# A Constructive Heuristic Approach for Single Airport Slot Allocation Problems

Sha Wang*, John H. Drake†, Jamie Fairbrother‡, John R. Woodward*
*School of Electronic Engineering and Computer Science, Queen Mary University of London, London, E1 4NS
†School of Informatics, University of Leicester, Leicester, LE1 7RH
‡Management School, Lancaster University, Lancaster, LA1 4YW
Email: sha.wang@qmul.ac.uk

***Abstract*—With growth in air transportation expected to continue, the mitigation of operational inefficiency and consequent delays is becoming increasingly important. Slot allocation, as a means of demand management at congested airports, has a significant impact on wider airport operations. This requires sophisticated approaches, to intelligently allocate scarce airport resources to unevenly distributed traffic demand for the use of airport facilities. This paper presents a novel heuristic approach to solve the single airport slot allocation problem. The approach has been tested on real-world data from three medium-sized airports. We introduce a constructive heuristic framework which is able to generate feasible solutions to the model. Within this framework, a number of ordering heuristics are developed to order slot requests to be scheduled, and an allocation algorithm is developed to allocate slots to a request determined by the ordering heuristic. Experimental results suggest that the order in which slot requests are scheduled has a significant impact on the solution quality.**

***Keywords*—Airport slot allocation; Airport demand management; Constructive heuristics**

## I. Introduction

According to [1], in 2040, air traffic in Europe is expected to grow to 16 million flights. This is a total growth of 53% compared to 2017. However, the expansion of airport capacity is restricted by a series of political, environmental, physical and other factors. The severe imbalance between demand and capacity, added to the difficulty of increasing airport capacity in the short term, make it increasingly hard to access airport facilities.

In order to allocate scarce airport resources more efficiently, slot allocation was introduced by the International Air Transport Association (IATA) to manage runway capacity. A slot corresponds to the right of a flight to use the airport facilities for landing or take-off on a specific date and time interval. It is currently practised at 175 'slot coordinated airports' (also known as 'level 3' airports), serving over 1.5 billion passengers each year [2], including many of the most busy airports around the world outside of the United States.

This study focuses on strategic-level slot allocation, which is carried out under the regulations of the Worldwide Slot Guidelines (WSG) twice a year for the summer and winter coordination seasons [3]. About five months before each season, airlines provide initial slot requests to the appointed airport slot coordinators. Less than one month after receiving the requests, airport slot coordinators must provide an initial slot allocation solution to all airlines, in a 'neutral, transparent and non-discriminatory' way and apply the following priorities [3]: *Historic requests* (a request for a series of slots, which were operated by the same airline in the last equivalent season at least 80% of the time, will be given first priority, also known as historic precedence or the grandfather right, of holding the same slots for the next equivalent season), *Changes to historic requests* (any *Historic request* that plans to operate with a different time, aircraft, terminal etc., from the previous equivalent season should have lower priority than *Historic request*) and *New entrants requests* (if a carrier requests no more than four slots on any day at the airport, the airline's requested slots will be granted *New entrant* status). Half of the remaining available slots, after *Historic* and *Change-to-historic* slots have been allocated, must be allocated to *New entrants* to the extent that such requests are outstanding. Slot requests that do not fall into above three priority categories are considered as *other requests* with no priority. They can only be scheduled once all requests with priorities have been allocated. The initial slot allocation plan will then be discussed at the Slot Conference. Adjustments and negotiations are made during the conference to resolve conflicts stemming from slot allocation decisions made at multiple airports, and disputes among airlines competing for the same slots [3].

## II. Previous Related Work

Slot allocation problems have been studied in the context of both single airport [4]–[9] and airport network level [10]–[12], and solved using both exact and heuristic approaches. Only the work that considers single airport slot allocation problems is reviewed here. For a survey paper, see [13].

### A. Models

Single airport slot allocation problems have been modelled as single objective, bi-objective or multi-objective Integer Linear Programming (ILP) problems in the existing literature. Zografos et al. [4] first formulated the single airport slot allocation problem as a single objective ILP model. The optimisation objective of this model was to minimise the total schedule displacement (i.e., sum of time difference, measured by number of slots, between the allocated time and the requested time of all slot requests). In order to take into

account the request priority classes, the model was applied hierarchically for *historic*, *new entrants* and *other* requests. This model was later extended to incorporate other objectives such as fairness [6], [9], maximum displacement and number of requests scheduled within an acceptable time window [14]. More recently this model formed the basis of a mechanism to incorporate airline preferences for which of its requests should be displaced [9].

Ribeiro et al. [7] proposed an alternative multi-objective formulation which they call the Priority-based Slot Allocation Model (PSAM). The model adopted a weighted-based objective function, which considered four objectives: the number of slots rejected, maximum individual displacement (i.e., displacement imposed on any slot), total schedule displacement and the number of slots displaced. Each component of the objective function was weighted according to its relative importance. This approach therefore largely relies on stakeholder's preferences. Recently, Ribeiro et al. [8] extended their model to a new model, PSAM-ATR, which incorporates apron and terminal capacity. The PSAM-ATR model employs a weighted-based bi-objective function which prioritises the minimisation of the maximum individual displacement over the total schedule displacement.

