Implementation of a Runway Scheduling Problem MILP solution

OPERATIONS OPTIMIZATION

AE4441-16

Authors:

Rami Sahtout (5210933)
Feitong Wang (6200303)
Piet Bouwens (5101875)

February 11, 2025

TU Delft Delft University of Technology

Contents

Introduction

Deciding the optimal sequence for landing airplanes at an airport is a complex task, and is commonly known as the "Runway scheduling problem" (RSP). Many approaches for solving this problem have been proposed, and a common way to solve this problem is through "Mixed-Integer Linear Programming" (MILP) [1]. MILP is a way of describing complex problems using a cost function, decision variables and constraints such that it can be solved efficiently using methods like the simplex method.

In this paper, an existing MILP is implemented and solved using Gurobi [2], to conduct a verification and validation analysis. The specific approach implemented in this paper is based on the study "Scheduling Aircraft Landings — The Static Case" [3]. This is one of the most cited papers pertaining to runway scheduling, and provides a flexible formulation of the RSP that has been improved over time. The approach used in this paper aims at optimizing the landing times of each aircraft such that airport operations proceed safely, while minimizing the deviation from the "target landing time". The model is applicable to both single- and multiple-runway operations.

Section 1 explains the RSP in more detail, as well as some of the considerations that need to be taken into account to create the MILP model. Section 2 shows how the mathematical model, including the cost function and constraints, is set up. On top of that, the Gurobi implementation is clarified. Section 3 goes through the verification and validation that was done to show how well this MILP formulation works. Finally, section 4 gives the conclusions about the model and the results.

1 Problem definition

The runway scheduling problem involves finding an optimal sequence of takeoffs and landings at an airport. Multiple safety restrictions and interactions between aircraft need to be taken into account, which give rise to constraints on the possible available solutions.

The optimality of a certain sequence of landings and takeoffs can be defined in multiple ways. The paper considered in this report, by Beasley et al. [3], considers the fact that aircraft have an ideal landing time, where they can stay at or around cruise speed, which is the most efficient speed, for the longest time. In this way, only the actual landing time has to be considered as a "decision variable" in the optimization problem. The deviation from the target landing time is related to fuel consumption (hence costs for the airline and increased environmental impacts) as well as delays for passengers.

Each landing aircraft has an earliest possible landing time (E_i), a latest possible landing time (L_i) and a target landing time (T_i). The actual landing time (x_i) should be as close as possible to T_i, while respecting proper clearances to other aircraft. Deviation from T_i results in a cost increase, as shown in figure 1. The cost function is designed to be piecewise linear, so that it can be used in the MILP definition.

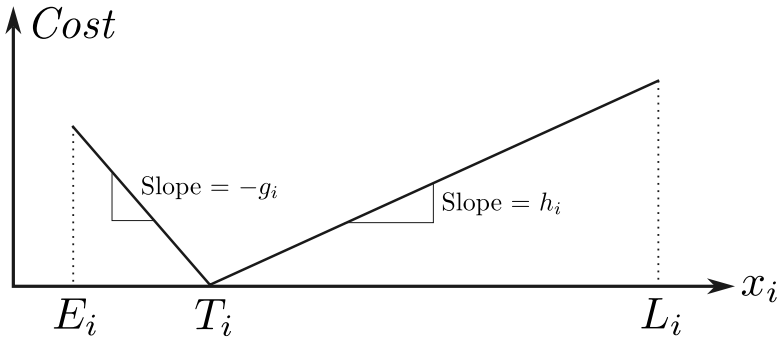


Figure 1: Cost for plane i, based on landing time x_i

There are also other possibilities for the cost function. For example, the airport / air traffic management might want to optimize the runway utilization. This way, the airport's capacity is used to its maximum, thus being more cost-efficient for the airport.

Another study found that trying to minimize the total delay of all flights normally leads to close-to-optimal runway utilization as well [4]. The other way around however, can lead to large delays. This means that the original cost function formulated by Beasley et al. is still very much applicable, and indeed it is used in many other studies as well.

Landing and departing aircraft are subject to certain "separation" requirements. These requirements are regulated by organizations like EASA (Europe) and the FAA (US), and are based on the class of the leading and following aircraft. EASA categorizes aircraft according to their weight [5], since this gives a good indication of the amount of effect that wake turbulence will have on the following aircraft. The requirements prescribe a minimum amount of time that must pass after an aircraft has landed / departed, before another aircraft can land / take off.

Large airports often use multiple runway at the same time, increasing the capacity of the airport. In the US, it is possible for 2 aircraft to land simultaneously on parallel runways, if they are far enough apart [6]. This is also true in europe [5], though there are more rules than in the US. Nevertheless, using more than 1 runway can significantly increase the possibilities for creating optimal arrival / departure schedules.

The optimization problem can be simplified by treating departing and landing aircraft the same, since both categories have a certain time that they are using the runway and require some separation because of safety and wake turbulence. This means only the runway usage time is relevant, regardless of whether the aircraft is landing or taking off. For the remainder of this paper, all aircraft will be referred to as "landing" for simplicity, though the term applies equally to departures.

It is not always possible for a plane to land at its desired target time, due to the separation requirements and other constraints. To construct constraints that can be used in the MILP definition, pairs of aircraft (i, j) are divided into 3 sets: U, V and W. These sets are defined as follows:

- U: Pairs of (i, j) for which it is possible that plane i lands before plane j or vice versa
- V: Pairs of (i, j) for which plane i lands before plane j, but there might not be enough time separation (S_i) between the two
- W: Pairs of (i, j) for which plane i lands before plane j and there definitely is enough time separation (S_i) between the two

Figure 2 shows an example of a pair (i, j) from each set.

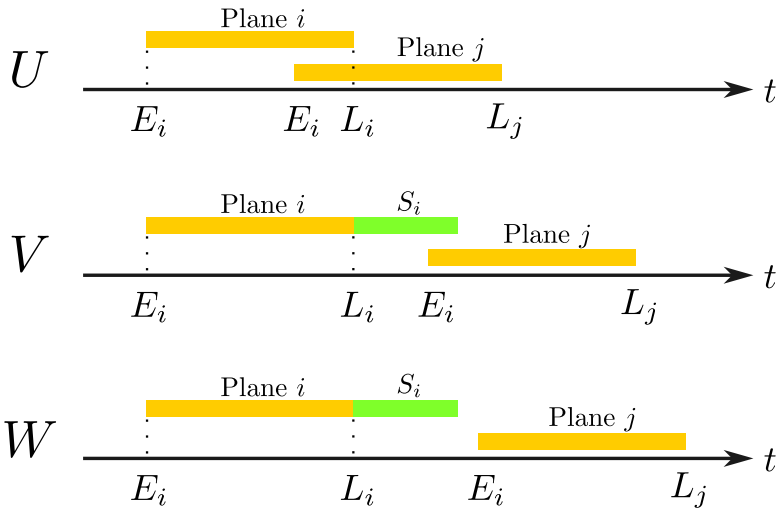


Figure 2: Categorization of plane pairs, based on possible landing sequences

2 Mixed linear integer programming model

This section describes how the MILP model is set up, such that it can be solved with linear programming optimization techniques.

2.1 Parameters

Table 1 provides an explanation for each parameter that will be used in the remainder of this section. Additionally, each parameter is explained in the text when it is first introduced.

Table 1: Overview of parameters

Parameter	Description
P	Set of all planes to be considered
R	Set of available runways
i, j	Index of plane, where $i, j \in P$
r	Index of runway, where $r \in R$
U	Set of pairs (i, j) where it's not certain if i lands before j
V	Set of pairs (i, j) where i lands before j, but there might not be enough separation
W	Set of pairs (i, j) where i lands before j and there is definitely enough separation
E_i	Earliest possible landing time for plane i
T_i	Target landing time for plane i
L_i	Latest possible landing time for plane i
S_{ij}	Required landing separation time between plane i and j
s_{ij}	Required separation time between planes i and j, which land on different runways
g_i	Cost per unit time for landing before target time of plane i
h_i	Cost per unit time for landing after target time of plane i
x_i	Actual landing time for plane i
α_i	How long plane i lands before T_i ($\alpha_i = T_i - x_i$)
β_i	How long plane i lands after T_i ($\beta_i = x_i - T_i$)
δ_{ij}	Equal to 1 if i lands before j, 0 otherwise
y_{ir}	Equal to 1 if i lands on runway r, 0 otherwise
z_{ij}	Equal to 1 if i and j land on same runway, 0 otherwise

2.2 Cost function and constraints

As explained in section 1, the cost for each individual airplane depends on how long it lands before / after the target time. The total cost for the MILP problem is simply the costs of each individual aircraft summed together. This results in the cost function seen in equation 1.

$$\text{Minimize} \quad \sum_{i=1}^{P} (g_i \alpha_i + h_i \beta_i) \tag{1}$$

Here, α_i is how soon plane i lands before T_i and β_i is how soon plane i lands after T_i. g_i and h_i determine the cost increase per time, as seen in figure 1.

Single runway

To make it more clear how the constraints are built up, first the single runway case is considered. Then, the constraints are extended to accommodate multiple runways.

For the cost function to work, α_i and β_i must not work against each other. The non-negativity constraints ensure that $\alpha_i, \beta_i \geq 0$. On top of that, equations 2 through 6 relate α_i, β_i, x_i and T_i with each other.

$$\alpha_i + x_i \geq T_i \qquad \forall\, i \in P \tag{2}$$
$$\alpha_i \leq T_i - E_i \qquad \forall\, i \in P \tag{3}$$
$$x_i - \beta_i \leq T_i \qquad \forall\, i \in P \tag{4}$$
$$\beta_i \leq L_i - T_i \qquad \forall\, i \in P \tag{5}$$
$$x_i + \alpha_i - \beta_i = T_i \qquad \forall\, i \in P \tag{6}$$

Constraints 2 through 5 ensure that α_i and β_i are bounded by the difference of T_i & x_i, and the earliest/ latest possible landing time. Equation 6 relates α and β to the actual landing time x_i. It's possible to see that if i lands before the target time $\alpha_i = T_i - x_i$ (since $\beta_i \geq 0 \ \rightarrow \beta_i = 0$), and similarly for β_i.

Constraints 7 and 8 simply state that the actual landing time of plane i (x_i) should be between the earliest and latest possible landing time.

$$x_i \geq E_i \qquad \forall\, i \in P \tag{7}$$
$$x_i \leq L_i \qquad \forall\, i \in P \tag{8}$$

The variables δ_{ij} indicate for each pair (i, j) if i lands before j ($\delta_{ij} = 1$) or vice versa ($\delta_{ij} = 0$). Since only one of δ_{ij} and δ_{ji} can be equal to 1, constraint 9 must be satisfied.

$$\delta_{ij} + \delta_{ji} = 1 \qquad \forall\, i, j \in P;\ j > i \tag{9}$$

Remembering the definitions of the sets U, V and W described in section 1 and figure 2, it is clear that for sets V and W that aircraft i lands before aircraft j, so $\delta_{ij} = 1$ (equation 10). Also for the planes in set V, the separation must be considered, such that the landing time of j is larger than the landing time of i plus the minimum required separation. This leads to constraint 11.

$$\delta_{ij} = 1 \qquad \forall\, (i, j) \in V \cup W \tag{10}$$
$$x_j - x_i \geq S_{ij} \qquad \forall\, (i, j) \in V \tag{11}$$

For the planes in U, a similar constraint to 11 is required. Here however, the constraint should become inactive when i lands *after* j, since that means the separation from j to i is now important (S_{ji}). Hence, the constraint looks like $x_j - x_i \geq S_{ij} - M\delta_{ji}$ for some big M. In other words, when j lands before i ($\delta_{ji} = 1$), the right hand side of the constraint becomes very negative and any values for x_i and x_j will satisfy the constraint.

