An Evolutionary Algorithm for the Over-constrained Airport Baggage Sorting Station Assignment Problem

Amadeo Ascó[1], Jason A. D. Atkin[1], and Edmund K. Burke[2]

[1] School of Computer Science, University of Nottingham
[2] University of Stirling

Abstract. Airport baggage sorting stations are the places at which the baggage for each flight accumulates for loading into containers or directly onto the aircraft. In this paper the multi-objective and multi-constraint problem of assigning Baggage Sorting Stations in an Airport is defined and an Evolutionary Algorithm is presented, which uses a number of different operators to find good solutions to the problem. The contributions of these different operators are studied, giving insights into how the appropriate choice may depend upon the specifics of the problem at the time.

1 Introduction

Passengers at an airport proceed to the check-in desks assigned to their flight where they leave their baggage to be processed. The baggage enters the baggage system at this point, where it is processed and delivered to baggage sorting stations (BSSs). There it is temporarily stored before being sorted and transported by carts to the aircraft.

This paper considers the task of assigning baggage sorting stations to flights at an airport, which will already have been allocated to stands along piers around the terminals. The aim is to maximise the number of flights which are assigned to sorting stations, using the sorting stations which are most conveniently situated for the stands when possible, while ensuring that the gaps between sorting station usage are at least as large as a specified buffer time, to cope with real world perturbations and uncertainties.

Research into a similar problem was previously performed by [1], who described the problem and applied the activity selection algorithm. In [2], we studied the quality of solutions which could be obtained from a variety of constructive algorithms. Different algorithms were found to prefer different objectives and a selection of these constructed solutions has been used for initial solutions of the evolutionary algorithm which is described in this paper.

The Airport Baggage Sorting Station Assignment Problem (ABSSP) has many similarities to the Airport Gate Assignment Problem (AGAP), which has had considerable study in the past. The basic gate assignment problem is a quadratic assignment problem, shown in [3] to be NP-hard. [4] used a Genetic

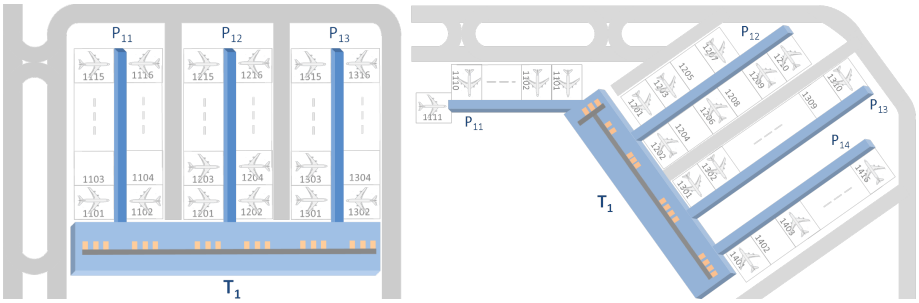
(The final publication is available at link.springer.com)

Algorithm (GA) for the AGAP with uniform crossover. [5] applied a GA, using an elitist selection method, with an additional mechanism composed of mutation operators to repair the infeasible solutions which resulted from applying the crossover operator.

We initially applied CPLEX and Gurobi to the Integer Linear Programming (ILP) representation of the ABSSP, both with and without an initial solution chosen as the fittest of the constructed solutions presented in [2]. Both systems were allowed a maximum runtime of 24 hours. Although CPLEX ran out of memory (using over 84GB of disk space) after 16hrs on a 2.4GHz Windows 7, 64bit machine with 4GB RAM. Results were inadequate, so we developed the evolutionary algorithm which is described in this paper. This has turned out to perform much better, even with a much shorter search time.

2 Airport Baggage Sorting Station Problem

Aircraft are usually parked at their allocated stand, around an airport terminal. Two example layouts which are considered in this paper are shown in Figure 1, showing aircraft parked around the piers of a terminal, and illustrating the position of baggage sorting stations at the bases of the piers. The aim of this work is to allocate each flight to a sorting station. In general, it is better to allocate a flight to a sorting station on the same pier, but it is not usually important which of the sorting stations in that group the flight is allocated to. Furthermore, a minimum gap should be ensured between flights.



