Heuristics to solve NP-Hard aircraft gate scheduling problem

Tugce Sahan, Ram Parikh

Problem Description

Multiple Objectives:

1. $\mathbf{P_m|p_j, r_j, d_j|\ \Sigma T_j}$ to minimize tardiness at gates for a set of flights whose ready time (release time), occupation time (processing time) and due date are known.

2. This is an NP-hard problem and hence heuristic approaches are involved to reach a close to optimal solution.

3. Among multiple solutions for the first objective, ties are broken based on the minimum value of total walking distances. That is the best solution gives maximum number of passengers to be in the gate which is closer to the exit.

Problem Description

Assumptions

- Walking distance is considered as between the gates and exit.
- All gates are identical.
- Only arriving flights are considered.
- Setup times between flights are not included.

Methods

Parameters:

n: number of flights

m: number of gates

a_i : start point of flight i's time window

b_i : end point of flight i's time window

d_i : gate occupation duration of flight i

assigned $gate_i$: holds the gates assigned for each flight

dis_j : distance from gate j to exit

p_i : number of passengers in flight i

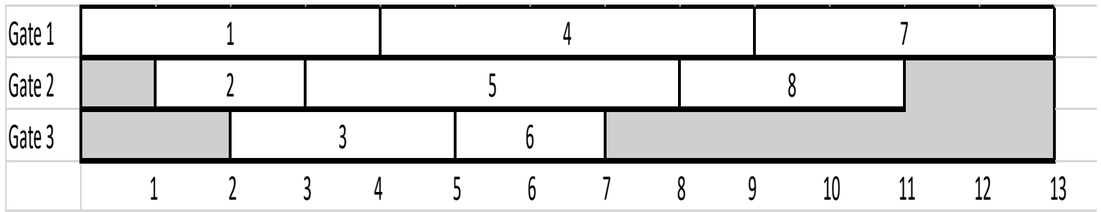
Algorithm 1: Earliest Release Date

For each gate k choose the flight with the earliest start time. Break ties according to the job with the earliest end time scheduled first. Make sure flight does not start before its release time.

Example:

flight	1	2	3	4	5	6	7	8
a	0	1	2	2	3	4	4	6
b	5	4	7	8	9	7	9	10
d	4	2	3	5	5	2	4	3

Solution:



Total Tardiness = 6

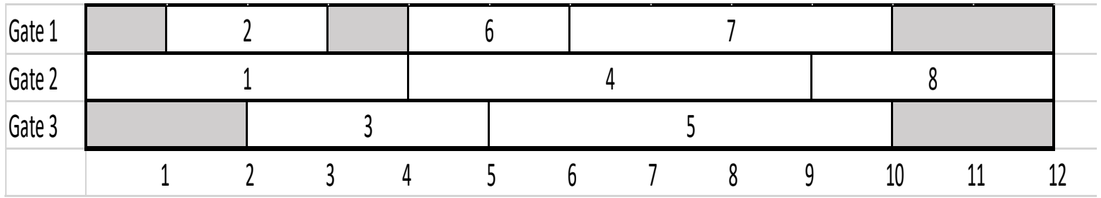
Algorithm 2: Earliest Due Date

For each gate k choose the flight with the earliest end time. Break ties according to the job with the earliest start time scheduled first. Make sure flight does not start before its release time.

Example:

flight	2	1	3	6	4	5	7	8
a	1	0	2	4	2	3	4	6
b	4	5	7	7	8	9	9	10
d	2	4	3	2	5	5	4	3

Solution:



Total Tardiness = 5

Algorithm 3: Forward Algorithm to choose earliest available gate with earliest starting flight

Step 1: Reorder flights such that $a_1 \leq a_2 \leq a_3 \ldots \leq a_i$. Break ties according to the job with the earliest end time b_i scheduled first.

Step 2: For each flight i choose the gate with the smallest elapsed time. Break ties randomly.

Step 3: Update the time elapsed at gate.

Step 4: Record the starting time of the flight accordingly:

starting time for each flight(i)= max(a(i),time at gate(assigned gate(i)));

Step 5: Record ending time of the flight accordingly:

ending time for each flight(i)= starting time for each flight(i) + gate occupation duration(i);

Step 6: Calculate tardiness:

max(0,ending time of the flight(i) - start time of the flight (i))

Algorithm 3: Forward Algorithm to choose earliest available gate with earliest starting flight

Example:

flight	1	2	3	4	5	6	7	8
a	0	1	2	2	3	4	4	6
b	5	4	7	8	9	7	9	10
d	4	2	3	5	5	2	4	3

Solution:

Total Tardiness = 3

Algorithm 4: Forward Algorithm to choose earliest available gate with earliest departing flight

Step 1: Reorder flights such that $b_1 \leq b_2 \leq b_3 \ldots \leq b_i$.

