Solving the Slot Allocation Problem with Flexible Scheduling of Series of Slots

Jamie Fairbrother\textsuperscript{1}  Stephen Maher\textsuperscript{1}  Konstantinos Zografos\textsuperscript{1}  Kevin Glazebrook\textsuperscript{1}

\textsuperscript{1}Centre for Transport and Logistics, Lancaster University

INFORMS Annual Conference, 2018-11-04
Introduction
Airport congestion

- Outside of the US, demand at congested airports is managed through the IATA WSG.
- Airports are designated as *coordinated* and airlines must obtain slots to use an airport.
- A slot is a time interval during which an aircraft can use an airport infrastructure for the purposes of landing or take-off.
- A *coordinator* proposes an initial allocation for slots to airlines based on their requests.
Slot Allocation Problem: Key Features

- Slots at a coordinated airport are allocated for a six month season
- Requests are typically made in series:
  - A series of slots consists of 5 or more requests for the same time and (usually) the same day
  - Request series are mostly allocated slots the same time
- Allocation should satisfy airport capacity constraints
- Allocation should satisfy turnaround constraints
- Slots are allocated according to the following main priority groups
  - Historics
  - New entrants
  - Others
Schedule Displacement

- Capacity restrictions mean that not all requests can be allocated their preferred slot.
- The difference between an preferred and allocated time is called the *schedule displacement*.
- The allocation should in some way minimize the total *schedule displacement*. 
• The complexity of allocating slots under IATA scheme has lead to the development of integer programming models

• Main features of IATA scheme first modeled as an MIP in [Zografos et al., 2012] which minimized total displacement

• Incorporation of fairness:
  • Displacement proportional to requests [Zografos and Jiang, 2017]
  • Displacement proportional to peak-time requests [Fairbrother and Zografos, 2018]

• [Zografos et al., 2017] introduced objectives to improve acceptability:
  • Minimize maximum displacement
  • Maximize slots allocated to acceptable time windows

• [Ribeiro et al., 2018] proposed new multiobjective model with:
  • Multiple objectives: number of rejections maximum displacement, total displacement, number of displacement
  • Differentiation between “historical” and “change to historical” requests
Inflexibility and Flexibility

- A request series which coincides on at least one day may block another for all it’s requested days
- Toy example:
  - One historical request series (2 days)
  - One new entrant request series (4 days)
  - Both prefer slot at time period 2
  - Capacity of 1 slot per time period

<table>
<thead>
<tr>
<th>Day</th>
<th>Historical Request Series</th>
<th>New Entrant Request Series</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Historical request series is given priority and blocks new entrant
Inflexibility and Flexibility

- A request series which coincides on at least one day may block another for all it’s requested days
- Toy example:
  - One historical request series (2 days)
  - One new entrant request series (4 days)
  - Both prefer slot at time period 2
  - Capacity of 1 slot per time period

- Historical request series is given priority and blocks new entrant
- Allowing flexibility, new entrant series can be allocate slots for days not used by historic request series
Inflexibility and Flexibility

- A request series which coincides on at least one day may block another for all its requested days.

- Toy example:
  - One historical request series (2 days)
  - One new entrant request series (4 days)
  - Both prefer slot at time period 2
  - Capacity of 1 slot per time period

- Historical request series is given priority and blocks new entrant.
- Allowing flexibility, new entrant series can be allocate slots for days not used by historic request series.