For the optimization algorithm, called branch and bound, it is most efficient to keep M as small as possible since this causes the bounds to be tighter, hence reducing the number of branches that need to be explored. In the worst case, $x_i = L_i - E_j$. So using $M = (L_i + S_{ij} - E_j)$ and rewriting $S_{ij} - S_{ij}\delta_{ji} = S_{ij}\delta_{ij}$ gives equation 12.

$$x_j - x_i - S_{ij}\delta_{ij} + (L_i - E_j)\delta_{ji} \geq 0 \qquad \forall\, (i, j) \in U \tag{12}$$

Multiple runways

Equations 11 and 12 are only valid when a single runway is used. If multiple runways are used, these constraints change and some additional constraints are added. To do this, some new parameters are introduced. The set of all available runways is called R, with indices r (so $r \in R$). As discussed in section 1, it is possible to land on parallel runways with decreased separation times. The required separation time for two aircraft i and j landing on different runways is denoted by s_{ij}. If i and j land on the same runway $z_{ij} = 1$, otherwise $z_{ij} = 0$. Finally if plane i lands on runway r then $y_{ir} = 1$, otherwise it's 0. Using these parameters, three new constraints are created.

$$\sum_{r=1}^{\max R} y_{ir} = 1 \qquad \forall\, i \in P;\ \forall r \in R \tag{13}$$

$$z_{ij} - z_{ji} = 0 \qquad \forall\, i, j \in P;\ j > i \tag{14}$$

$$y_{ir} + y_{jr} - z_{ij} \leq 1 \qquad \forall\, i, j \in P;\ j > i;\ \forall r \in R \tag{15}$$

Equation 13 states that any plane i can only land on 1 runway, and equation 14 indicates that if i and j land on the same runway, so do j and i. Equation 15 relates y_{ir}, y_{jr} and z_{ij} by stating that if both plane i and j land on the same runway r ($y_{ir} = y_{jr} = 1$), then $z_{ij} = 1$.

Equation 11, which ensured proper separation for the planes in set V, has to be adjusted. If i and j land on different runways, the minimum separation should be s_{ij} rather than S_{ij}. This is done with equation 16.

$$x_j - x_i - S_{ij} z_{ij} - s_{ij}(1 - z_{ij}) \geq 0 \qquad \forall\, (i, j) \in V \tag{16}$$

The same is true for equation 12. When $x_i > x_j$ however (i lands after j), the big number M could now be insufficiently large. This means that to be safe, $M = (L_i + \max(S_{ij}, s_{ij}) - E_j)$. Combined, this gives equation 17.

$$x_j - x_i - S_{ij} z_{ij} - s_{ij}(1 - zij) + (L_i + \max(S_{ij}, s_{ij}) - E_j)\delta_{ji} \geq 0 \qquad \forall\, (i, j) \in U \tag{17}$$

So to conclude, the final set of constraints is given by equations 2 - 10, and 13 - 17.

2.3 FCFS and heuristic solution

To help with some optimizations explained in section 2.4, an upper bound for the cost is needed. This needs to be as low as possible, so it's beneficial to have a "smart guess" rather than a straight forward solution. The Beasley paper introduces a heuristic based on the "first come, first serve" algorithm. First come first serve applied to multiple runways can be described as follows:

1. Start with empty sets $A_r \forall r \in R$, which will hold the aircraft landing on runway r
2. Order the planes based on their target times T_i. For each plane i:
   (a) Calculate the earliest time they can land on each runway, B_r. This is based on T_i and the planes that already landed before plane i, which are stored in the sets A_r:

$$B_r = \max(T_i,\ \max(x_j + S_{ji} \mid \forall j \in A_r),\ \max(x_j + s_{ji} \mid \forall j \in A_u; \forall u \in R; u \neq r)) \tag{18}$$

   (b) Take the runway r with the lowest landing time B_r and land there. The corresponding runway is called γ. Now, $x_i = B_\gamma$ and index i is added to A_γ

Using this method, planes are landed at their desired target times or as soon as possible after T_i. Because of this, planes will always land at or after T_i, which might not be optimal. Therefore, Beasley introduces a 3rd step, to also consider landing before T_i. They take the solution provided by FCFS and don't change the order of landings or the used runways. This means all binary variables are pre-determined and the resulting problem is just a normal LP problem (rather than MILP). This can be quickly solved with the simplex method.

The new model uses the single runway formulation as described in section 2.2. Only the variables α, β and x are considered. The constraints consist of equation 2 - 8 and the new constraint 19.

$$x_j - x_i \geq S_{i,j} \qquad \forall\, i, j \in P;\ j > i \tag{19}$$

The result is a more optimal distribution of landing times per runway, and thus a decreased cost compared to FCFS.

2.4 Additional constraints

The above heuristic function can be used to simplify the MILP problem using "time window tightening". The cost of the heuristic solution counts as an upper bound for the MILP problem (Z_{ub}), since the optimal solution can only produce a lower (more optimal) cost. If a plane would land longer than Z_{ub}/g_i seconds before T_i, you'd exceed the upper bound of optimal solution. So you can disregard landing times earlier than $T_i - Z_{ub}/g_i$. The same reasoning can be made for Z_{ub}/h_i. This results in the new definitions for E and L given in equations 20 and 21. Doing this decreases the size of the U and V sets, giving less constraints.

$$E_i = \max(E_i, T_i - Z_{ub}/g_i) \qquad \forall\, i \in P \tag{20}$$

$$L_i = \min(L_i, T_i + Z_{ub}/h_i) \qquad \forall\, i \in P \tag{21}$$

The cost functions and constraints described in 2.2 are already sufficient to form a Mixed-Integer Program. In this section, however, several additional constraints are proposed. These additional constraints help to strengthen the relaxed zero-one variables δ_{ij}, y_{ir}, z_{ij} when in continuous form, which are redundant and automatically satisfied in zero-one space. The following is a list and explanation of these additional constraints.

$$\delta_{ij} \geq \frac{(x_j - x_i)}{(L_j - E_i)} \quad \forall (i,j) \in U, \tag{22}$$

$$\sum_{i=1}^{P} \sum_{j=1, j\neq i}^{P} \delta_{ij} = \frac{P(P-1)}{2}, \tag{23}$$

$$\delta_{ij} \geq 1 - \frac{\beta_i + \alpha_j}{T_j - T_i} \quad \forall (i,j) \in U \quad \text{with} \quad T_i < T_j, \tag{24}$$

$$\begin{aligned}(\alpha_i + \beta_i) + (\alpha_j + \beta_j) \geq\ & [S_{ij} - (T_j - T_i)]\delta_{ij} + [(T_j - T_i) + S_{ji}]\delta_{ji} \\ & - \max\{[S_{ij} - (T_j - T_i)], [(T_j - T_i) + S_{ji}]\}\,(1 - z_{ij}), \\ & \forall (i,j) \in U \quad \text{with} \quad T_i < T_j \quad \text{and} \quad (T_j - T_i) < S_{ij},\end{aligned} \tag{25}$$

$$U^* = \{(i,j) \mid (i,j) \in U \text{ and } E_j + S_{ji} > L_i\}, \tag{26}$$

$$\delta_{ij} + z_{ij} \leq 1 \quad \forall (i,j) \in U^*, \tag{27}$$

$$U^{**} = \{(i,j) \mid (i,j) \in U \text{ and } E_j + s_{ji} > L_i\}, \tag{28}$$

$$\delta_{ji} + (1 - z_{ij}) \leq 1 \quad \forall (i,j) \in U^{**}, \tag{29}$$

$$\sum_{i=1}^{P} \sum_{j=1, j\neq i}^{P} z_{ij} \geq K, \tag{30}$$

Equation 22 imposes a constraint on δ_{ij} when plane j land after plane i. Equation 23 provides the theoretical sum of all δ_{ij}. Equation 24 determines the order of i and j by considering whether the target time gap $T_j - T_i$ can be closed. Similarly, Equation 25 states that if two planes are to land on the same runway, i.e. $z_{ij} = 1$, there must be enough adjustable time to rearrange in order to meet the separation requirements. Equation 26 through Equation 29 provide additional constraints on the order of two planes in set U, by considering the time windows and the separation constraints. Finally, Equation 30 established a lower bound, K, on the number of plane pairs landing on same runway, which guarantees that the maximum capacity of each runway is utilized. K here is the minimum value of

$$\sum_{r=1}^{R} (m_r)^2 - P \tag{31}$$

where m_r is the number of planes landing on runway r. In section 3.4, the effects of additional constraints are discussed.

2.5 Setup in Gurobi

Using the python library `gurobipy` developed by Gurobi[2], one can implement various types of variables, constraints, and objective functions into the model and utilize the Gurobi optimizer to solve the problem effectively. As an example, the following code snippets explain how the variables in single runway situation are added:

```
x[i] = model.addVar(lb=E[i], ub=L[i], vtype=gp.GRB.CONTINUOUS,
name=f"landing time for plane{i}")

alpha[i] = model.addVar(lb=0, vtype=gp.GRB.CONTINUOUS,
name=f"how soon plane {i} land before T_{i}")

beta[i] = model.addVar(lb=0, vtype=gp.GRB.CONTINUOUS,
name=f"how soon plane {i} land after T_{i}")

delta[i,j] = model.addVar(vtype=gp.GRB.BINARY,
name=f"plane{i} land before plane{j}")
```

Here, `gp.GRB.CONTINUOUS` and `gp.GRB.BINARY` specify the type of variables as continuous or binary (0/1), respectively. The constraints regarding the upper and lower bounds of the landing time are also incorporated when adding each variable `x[i]`.

Considering constraints, three distinct sets, `U`, `V`, and `W`, are defined, and constraints related to these sets are added in the following steps:

```
# W-set: certain of sequence, separation constraint is satisfied
if L[i]<E[j] and (L[i]+S[i,j]) <= E[j]:
    W.append((i,j))

# V-set: certain of sequence, but need to check separation time
elif L[i]<E[j] and (L[i]+S[i,j]) > E[j]:
    V.append((i,j))

# U-set: uncertain of sequence
elif E[j]<=E[i]<=L[j] or E[j]<=L[i]<=L[j]
    or E[i]<=E[j]<=L[i] or E[i]<=L[j]<=L[i]:
    U.append((i,j))

# those order can already be determined
for element in set(W) | set(V):
    i,j = element
    model.addConstr(delta[(i,j)] == 1, name = f"plane{i} must lands before plane{j}")
for element in V:
    i,j = element
    model.addConstr(x[j] - x[i] - S[(i,j)] >= 0,
    name=f"plane{i} lands before plane{j}, need to check separation")

# those order need to be checked
for element in U:
    i,j = element
    model.addConstr(x[j] - x[i] - S[(i,j)]*delta[(i,j)]
    + (L[i] - E[j])*delta[(j,i)] >= 0,
    name=f"separation coonstraint for plane{i} and plane{j} in U")
```

Other constraints are straightforward:

```
model.addConstr(alpha[i] + x[i] - T[i] >= 0,
    name=f"put limit on alpha{i}")

model.addConstr(alpha[i] + E[i] - T[i] <= 0,
    name=f"put limit on alpha{i}")

model.addConstr(beta[i] - x[i] + T[i] >= 0,
    name=f"put limit on beta{i}")

model.addConstr(beta[i] - L[i] + T[i] <= 0,
    name=f"put limit on beta{i}")
```

```
model.addConstr(x[i] + alpha[i] -beta[i] - T[i] == 0,
    name=f"relationship between alpha, beta, x, T for plane{i}" )
```

Finally, the objective function is:

```
obj = gp.LinExpr()

obj += g[i]*alpha[i] + h[i]*beta[i]

model.setObjective(obj, gp.GRB.MINIMIZE)
```

Here, the problem type is specified as minimization using `gp.GRB.MINIMIZE` and the objective is set to be linear expression using `gp.LinExpr()`.

3 Verification and validation

3.1 Verification

In this section, some test data are generated to verify the model implemented in Gurobi. Start with simple case, the data are created as in Table 2.

Table 2: Test data 1

plane	Target landing time	Earliest landing time	Latest landing time	Separation time
0	5	10	15	20
1	20	25	30	20

The penalty cost per unit of time for landing before the target time for each plane is set to 1, while the penalty cost for landing after the target time is set to 2. Given that the latest landing time for Plane 0 is earlier than the earliest landing time for Plane 1, Plane 0 must land before Plane 1 to satisfy the scheduling constraints.

Next, considering the separation constraint, the difference between the target landing times of these two planes is 15, which is smaller than the required separation time of 20. Therefore, the landing times must be rescheduled. Since the unit penalty cost for Plane 0 to land earlier than its target time is 1, and the unit penalty cost for Plane 1 to land later than its target time is 2, it is preferable to adjust Plane 0's landing time. However, Plane 1's earliest allowable landing time imposes a limit on how far Plane 0's time can be moved forward.

In this scenario, Plane 0's landing time can be shifted forward by up to 5 units, resulting in a new landing time of 5. With Plane 1's landing time remaining unchanged, this adjustment satisfies the separation requirement while minimizing the total penalty cost.