### B. Solution approaches

Solution approaches to single airport slot allocation problems can be broadly classified into two categories: exact approaches and heuristic approaches. Zografos et al. [4] developed an ILP-based algorithm which dynamically added capacity constraints to the model as they are needed. They tested their model on real-world data from three medium-sized airports. Results showed that the algorithm can effectively solve small-scale slot allocation problems and find the optimal solution within a few seconds. It was more difficult to solve the last priority class of a moderate size problem (optimality gap of 1.58% after a few minutes). The PSAM model [7] was solved by direct application of the CPLEX solver. It was applied to real-world instances of two medium-sized airports, Madeira and Porto, and could be solved to optimality within a few minutes.

Ribeiro et al. [8] solved the PSAM-ATR model at three airports with two methods: CPLEX and a 'matheuristic' approach. CPLEX found the optimal solution for two medium-sized airports in a few minutes or hours. However, it remains intractable for a large airport with over 200,000 movements a year (optimality gap of 2-5% after 7 days). The proposed matheuristic approach combines a constructive heuristic and an improvement heuristic based on the large-scale neighbourhood search algorithm. Request series of each priority class were sorted by decreasing order of the number of days. The constructive heuristic then divided requests into a certain number of groups, and solved the PSAM-ATR model for each group using CPLEX, in order to obtain an initial feasible solution. Starting from this solution, the improvement heuristic implemented a 'destroy and repair' process to iteratively improve the quality of the solution. Results showed that a trade-off between the number of groups and the solution quality exists, and may vary for different problem sizes and features (e.g., demand-capacity imbalances). For the same large airport, a near-optimal solution with optimality gap of 2-5% could be found in 30 minutes and a solution with optimality gap of 0-0.03% was found in 10 hours.

Considering the decision horizon in which the initial slot allocation plan needs to be made (less than one month in practice) [3], more effective and computationally efficient solution approaches are required to solve large-scale single airport slot allocation problems. As for large-scale slot allocation exact methods are too slow and computationally infeasible. Motivated by this, the aim of this paper is to develop heuristic approaches to scale better to larger instances than exact methods do, but not to tackle large-scale problems due to a lack of data availability. Not only will these enable the fast construction of efficient schedules, but these could also be used to speed-up exact methods by providing warm-start solutions. In addition, the order in which slot requests are processed during construction of the initial solution is only considered in one existing paper [8], where requests were simply ordered by number of operational days the slot is desired. To address this issue, this paper develops more request ordering methods and a different constructive heuristic approach.

## III. Slot allocation model

### A. Problem statement

About five months before each coordination season, airlines make requests for pairs of arrival and departure slots to the airport coordinators using a standardised format. Each of these requests specify, inter alia, arrival and departure times, the dates for which the request applies, and the priority category of the request. For a more detailed description of the input request data, see [7].

If the request applies for five or more weeks, then the slots should be allocated in series, that is, the slots allocated for each arrival and departure of the request is for the same time. Requests which fall below this threshold may be scheduled independently on different days. In order to allow an arriving aircraft to prepare for a subsequent departure, the arrivals and departures of a request must be scheduled with sufficient *turnaround time*. Runway constraints limit the number of arrivals, departures or total number of movements that can be scheduled in a given period. These typically take the form of *rolling capacity constraints* which limit runway movements for a given duration rather than a specific interval. Finally, requests should be scheduled according to the priority classes described in the Introduction above. That is, historic requests should be scheduled first, followed by change to historic requests and so on.

The quality of a feasible solution is usually measured in terms of *displacement*, which is the absolute time difference between requested and allocated slots. The aim of the slot allocation problem is to allocate slots to as many requests as possible, in a way which minimises some aggregate measure of

displacement, while respecting all of the constraints described above.

### B. Model Formulations

The model formulation used for this slot allocation problem is based mainly on the model presented in [4]. Unlike the previous model, the model we use here considers two additional cases. In our model it is possible for one or more requests to be rejected, i.e. the request is not allocated a slot. Additionally we consider the case of 'split days' requests, where there the departure occurs the day after the arrival. Both arrival and departure flights are only allowed to be scheduled on the requested operational days. For a detailed description of the model used, see Appendix A.

## IV. Solution Approach

The proposed constructive heuristic approach aims to build a feasible solution incrementally from an empty solution, with the aim of minimising the total schedule displacement. Specifically, the solution construction process relies on a number of ordering heuristics and an allocation algorithm. The ordering heuristics aim to order the requests that have not been scheduled according to the difficulty of allocating them to feasible slots. The allocation algorithm then decides which slots to assign each request to, given the current solution.

### A. Solution construction procedure

The framework of the constructive heuristic approach is described in Algorithm 1. The constructive heuristic starts with an empty solution, and processes requests one by one to gradually build a complete solution. Firstly, Algorithm 1 takes a list of request series to be scheduled as input. Secondly, a static ordering heuristic (line 2) or a dynamic ordering heuristic (line 4) is employed to sort the list according to one or multiple criteria of each request. Next, the algorithm schedules requests in the list iteratively until all requests have been considered. Specifically, for each iteration, the algorithm selects the first request on the list (determined by the current ordering heuristic) and removes it from the list. An allocation algorithm is then employed to schedule or reject the request under consideration.

