Optimum Scheduling of Series of Slots with Scheduling Season Segmentation

Jamie Fairbrother\textsuperscript{1} \hspace{1em} Konstantinos Zografos\textsuperscript{1}

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

2nd IMA and OR Society Conference on Mathematics of Operational Research, Friday 26th April 2019
Introduction
Airport congestion

- Outside of the US, demand at congested airports is managed through the IATA Worldwide Slot Guidelines (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 requests for the same time and (usually) the same day over at least 5 weeks.
  - 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 led to the development of integer programming models.

Main features of IATA scheme first modeled as an ILP 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
Schedule blocking from series

- 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 entrants from slot on all days
- Scheduling in series thus leads to more displacement compared to scheduling requests independently
Aims

• Investigate ways of mitigating “blocking” for scheduling in series:
  1. Modifying threshold for weeks
  2. Segment scheduling season
Changing series threshold
Requests and scheduling in series

- Requests are made in arrival-departure pairs and take the following form:

<table>
<thead>
<tr>
<th>ID</th>
<th>Start Date</th>
<th>End Date</th>
<th>Arr. Time</th>
<th>Dep. Time</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
<th>Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4th March</td>
<td>1st April</td>
<td>12:15</td>
<td>13:15</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>B</td>
<td>5th March</td>
<td>16th April</td>
<td>09:35</td>
<td>10:45</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Table**: Simplified request examples

- Currently duration threshold for scheduling in series is 5 weeks
- Otherwise, each request of the series can be scheduled independently
- Request A has duration of 4 weeks: not scheduled in series
- Request B has duration of 6 weeks: scheduled in series
Changing the threshold

- A recent industry report [Europe, 2018] has suggested increasing the duration threshold for requests to be scheduled in series

![Histogram of length of requests (weeks) for a medium-sized airport](image)

**Figure:** Histogram of length of requests (weeks) for a medium-sized airport

- Fewer requests series relaxes the slot allocation problem and thus could lead to less displacement
Changing the threshold

- A recent industry report [Europe, 2018] has suggested increasing the duration threshold for requests to be scheduled in series.

Figure: Histogram of length of requests (weeks) for a medium-sized airport

- Fewer requests series relaxes the slot allocation problem and thus could lead to less displacement.
Changing the threshold

- A recent industry report [Europe, 2018] has suggested increasing the duration threshold for requests to be scheduled in series.

![Histogram of length of requests (weeks) for a medium-sized airport](image)

**Figure:** Histogram of length of requests (weeks) for a medium-sized airport

- Fewer requests series relaxes the slot allocation problem and thus could lead to less displacement.
Segmentation of scheduling season
Motivation - Scheduling with flexibility

- Historical request series is given priority and blocks new entrant.
Motivation - Scheduling with flexibility

- 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
Scheduling requirements

- Consider the arrival schedules for the following request series:

<table>
<thead>
<tr>
<th>ID</th>
<th>Start Date</th>
<th>End Date</th>
<th>Arr. Time</th>
<th>Dep. Time</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
<th>Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>4th March</td>
<td>8th August</td>
<td>14:45</td>
<td>16:00</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Week</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Week</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Week</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Week</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14:30</td>
<td>14:30</td>
<td>14:30</td>
<td>1</td>
<td>14:30</td>
<td>14:45</td>
<td>14:15</td>
<td>1</td>
<td>14:30</td>
<td>14:30</td>
<td>14:30</td>
<td>1</td>
<td>14:30</td>
<td>14:30</td>
<td>14:30</td>
</tr>
<tr>
<td>2</td>
<td>16:00</td>
<td>16:00</td>
<td>16:00</td>
<td>2</td>
<td>14:35</td>
<td>14:20</td>
<td>14:40</td>
<td>2</td>
<td>14:30</td>
<td>14:30</td>
<td>14:30</td>
<td>2</td>
<td>14:30</td>
<td>14:30</td>
<td>14:30</td>
</tr>
<tr>
<td>3</td>
<td>13:00</td>
<td>13:00</td>
<td>13:00</td>
<td>3</td>
<td>14:45</td>
<td>14:45</td>
<td>14:25</td>
<td>3</td>
<td>14:45</td>
<td>14:45</td>
<td>14:45</td>
<td>3</td>
<td>14:45</td>
<td>14:45</td>
<td>14:45</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Table:** Schedule 1  
Time slots vary too widely

**Table:** Schedule 2  
Time changes too often

**Table:** Schedule 3  
Ideal
Segmentation Scheduler - Idea

- We propose to segment the scheduling season
- Scheduling season consists of days $\mathcal{D} = [0, \ldots, D - 1]$
- Segmentation of season $\mathcal{S} = \{s_0, \ldots, s_{n-1}\}$ described by a vector $\Sigma = [d_0, d_1, \ldots, d_n]$ where $d_0 = 0$ and $d_n = D$.
- Each segment $s_i$ corresponds to half-open interval $[d_i, d_{i+1}) \subset \mathcal{D}$.

