Runway Assignment Optimisation Model for Istanbul Airport

Considering Multiple Parallel Runway Operations

Based on Güven et al. (2024)

December 20, 2025

The Aeronautical Journal — Optimization Implementation

Outline

Introduction & Problem Definition

Motivation: Istanbul Airport (IST)

- **Context:** IST is one of the busiest airports globally (approx. 1,000 daily flights).
- **Configuration:** 5 Parallel Runways ($34L, 34R, 35L, 35R, 36$).
- **The Problem:**
  - Long taxi times due to vast airport area.
  - High fuel consumption during ground operations.
  - Complex separation constraints (Wake Turbulence + Geometry).
- **Current State:** Fixed Runway Assignment Approach (FRAA) based on gate proximity.

Goal:
Minimize Total Fuel Consumption via Optimized Assignment (MILP)

Methodology Overview

- **Model Type:** Mixed Integer Linear Programming (MILP).
- **Data Sources:**
  - Actual traffic data (September 2021).
  - 30,000 flight operations analyzed for taxi times.
  - 47 distinct aircraft types modeled for fuel flow (f_i).
- **Optimization Engine:** Gurobi Optimizer.

Mathematical Model

Sets and Parameters

Sets:

- $I = \{1, \ldots, n\}$: Set of Aircraft.
- $J = \{1, \ldots, r\}$: Set of Runways (e.g., 34L, 36).
- K: Set of Parking Positions (Gates).

Key Parameters:

- f_i: Fuel flow rate for aircraft i (kg/min).
- rt_i: Actual runway time (from data).
- $ra_{ox_i,j}$: Availability (1 if operation type ox_i can use runway j).
- dpr_{j_1,j_2}: Dependent Parallel Runway indicator.

Decision Variables

The core decisions revolve around assignment and sequencing:

$x_{i,j}$	Binary. 1 if aircraft i is assigned to runway j.
$e1_{i_1,i_2}$	Binary. 1 if aircraft i_1 uses runway before i_2.
rut_i	Continuous. Runway Use Time for aircraft i.
gut_i	Continuous. Gate Use Time (Reach/Leave).
aw_i, gw_i	Continuous. Waiting times (Arrival/Departure).

Objective Function (Eq. 12)

Minimize total fuel consumption considering taxi duration and waiting times:

Minimization Goal

$$\min \sum_{i \in I} (taxia_i + taxid_i + aw_i + gw_i) \cdot f_i$$

Where:

- $taxia_i$: Taxi time for arrival ($rut_i \rightarrow gut_i$).
- $taxid_i$: Taxi time for departure ($gut_i \rightarrow rut_i$).
- f_i: Specific fuel consumption rate for aircraft type (from ICAO databank).

Critical Constraints: Runway Separation

Ensuring safety between aircraft i_1 and i_2 on dependent runways ($dpr_{j_1,j_2} = 1$).

Eq. 6 (Wake Turbulence & Geometry):

$$
\begin{aligned}
rut_{i_2} - rut_{i_1} \geq\ & tsep_{j,p_{i1},p_{i2}} \\
& - (1 - e1_{i_1,i_2}) \cdot M \\
& - (2 - x_{i_1,j_1} - x_{i_2,j_2}) \cdot M
\end{aligned}
$$

Note: M is a large constant ("Big M") to relax constraints when aircraft are not interacting or sequence is swapped.

Algorithmic Implementation

Modeling in Python (Gurobi)

1. Data Structures & Parameters: Mapping the paper's separation logic (Table 2 & 3) to code functions.

```python
# Table 2: Runway separation minima (NM)
def get_geo_sep_nm(arr_rwy_idx, dep_rwy_idx):
    arr_name = RUNWAY_NAMES[arr_rwy_idx]
    dep_name = RUNWAY_NAMES[dep_rwy_idx]
    sep = 0.0
    # Row 1: Arr 34L/36, Dep 34R/35L
    if dep_name in ['34R', '35L']:
        if arr_name == '34L': sep = max(sep, 8.0)
        if arr_name == '36':  sep = max(sep, 4.0)
    # ... (Logic continues for all Table 2 rows)
    return sep
```

Implementing Separation Constraints

The "Big M" constraint (Eq 6/7) translation into Gurobi:

```
# Eq 6: if i1 assigned j1, i2 assigned j2, and i1 precedes i2
m.addConstr(
    rut[i2] - rut[i1] >= tsep_12
    - M_VAL*(1 - e1[i1, i2])
    - M_VAL*(2 - x[i1, j1] - x[i2, j2]),
    name=f"Eq6_{i1}_{i2}_{j1}_{j2}"
)
```

- `tsep_12`: Calculated dynamically based on Wake Turbulence Categories (Heavy/Medium/Light) and Geometric interactions.
- `M_VAL`: Derived from the max time horizon.

Objective Definition

Aggregating fuel costs across all operational phases:

```
# Eq 12: Objective Function
obj = gp.LinExpr()
for i in I_SET:
    # Total time = Taxi In + Taxi Out + Wait (Gate/Rwy)
    total_duration = taxia[i] + taxid[i] + aw[i] + gw[i]

    # Cost = Time * Fuel Flow Rate
    obj += total_duration * traffic_data[i]['f']

m.setObjective(obj, GRB.MINIMIZE)
```

Experiments & Results

Experimental Scenarios

- **Baseline:** Fixed Runway Assignment Approach (FRAA).
  - Runway assigned strictly by proximity to parking gate.
- **Proposed Model (PMM):**
  - Dynamic assignment considering the entire traffic flow.
- **Scope:** 8 Scenarios representing busy 6-hour windows.

Metric	FRAA (Baseline)	PMM (Optimized)	Improvement
Avg. Taxi (Arr)	12.4 min	10.1 min	**18.6%**
Avg. Taxi (Dep)	16.7 min	14.9 min	**10.8%**
Total Fuel	168,347 kg	144,109 kg	**14.4%**

Table 1: Scenario 1 Results Summary (Highest Traffic)

Key Findings

1. **Fuel Reduction:** Between **6.6%** and **14.4%** total fuel savings depending on traffic density.
2. **Load Balancing:**
   - Model shifts traffic from congested 34L/35R to underutilized 35L/36 when feasible.
   - Prevents "bottlenecks" at taxiway intersections.
3. **Aircraft Type Impact:**
   - Heavy aircraft prioritized for shorter taxi routes due to higher f_i.
   - Significant reduction in *Gate Waiting Time* (gw_i).

Conclusion

- **Summary:** An MILP model effectively integrated with real-world IST constraints (Wake Turbulence, Geometric Separation).
- **Impact:** Demonstrates that abandoning fixed gate-to-runway rules yields significant economic and environmental benefits.
- **Future Work:**
    - Integration with Approach Control (TMA) operations.
    - Real-time dynamic re-optimization (Rolling Horizon).
    - Consideration of noise pollution constraints.

Questions?

Thank you for your attention.