**Algorithm 1** Framework of the constructive heuristic

Input: a list of slot requests, airport parameters
Output: slot allocation solution

1: initialising an empty solution
2: sort the list according a static ordering heuristic
3: **while** request list is not empty **do**
4: sort the list according to a dynamic ordering heuristic
5: select the first request series $m$ in the list
6: remove $m$ from the list
7: call the *allocating algorithm*
8: **end while**

### B. Ordering heuristics

The rationale of the ordering heuristics is that requests which are more difficult to schedule should be allocated first, with the hope that the easier requests can fit around the difficult ones while maintaining feasibility [15]. Based on this rationale, we developed a number of static and dynamic ordering heuristics to order slot requests to be scheduled. The difference between a static and dynamic ordering heuristic is that, the former orders requests based on immutable information (e.g., flight type or the number of operation days), thus generating a fixed order. The latter orders requests according to the problem state at that time, thus the order changes dynamically as the solution is constructed. It is worth noting that, in order to respect priority classes, requests are first ordered according to priorities, and then within each class, requests are ordered by one of the ordering heuristics.

*1)* ***Static deterministic ordering heuristics****:*

- *Most days first (MD)*: requests are ordered, in a non-increasing order, in terms of the number of operation days requested.

*2)* ***Dynamic deterministic ordering heuristics****:*

- *Most conflicts first (MC)*: requests are firstly ordered, in a non-increasing order, in terms of the number of operation days, and secondly ordered, in a non-increasing order, in terms of the number of conflicts that they have with the solution at that time. Specifically, the number of conflicts is determined as the number of flights that have already been scheduled at the requested time periods or the feasible time periods with minimum displacement. For example, a departure request for the $55^{\text{th}}$ time period on a number of days needs to be scheduled next. If its paired arrival flight has been scheduled at the $60^{\text{th}}$ time period, and the required minimum turnaround time for this flight is 4 time intervals, then the number of conflicts of this departure request is the total number of flights that have already been scheduled at the $64^{\text{th}}$ time period on corresponding days.
- *Least residual degree first (LR)*: requests are firstly ordered, in a non-increasing manner, in terms of the number of operation days, and secondly, in non-decreasing order, in terms of the remaining capacity left in the requested time periods or the feasible time periods with minimum displacement.
- *Largest displacement cost first (LDC)*: requests are ordered, in a non-increasing order, by the cost of scheduling them into the current solution. If the requested slots of a request are currently feasible, there is no scheduling cost of this request. Otherwise, the cost of this request is the minimum schedule displacement multiples the corresponding number of days.

*3)* ***Dynamic stochastic ordering heuristics****:*

- *Random ordering based on number of days (ROD)*: requests are firstly ordered, in a non-increasing order, by the number of operation days, and then requests with same number of days are scheduled randomly.

TABLE I
SUMMARY OF SLOT REQUESTS

| Airport | Total | Historic | Change to historic | New entrants | Others |
|---|---|---|---|---|---|
| 1 | 14,956 | 4,266 | 2,966 | 466 | 7,080 |
| 2 | 33,904 | 5,460 | 8,448 | 2,668 | 17,328 |
| 3 | 46,900 | 10,448 | 11,522 | 4,080 | 20,850 |

- *Random ordering (RO)*: requests that have not been scheduled, are ordered randomly at each step of the solution construction.

### C. Allocation algorithm

The allocation algorithm described in Algorithm 2 aims to find a feasible slot time closest to the requested slot time for a selected request series $m$. The allocation algorithm first checks the feasibility of scheduling $m$ at $t_m$ with respect to turnaround time and the declared runway capacity constraints. If all of the constraints remain satisfied, the algorithm directly allocates request $m$ slots at time $t_m$ and updates the solution and capacity state. If any of the constraints are violated, the current request may be scheduled earlier or later than its requested slot time, or may be rejected (not scheduled at any time) if no feasible slots can be found. Specifically, the allocation algorithm firstly searches for feasible slots that are later than the requested slot time $t_m$ (line 5-11), before searching for feasible slots earlier than $t_m$ (line 12-18). After searching in both directions, if no feasible slots can be found, the current request and its paired request will be rejected at the same time and removed from the request list to be scheduled and the solution. Otherwise, feasible slots with smaller displacement will be assigned to the current request.

**Algorithm 2** Framework of the allocating algorithm

Input: request series $m$
Output: allocated slot time $t$ for $m$

1: **if** all constraints are met when $x_m^{t_m} = 1$ **then**
2: $t \leftarrow t_m$
3: update the solution and capacity state
4: **else**
5: **for** $i = t_m + 1$; $i \leq T$; $i$++ **do**
6: **if** all constraints are met **then**
7: $forward \leftarrow i - t_m$
8: feasible slots found
9: break
10: **end if**
11: **end for**
12: **for** $i = t_m - 1$; $i \geq 0$; $i$-- **do**
13: **if** all constraints are met **then**
14: $backward \leftarrow i - t_m$
15: feasible slots found
16: break
17: **end if**
18: **end for**
19: **if** no feasible slot found **then**
20: reject request $m$ and its paired request
21: update the solution and capacity state
22: **else if** $|forward| < |backward|$ **then**
23: $t \leftarrow t_m + forward$
24: update the solution and capacity state
25: **else**
26: $t \leftarrow t_m + backward$
27: update the solution and capacity state
28: **end if**
29: **end if**

## V. DATA AND EXPERIMENTAL RESULTS

The constructive heuristic approach was applied to solve real-world slot allocation problems from three airports. Table I summarises the number of slots requested for each airport and the distribution of slot priority categories.