The model results are provided below and align with the discussion above. Only variables with non-zero values are shown for clarity.

```
Penalty cost: 5.0
landing time for plane0: 5.0
how soon plane 0 lands before T_0: 5.0
plane0 lands before plane1: 1.0
plane0 lands on runway0: 1.0
landing time for plane1: 25.0
plane1 lands on runway0: 1.0
```

Next, consider another example as Table 3 suggests.

Table 3: Test data 2

plane	Target landing time	Earliest landing time	Latest landing time	Separation time
0	20	10	30	20
1	15	5	25	20

In the case of a single runway, since the latest landing time of plane 1 falls within the range of plane 0's earliest and latest landing times, the landing order of these two planes cannot be determined yet, i.e., this situation belongs to set U. Further analysis reveals that to satisfy the separation time requirement and minimize the penalty cost, plane 1 should land as soon as possible, since the penalty for landing before the target time is lower. Meanwhile, plane 0 should land after the target time to meet the separation time requirement. The model's outputs, as shown below, are consistent with this reasoning.

```
Penalty cost: 20.0
landing time for plane0: 25.0
```

```
how soon plane 0 lands after T_0: 5.0
landing time for plane1: 5.0
how soon plane 1 lands before T_1: 10.0
plane1 lands before plane0: 1.0
```

In the case of two runways, the cost should obviously be zero, as each plane lands at its scheduled time. This is because each aircraft can now land separately on different runways, eliminating the need to adjust the target landing times with respect to the separation principle. The results are as follows.

```
Penalty cost: 0.0
landing time for plane0: 20.0
plane0 lands on runway0: 1.0
landing time for plane1: 15.0
plane1 lands before plane0: 1.0
plane1 lands on runway1: 1.0
```

To further validate, the model is ran on Beasley's dataset and results are compared with results shown in Beasley's paper in next section.

3.2 Validation

In order to validate that the implementation in Gurobi (see section 2.5) actually is the same as the one from Beasley (which used CPLEX [7]), and that the results are realistic w.r.t. real life, results were compared for a number of test cases. To do this, 2 datasets were used. The first one was used by the original Beasley paper and contains different scenarios ranging from 10 to 500 aircraft[1]. The second one uses real data from Paris Charles de Gaulle airport, constructed by the authors of [1][2]. The "airlandX" datasets are from Beasley, while the "alp_X_XX" datasets are from Charles de Gaulle. It is important to note that the time units for the "airlandX" datasets are undefined and the data will just be considered to be in "[time units]". A visualization of an input data file is shown in figure 3 below.

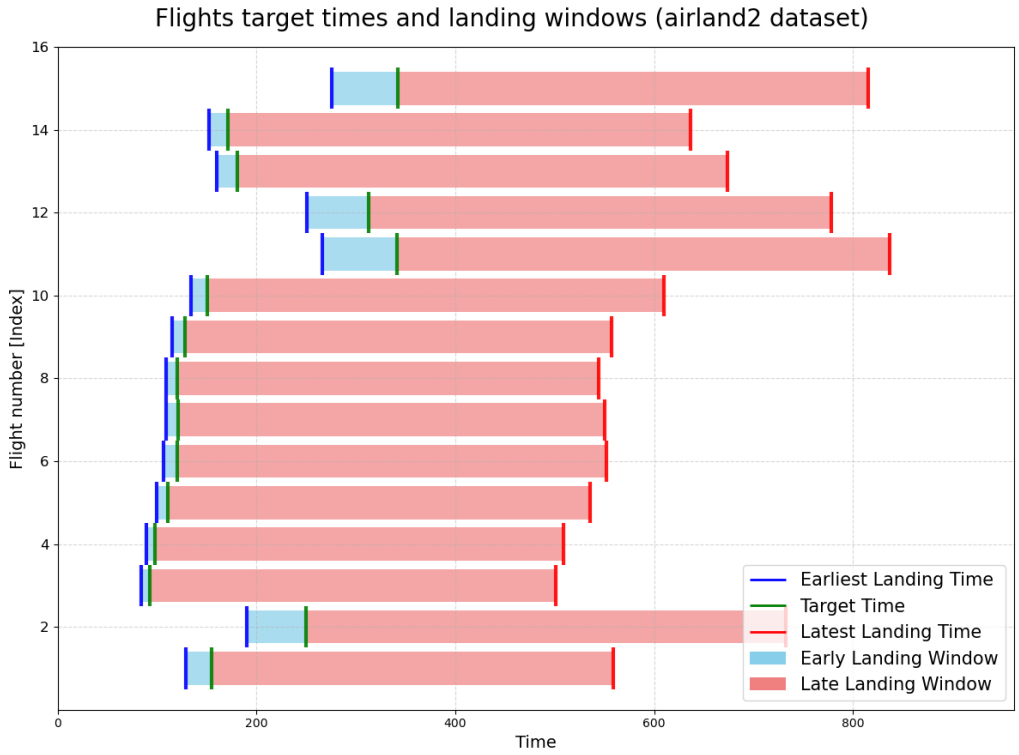


Figure 3: Example input data from the Beasley dataset. This specific file is Airland2 with data for 20 flights. The data includes the earliest and latest possible landing time and the target time

Tests were run for 1, 2 and 3 available runways. A time limit of 25 minutes was applied to find the solution. The results are only known for the first 8 datasets of Beasley, for the other datasets there is no available information about the cost. Table 4 shows the results for 1, 2 and 3 runways.

[1] https://people.brunel.ac.uk/~mastjjb/jeb/orlib/airlandinfo.html
[2] https://data.recherche.enac.fr/ikli-alp/

Table 4: Results of select datasets

Dataset	Time taken [s]	Number of planes	Number of runways	FCFS cost	Heuristic cost	Optimal cost	Same as literature
airland1	0.03	10	1	1210	700	700	✓
	0.01		2	120	90	90	✓
	0		3	0	0	0	✓
airland2	0.08	15	1	2030	1500	1480	✓
	0.03		2	210	210	210	✓
	0		3	0	0	0	✓
airland3	0.07	20	1	2870	1730	820	✓
	0.03		2	60	60	60	✓
	0.01		3	0	0	0	✓
airland4	0.62	20	1	4480	2520	2520	✓
	0.24		2	680	640	640	✓
	0.04		3	130	130	130	✓
airland5	2.89	20	1	7120	5420	3100	✓
	0.45		2	1640	1190	650	✓
	0.12		3	240	240	170	✓
airland6	0	30	1	24442	24442	24442	✓
	0.05		2	1034	888	554	✓
	0.01		3	0	0	0	✓
airland7	0.04	44	1	3974	1550	1550	✓
	0.01		2	0	0	0	✓
	0.01		3	0	0	0	✓
airland8	0.28	50	1	4390	2480	1950	✓
	0.16		2	260	135	135	✓
	0.07		3	0	0	0	✓
airland9	$>$ 1500	100	1	14265.9	7310.2	5611.7*	Not available
	0.72		2	617.1	545.5	444.1	Not available
	0.2		3	89	75.75	75.75	Not available
alp_7_30	1029	30	1	50065	43087	17684.2	Not available
	7.89		2	4793.9	4134.4	3036.7	Not available
	1.06		3	2020	1192	740.1	Not available

All results were the same as the values in the original paper, meaning the MILP was correctly implemented in Gurobi. The results very nicely show that FCFS cost $\geq$ heuristic cost $\geq$ optimal cost. This indicates that the full formulation of the MILP problem and solving it to optimality can significantly improve the quality of the solution compared to heuristic solutions.

Since calculating the actual result for airland9 (single runway) took too long, the value in the table is just the value of the incumbent when the solver was stopped.

To ensure that the resulting schedules are indeed feasible, visualizations of the output of the model were made. The output for airland2 is shown in figure 4 for 1, 2 and 3 runways. The separation time is marked between the

actual landing times of two aircraft. The rest of the visualizations can be found in appendix A.

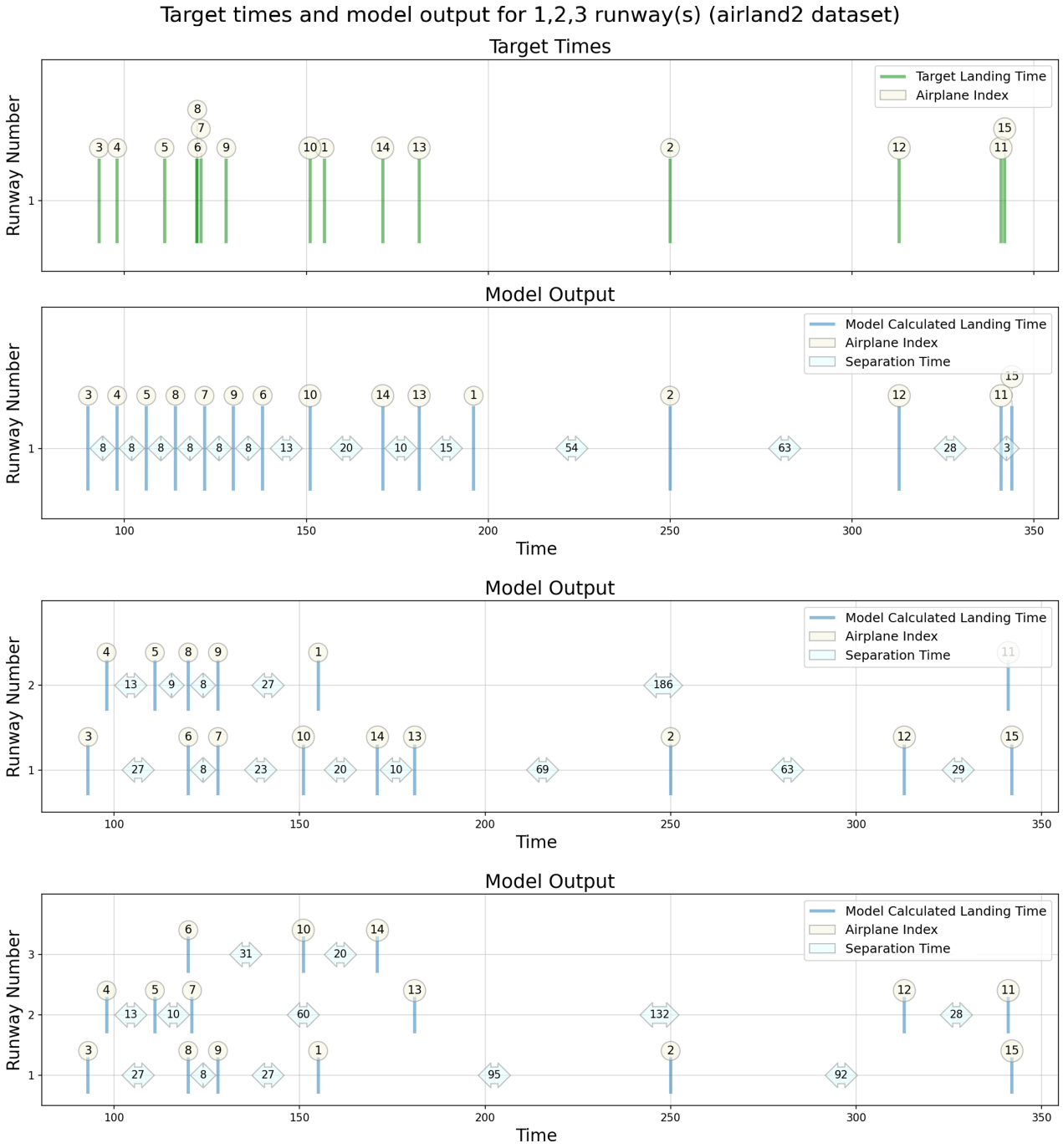
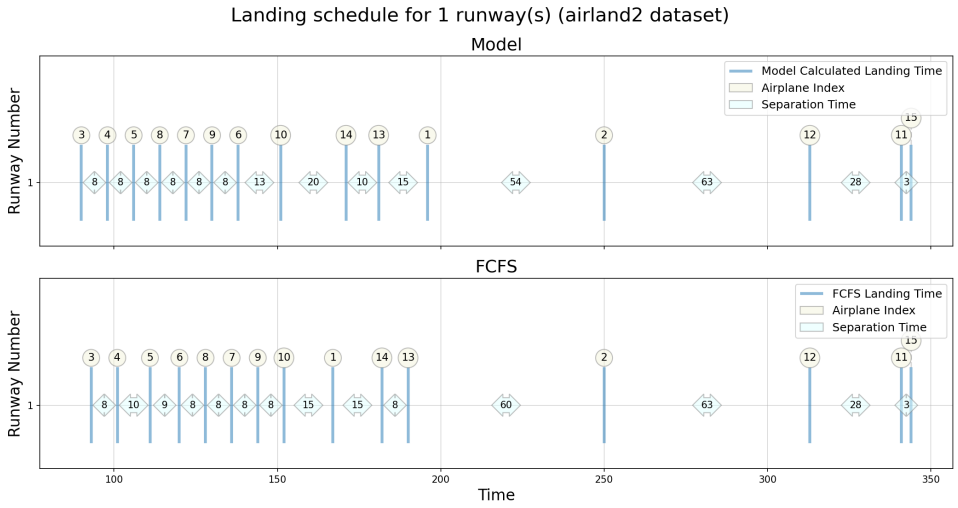
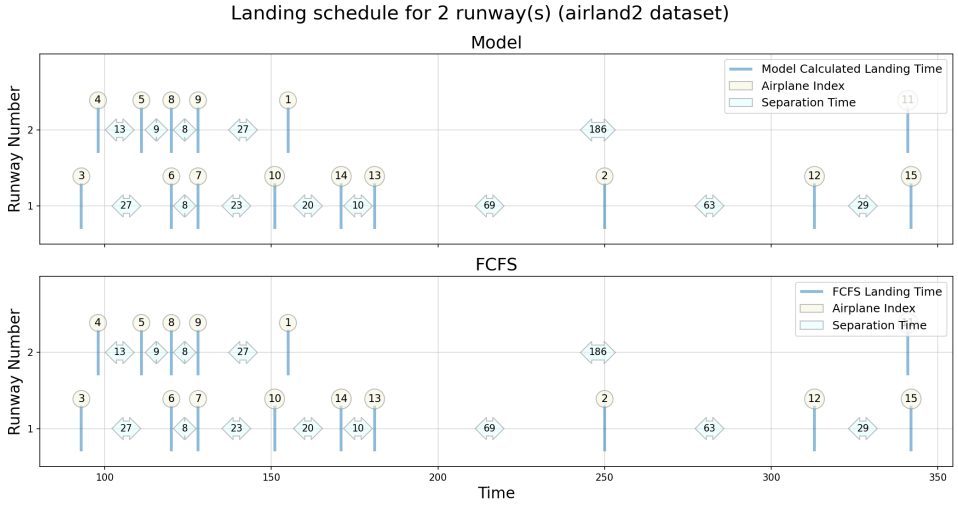