- By allowing some flexibility in the slots allocated to a series, we can reduce significantly the overall displacement.
Aims

1. Develop a model which allows some flexibility in the slots allocated to a series
2. Incorporate demand-based fairness into new model
3. Study trade-off between flexibility, fairness and efficiency using real slot request data
Flexibility Model and Demand-based Fairness
Slot assignment variables

- Model is generalisation of that of [Zografos et al., 2012]
- Let:
  \[ M = \text{set of request (series)} \]
  \[ D = \text{set of days in scheduling season} \]
  \[ D_m \subseteq D = \text{set of days which make up request } m \]
  \[ T = \{1, \ldots, T\} = \text{set of time coordination intervals} \]

- Then, the main decision variables for our model are:
  \[ y_{dt}^m = \begin{cases} 
    1 & \text{if request } m \text{ is allocated slot at time } t \text{ on day } d, \\
    0 & \text{otherwise} 
  \end{cases} \]
Range Constraints

- It is necessary that the slots allocated to a request series are close to each other
- Introduce auxiliary variables:
  \[\tau_m = \text{earliest slot allocated to request series } m\]
  \[\bar{\tau}_m = \text{latest slot allocated to request series } m\]

- We require that each request series is assigned slots at times within a given range of each other:
  \[\bar{\tau}_m - \tau_m \leq r_m, \ m \in \mathcal{M}\]

  where \(r_m\) is a tolerance specified by the airline making the request \(m\)
Fairness - Peak Request Proportionality Principle

- For a schedule to be acceptable it is important that it distributes schedule displacement in a fair way to the airlines
- We define fairness based on the following principle [Fairbrother and Zografos, 2018]
  \[ \text{proportion of schedule displacement} \approx \text{proportion of peak requests} \]
- Solving base model may yield schedule which distributes schedule displacement disproportionately
Fairness indices and constraints

• For each airline, we define the following fairness index:

\[ \mu_a = \frac{\text{proportion of schedule displacement}}{\text{proportion of peak requests}} \]

• Fairness is then ensured through the addition to the base model of a constraint on the maximum deviation from absolute (MDA) fairness:

\[ \max_{a \in A} |\mu_a - 1| \leq \epsilon \]
Base Model

minimize \( \sum_{m \in M} |t - t_m| y_{td}^m \)

(subject to \( \sum_{t \in T} y_{td}^m = 1, \ m \in M, \ d \in D_m \))

\( \sum_{m \in M} \sum_{t \in T_c} a_m b_m c_{yd}^t \leq u_c^d, \ c \in C, \ d \in D, \ s \in T_c \)

\( l_{m_1 m_2} \leq \sum_{t \in T} t y_{td}^m - \sum_{t \in T} t y_{td}^t \leq \bar{l}_{m_1 m_2}, (m_1, m_2) \in P, \ d \in D_m \)

\( \bar{\tau}_m \leq \sum_{t \in T} t y_{td}^m \leq \bar{\tau}_m, \ m \in M, \ d \in D_m \)

\( \bar{\tau}_m - \bar{\tau}_m \leq r_m, \ m \in M \)

\( \max_{a \in A} |\mu_a - 1| \leq \epsilon \)

\( y_{td}^m \in \{0, 1\} \)

(minimize total displacement)

(assign slot to each request)

(rolling capacity constraints)

(turnaround time constraints)

(definition of \( \bar{\tau}_m, \bar{\tau}_m \))

(range constraints)

(fairness constraints)
### Notation

#### Sets

<table>
<thead>
<tr>
<th>Set</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{A} )</td>
<td>set of airlines</td>
</tr>
<tr>
<td>( \mathcal{D} (\mathcal{D}_m) )</td>
<td>set of days (for which movement ( m ) requested)</td>
</tr>
<tr>
<td>( \mathcal{M}(\mathcal{M}_a) )</td>
<td>set of movement requests (by airline ( a ))</td>
</tr>
<tr>
<td>( \mathcal{P} \subset \mathcal{M} \times \mathcal{M} )</td>
<td>set of arrival-departure pairs ((m_a, m_d))</td>
</tr>
<tr>
<td>( \mathcal{C} )</td>
<td>set of airport capacity constraints</td>
</tr>
<tr>
<td>( \mathcal{T} = {1, \ldots, T} )</td>
<td>set of coordination time intervals</td>
</tr>
</tbody>
</table>

