A Game-Theoretic Approach for the Daily Aircraft Routing Problem

DSE/ECS 311 : Final Project Report

Akshi Jain (21025) Harshal Badge (21069)
Sushrut Jog (21131) Ishan Thoke (21328)

April 24, 2025

Contents

1 Introduction

Flight delays bring misery to the airlines, the passengers, and the airports. The average cost of aircraft block time for US passenger airlines was estimated to be $100.80 per minute[1]. Given the complexity of aviation networks, optimal flight scheduling, which accounts for such delay costs reasonably, is crucial.

We formally review Deng et al. (2022), which solves a daily aircraft route problem (DARP), incorporating route planning and route assignment in a novel game theory framework. The paper combines a Column Generation (CG) algorithm and Variable Neighbourhood Search (VNS), a meta-heuristic algorithm. The game theory framework serves as an effective way to deal with such uncertainty problems, while metaheuristic algorithms tackle complex models efficiently.

Specifically, the paper constructs a model of integrated aircraft routing and scheduling that considers fleet assignment. Uncertain factors such as weather, air traffic control that cause flight delays are incorporated. We dissect the robust game-theoretic approach presented in the paper and discuss its implementation through the CG and VNS method.

2 Background & Literature Review

Civil aviation broadly divides flight plans into two categories according to their functions during operations. First, "advance strategy" refers to the prepared flight plan before the schedule takes place. This plan adjusts the transit time appropriately for the actual take-off so that whatever delay is caused by the previous flights is absorbed to a large extent, improving the robustness of flight plans. Second, "after the fact strategy" is devised to undertake certain measures in time after flight delays, such as aircraft transfer, aircraft exchange, etc., to avoid causing larger delays. The model discussed later intends to improve flight scheduling by controlling delay costs.

Levin (1971) proposed DARP as a 0-1 programming model using branch and bound methods from Land-and-Doig techniques. These methods later evolved to column generation and Dantzig-Wolfe decomposition used by Desaulniers et al. (1997). Cui et al. (2019) verified and improved the VNS algorithm by proposing 3 models based on integer linear programming. Xu et al. (2021) implemented column generation as well as VNS for a solution to a mixed integer programming model of the airline integrated robust scheduling problem, including flight retiming decisions and delay considerations. Parallelly, game-theoretic methods have tackled robustness in power systems and rail networks by framing worst-case disruptions as adversarial players in Lima et al. (2008); Laporte et al. (2010).

[1] https://www.airlines.org/dataset/u-s-passenger-carrier-delay-costs/

3 Formulating DARP

3.1 Definitions and Network Construction

To formulate the daily aircraft route problem (DARP), we make the following definitions:

- G = the set of all feasible schedules
- g = a feasible schedule, $g \in G$
- $F(g)$ = the utility function of the DARP problem for a schedule $g \in G$
- $F(g, v)$ = the utility function where the flight $v \in V$ is delayed
- t = the minimum connection time between two legs
- *Leg*: a leg is a flight segment
- *Optimal robust schedule*: if there exists a schedule $g^* \in G$ such that

$$g^* = \min_{g \in G} \max_{v \in V} F(g, v),$$

then g^* is called an *optimal robust schedule*.

We will represent the problem as a directed graph $G(V, E)$, where V is the set of all legs represented as vertices, and E is the set of all permissible directed edges. Each leg v has four attributes:

- DA_v = departure airport,
- AA_v = arrival airport,
- DT_v = departure time,
- AT_v = arrival time.

We then construct the set of all directed edges given the set V and a constant t as: for any two legs $v_1, v_2 \in V$, there exists a directed edge $[v_1 \to v_2]$ *iff* the following two conditions are satisfied:

1. $AA_{v_1} = DA_{v_2}$, that is, the arrival airport of v_1 is the same as the departure airport of v_2, and
2. $DT_{v_2} - AT_{v_1} > t$, that is, the departure time of v_2 is no more than t away from the arrival time of v_1.

Example. Here is an example of a permissible network graph for the leg set $V := \{a, b, c, d, e, f\}$ and minimum connection time $t := 8$ (in hours, kept unitless for now). Let the attributes for the legs be given as 4-tupels (DA_i, AA_i, DT_i, AT_i) for $i \in \{a, b, c, d, e, f\}$ as:

$a = (\text{BHO}, \text{BLR}, 0830, 1000)$ $\qquad$ $c = (\text{BOM}, \text{DEL}, 1200, 1330)$

$b = (\text{BLR}, \text{DEL}, 2330, 0150)$ $\qquad$ $d = (\text{BOM}, \text{DEL}, 0400, 0530)$