The minimum turnaround time, denoted by $\ell_p$, depends on the aircraft, airline policy and airport facilities. Due to a lack of explicit data on minimum turnaround time we set it using the following rule. If the requested turnaround time for a pair of flights is less than one hour, then $\ell_{m_1 m_2}$ takes the value of the actual requested turnaround time, that is $(Tv_{m_1 m_2} + t_{m_2} - t_{m_1})$; otherwise, it equals the number of time periods equivalent to one hour.

In addition to the total schedule displacement, we employed three other metrics to measure the schedule efficiency (see Table II). Note that these metrics are observation metrics and are not objectives that have been optimised. The first metric (1) is the maximum displacement imposed on any scheduled slot (also known as the maximum individual displacement), which is motivated by literature [7], [8], [14]. It is calculated as the absolute value of the difference between the allocated time $t$ and the requested time $t_m$. Since a request series is scheduled at the same time, the displacement of each individual request in the series is the same. The second metric (2) reflects the total number of slots rejected due to infeasibility. This metric has been considered in previous works [7], [8], [10], [12]. The last metric (3) reflects the total number of slots which are displaced. This metric has been only considered previously by Ribeiro et al. [8].

TABLE II
SLOT ALLOCATION EFFICIENCY METRICS

$$\max_{m \in \mathcal{M}} \sum_{t \in \mathcal{T}} |t - t_m| x_m^t, \forall m \in M \tag{1}$$

$$\sum_{m \in M} |\mathcal{D}_m| r_m \tag{2}$$

$$\sum_{m \in M} |\mathcal{D}_m| : \sum_{t \in \mathcal{T}} |t - t_m| x_m^t > 0 \tag{3}$$

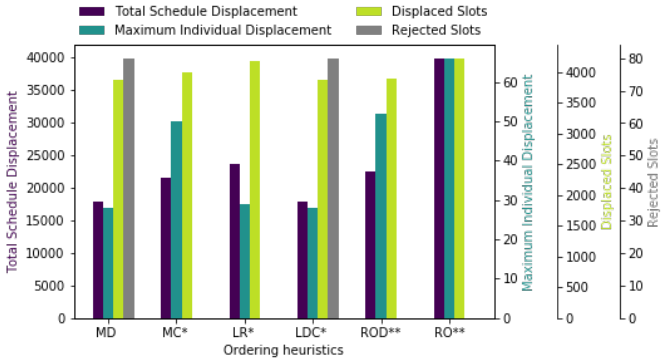
Here we investigate the performance of different ordering heuristic on the quality of the initial feasible solution obtained. This helps us to better understand the features of each ordering

TABLE III
PERFORMANCE OF DIFFERENT ORDERING HEURISTICS ON THE SCHEDULE EFFICIENCY

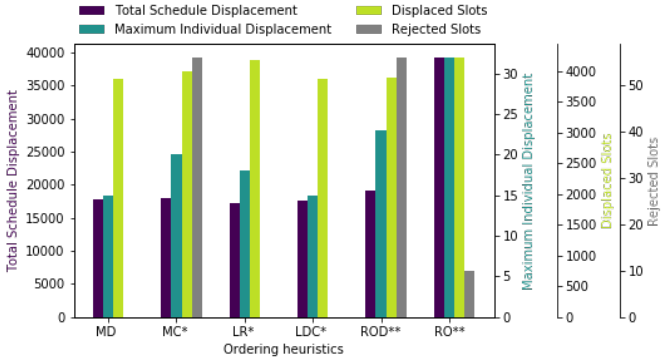
| Ordering heuristics | Rejected slots | Max Disp. (slot) | Total Disp. (slot) | Displaced slots |
|---|---|---|---|---|
| | | **Airport 1** | | |
| *MD* | **80** | **28** | **17,753** | **3,882** |
| *MC** | **0** | **50** | **21,487** | **4,005** |
| *LR** | **0** | **29** | **23,587** | **4,180** |
| *LDC** | **80** | **28** | **17,753** | **3,882** |
| *ROD*** | **0** | **52** | **22,413** | **3,897** |
| *(median)* | | | | |
| *(s.d.)* | *5.8* | *6.9* | *2,004.6* | *90.3* |
| *RO*** | **0** | **66** | **39,772** | **4,230** |
| *(median)* | | | | |
| *(s.d.)* | *0.0* | *7.7* | *6,238.0* | *209.4* |
| | | **Airport 2** | | |
| *MD* | **0** | **15** | **17,733** | **6,089** |
| *MC** | **56** | **20** | **17,976** | **5,668** |
| *LR** | **0** | **18** | **17,155** | **5,901** |
| *LDC** | **0** | **15** | **17,558** | **6,108** |
| *ROD*** | **56** | **23** | **19,160** | **5,439** |
| *(median)* | | | | |
| *(s.d.)* | *27.9* | *3.6* | *677.1* | *213.3* |
| *RO*** | **10** | **32** | **39,315** | **7,142** |
| *(median)* | | | | |
| *(s.d.)* | *47.8* | *14.9* | *7,609.4* | *389.8* |
| | | **Airport 3** | | |
| *MD* | **0** | **95** | **141,170** | **13,495** |
| *MC** | **0** | **94** | **144,346** | **14,702** |
| *LR** | **0** | **94** | **142,995** | **15,011** |
| *LDC** | **0** | **86** | **138,530** | **15,169** |
| *ROD*** | **0** | **204** | **147,908** | **13,548** |
| *(median)* | | | | |
| *(s.d.)* | *0.0* | *22.9* | *8,710.2* | *309.8* |
| *RO*** | **0** | **195** | **310,390** | **13,898** |
| *(median)* | | | | |
| *(s.d.)* | *0.0* | *13.3* | *45,611.0* | *350.7* |