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_m )</td>
<td>requested time for movement ( m )</td>
</tr>
<tr>
<td>( \delta_c )</td>
<td>duration of constraint ( c )</td>
</tr>
<tr>
<td>( l_p(l_p) )</td>
<td>minimum (maximum) turnaround time for movement pair ( p \in \mathcal{P} )</td>
</tr>
<tr>
<td>( a^d_m )</td>
<td>indicates whether constraint ( c \in \mathcal{C} ) is active on day ( d \in \mathcal{D} )</td>
</tr>
<tr>
<td>( b_{mc} )</td>
<td>contribution of movement ( m ) to constraint ( c )</td>
</tr>
<tr>
<td>( v^{ds}_c )</td>
<td>capacity for constraint ( c \in \mathcal{C} ) on day ( d \in \mathcal{D} ) at time period ( s )</td>
</tr>
<tr>
<td>( r_m )</td>
<td>maximum range of slot times for request ( m )</td>
</tr>
<tr>
<td>( \nu_m )</td>
<td>marginal cost of request ( m )</td>
</tr>
</tbody>
</table>

#### Decision variables

<table>
<thead>
<tr>
<th>Decision Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y_{td}^m )</td>
<td>indicates whether movement ( m ) is assigned slot ( t ) on day ( d )</td>
</tr>
<tr>
<td>( \tau_m(\overline{\tau}_m) )</td>
<td>earliest (latest) slot time assigned to ( m )</td>
</tr>
<tr>
<td>( \mu_a )</td>
<td>fairness index for airline ( a )</td>
</tr>
</tbody>
</table>
Solution Approaches
Problem Size

• The flexibility slot allocation model has $O(|T| \sum_{m \in M} |D_m|)$ variables as opposed to $O(|T||M|)$ variables in previous models [Zografos et al., 2012, Ribeiro et al., 2018]

• It also has many more turnaround constraints as these are now required for every day and every movement ($\sum_{m \in M} |D_m|$) rather than just every movement
The base model assumes that a request can be allocated a slot any time in the day.

In reality, slots will only be acceptable if they are close to the requested time.

We therefore enforce that each request series is allocated slots in the time window \([t_m - R, t_m + R]\).

\[ x = \text{preferred slot} \]

\[ = \text{Allowable Time Slot} \]

This reduces number of variables to \(O(2R \sum_{m \in M} |D_m|)\).

Suitable value for \(R\) can be found by solving inflexible problem.
The base model assumes that a request can be allocated a slot any time in the day. In reality, slots will only be acceptable if they are close to the requested time. We therefore enforce that each request series is allocated slots in the time window \([t_m - R, t_m + R]\).

\[ x = \text{preferred slot} \]

\[ \text{Shaded} = \text{Allowable Time Slot} \]

This reduces number of variables to \(O(2R \sum_{m \in M} |D_m|)\).

Suitable value for \(R\) can be found by solving inflexible problem.
Warmstart - Improvement Heuristic

- Warm-start can help speed-up solution significantly
- Heuristic takes solution to inflexible problem and improves solution greedily

```
input : Schedule
output: Improved schedule
Order requests by duration;
repeat
    for each request do
        Remove request from current schedule;
        Find minimum cost insertion of request into schedule;
        Reinsert request into schedule;
    end
until No improvement made;
```

- Minimum displacement insertions found by solving small ILP
Warmstart - Improvement Heuristic

- Warm-start can help speed-up solution significantly
- Heuristic takes solution to inflexible problem and improves solution greedily

**Input:** Schedule

**Output:** Improved schedule

Order requests by duration;

**repeat**

- **for each request do**
  - Remove request from current schedule;
  - Find minimum cost insertion of request into schedule;
  - Reinsert request into schedule;

**until** No improvement made;

- Minimum displacement insertions found by solving small ILP
Branch-and-cut

- For most time periods in the scheduling season, there is sufficient capacity to meet demand, and so most capacity constraints are inactive.
- We can speed up the solution of our model by adding capacity constraints as *lazy constraints*.
- This is achieved through a callback in the branch-and-bound algorithm.
Overview of Solution Approach