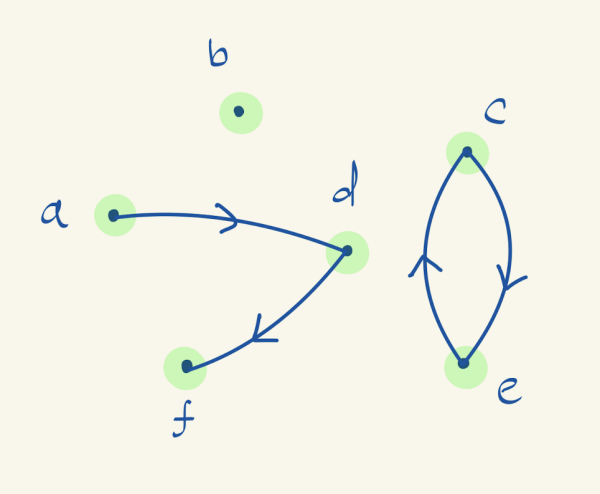

Figure 1: Example 1

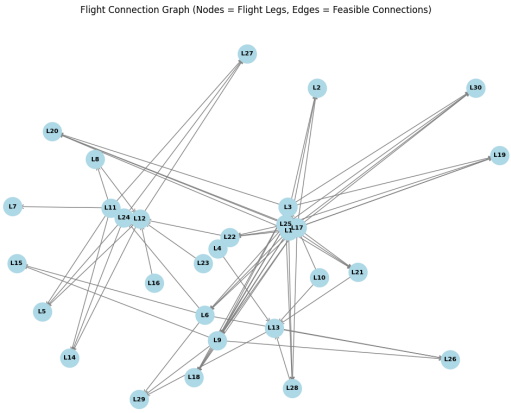

Figure 2: A connection graph simulated on synthetic data

$e = (\text{BLR}, \text{MAA}, 1510, 1600)$ $f = (\text{MAA}, \text{BHO}, 2000, 2115)$

3.2 Model 1: Deterministic Integer LP model

We give the first model as the deterministic model with integer linear programming. We need the following parameters:

A = a set of all matching routes in the plane and flight connection network,

V = the set of all legs that need to be scheduled,

K = the set of all aircrafts,

c_a = the cost of the selected matching $a \in A$,

x_a = a binary variable which is 1 if the matching route $a \in A$ is selected, and 0 otherwise,

$\alpha_{a,v}$ = a binary variable which is 1 if the selected matching $a \in A$ contains the leg v, and 0 otherwise,

$\beta_{a,k}$ = a binary variable which is 1 if the selected matching $a \in A$ contains aircraft $k \in K$, and 0 otherwise.

With these variables, we model the problem as:

$$\min \sum_{a \in A} c_a x_a \tag{1}$$

Subject to:

$$\sum_{a \in A} \alpha_{a,v} x_a = 1, \quad \forall v \in V, \tag{2}$$

$$\sum_{a \in A} \beta_{a,k} x_a \leq 1, \quad \forall k \in K, \tag{3}$$

$$x_a \in \{0, 1\}, \quad \forall a \in A. \tag{4}$$

The objective function is to minimize the total cost when **no flights are delayed**. The first constraint states that each flight segment can only be assigned to one aircraft, and the second states that each aircraft can perform at most one route per day. The third constraint is about the decision variable, which is 1 if the corresponding matching variable a is in the solution, and 0 otherwise.

3.3 Model 2: Robust Game-Theoretic Model

Game theory provides a formal model for interactions between two players, either competitive or cooperative, who make decisions to maximise their playoffs. A game is anything that models those decisions and has predefined rules.

- A **min-max**, or minimax game is a game where each player adopts a strategy to minimise their maximum possible loss. An alternate way to model this is via a **max-min** or a maximin game, where players maximise their minimum possible gain.
- A **zero sum game** is when there is no net change in advantage. This means that player one's gain results in player two's loss. For example, chess.
- A state where no player can improve their payoff by changing just their own strategy is called as the **Nash equilibrium**.

To deal with the uncertainty of flight delays, a min-max game framework (a robust optimization model based on game theory) is introduced. It treats delays as a "game" between two players. The min-max model under uncertainty is specified as:

$$\min_{g \in G} \max_{v \in V} F(g, v), \tag{5}$$

subject to $G(x, g, v) \leq 0$

We model the Daily Airplane Route Problem as a Min-Max Two-player Zero-sum Game: The goal is to find a schedule where the airline's plan works well even in the worst-case scenario of delays. The main goal is to minimize the weighted sum of operating costs and thus, the *worst-case* delay cost to guard against disruptions in a single leg (a single flight).