*Note*: static/dynamic/stochastic ordering heuristic(*/*/**)

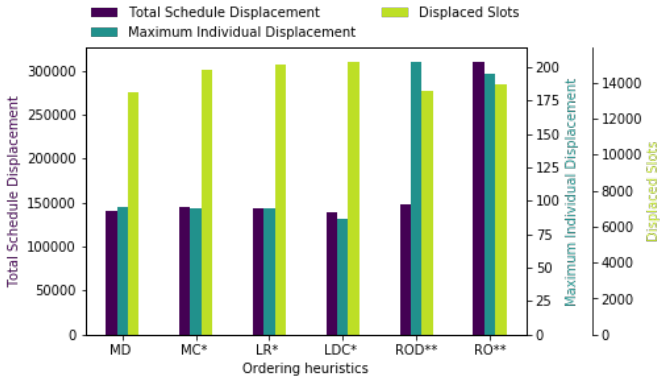
Fig. 1. Comparison of ordering heuristics




(a) Airport 1




(b) Airport 2

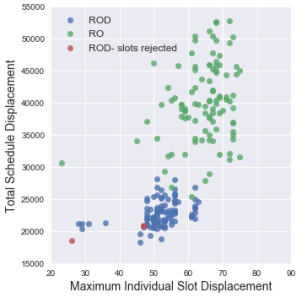



(c) Airport 3

heuristic and the relationships between schedule efficiency metrics. All six ordering heuristics were tested on each of the three airports. For stochastic heuristics (*ROD* and *RO*), 101 runs with distinct seeds were carried out, and the median and standard deviation of each metric are presented in Table III for Airport 1, 2 and 3.

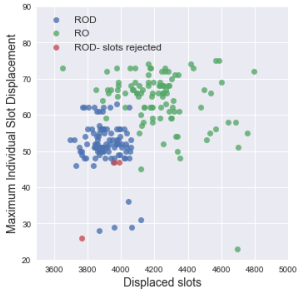
When slot requests in each priority class are scheduled in a random order (*RO* heuristic is applied), the obtained solutions were much worse than the solutions achieved by using other ordering heuristics. This indicates the existence of a 'best order' of requests to be scheduled, which may lead to optimal or near-optimal solutions. For the small Airport 1, both *MD* and *LDC* achieved a solution with the smallest total schedule displacement, maximum individual displacement and the number of displaced slots. However, two request series (corresponding to 80 out of 14,956 requested slots) were rejected. When considering randomness upon *MD*, the result of *ROD* shows that no requests were rejected and the total displaced slots remained low, but the total schedule displacement and maximum individual displacement increased by 28% and 96% respectively. The *MC* heuristic shows competitive performance as it allocated all slot requests, and it outperforms *LR* with respect to the total schedule displacement and the number of displaced slots, but the maximum individual displacement of *MC* is more than 1.78 times that obtained by *LR*. For Airport 2, results also suggest that deterministic ordering heuristics perform much better than the stochastic ordering heuristics. *LR* performs best in terms of the total schedule displacement and number of slots displaced. *MD* and *LDC* lead to solutions with smallest maximum individual displacement. The number of displaced slots tend to be reduced by using *MC* and *ROD*, even when rejected slots are counted. For Airport 3, no requests were rejected in every algorithm execution. *LDC* achieved a solution with the smallest total schedule displacement and maximum individual displacement. However, the total dis-

Fig. 2. Relationship between schedule efficiency metrics among solutions obtained by using stochastic ordering heuristics, Airport 1



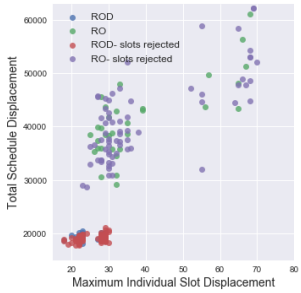

(a) Total displacement vs. Maximum displacement, $\rho = 0.488(ROD)$, $\rho = 0.219(RO)$


(b) Total displacement vs. Displaced slots, $\rho = 0.151(ROD)$, $\rho = 0.011(RO)$

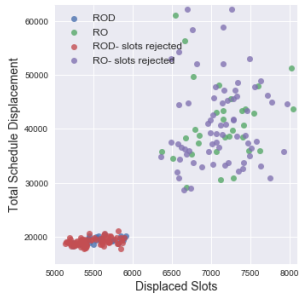



(c) Maximum displacement vs. Displaced slots, $\rho = -0.104(ROD)$, $\rho = -0.201(RO)$

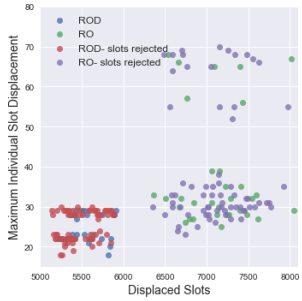
Fig. 3. Relationship between schedule efficiency metrics among solutions obtained by using stochastic ordering heuristics, Airport 2




(a) Total displacement vs. Maximum displacement, $\rho = 0.343(ROD)$, $\rho = 0.609(RO)$




(b) Total displacement vs. Displaced slots, $\rho = 0.424(ROD)$, $\rho = 0.141(RO)$




(c) Maximum displacement vs. Displaced slots, $\rho = 0.194(ROD)$, $\rho = 0.005(RO)$

placed slots is 12.4% more than the smallest value achieved by the *MD* heuristic.