1. Impose time windows on slot allocations
2. Aggregate dates in preprocessing stage
3. Solve inflexible problem and run improvement heuristic to construct warm start solution
4. Run branch-and-bound with lazy capacity constraints
Numerical Tests
Problem Data

- Slot allocation problem solved using real request data for two medium-sized airports

Airport 1:

<table>
<thead>
<tr>
<th>Requests</th>
<th>Total movements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Historics</td>
<td>252</td>
</tr>
<tr>
<td>Changes to Historics</td>
<td>154</td>
</tr>
<tr>
<td>New Entrants</td>
<td>48</td>
</tr>
<tr>
<td>Others</td>
<td>444</td>
</tr>
<tr>
<td>Total</td>
<td>898</td>
</tr>
</tbody>
</table>

(a) Number of requests

<table>
<thead>
<tr>
<th>15 minutes</th>
<th>1 hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrivals</td>
<td>-</td>
</tr>
<tr>
<td>Departures</td>
<td>-</td>
</tr>
<tr>
<td>Total</td>
<td>3 (4)</td>
</tr>
</tbody>
</table>

(b) Rolling Capacity

Airport 2:

<table>
<thead>
<tr>
<th>Request</th>
<th>Total Movements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Historics</td>
<td>266</td>
</tr>
<tr>
<td>Changes to Historics</td>
<td>432</td>
</tr>
<tr>
<td>New Entrants</td>
<td>76</td>
</tr>
<tr>
<td>Others</td>
<td>1090</td>
</tr>
<tr>
<td>Total</td>
<td>1864</td>
</tr>
</tbody>
</table>

(a) Number of requests

<table>
<thead>
<tr>
<th>15 minutes</th>
<th>1 hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrivals</td>
<td>-</td>
</tr>
<tr>
<td>Departures</td>
<td>-</td>
</tr>
<tr>
<td>Total</td>
<td>5</td>
</tr>
</tbody>
</table>

(b) Rolling Capacity
Solution Approach

Hierarchical solve

- To take into account the different priority classes we solve the model *hierarchically*
- That is, problem is solved first for highest priority group, the airport capacities are updated, the problem is solved for the next highest priority group and so on.

Preprocessing

- 181 (out of 212) scheduling days after date aggregation for Airport 1
- 203 (out of 212) scheduling days after date aggregation for Airport 2

Parameters

- One coordination time interval corresponds to 15 minutes
- Problems solved using Gurobi 8.1 on Intel(R) Core(TM) i5-4690 CPU @ 3.50GHz
- For time windows we use $R = 3$ hours
Total Displacement - Airport 1

Historics

Changes to Historics

New Entrants

Others
Total Displacement - Airport 2

Changes to Historics

New Entrants

Others
Fairness - Airport 1

Historics

- Flexibility Range 0
- Flexibility Range 1

Schedule Displacement vs. MDA Fairness

Change to Historics

- Flexibility Range 0
- Flexibility Range 1

Schedule Displacement vs. MDA Fairness

New Entrants

- Flexibility Range 0
- Flexibility Range 1

Schedule Displacement vs. MDA Fairness

Others

- Flexibility Range 0
- Flexibility Range 1

Schedule Displacement vs. MDA Fairness
Conclusions

• We introduced a new model which incorporates efficiency, fairness and flexibility
• Flexibility in slots allocated to request series subject to range constraints
• Model tested using request data for medium-sized airports:
  • Flexibility significantly reduces displacement, but benefit attenuates for greater ranges
  • Flexibility also improves outcomes of fairness
Future Work

- Develop faster algorithms to allow solution for larger problems:
  - Benders decomposition by scheduling day (in progress)
  - Dantzig-Wolfe decomposition by request series
- Development of model which ensures greater schedule consistency


*Transportation Research Part C: Emerging Technologies, 21(1):244–256.*