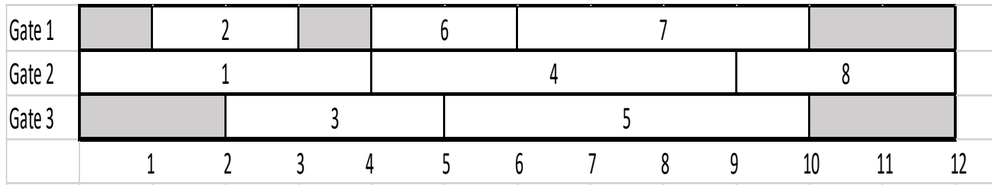
Break ties according to the job with the earliest start time a_i scheduled first.

Step 2, 3, 4, 5, 6 are same as Algorithm 3.

Example

flight	2	1	3	6	4	5	7	8
a	1	0	2	4	2	3	4	6
b	4	5	7	7	8	9	9	10
d	2	4	3	2	5	5	4	3

Solution:



Total Tardiness = 5

Algorithm 5: Backward Algorithm to choose latest departing flight to the end

Step 1: Reorder flights such that $b_1 \geq b_2 \geq b_3 \ldots \geq b_i$. Break ties according to the job with the latest start time a_i scheduled first.

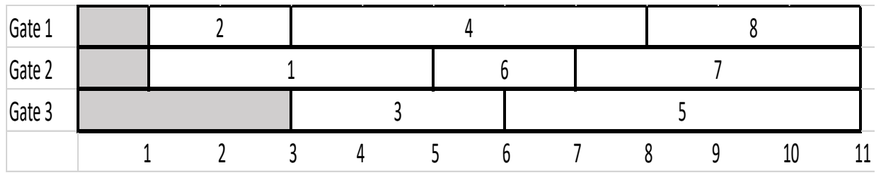
Step 2: Use C_{max} obtained from the forward algorithm to define starting point of each gate.

Step 3: For each flight i choose the gate with the maximum end time. Break ties randomly.

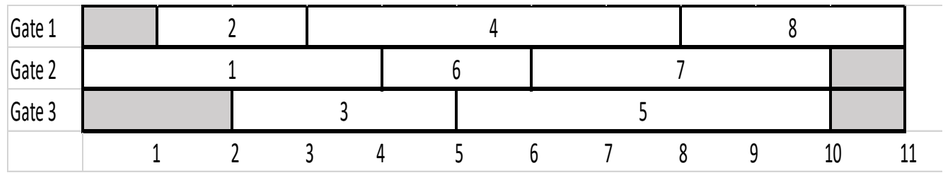
Example

flight	8	7	5	4	6	3	1	2
a	6	4	3	2	4	2	0	1
b	10	9	9	8	7	7	5	4
d	3	4	5	5	2	3	4	2

Solution:



Solution after shift:



Total Tardiness = 3

Objective 2: Minimum Total Walking Distance

- Each algorithm gives multiple solutions. All these solutions are then multiplied by the gate distances and number of passengers in each flight to find the best schedule with minimum tardiness and shortest total walking distance.

- Total weighted distance = $\sum_{i=1}^{n} \sum_{j=1}^{m} dis_j . p_i$

- Among solutions which gives minimum tardiness, the one which gives minimum total walking distance is chosen.

Algorithm Complexities

- Structure of each algorithm consists of 3 parts: Sorting, scheduling, calculating weighted distance.