Figure 4: Visualization of the model output for 1, 2 and 3 runway(s). Input data is the Beasley airland2 dataset with 20 flights. It is clear that using more runways, more flights land on the target time.

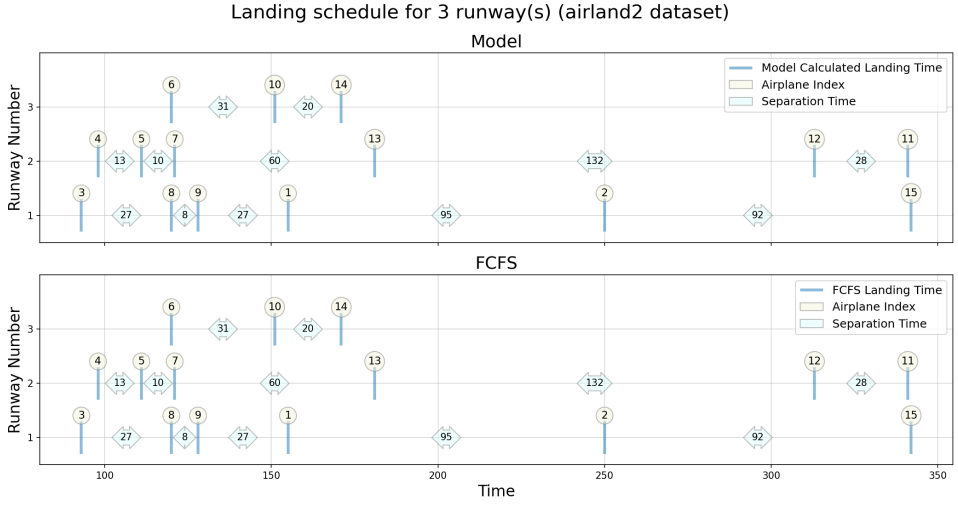
It is possible to compare the output of the model to just using the FCFS algorithm (Figure 5). From the figure it can be seen that the implementation of the Beasley model differs more from the FCFS output, especially when less runways are available. Which makes sense, as the more runways there are, the less "careful" you have to be with our resources (time).



(a) Comparison using 1 runway



(b) Comparison using 2 runways



(c) Comparison using 3 runways

Figure 5: Comparison between output schedule by the FCFS algorithm (bottom) and the output schedule of our implementation (top)

3.3 Sensitivity analysis

To analyse how the MILP solution responds to changes in parameters, a sensitivity analysis was performed. Here, all of the various input parameters were varied to see how the solution would change. Figure 6 shows the results for parameters g, h, E, L and S. The base problem here is airland5, one of the datasets studied in section 3.2. Each parameter was varied from there original value by a certain percentage. Since each individual aircraft has different values for g, h, etc. all variables were varied at the same time. So a 5% increase of g would mean g_i for every plane i is increased by 5%.

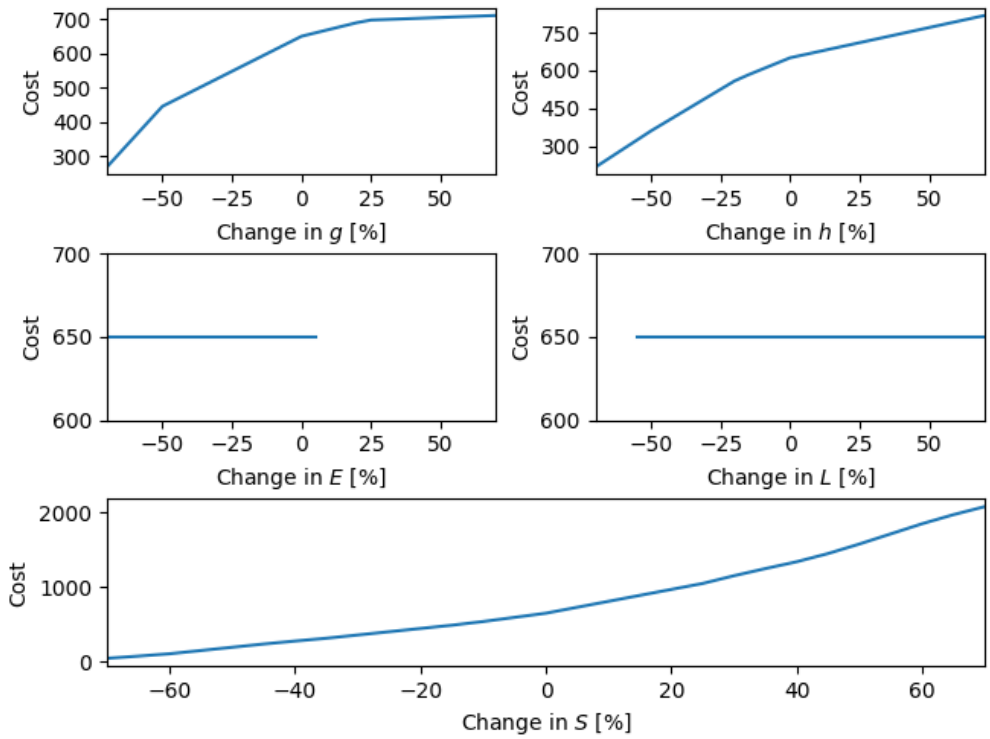


Figure 6: Change of cost for different parameter values

The graphs show some interesting results. As expected, lowering the "cost slopes" g and h decreases the total cost. This is shown by the first linear part of the graphs ($-50\% \leq \Delta g \leq 0\%$ and $-20\% \leq \Delta h \leq 0\%$). After that, the graph hits an inflection point where a totally new solution / landing order is more optimal. This is the second linear part ($-70\% \leq \Delta g \leq -50\%$ and $-70\% \leq \Delta h \leq -20\%$). In other words, the cost decreases at first because it's simply scaled to be lower, then decreases because a new solution is found.

Increasing g and h shows similar behaviour but in reverse. However, the increase in cost levels out because it becomes so important to stay close to the target times that only a couple of are forced to land long before / after their specific target time. This means the cost does not increase much.

The earliest and latest landing times E and L show no effect on the cost, only on the feasibility of the problem. Values for which there is no feasible solution are indicated by the "gaps" where no data is present. Clearly, increasing the earliest landing time or decreasing the latest landing time makes the landing windows smaller, explaining why the problem eventually becomes infeasible.

Finally, decreasing the separation time S leads to more optimal solutions (lower cost). This is because aircraft can simply land closer to their desired target time. Increasing S first leads to a linear increase in cost because aircraft must land further from their target time. Then, at the inflection point $S = 45\%$, the current solution becomes infeasible (with the current order and separation time, it is impossible to land all planes within their time windows). At this point another (feasible) landing order becomes optimal, which is seen for $S \geq 45\%$.

Another input variable is the number of runways. The cost for airland 1 through 8 for different runway amounts is plotted in figure 7.

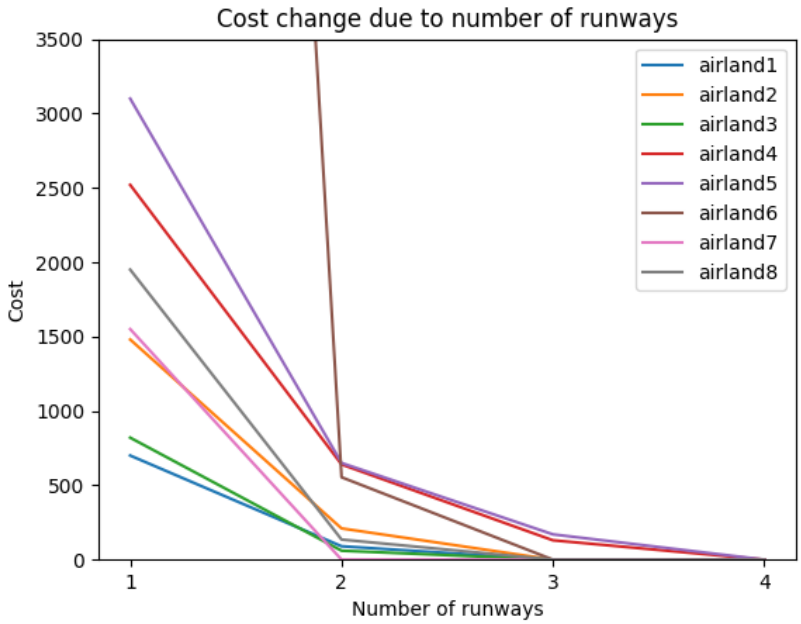


Figure 7: Change of cost for different amount of runways

This plot clearly shows that in every case, the cost will decrease if more runways are available until the cost reaches 0. This happens when all planes can land exactly at their target time. More runways being available means that the separation constraints for aircraft landing on the same runway can be avoided, therefore decreasing the cost.

Generally speaking, the absolute slope between runway numbers and costs increases as the number of aircraft and the cost per single runway increase. In airland1, where there are 10 aircraft and the cost for a single runway is 700, the slope is -610 unit cost per runway. In contrast, in airland5, with 20 aircraft and a single runway cost of 3100, the slope is -2450 unit cost per runway.

Another notable feature regarding the slope is that the absolute value of the slope decreases as the number of runways increases, which aligns with the principle of diminishing marginal returns.

This relationship implies that costs can be reduced most significantly in airports with busy landing schedules, with the greatest cost-saving effect observed when adding the first new runway. Therefore, managers must carefully weigh the benefits of reduced aircraft scheduling costs against the investment required to construct additional runways.

3.4 Computational efficiency

Table 4 shows the time taken to calculate each result. The Beasley paper only tested relatively small problems, with up to 50 aircraft. This is because they used a regular branch and bound method. To improve the performance, they introduced the additional constraints shown in section 2.4. These are aimed at reducing the amount of nodes that need to be evaluated, thus decreasing the time to solve the problem.

The times in table 4 are without use of the additional constraints. When they were implemented, they were shown to either have no effect or increase the solver time. This is likely due to the optimization method Gurobi uses. The basis of this is a branch-and-cut method, which is more advanced than branch-and-bound. On top of that, Gurobi uses many (proprietary) tricks and innovations to improve efficiency. This includes pre-processing steps, multi core processing, prioritizing decreasing the incumbent more quickly over just exploring all nodes, and other advanced MILP algorithms. All of this means that the additional constraints are unnecessary since they are already effectively implemented (but better) within the Gurobi solver. In fact, they often increase the amount of nodes that the solver has to go through until it is certain that the solution is optimal.