(a) 3 piers and 48 stands. (b) 4 piers and 46 stands.

Fig. 1. Topologies.

Let $i \in \{0, \ldots, N\}$ denote a sorting station where sorting station 0 represents a dummy sorting station to which flights are assigned, if they cannot be assigned to a real sorting station. Let $j \in \{1, \ldots, M\}$ denote a flight. Let T_j denote the required service time for flight j (1 hour for short haul and $1\frac{1}{3}$ hours for long haul) and let B_j denote the desired buffer time for flight j - the time for which its

sorting station should be idle prior to this flight being serviced. B_j is assumed to be 15 minutes for short haul and 30 minutes for long haul flights. Let e_j denote the end service time for flight j and let t_j denote the target starting service time for flight j ($t_j = e_j - T_j - B_j$). Let y_{ij} be a decision variable defined to be 1 if flight j is assigned to sorting station i, and 0 otherwise. The aim is to find y_{ij} values such that the objective function (4) is maximised, subject to the constraints expressed by (1), (2) and (3).

$$\sum_{i=0}^{N} y_{ij} = 1 \quad \forall j \in \{1, \ldots, M\} \tag{1}$$

$$y_{ij} + y_{il} \leq 1 \;\; \forall i \in \{1, \ldots, N\}, j \neq l, t_l + B_l < e_j \leq e_l \tag{2}$$

Equation (1) states that each flight can be assigned either to exactly one real sorting station or to the dummy sorting station. Inequality (2) states that flights cannot be assigned to the same real sorting station if their service times overlap.

If the service periods of two flights are closer together than the buffer time of the second flight, then they can still be allocated to the same sorting station, but the buffer time will have to be reduced. The required reduction in buffer time for flight l is denoted r_l and can be calculated using (3), where $0 \leq r_l \leq B_l \; \forall \; l \in \{1, \ldots, M\}$.

$$r_l \geq (y_{ij} + y_{il} - 1) \cdot (e_j - t_l) \;\; \forall i \in \{1, \ldots, N\} \tag{3}$$

Although this is inherently a multi-objective problem, the importance of ensuring maximal assignment of flights to sorting stations (top priority) and the relative importance of keeping reasonable buffer times (second priority) allows these objectives to be combined into a single compound objective (4) with weights W_1, W_2 and W_3 chosen to implement these priorities.

$$\max \left(W_1 \cdot \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} - W_2 \cdot \sum_{j=1}^{M} r_j - W_3 \cdot \sum_{i=1}^{N} \left(C_j \cdot \sum_{j=1}^{M} y_{ij} \cdot d_{ij} \right) \right) \tag{4}$$

The first element in (4) aims to maximise the assignment of flights to sorting stations, the second aims to minimise the reduction in buffer times and the third aims to minimise some distance cost associated with the assignments, where C_j is a factor related to the amount of baggage for flight j (assumed to be 1 in all cases for this paper) and d_{ij} is a measure of the distance or cost incurred from assigning sorting station i to flight j. This aims to ensure that flights are allocated to appropriate sorting stations.

3 Algorithm

In this paper we describe an Evolutionary Algorithm which we have developed for solving the Airport Baggage Sorting Station Assignment Problem (ABSSAP).

Our algorithm uses a number of custom problem-specific operators and we have categorised them as either mutation operators, if they modify a single existing solution, or crossover operators, if they combine multiple solutions. We describe the algorithm, selection methods and operators in this section.

3.1 Steady State Evolutionary Algorithm

A steady state GA does not replace all the parent solutions by their children as in generational GAs. In our Steady State Evolutionary Algorithm (SSEA) the next population is obtained by applying the population selector (S_p) to the current population. One of the operators is applied to the required number of individuals which are chosen by applying the member selector (S_m) to the population. This selection and modification is called an iteration and is repeated L times for each population. The newly obtained individuals are then added to the population which will then constitute the new current population and a population selector is used to determine which population members to keep. This is repeated until the termination condition is reached, as shown in Algorithm 1.