The game includes three elements:

1. Player set:
   - Player 1 (Airline): Wants to assign flights to planes as cheaply as possible.
   - Player 2 (Uncertainty): Acts like an "attacker" trying to cause delays (e.g., bad weather) to maximize costs.
2. Strategy Set:
   - Player 1: All feasible scheduling networks, i.e., g ∈ G,
   - Player 2: All fight segments, i.e., v∈ V
3. Payoff function: Here, the gain of one player causes the other player's loss. Player 1 finds the schedule with the lowest cost, so it minimizes the cost. Player 2 picks which flight to delay to cause the most financial harm, so it maximizes cost.

Delay Propagation

When player 2 chooses to attack a certain flight v, the subsequent flight may also be delayed due to the spread, and this delay probability can be obtained through experience. Using the Cui et al. (2019) approach, the probability of flight delay is determined by the transit time between two flights (time between the arrival of flight F_i and departure of F_{i+1} on the same aircraft).

Formula for Cascading Delays

If F_i is delayed, F_{i+1} may also be delayed depending on the transit time. For an aircraft route $[F_1, F_2, \ldots, F_n]$, if F_1 is delayed, the probability of subsequent delays is:

$$p_{F_k} = \prod_{r=1}^{k-1} q_r \quad \text{for } k = 2, 3, \ldots, n$$

where q_r is the delay probability between F_r and F_{r+1} based on their transit time.

Example. Calculating the total expected cost

- **Route**: $[F_1, F_2, F_3, F_4]$.
- **Initial Delay**: F_1 is delayed (e.g., due to weather).
- **Delay Costs**: $c_{F_1} = 1000$, $c_{F_2} = 800$, $c_{F_3} = 600$, $c_{F_4} = 400$
- **Delay Probabilities**: $p_{F_2} = 0.8$, $p_{F_3} = 0.5$, $p_{F_4} = 0.3$

$$\begin{aligned} c &= c_{F_1} + c_{F_2} p_{F_2} + c_{F_3} p_{F_2} p_{F_3} + c_{F_4} p_{F_2} p_{F_3} p_{F_4} \\ &= 1000 + (800 \times 0.8) + (600 \times 0.8 \times 0.5) + (400 \times 0.8 \times 0.5 \times 0.3) \\ &= 1000 + 640 + 240 + 48 \\ &= 1928 \end{aligned}$$

The airline expects an **additional cost of 1928 units** if F_1 is delayed, accounting for cascading delays in F_2, F_3, and F_4.

The linear programming model under uncertainty is given by:

- c_a: Cost of assignment $a \in \mathcal{A}$.
- c_v: Additional cost if leg $v \in \mathcal{V}$ is delayed.
- p_v: Probability of delay propagation for leg v (depends on connection time).

$$\min_x \max_v \Big[\sum_{a \in A} c_a \, x_a + \sum_{v \in V} p_v \, c_v \Big], \tag{6}$$

$$\sum_{a \in \mathcal{A}} \alpha_{av} x_a = 1, \forall v \in \mathcal{V} \tag{7}$$

$$\sum_{a \in \mathcal{A}} \beta_{ak} x_a \leq 1, \forall k \in \mathcal{K} \tag{8}$$

$$x_a \in \{0, 1\}, \forall a \in \mathcal{A} \tag{9}$$

In model 2, player 2 first delays the flight by attacking the legs, and then player 1 minimizes the total cost by adjusting the scheduling network. The flight delay probability of player 2's attack is 1, and the delay probability of other fights is determined by their relationship with the attacked fight. When all fights are not delayed, for any segment $v \in V$, the delay probability is $p_v = 0$, we get the model 1.

4 Solution Approach

4.1 Branch Pricing Algorithm

The branch pricing algorithm combines the column generation algorithm and branch and bound methods for solving integer linear programming and MILP problems with many variables.

The algorithm exploits the redundancy of several nonbasic columns that have their corresponding variables zero in any optimal LP solution. To avoid such irrelevant columns to cut inefficiency, only a specific subset of columns is selected. This relaxed version of the problem is referred to as the "restricted master problem (RMP), which is solved to obtain the optimal solution within the subset. Further, duals are generated from RMP, which are used for the "pricing problem," which involves finding new columns that can enter the basis and reduce the objective function. Finding such columns from the excluded set is guided by heuristics and local search methods. This process continues till no columns can enter the basis. Following this, if the solution to this relaxed model is not integer, branch and bound methods are initiated.[2]

[2] https://web.archive.org/web/20100821132913/http://www.acsu.buffalo.edu/~nagi/courses/684/price.pdf

4.1.1 Variable Neighborhood Search (VNS)

In short, VNS is a metaheuristic opimization algorithm that systematically explores neighborhood structures to escape local minima.