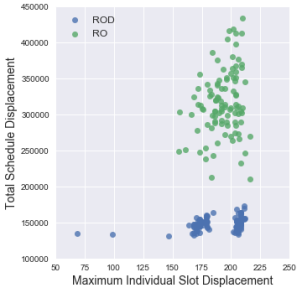
We may then conclude from the above results that: (i) the existence of rejected slots depends on the order of requests to be scheduled; (ii) stochastic ordering heuristics (*RO* and *ROD*) perform significantly worse than the deterministic ordering heuristics (*MD*, *LR*, *MC*, *LDC*), which demonstrates that the order of requests to be scheduled has a significant impact on the solution quality; (iii) in general, slot requests with the most operation days should be scheduled first in order to achieve better solutions; (iv) by using *LDC*, the total schedule displacement and the maximum schedule individual displacement are more likely to be reduced. The reason is that when calculating the potential minimum displacement cost of each request, feasibility of slots in adjacent time periods of the requested slots are also considered. In contrast, *MD* does not consider the dynamic solution state, and *MC* tends to make shortsighted decisions because it does not consider the number of flights that have already been scheduled in the adjacent time periods of the requested slots; (v) by using *ROD*, the number of displaced slots is more likely to be reduced.

The observations above provide the motivation to investigate the correlation between the different schedule efficiency metrics. Figure 2, Figure 3 and Figure 4 show a number of scatter plots, illustrating the relationships between the schedule efficiency metrics of the three airports respectively. Each green or blue dot represent a solution obtained when *RO* or *ROD* was applied. The red and purple dots represent there are slots rejected in the corresponding solution. Plots (a), (b) and (c) show the relationship between two of the three schedule efficiency metrics: total schedule displacement, maximum individual displacement and number of slots displaced. The corresponding correlation coefficients is given below each plot. We can clearly see from the plots (a) and (b) for all airports that the green dots are distributed in a distinct region to the blue dots, which means that by using *ROD*, the total schedule displacement of the initial solutions can be significantly reduced compared to *RO*. In addition, we found that there is a positive correlation between the total schedule displacement and the maximum individual displacement (the Spearman correlation coefficient $\rho$ ranged from 0.343 to 0.488 of the three airports) when *ROD* is applied. Also, a positive correlation was found between the total schedule displacement and displaced slots in Airport 2. No significant correlation was found between the maximum individual displacement and the displaced slots.

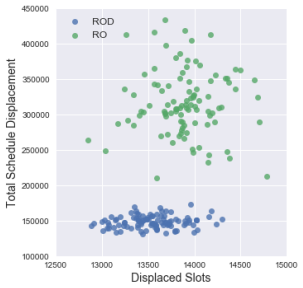
## VI. Conclusion

In this paper, we have proposed and investigated a constructive heuristic approach which is able to generate initial feasible

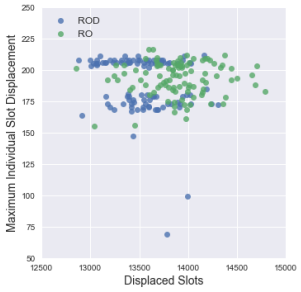
Fig. 4. Relationship between schedule efficiency metrics among solutions obtained by using stochastic ordering heuristics, Airport 3




(a) Total displacement vs. Maximum displacement, $\rho = 0.369(ROD)$, $\rho = 0.177(RO)$




(b) Total displacement vs. Displaced slots, $\rho = 0.100(ROD)$, $\rho = 0.048(RO)$




(c) Maximum displacement vs. Displaced slots, $\rho = -0.161(ROD)$, $\rho = -0.010(RO)$

solutions to a single airport slot allocation problem. In our constructive heuristic framework, six static or dynamic request ordering heuristics were employed, and the approach was tested on real-world data from three airports. We investigated the impact of the order of requests to be scheduled on the quality of the initial feasible solution. Our hypothesis that the requests which are more difficult to schedule should be allocated first is experimentally supported. Experimental results indicate that requests with most number of operation days should be scheduled first. In addition, results show that by using the *Largest minimum displacement cost first* ordering heuristic, the total schedule displacement can be reduced by 1% and 1.87% (for Airport 2 and 3 respectively) compared to the *Most number of days first*. We also found that there is a positive correlation between the total schedule displacement and the maximum individual displacement of a solution. No significant correlation was found between the maximum individual displacement and the displaced slots.

In practice, the declared capacity is not fixed and can vary at different times such as peak hours, weekends or holidays [8]. Future work will extend our to incorporate dynamic declared capacity profiles. It might also be interesting to extend the current constructive heuristic framework to solve a network-level airport slot allocation problem.