Algorithm 1: Steady State Evolutionary Algorithm

input : Initial population P_0
input : Number of iterations in a generation $L \in \mathbb{Z}^+, L > 0$
input : Operators; $O_1 : p_1, O_2 : p_2, \cdots$ and $O_R : p_R$ where $0 < p_k \leq 1 \; \forall \; k \in (1, \cdots, R)$ and $\sum_{k=1}^{R} p_k = 1$
input : Population selector, S_p
input : Population member(s) selector, S_m

```
1  begin
2      P ← P_0; // initialise population
3      repeat
4          P ← S_p(P); // get the current population for this generation
5          P_t ← ∅; // empty temporary population
6          i ← 0; // initialise the generation
7          repeat
8              Select randomly an operator, O_k; // roulette wheel selector
9              Q ← S_m(P, O_k); // select population member(s)
10             Q ← O_k(Q); // generate new solution(s)
11             P_t ← P_t ∪ Q; // add new solution(s)
12             i ← i + 1;
13         until i = L or Termination Condition;
14         P ← P ∪ P_t; // add new solutions to the current population
15     until Termination Condition;
16     return P;
17 end
```

3.2 Population Selectors

The population selector (S_p) has the responsibility of selecting the solutions within the current population that will form part in the next generation. A comprehensive analysis of selection schemes used in Evolutionary Algorithms can be found in [6]. We present below an overview of the population selection algorithms used.

Elitist Sampling (ES): Select the fittest μ population members from the current population.

Stochastic Universal Modified Sampling (SUMS): Stochastic Universal Sampling was introduced by [7]. SUMS was used since the order of magnitude of the fitness values are much larger than the differences between the fitness values, so it was not appropriate to use Stochastic Universal Sampling. The population members are mapped by sections, as in Roulette Tournament Selection, $[p_{i-1}, p_i)$ with $p_0 = 0$ for $p_i = \frac{\sum_{j=1}^{i} f_j - F}{\sum_{j=1}^{\lambda} f_j - F}$ where $F = f_\lambda - (f_{\lambda-1} - f_\lambda)$, f_j corresponds to the fitness of solution j and λ is the current population size. μ individuals are selected by obtaining an initial random number (rnd) within $[0, \frac{1}{\mu})$ and subsequent ones spread $\frac{1}{\mu}$ from the previous one. Solution i is selected once per each $p_{i-1} \leq rnd + \frac{j-1}{\mu} < p_i \forall j \in \{1, \ldots, \mu\}$.

Remainder Stochastic Sampling (RSS): This method was also investigated due to its diversity retention properties (see [8]), but provided worse results, so we have not discussed the results in this paper.

3.3 Member Selectors

The member selectors (S_m) distribute the chances that the individuals have to take part in the production of new offsprings in a generation. The roulette wheel selection method was used as the member selector on this paper.

3.4 Operators

Whenever a time has to be determined (for instance for a 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.

Mutation. Guided mutation operators are introduced here which are based upon local search operators which guarantee to generate feasible solutions.

Dummy Single Exchange Mutation Operator (DSEMO). This operator is equivalent to the 'Apron Exchange Move' used by [9] and [10]. 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' sorting station to a randomly selected sorting station, potentially moving another flight back onto the 'dummy' sorting station if it can no longer fit in. This operator may increase the number of assignments if the operation does not move a flight back onto the 'dummy' sorting station. It requires some flights to be unassigned in the parent solution.