The only change to the original MILP problem that was shown to help was the use of time window tightening. For example, one of the more difficult cases (single runway alp_7_30) took 1029 seconds normally but only

830 seconds after window tightening.

Although it can take a long time to completely optimize the problem, the incumbent converges to the optimal value quite quickly. For alp_7_30, the optimal value is already reached after 100 seconds, and an almost optimal value (17684.1200 instead of 17684.1199) after only 2 seconds. So although it might look impossible to extrapolate this method to larger datasets or use it for real time operations, cutting off the optimization after a short time likely already gives a very optimal solution.

4 Conclusion

Despite being released in 2000, the RSP formulation by Beasley et al. is still very influential and relevant today. The MILP formulation is still used as a basis for many other studies, as seen in [1]. The cost function and constraints as seen in section 2 are applicable to both landing and departing aircraft, on single- or multiple-runway airport layouts.

After implementing the MILP in Gurobi, the model was verified for simple cases in section 3.1 and validated for more realistic / complex cases in section 3.2. Evaluating the costs of each solution and visualizing the resulting schedules shows that the results are indeed feasible and optimal. Also, it shows that there are large benefits to solving the full MILP problem compared to FCFS or heuristic approaches.

Additional constraints, which do not influence the results but are aimed at improving solver times were shown not to work due to differences in the optimizer algorithm ("branch and bound" compared to custom "branch and cut" based algorithm in Gurobi), except for the inclusion of Beasley's heuristic and time-window tightening (section 3.4).

The visualizations also show that the chosen cost function, aimed at decreasing deviation from a target landing time, will simultaneously lead to a solution that optimizes runway capacity. This shows that the approach is not only beneficial for airlines (decreased fuel costs, improved customer satisfaction) but also for the airport (more runway utilization means more income, less need for expansion).

References

[1] Sana Ikli et al. “The aircraft runway scheduling problem: A survey.” In: *Computers & Operations Research* 132 (2021), p. 105336. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2021.105336. URL: https://www.sciencedirect.com/science/article/pii/S0305054821001192.

[2] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2024. URL: https://www.gurobi.com.

[3] John Beasley et al. “Scheduling Aircraft Landings—The Static Case.” In: *Transportation Science* 34 (May 2000), pp. 180–197. DOI: 10.1287/trsc.34.2.180.12302.

[4] Waqar Malik and Yoon Jung. “A Mixed Integer Linear Program for Airport Departure Scheduling.” In: (Sept. 2009). DOI: 10.2514/6.2009-6933.

[5] European Union Aviation Safety Agency (EASA). *Easy Access Rules for Air Traffic Management/Air Navigation Services (ATM/ANS) and other Air Traffic Management Network Functions — Regulation (EU) 2017/373*. Consolidated version including amendments. 2023. URL: https://www.easa.europa.eu/document-library/easy-access-rules/easy-access-rules-air-traffic-managementair-navigation-services.

[6] Federal Aviation Administration. *FAA Order JO 7110.65AA - Air Traffic Control*. Washington, D.C.: Federal Aviation Administration, 2024. URL: https://www.faa.gov/air_traffic/publications/atpubs/atc_html/chap3_section_8.html.

[7] IBM ILOG Cplex. “V12. 1: User’s Manual for CPLEX.” In: *International Business Machines Corporation* 46.53 (2009), p. 157.