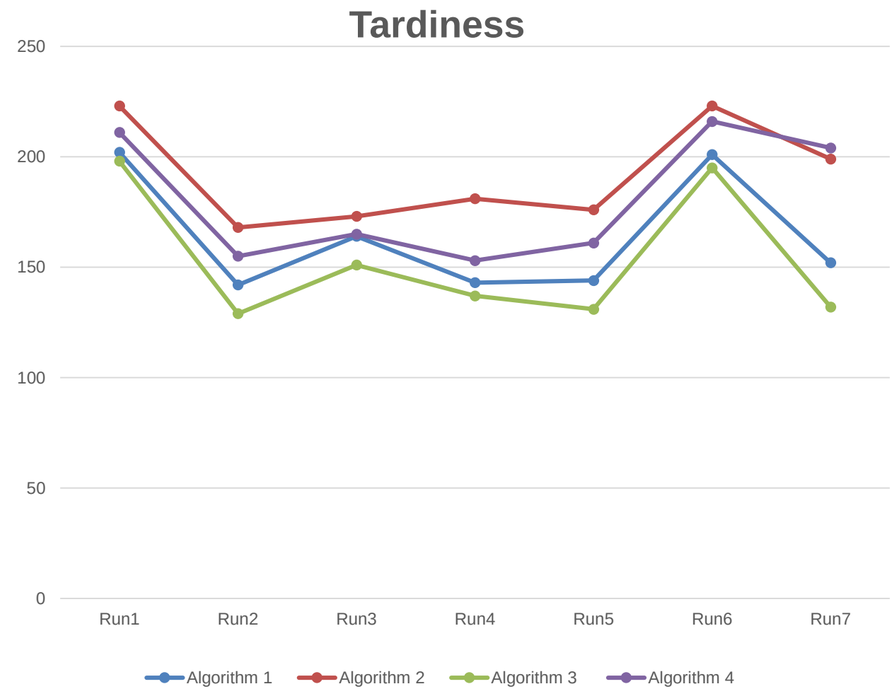
- n: number of flights, m: number of gates

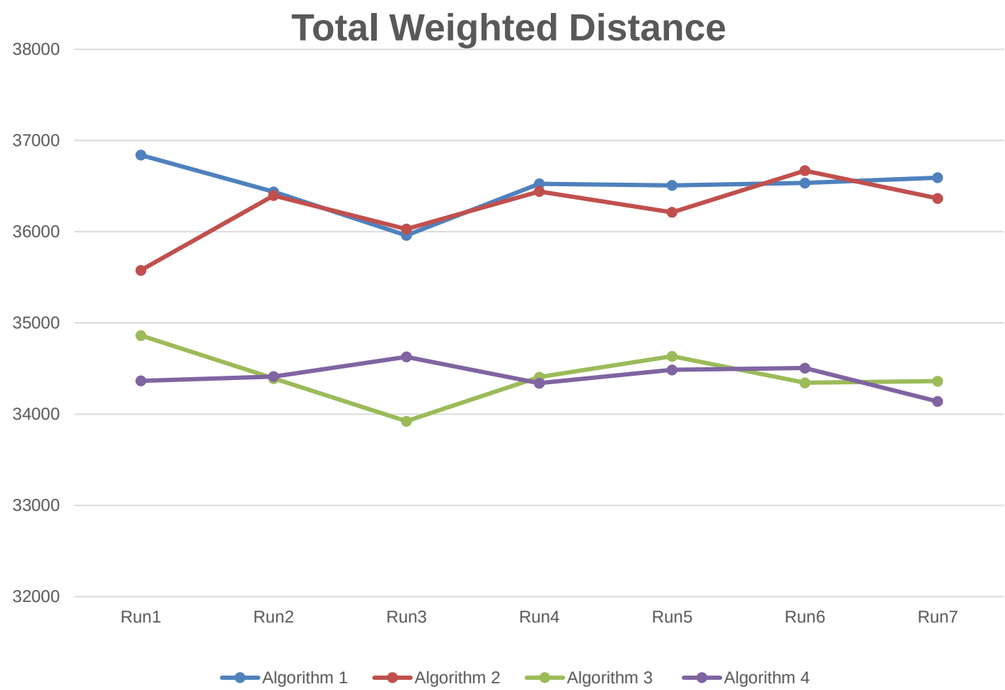
  $O(nlog(n)) + O(n.m) + O(n.m) = O(n.m)$

- Backward algorithm has additional parts: calculation of shift amount and updating start and end times which have the complexity of O(n) + O(n). Total complexity for the backward algorithm stays the same which is O(n.m).

Results

- The release times, due dates, occupations times, walking distances and number of passengers were randomly generated for 4 gates and 40 flights.
- 7 different test cases were run with 10,000 iterations to compare the efficiency of each algorithm.





Conclusion

- We can see that the Algorithm 3 performs best for all the runs to minimize the tardiness and the corresponding schedules are used to find the minimum walking distance.
- 7 different runs are performed to validate the performance of the various heuristics as the performance can tend to be data dependent in some cases.
- Objective can also be modified by giving weights to the 2 objectives rather than having a primary and secondary objective. **($\alpha \mid \beta \mid \theta_1\Upsilon_1 + \theta_2\Upsilon_2$ with $\theta_1 + \theta_2 = 1$)**

Thank you