Multi Exchange Mutation Operators. A time period is randomly selected, t_{rs} to t_{re}. All assignments for which the base service times are entirely within the time period are then moved to the next sorting station in the set, as shown in Figure 2, as long as they will fit. These operators generalise the 'Interval Exchange Move' which was presented by [10], and cannot increase the number of assignments. Two variants have been developed:

1. *Multi Exchange with Fixed Number of Resources (MEFNRn)*: The number of sorting stations to exchange flights between is fixed at n, where $2 \leq n \leq N$.
2. *Multi Exchange with Random Number of Resources (MERNRn)*: The number of sorting stations to exchange flights between is randomly chosen each time, between 2 and n, where $2 \leq n \leq N$.

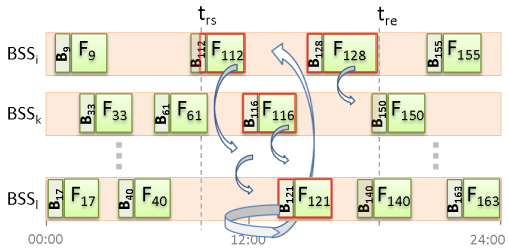


Fig. 2. Multi Exchange Mutation Operator.

Multi Exchange By Pier Mutation Operators. These operators are a specialised case of the Multi Exchange Mutation Operators, where the sorting station selection element ensures that no two consecutive sorting stations in the set are on the same pier. The idea is to improve the distance objective by encouraging the movement of assignments between piers. Again, this operator cannot increase the number of assignments. As for the Multi Exchange Mutation Operators, two variants have been created: *Multi Exchange By Pier with Fixed Number of Resources* (MEBPFNRn) and with Random Number of Resources (MEBPRNRn).

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. Such flights are added to the set of flights which will be considered for assignment to the next sorting station, potentially reducing this number. Again, this operator cannot increase the number of assignments. As for the Multi Exchange Mutation Operators, two variants have been created: *Multi Exchange with Fixed Number of Resources* (RMEFNRn) and with Random Number of Resources (RMERNRn).

Crossover (reproduction). The crossover operators involve the generation of new solutions from multiple parents. Each parent will be chosen using the Population Member Selector (S_m) and multiple child solutions will be generated.

2-Point Crossover (C2P): Two points in time are randomly selected to generate a time window. A child solution is created from each parent by re-assigning all flights within the time window to the sorting station they were assigned to in the other parent. Although the flight timings are identical across all solutions, flights in the swapped region may overlap flights which are not swapped for some sorting stations. Such overlapping flights in the swapped region are re-assigned to other sorting stations, if possible, or assigned to the dummy sorting station otherwise.

1-Point Crossover (C1P): This is a specific case of the above 2-Point Crossover, where the window extends to the end time of the solution.

4 Results

Since it would be unrealistic to assume that baggage from a flight at a stand in one terminal is serviced by a baggage sorting station in another terminal (e.g. passengers usually go through security and board flights from the same terminal at which their baggage was checked in), this paper is centred on a single terminal.

For this paper, we assumed that all flights require only one sorting station. Two datasets, provided by NATS (formerly National Air Traffic Services) were used, from 16^{th} December 2009 (194 flights) and from 1^{st} March 2010 (163 flights) for Heathrow terminal 1. All the experiments were executed for 800,000 iterations. The same random number generator was used throughout.

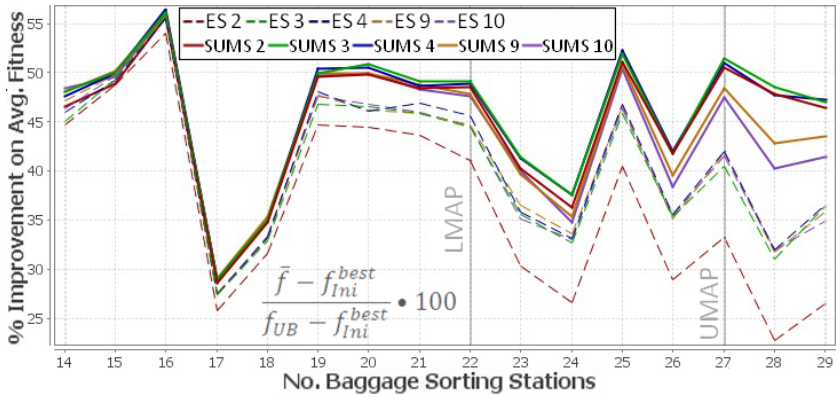
The Steady State Evolutionary Algorithm (SSEA) which was used in these experiments had $L = 1$ and a population size of 30. The value of W_1 was determined by running initial experiments with our Evolutionary Algorithm (EA) using different values, from 15 to 100, for the data set of 16^{th} December 2009, 3-pier topology, a fixed W_2 of 0.008 and W_3 of 1. A value of $W_1 = 90$ appeared to give an appropriate balance between the objectives and was adopted. The SSEAs were run with just one of the presented operators at a time, to show their individual behaviour in the considered landscapes. A distance of $1 unit$ is assumed between sides of the same pier, $2 units$ to move between piers and a distance of zero is assumed for those assignments that are as close as possible.