A Complete results

A.1 1 Runway

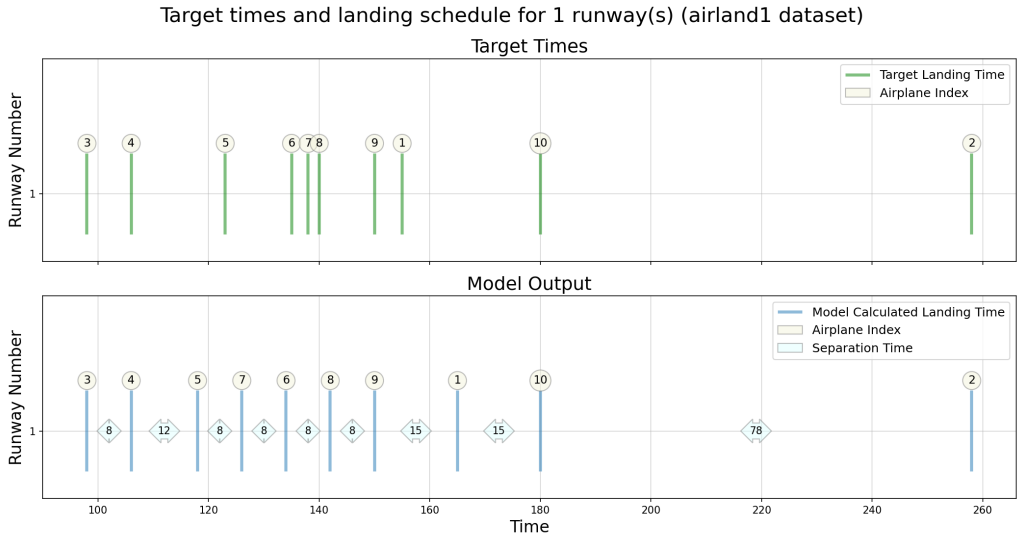


Figure 8: Dataset: airland1

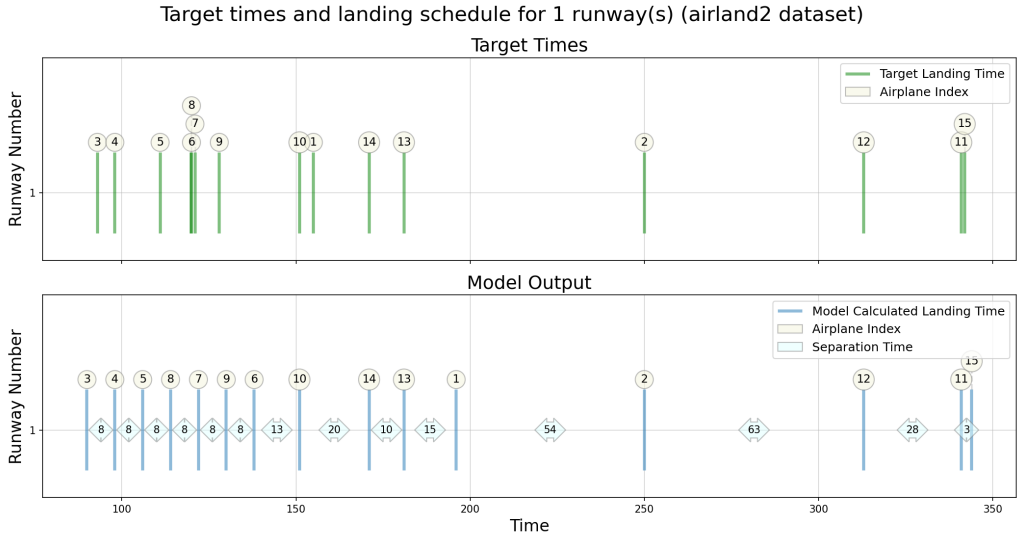


Figure 9: Dataset: airland2

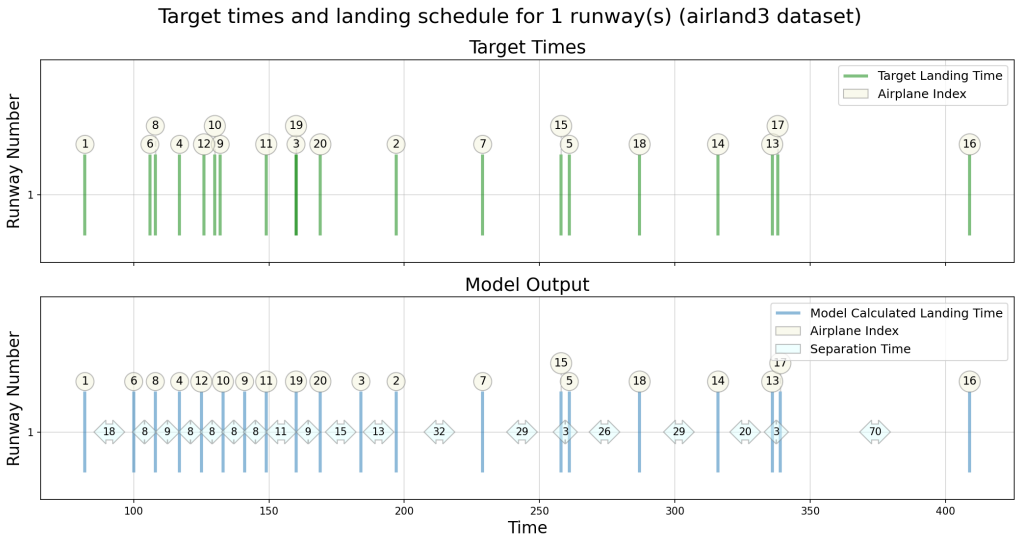


Figure 10: Dataset: airland3

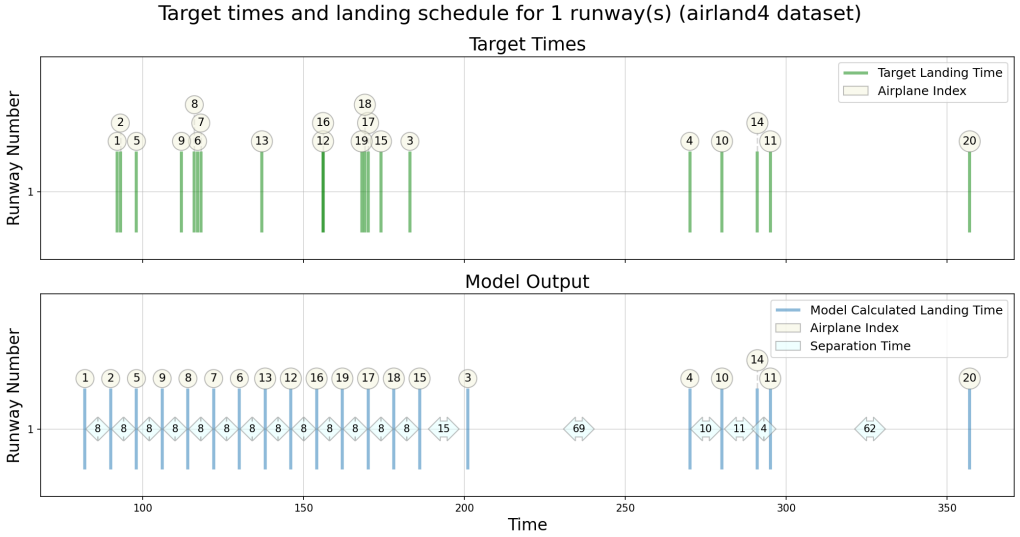


Figure 11: Dataset: airland4

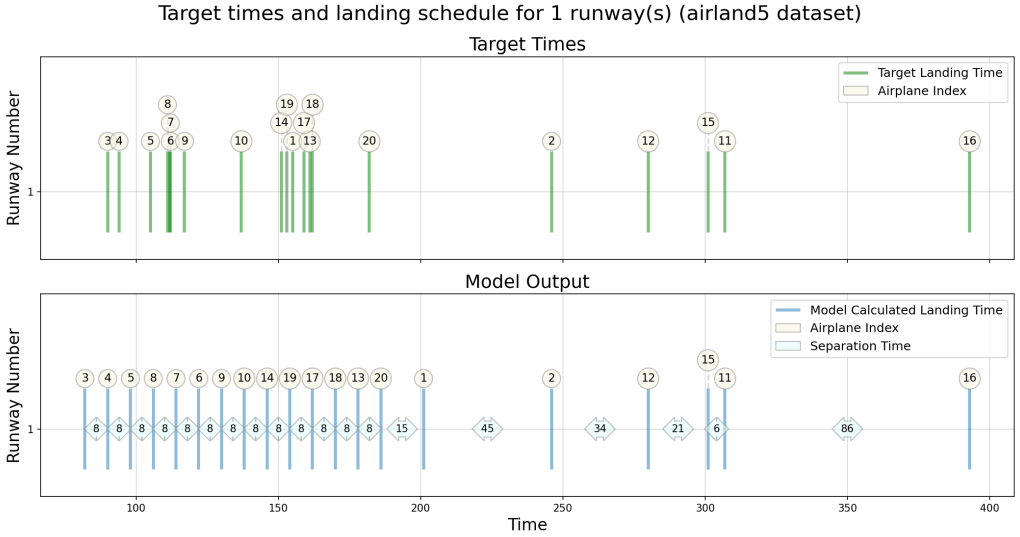


Figure 12: Dataset: airland5

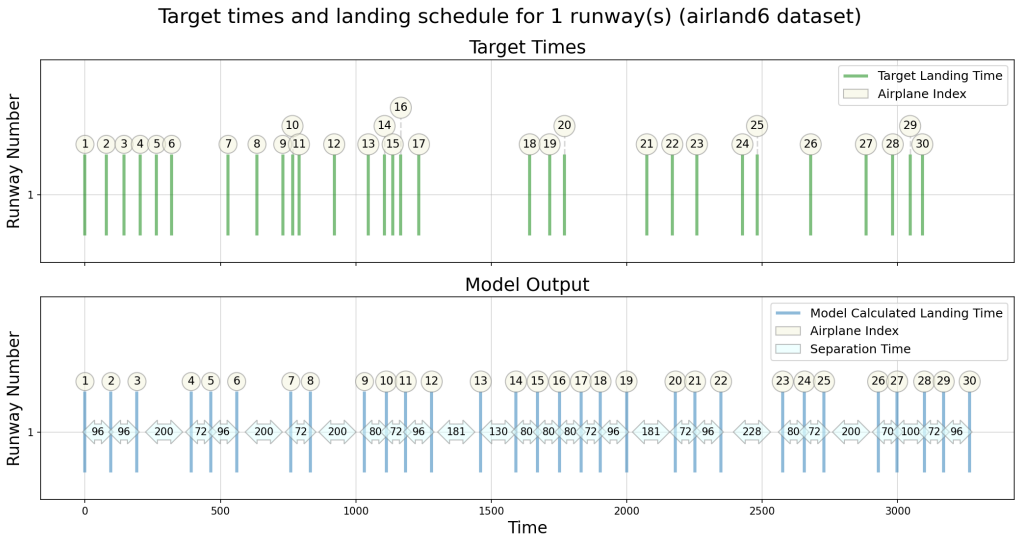


Figure 13: Dataset: airland6

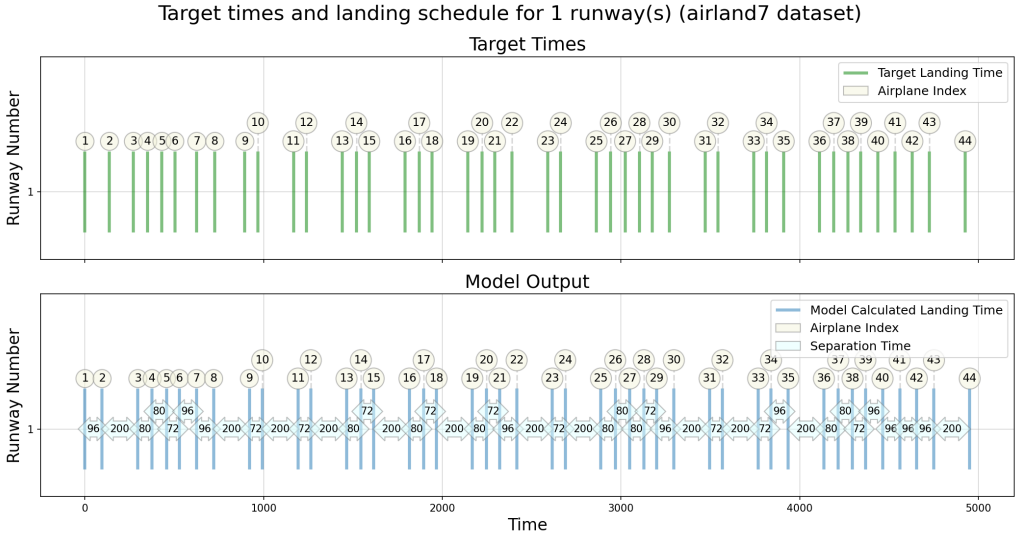


Figure 14: Dataset: airland7

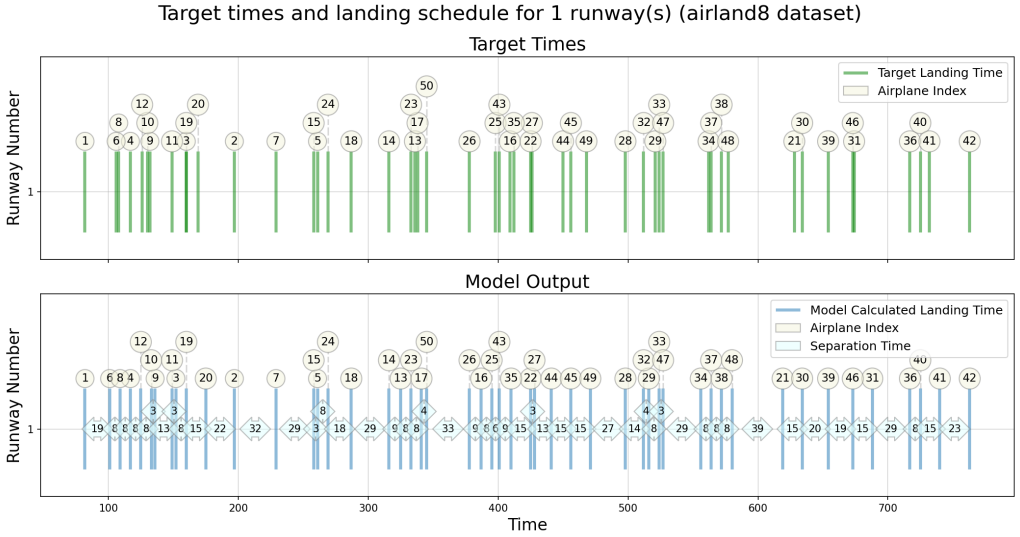


Figure 15: Dataset: airland8

