Wallace. On the tendency of species to form varieties; and on the perpetuation of varieties and species by natural means of selection. *Journal of the Proceedings of the Linnean Society of London*. 1858;3(9):45-62.
27. G Mendel. Experiments in Plant Hybridisation. The Nature Research Society of Brunn. 1865.
28. H Ding, AL Rodrigues, Y Zhu. The over-constrained airport gate assignment problem. *Computers and Operations Research*. 2005;32(7):1867-1880.
29. H Ding, A Lim, B Rodrigues, Y Zhu. Aircraft and gate scheduling optimization at airports. *System Sciences. Proceedings of the 37th Annual Hawaii International Conference*. 2004;1530-1605.
30. JH Holland. Adaptation in Natural and Artificial Systems. University of Michigan Press. 1975.