The Lower Maximum Assignment Point (LMAP) is the minimum number of sorting stations needed to assign all flights without buffer times. The Upper Maximum Assignment Point (UMAP) is the minimum number of sorting stations needed to assign all flights without reducing the service time. The LMAP and UMAP points will be observed to be useful when interpreting the results.

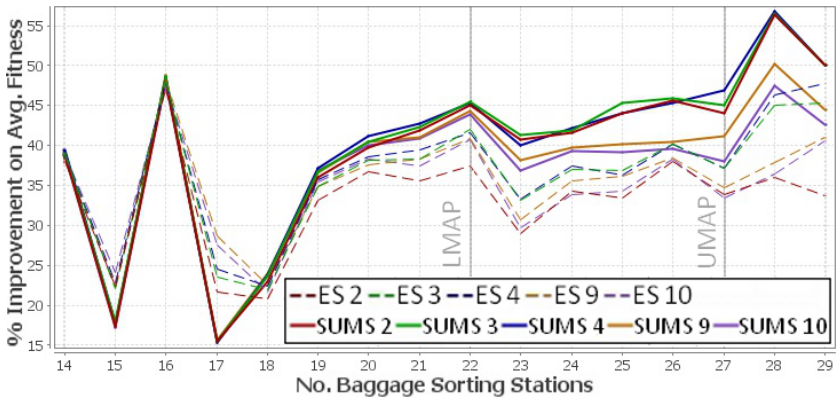
4.1 Experiments

Figure 3(a) shows the percentage improvement in average fitness of the results for the Multi Exchange with Fixed Number of Resources (MEFNRn) operator, $n \in \{2, 3, 4, 9, 10\}$, with the population selectors Elitist (ES) and Stochastic Universal Modified Sampling (SUMS), applied to different numbers of baggage

sorting stations (BSSs) with respect to the best initial solution (f_{Ini}^{best}, 0%) and the Upper Bound obtained by CPLEX (f_{UB}, 100%). The achieved improvements range from 25% to over 55%, depending on the number of BSSs. It should be noted that 100% improvement corresponds to solutions which reach the upper bound for their specific case, whereas 0% corresponds to no improvement over the best initial (constructed) solution.



(a) 4-pier and 46 stands.



(b) 3-pier and 48 stands.

Fig. 3. Improvement on average fitness for 194 flights, MEFNRn, $n \in \{2, 3, 4, 9, 10\}$.

When the number of sorting stations is lower than the Lower Maximum Assignment Point (LMAP), it is impossible to fully assign all of the flights. Since the initial solutions already provide at least one solution which is guaranteed to achieve the maximum number of assignments for the given number of baggage sorting stations ([2]), the EA can only improve upon this by improving the

'distance' and 'reduction in service' objectives. For example, it may be possible to swap out flights in order to lower the reduction in service time. The results show that the largest benefits in this region are due to the ability to swap which flights are assigned, however, in practice it is unlikely for an airport to operate in this mode - where some flights cannot actually be serviced.