## ACKNOWLEDGMENT

The work reported in this paper has been supported by the Engineering and Physical Sciences Research Council (EPSRC) through Programme Grant EP/M020258/1 'Mathematical Models and Algorithms for Allocating Scarce Airport Resources (OR-MASTER)' and Programme Grant EP/N029496/1 'TRANSIT: Towards a Robust Airport Decision Support System for Intelligent Taxiing'. The views expressed in this paper are the authors' own opinions.


## REFERENCES


[1] EUROCONTROL, "European aviation in 2040: Challenges of growth," https://www.eurocontrol.int/sites/default/files/content/documents/official-documents/reports/challenges-of-growth-2018.pdf, 2018.

[2] IATA, "Worldwide airport slots," https://www.iata.org/policy/slots/Pages/index.aspx, 2019.

[3] WSG, "Wsg edition 9 - english version (pdf) official version," https://www.iata.org/policy/slots/Documents/wsg-edition-9-english-version.pdf, 2019.

[4] K. G. Zografos, Y. Salouras, and M. A. Madas, "Dealing with the efficient allocation of scarce resources at congested airports," *Transportation Research Part C: Emerging Technologies*, vol. 21, no. 1, pp. 244 – 256, 2012.

[5] K. Zografos and Y. Jiang, "Modelling and solving the airport slot scheduling problem with efficiency, fairness, and accessibility considerations," in *TRISTAN SYMPOSIUM 2016*, 6 2016.

[6] K. G. Zografos and Y. Jiang, "A bi-objective efficiency-fairness model for scheduling slots at congested airports," *Transportation Research Part C: Emerging Technologies*, vol. 102, pp. 336 – 350, 2019.

[7] N. A. Ribeiro, A. Jacquillat, A. P. Antunes, A. R. Odoni, and J. P. Pita, "An optimization approach for airport slot allocation under iata guidelines," *Transportation Research Part B: Methodological*, vol. 112, pp. 132–156, 2018.

[8] N. A. Ribeiro, A. Jacquillat, and A. P. Antunes, "A large-scale neighborhood search approach to airport slot allocation," *Unpublished*, 2019. [Online]. Available: https://ajacquil.heinz.cmu.edu/wp-content/uploads/sites/26/2018/09/Manuscript_Ribeiro2018.pdf

[9] J. Fairbrother, K. G. Zografos, and K. Glazebrook, "A slot scheduling mechanism at congested airports which incorporates efficiency, fairness and airline preferences," *Transportation Science*, 2019, accepted for publication.

[10] U. Benlic, "Heuristic search for allocation of slots at network level," *Transportation Research Part C: Emerging Technologies*, vol. 86, pp. 488–509, 2018.

[11] P. Pellegrini, L. Castelli, and R. Pesenti, "Metaheuristic algorithms for the simultaneous slot allocation problem," *IET Intelligent Transport Systems*, vol. 6, no. 4, pp. 453–462, 2012.

[12] P. Pellegrini, T. Bolić, L. Castelli, and R. Pesenti, "Sosta: An effective model for the simultaneous optimisation of airport slot allocation," *Transportation Research Part E: Logistics and Transportation Review*, vol. 99, pp. 34–53, 2017.

[13] K. G. Zografos, M. A. Madas, and K. N. Androutsopoulos, "Increasing airport capacity utilisation through optimum slot scheduling: review of current developments and identification of future needs," *Journal of Scheduling*, vol. 20, no. 1, pp. 3–24, Feb 2017.

[14] K. G. Zografos, K. N. Androutsopoulos, and M. A. Madas, "Minding the gap: Optimizing airport schedule displacement and acceptability," *Transportation Research Part A: Policy and Practice*, vol. 114, pp. 203 – 221, 2018.

[15] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, "A graph-based hyper-heuristic for educational timetabling problems," *European Journal of Operational Research*, vol. 176, no. 1, pp. 177–192, 2007.


## APPENDIX A
## MODEL FORMULATION

### A. Model inputs and parameters

Table I gives the inputs and parameters of the proposed model. Each day in the scheduling season is segmented into

$T$ intervals of equal length (also known as the coordination time interval), indexed by the set $\mathcal{T} = \{0, \ldots, T-1\}$. For example, if the length of the coordination time interval is 15 minutes then there are $24h * 60min/15min = 96$ intervals in a day. The set $\mathcal{D}$ consists the set of days of a slot allocation season. Each slot corresponds to an interval $t \in \mathcal{T}$ and a day $d \in \mathcal{D}$.

We use $\mathcal{M}$ to denote the request for a series of slots, and for each $m \in \mathcal{M}$ we denote by $t_m \in \mathcal{T}$ the time of the requested slots. By $\mathcal{D}_m \subseteq \mathcal{D}$ we denote the set of days to which request $m \in \mathcal{M}$ applies, and the binary parameter $b_m^d$ specifies whether or not a request series $m \in \mathcal{M}$ applies to day $d \in \mathcal{D}$. By $\mathcal{M}_{\text{arr}} \subset \mathcal{M}$ and $\mathcal{M}_{\text{dep}} \subset \mathcal{M}$ to represent respectively the set of arrival and departure requests.

The set $\mathcal{P} \subset \mathcal{M}_{\text{arr}} \times \mathcal{M}_{\text{dep}}$ consists of the pairs of arrival and departure request series which must be scheduled with appropriate turnaround time. The minimum turnaround time for the request pair $p \in \mathcal{P}$ is denoted by $\ell_p$. For some request pairs, the departure following an arrival occurs the next day. This is often the case if the arrival requests a slot near midnight. The binary parameter $v_p$ specifies whether this is the case for request pair $p \in \mathcal{P}$.