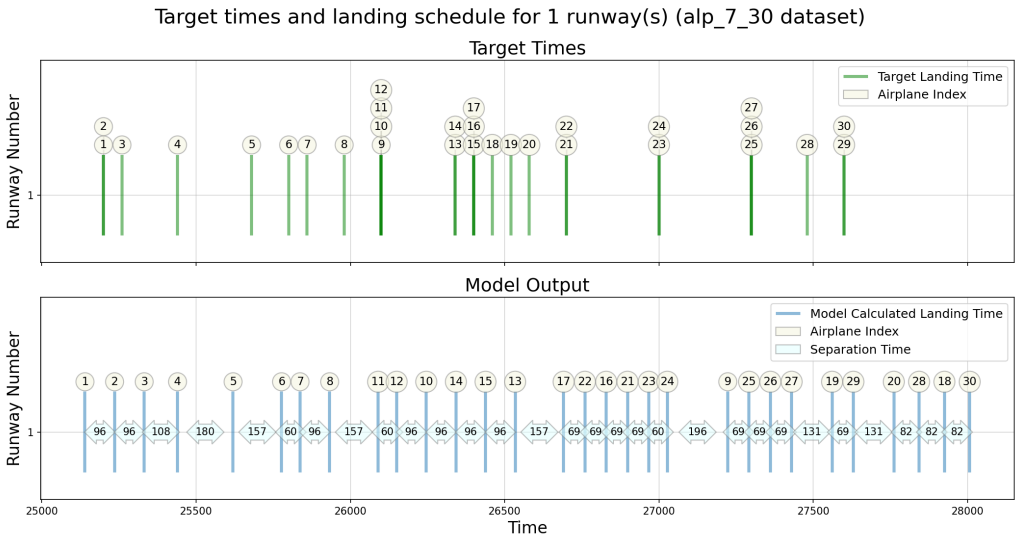


Figure 16: Dataset: alp_7_30

A.2 2 Runways

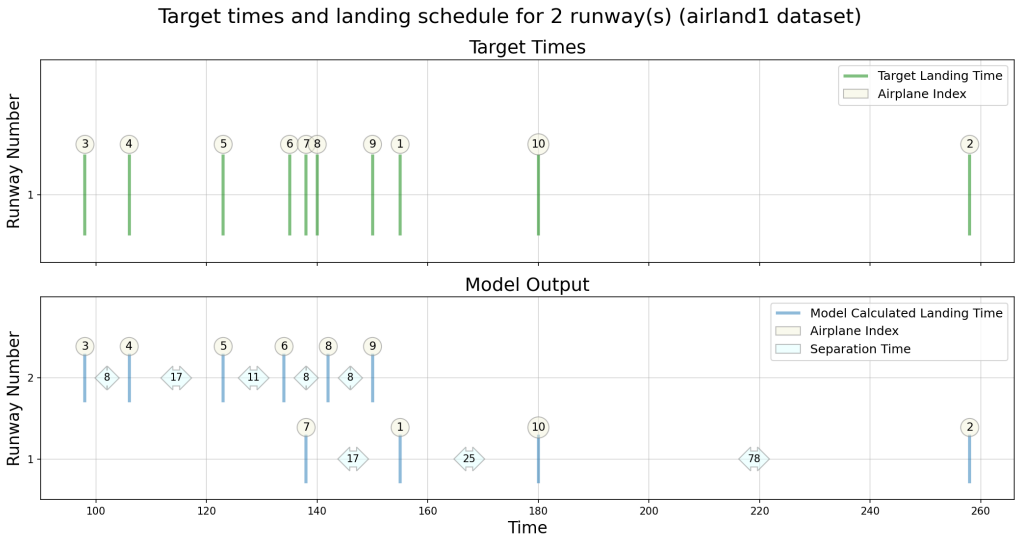


Figure 17: Dataset: airland1

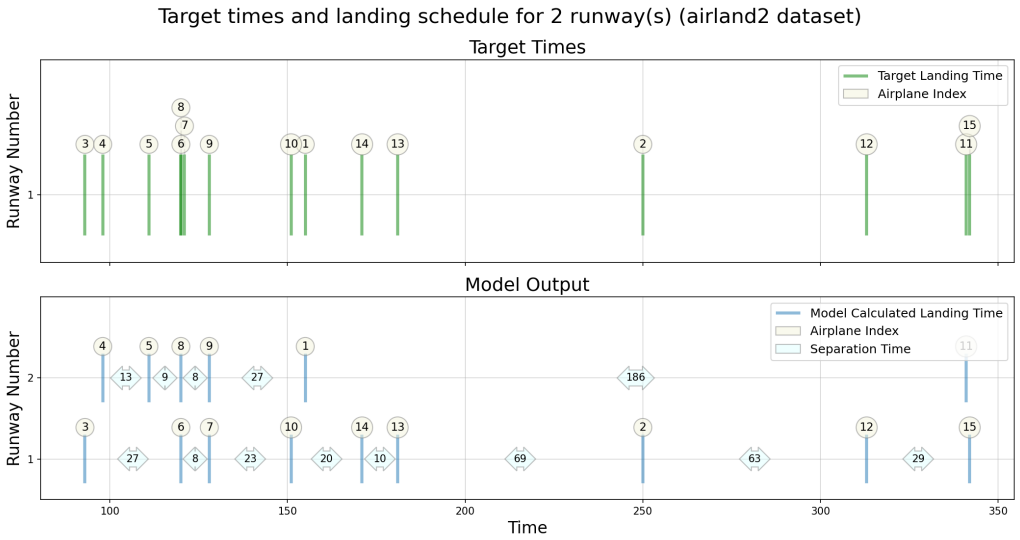


Figure 18: Dataset: airland2

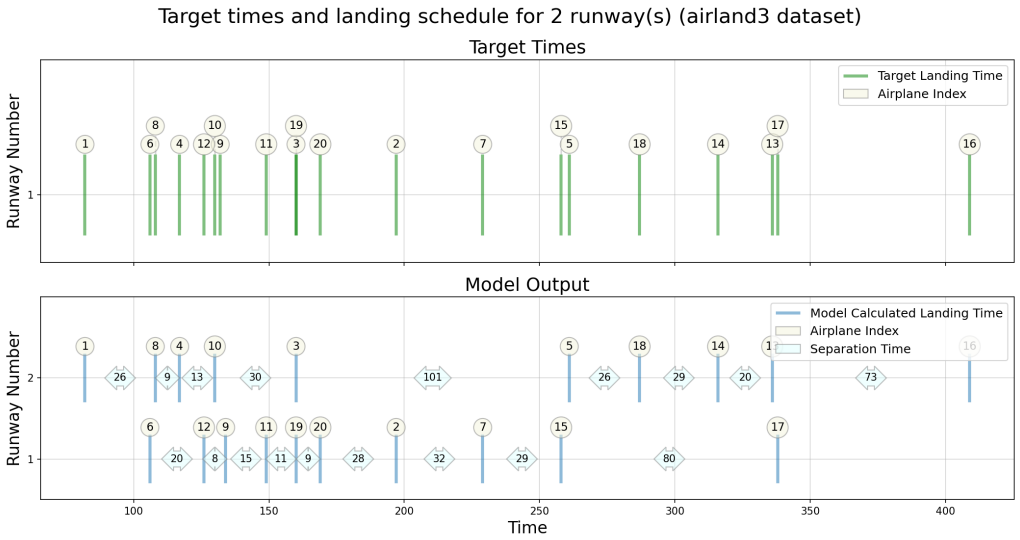


Figure 19: Dataset: airland3

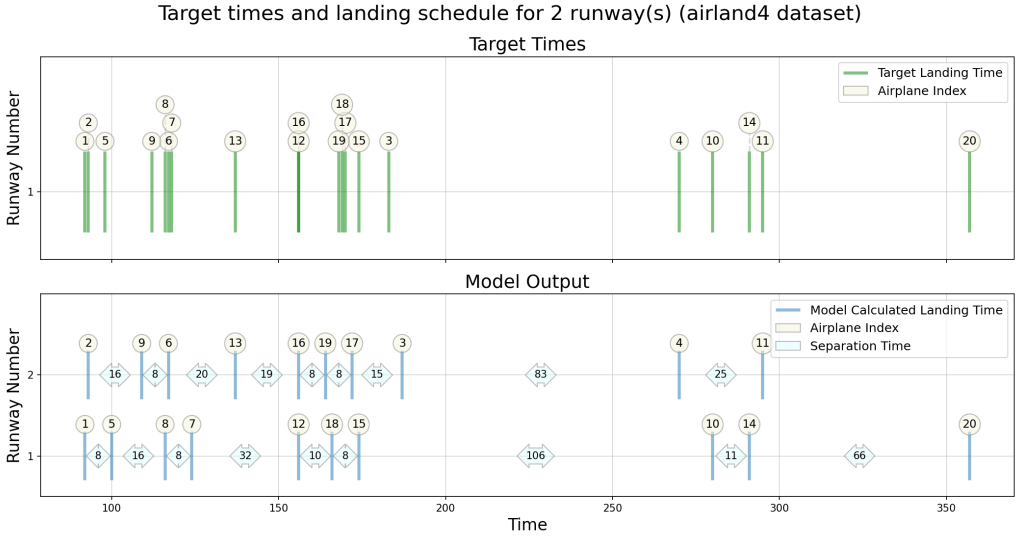


Figure 20: Dataset: airland4

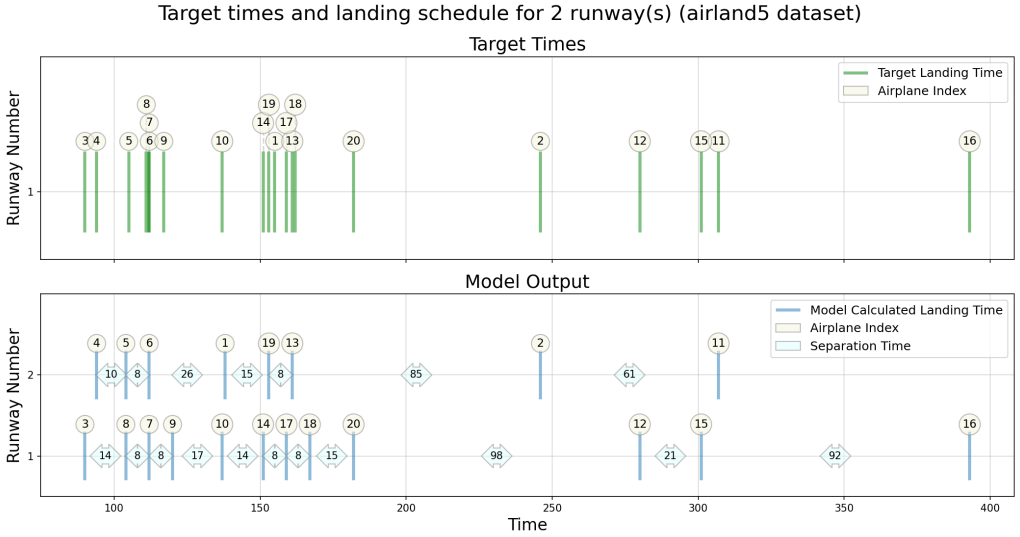


Figure 21: Dataset: airland5

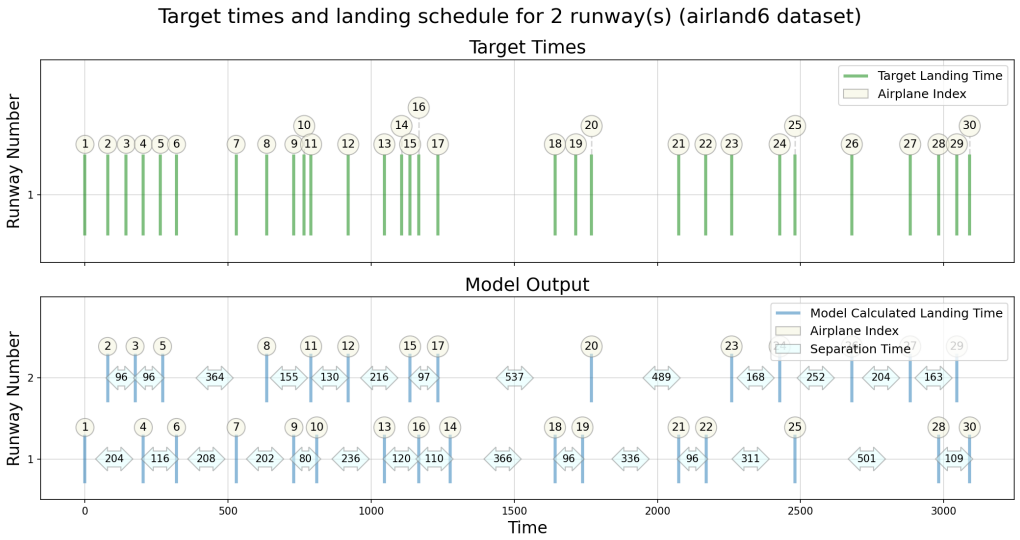


Figure 22: Dataset: airland6

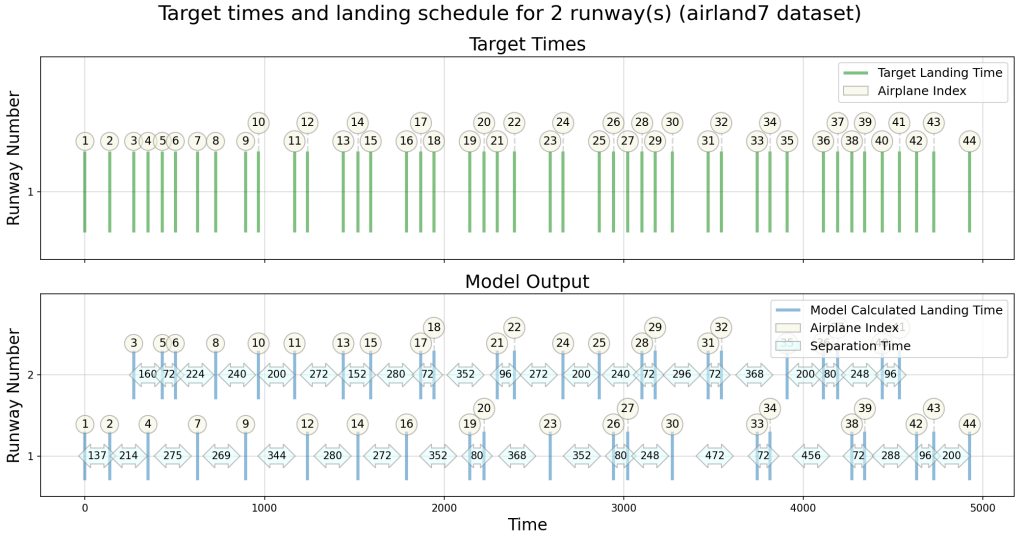


Figure 23: Dataset: airland7

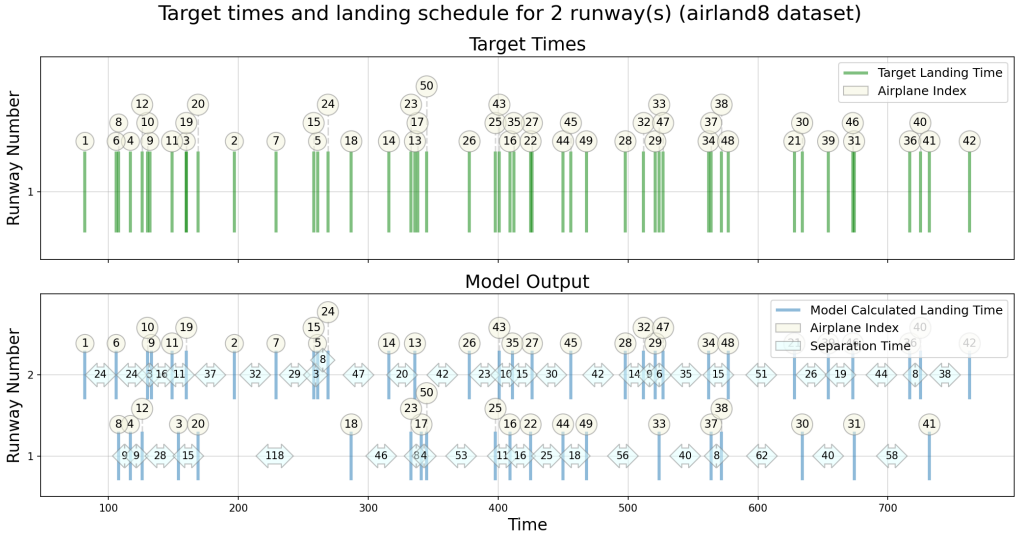