In contrast, full assignment can be achieved between the LMAP and Upper Maximum Assignment Point (UMAP), but only by reducing the service times. As long as there are solutions with unassigned flights, it may be advantageous to use the operators which use the dummy sorting station. These operators do not contribute once all solutions have full assignment, and should no longer be used.

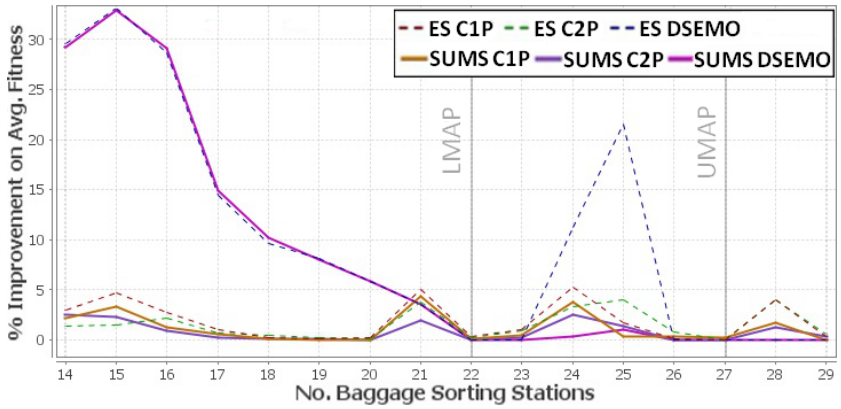
Finally, after the UMAP full assignment can be achieved without reduction in service time so the only objective that contributes to the fitness is the distance. The fitness is still improved by the EA but by a much smaller amount since the weights and the values of this objective are much lower than in the others.

All of the operators always improve upon the initial solutions by at least 10% except for the crossover and DSEMO operators (Figure 4(a)). Crossover alone could not always find improvements over the constructed initial solutions, since crossover operators are dependent upon the quality of the building blocks which are already present in the population. If the current solutions do not contain many useful differences, then the crossover will not be able to improve the solution. Similarly initial solutions of lower fitness may contain better building blocks from the point of view of the crossover operator, but may be removed too early in the search.

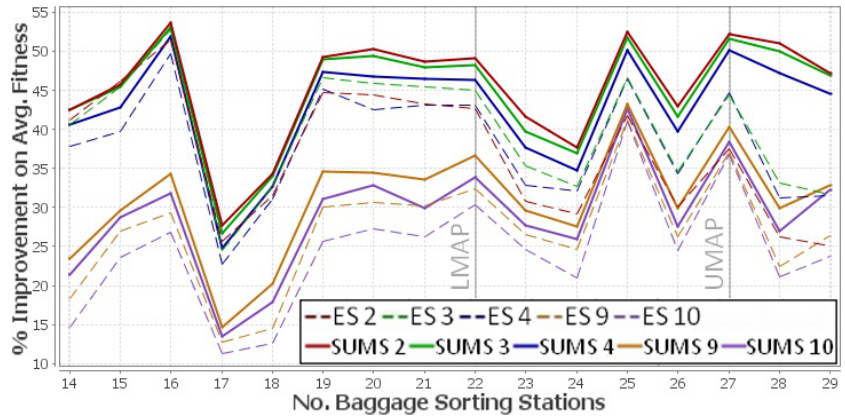
When $N < LMAP$, DSEMO provided better solutions in most cases than all of the crossover operators which are considered in this paper, irrespective of the selection method used, but the solutions are of lower fitness than those obtained by the other considered mutation operators.

For $N \geq LMAP$ the solutions obtained by DSEMO are generally no better than the initial solution and the main reason for this is that the highly fit solutions in this area will already have all of their flights assigned. However, not all individuals in the population will have maximal assignments, so in some of the instances, where the operator is applied to a lower fitness solution, doing so can reach better areas of the search space and achieve a better fitness. Thus, the use of DSEMO may help the algorithm to reach different parts of the search space when the initial population is composed of more diverse solutions. These results imply that it may be wise to adjust the use of DSEMO dynamically to accommodate to the changing structure of the population.