The set $C$ indexes the time scales corresponding of the runway capacities and the number of time periods for which a constraint is checked is denoted by $L_c$. The parameters $C_{tdc}^{\text{arr}}$, $C_{tdc}^{\text{dep}}$, $C_{tdc}^{\text{tot}}$ specify respectively the maximum number of arrivals, departures, and total number of movements that can be scheduled on day $d \in \mathcal{D}$ over time intervals $[t, t + L_c - 1]$.

TABLE I
MODEL INPUTS AND PARAMETERS

| | |
|---|---|
| $\mathcal{T} = \{0, ..., T-1\}$ | set of coordination time periods |
| $\mathcal{D}$ | set of coordination days |
| $\mathcal{M}$ | set of requests for slot series |
| $\mathcal{D}_m \subseteq \mathcal{D}$ | set of days to which request series $m \in \mathcal{M}$ applies |
| $\mathcal{M}_{\text{arr}}(\mathcal{M}_{\text{dep}}) \subset \mathcal{M}$ | set of arrival (departure) request series |
| $\mathcal{P} \subset \mathcal{M}_{\text{arr}} \times \mathcal{M}_{\text{dep}}$ | set of arrival-departure request pairs |
| $\ell_p$ | minimum turnaround time for request pair $p \in \mathcal{P}$ |
| $b_m^d$ | $b_m^d = 1$, if $m$ is requested for day $d$; otherwise, $b_m^d = 0$ |
| $v_p$ | $v_p$=1, if the departure occurs the next day of the arrival for request pair $p$; otherwise, $v_p$=0 |
| $\mathcal{C}$ | set of capacity time scales, indexed by $c$ |
| $L_c$ | length of time scale $c$, measured by the number of time periods |
| $C_{tdc}^{arr}$ | arrival capacity at time period $t$, day $d$ and time scale $c$ |
| $C_{tdc}^{dep}$ | departure capacity at time period $t$, day $d$ and time scale $c$ |
| $C_{tdc}^{total}$ | total capacity at time period $t$, day $d$ and time scale $c$ |

### B. Decision variables

The decision variables $x_m^t$ are binary indicators which specify whether or not request series $m \in \mathcal{M}$ is allocated slots at time $t \in \mathcal{T}$ on days $d \in \mathcal{D}_m$. Sometimes due to limited capacity, slots cannot be allocated to a request series, and in this case we say that the request is rejected. The decision variable $r_m^t$ is a binary indicator which specifies whether or not request series $m \in \mathcal{M}$ is rejected.

TABLE II
DECISION VARIABLES

| | |
|---|---|
| $x_m^t$ | $x_m^t = 1$, if request series $m$ is allocated to time period $t$; otherwise, $x_m^t = 0$ |
| $r_m$ | $r_m^t = 1$, if request series $m$ is rejected; otherwise $r_m = 0$ |

### C. Constraints

Constraint (A.1) ensures that every request series $m$ is allocated slots for one time interval or is rejected. Constraints (A.2) to (A.4) ensure that the runway capacity constraints for arrival, departure and all types of flights are satisfied. Constraint (A.5) ensures that each request pair is scheduled with sufficient turnaround time. Constraint (A.6) ensures the decision variables can only take a value of 0 or 1.

### D. Objective

The objective function (A.7) is concerned with minimising the total displacement across all slot requests. Note that rejected request series are not considered in the calculation of the objective function.

TABLE III
MODEL FORMULATIONS

$$\sum_{t \in \mathcal{T}} x_m^t + r_m = 1, \forall m \in \mathcal{M} \tag{A.1}$$

$$\sum_{m \in \mathcal{M}_{\text{arr}}} \sum_{s=t}^{t+L_c-1} b_m^d x_m^s \leq C_{tdc}^{\text{arr}}, \forall t = 0, \ldots, T - L_c + 1, d \in \mathcal{D}, c \in \mathcal{C} \tag{A.2}$$

$$\sum_{m \in \mathcal{M}_{\text{dep}}} \sum_{s=t}^{t+L_c-1} b_m^d x_m^s \leq C_{tdc}^{\text{dep}}, \forall t = 0, \ldots, T - L_c + 1, d \in \mathcal{D}, c \in \mathcal{C} \tag{A.3}$$

$$\sum_{m \in \mathcal{M}} \sum_{s=t}^{t+L_c-1} b_m^d x_m^s \leq C_{tdc}^{\text{total}}, \forall t = 0, \ldots, T - L_c + 1, d \in \mathcal{D}, c \in \mathcal{C} \tag{A.4}$$

$$\sum_{t \in \mathcal{T}} (T v_{m_1 m_2} + t) x_{m_2}^t - \sum_{t \in \mathcal{T}} t x_{m_1}^t \geq \ell_{m_1 m_2} (1 - r_{m_2}), \forall (m_1, m_2) \in \mathcal{P} \tag{A.5}$$

$$x_m^t, r_m \in \{0, 1\}, \forall m \in \mathcal{M}, t \in \mathcal{T} \tag{A.6}$$

$$minimise \sum_{m \in \mathcal{M}} \sum_{t \in \mathcal{T}} |\mathcal{D}_m| |t - t_m| x_m^t \tag{A.7}$$