Figure 24: Dataset: airland8

A.3 3 Runways

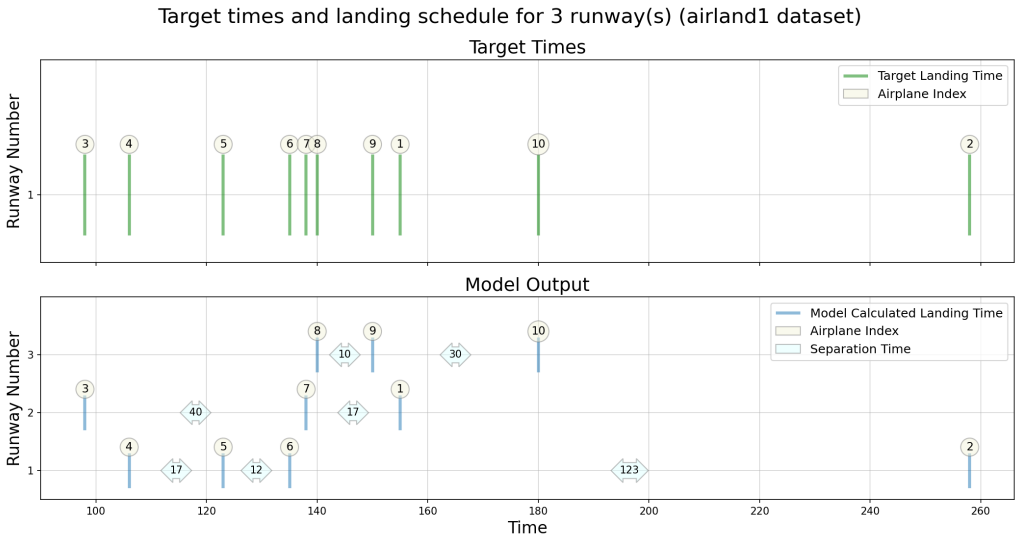


Figure 25: Dataset: airland1

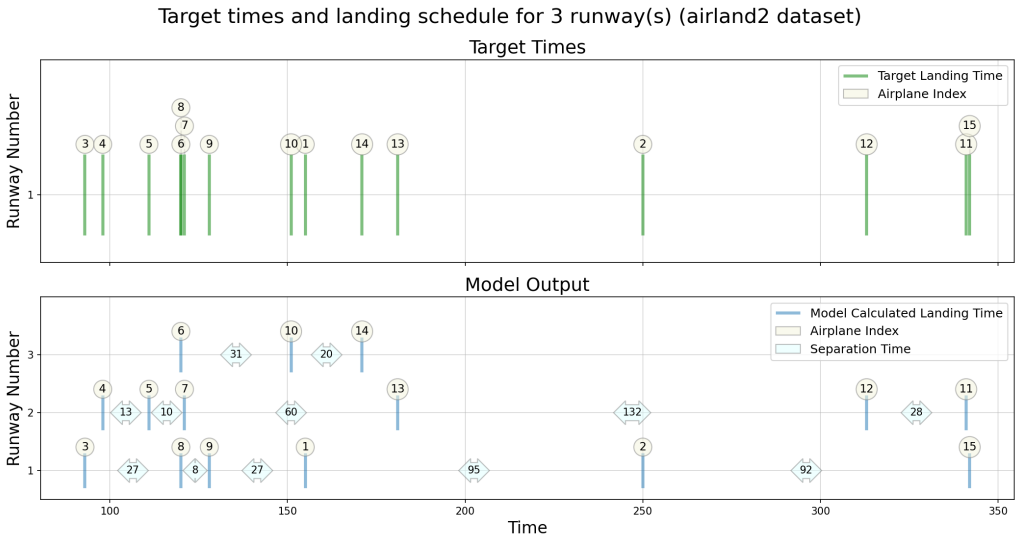


Figure 26: Dataset: airland2

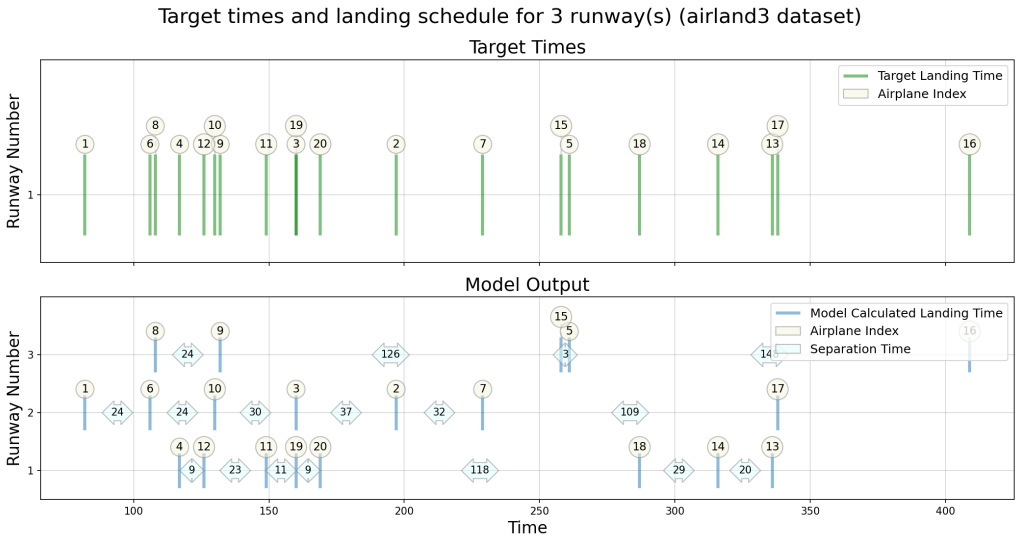


Figure 27: Dataset: airland3

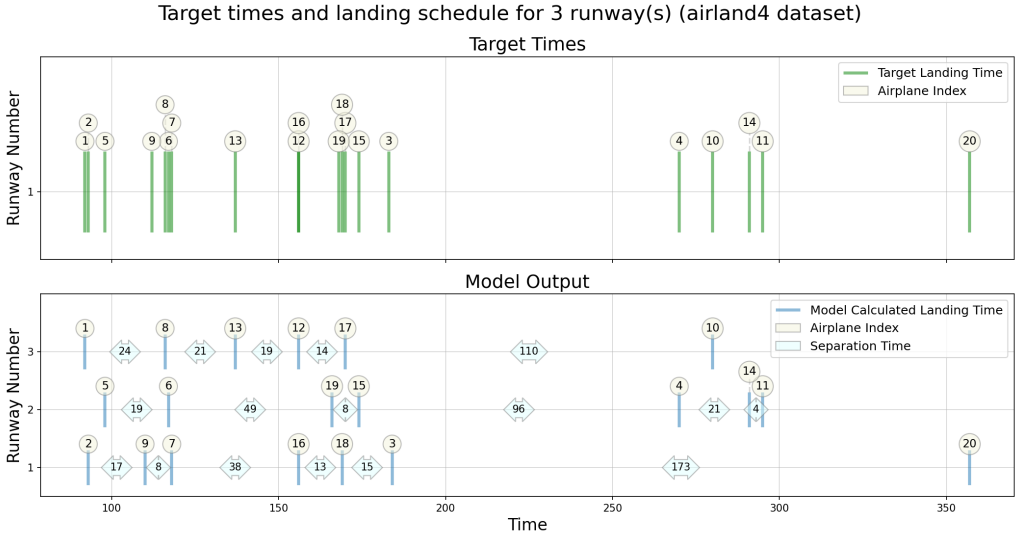


Figure 28: Dataset: airland4

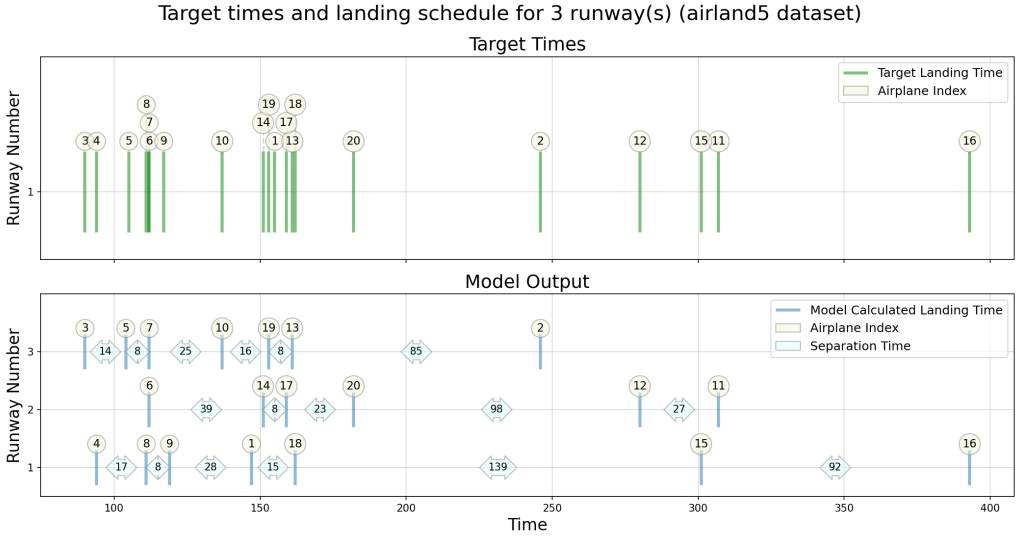


Figure 29: Dataset: airland5

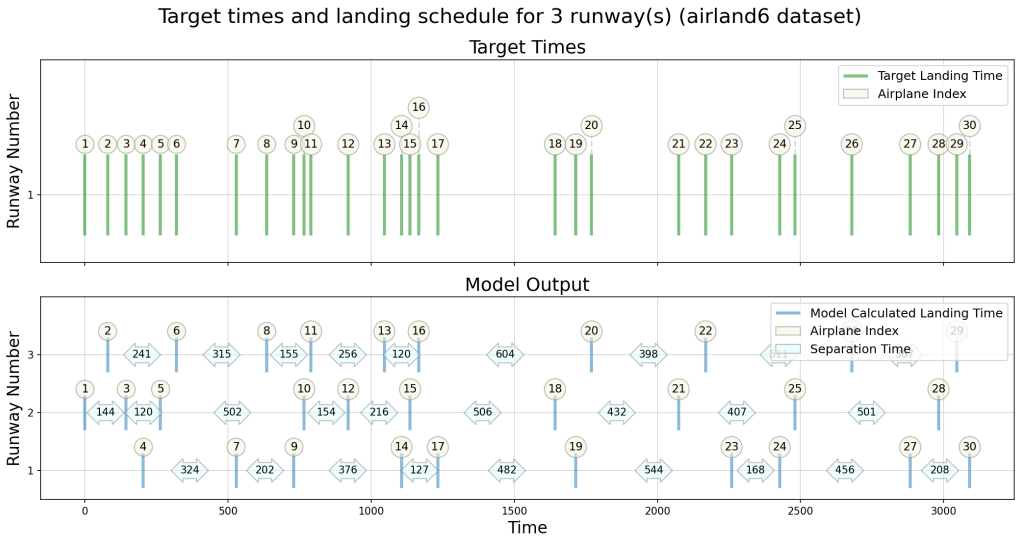


Figure 30: Dataset: airland6

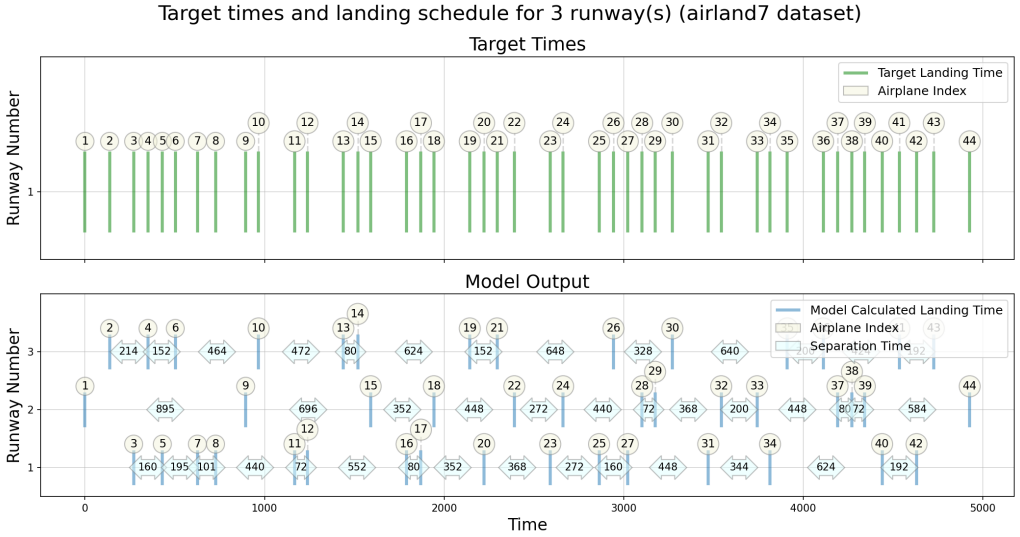


Figure 31: Dataset: airland7

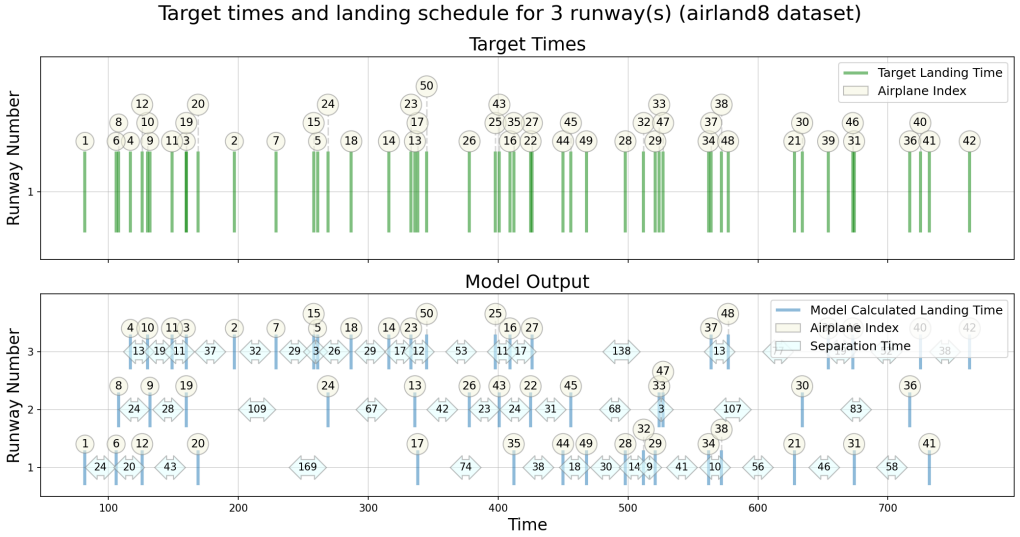


Figure 32: Dataset: airland8