With the Multi Exchange with Fixed Number of Resources (MEFNRn), the results observed for the different population selectors do not differ much on average for different values of n. This is not the case with the Range Multi Exchange with Fixed Number of Resources (RMEFNRn), where the obtained solutions deteriorate as n increases, so $n = 2$ and $n = 3$ provided the best solutions, as shown in Figure 4(b). When considering the execution speed of using



(a) C1P, C2P and DSEMO.



(b) RMEFNRn, $n \in \{2, 3, 4, 9, 10\}$.

Fig. 4. Improvement on average fitness for 4-pier and 194 flights.

these operators, lower values of n are also faster, as is also true for MEFNRn and MEBPFNRn.

Figures 3(a) and 4(b) show that SUMS provides better average results than an elitist strategy, which could be considered to be a consequence of SUMS preserving the diversity better than the ES.

In general the results did not differ across airport topologies (i.e. 3 or 4 pier models) or days (i.e. number of flights), however there were some minor differences. For example, in the 3-pier topology, with 194 flights, shown in Figure 3(b), Elitist selection performed better than SUMS for 17 sorting stations and 15 sorting stations (163 flights). We believe that this is because the constructive algorithms perform well on these problems, so that the initial population contains solutions which are much better, and the diversity which SUMS introduces is not beneficial for the search. This behaviour is not present in the 4-pier topology, for

either of the datasets for the range of sorting stations studied, shown in Figure 3(a).

5 Conclusions

The aim of this research was to gain more general insights into the appropriate operator choices, especially since some operators (such as crossover) are slower than others to apply, and the appropriate operator percentages may differ depending upon the situation.

The Dummy Single Exchange Mutation Operator (DSEMO) extends the search to other areas of the search space which may help to improve the solutions, but it is only useful when there are unassigned flights, eg. for N $<$ LMAP. Whereas for N $\geq$ LMAP, the DSEMO should only be used when the selected solution from the population has unassigned flights, most commonly nearer the start of the search.

Given the diverse ways the studied operators work it is expected that their combination will further improve the solutions. Furthermore, the combination of different operators together with an adaptive method of selecting operators seems to be the most promising approach for future work.

In further research we plan to shed some light upon the effects of changing the value of L, both statically and dynamically, as well as to consider statically and dynamically changing the proportions of usage of different operators.

Acknowledgments. We are grateful to NATS Ltd and to EPSRC for providing the funding for this project, and especially to John Greenwood (NATS) for his continuous support.

References

1. Abdelghany, A., Abdelghany, K., Narasimhan, R.: Scheduling baggage-handling facilities in congested airports. Journal of Air Transport Management **12**(2) (March 2006) 76 – 81
2. Asc, A., Atkin, J.A.D., Burke, E.K.: An analysis of constructive algorithms for airport baggage sorting station assignment. Presented to the Journal of Scheduling (2011)
3. Obata, T.: The quadratic assignment problem: Evaluation of exact and heuristic algorithms, Troy, New York (1979)
4. Hu, X., Di Paolo, E.: An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. In: Evolutionary Computation, 2007. CEC 2007. IEEE Congress on. (sept. 2007) 55 –62
5. Bolat, A.: Models and a genetic algorithm for static aircraft-gate assignment problem. The Journal of the Operational Research Society **52** (2001) 1107–1120
6. Blickle, T., Thiele, L.: A comparison of selection schemes used in evolutionary algorithms. Evolutionary Computation **4**(4) (Winter 1996) 361 – 394
7. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application, Hillsdale, NJ, USA, L. Erlbaum Associates Inc. (July 1987) 14–21

8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston, MA (1989)
9. Ding, H., Lim, A., Rodrigues, B., Zhu, Y.: Aircraft and gate scheduling optimization at airports. In: System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on. (jan. 2004) 1530 – 1605
10. Ding, H., Rodrigues, A.L., Zhu, Y.: The over-constrained airport gate assignment problem. Computers and Operations Research **32**(7) (July 2005) 1867 – 1880