![Figure: Segmentation of scheduling season](image)

- On different segment, requests in a request series can be scheduled at different times.
Notation

• $\mathcal{T} = [0, \ldots, T - 1] = \text{set of time slots in each day}$
• $\mathcal{M} = \text{set of requests}$
• $\mathcal{D}_m = \text{set of days for request } m$
Main decision variables

- $S_m = \{s \in S : D_m \cap s \neq \emptyset\}$
- For each $m \in M$, $s \in S_m$ and $t \in T$ we define:

$$y_{st}^m = \begin{cases} 
1 & \text{if } m \text{ allocated slot on segment } s \text{ time } t \\ 
0 & \text{otherwise} 
\end{cases}$$
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:
  \[ \bar{\tau}_m - \tau_m \leq r_m, \quad m \in \mathcal{M} \]
  where \( r_m \) is a tolerance specified by the airline making the request \( m \)
Full Model

minimize \[ \sum_{m \in M} |t - t_m| y_{st}^m \]

subject to \[ \sum_{t \in T} y_{st}^m = 1, \ m \in M, \ s \in S_m \]

\[ \sum_{m \in M} \sum_{t \in T_c^s} a^d_mb^mcy_{st}^m \leq u^d_c, \ c \in C, \ d \in D, \ s \in T_c \]

\[ l_{m_1m_2} \leq \sum_{t \in T} t y_{st}^m_{m_2} - \sum_{t \in T} t y_{st}^m_{m_1} \leq \tau_{m_1m_2}, \ (m_1, m_2) \in P, \ s \in S_m \]

\[ \bar{\tau}_m \leq \sum_{t \in T} t y_{st}^m \leq \tau_m, \ m \in M, \ s \in S_m \]

\[ \tau_m - \bar{\tau}_m \leq r_m, \ m \in M \]

\[ y_{st}^m \in \{0, 1\} \]
Solution approaches
Problem Size

• The segmentation slot allocation model has $O(|T| \sum_{m \in M} |S_m|)$ 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 ($\sum_{m \in M} |S_m|$) rather than just every movement
Warmstart - Greedy Heuristic

- Warm-start can help speed-up solution significantly

**Input:** Requests

**Output:** Initial schedule

Order requests by duration;

```
for each request do
    Find minimum cost insertion of request into schedule;
    Insert request into schedule;
end
```

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

- Warm-start can help speed-up solution significantly

input: Requests
output: Initial schedule
Order requests by duration;
for each request do
    Find minimum cost insertion of request into schedule;
    Insert request into schedule;
end

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

- Warm-start can help speed-up solution significantly

```
input: Requests
output: Initial schedule
Order requests by duration;
for each request do
    Find minimum cost insertion of request into schedule;
    Insert request into schedule;
end
```

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

- Warm-start can help speed-up solution significantly

**Input:** Requests

**Output:** Initial schedule

Order requests by duration;

for each request do

    Find minimum cost insertion of request into schedule;
    Insert request into schedule;

end

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

- Warm-start can help speed-up solution significantly

**Input:** Requests  
**Output:** Initial schedule  
Order requests by duration;  
for each request do  
  Find minimum cost insertion of request into schedule;  
  Insert request into schedule;  
end

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

- Warm-start can help speed-up solution significantly

input : Requests
output: Initial schedule
Order requests by duration;
for each request do
  Find minimum cost insertion of request into schedule;
  Insert request into schedule;
end

- Minimum displacement insertions found by solving small ILP
Lazy cuts

- 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.
Numerical tests
Set-up

- Slot allocation problem solved using real request data for a medium-sized airport

<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>
<tr>
<td></td>
<td>5480</td>
</tr>
<tr>
<td></td>
<td>8554</td>
</tr>
<tr>
<td></td>
<td>2680</td>
</tr>
<tr>
<td></td>
<td>18000</td>
</tr>
<tr>
<td></td>
<td>34714</td>
</tr>
</tbody>
</table>

(a) Number of requests

<table>
<thead>
<tr>
<th></th>
<th>15 minutes</th>
<th>1 hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrivals</td>
<td>-</td>
<td>8</td>
</tr>
<tr>
<td>Departures</td>
<td>-</td>
<td>12</td>
</tr>
<tr>
<td>Total</td>
<td>5</td>
<td>20</td>
</tr>
</tbody>
</table>

(b) Rolling Capacity

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
- Disregard priority classes (non-compliant with IATA WSG) for simplicity
Results - Change to series threshold

Figure: Sensitivity of displacement to week threshold
Results - Segmentation scheduler

**Figure**: Scheduling in series penalty for segmentation scheduler
Results - Segmentation and change in threshold

Figure: Scheduling in series penalty for segmentation scheduler with split threshold of 9 weeks
Discussion

- Changing scheduling in series threshold has only small effect on displacement
- The segmentation scheduler can reduce penalty of scheduling series by around 50%
- Approaches are complementary: increasing series threshold on top of segmentation scheduler will result in a greater reduction in schedule displacement
Conclusions
Conclusions

• Scheduling series of requests can result in blocking which increases the total displacement of a schedule
• We propose and implement two approaches to mitigate against this:
  • Changing the threshold for scheduling requests in series
  • Segmentation schedule
• Segmentation schedule is more effective at reducing displacement, but approaches are complementary
Future Work

- Test approaches for scheduling with priority classes (hierarchical or lexicographic)
- Investigate different (non-equally sized) segmentations