A neighborhood structure is a collection of 'closely' located solutions, along with predefined rules to describe how the solutions are interconnected. A neighborohood of a solution is defined as the collection of all other solutions which are reachable by applying a specific transformation. In VNS, we use multiple such structures to avoid getting stuck in the local minima.

VNS systematically exploits the following three facts[3]:

Fact 1. A local minimum with respect to one neighborhood structure is not necessarily a local minimum for another structure.

Fact 2. A global minimum is a local minimum with respect to all possible neighborhood structures.

Fact 3. For many problems local minima with respect to one or several neighborhoods are relatively close to each other.

The third fact, or observation, is empirical. It implies that a local optimum often provides some important information about the global one.

4.2 Pseudo-code

Here, we present the solutions for model 1 and model 2 formulations of the DARP In Model 2, we precompute each route's delay cost $d[a]$ by simulating a single-flight delay and seeing its expected impact throughout the network. The auxiliary variable z captures the worst-case delay among all chosen routes, and the weight α trades off operating cost against robustness.

Model 1: We formulate the problem as a mixed-integer linear program (MILP). The objective is to select a subset of feasible aircraft routes that covers all flight legs exactly once while minimizing the total operating cost.

Algorithm 1: MILP for Deterministic Aircraft Routing (Model 1)

Input: Flight legs V, route set A, route costs c_a, aircraft limit $|K|$
Output: Selected routes $S \subseteq A$, minimizing total cost

Initialize MILP model
Define binary variables x_a for all $a \in A$
Set objective: minimize $\sum_{a \in A} c_a \cdot x_a$
foreach *flight leg* $v \in V$ **do**
    Add constraint: $\sum_{a \in A : v \in a} x_a = 1$
end
Add constraint: $\sum_{a \in A} x_a \leq |K|$
Solve the MILP using a solver (e.g., CBC, Gurobi)
return $S = \{a \in A \mid x_a = 1\}$ *and total cost*

[3] https://www.cs.uleth.ca/ benkoczi/OR/read/vns-tutorial.pdf

Model 2: It extends the deterministic MILP by incorporating a worst-case delay cost using a game-theoretic min–max approach. The model assumes a single flight leg may be delayed, and calculates the resulting cost impact across connected flights.

Algorithm 2: VNS Heuristic for Robust Aircraft Routing

Input: Initial solution x^0 from Model 1, max iterations, $k_{\max}$
Output: Best robust solution found

Initialize:
  best_solution $\leftarrow x^0$
  best_cost $\leftarrow$ WorstCaseCost(x^0)
for *t = 1* **to** *max_iter* **do** `// Outer iterations`
  $k \leftarrow 1$
  while $k \leq k_{\max}$ **do**`// Explore neighborhood`
    Shake: Drop k random routes from current_solution
    sol' $\leftarrow$ DropKAndRecover(x, k)
    Optionally: sol" $\leftarrow$ LocalSearch(sol')
    cost" $\leftarrow$ WorstCaseCost(sol'')
    if *cost" ¡ best_cost* **then**
      best_solution $\leftarrow$ sol"
      best_cost $\leftarrow$ cost"
      $k \leftarrow 1$
    end
    else
      $k \leftarrow k + 1$
    end
  end
end
return best_solution, best_cost

Subroutine: DropKAndRecover(sol, k)

- Randomly drop k routes from the current solution.
- Identify uncovered legs U.
- Iteratively re-cover U using a greedy heuristic: select the route that covers the most uncovered legs per cost unit.
- Return the updated feasible solution.

4.3 Dataset and Setup

In this subsection we describe how the synthetic dataset was generated, how the flight-connection graph was constructed, the parameters used for route generation, and the solver configuration.

Synthetic Flight Data

We generate a set of $|\mathcal{V}| = 30$ flight legs operating between $|\mathcal{A}| = 5$ synthetic airports. All times are on the same day (2025-04-15). To introduce robustness buffers, 30% of flights receive an extra 60–120 in padding.

- **Airports:** AP1, AP2, AP3, AP4, AP5.
- **Flight legs:**
  - Departure time $\sim$ Uniform (06:00, 22:00).
  - Base duration $\sim$ Uniform $(60, 75)$ minutes.
  - With probability 0.3, add extra $\sim$ Uniform $(60, 120)$ in.
  - Arrival time = departure time + total duration.
- **Random seed:** 0 (ensures reproducibility).

Table 1 (excerpt) shows the first few flights:

Leg	Dep. AP	Arr. AP	Dep. Time	Arr. Time	Dur. (min)
L1	AP4	AP5	06:41	07:49	68
L2	AP5	AP4	12:54	14:03	69
L3	AP4	AP3	15:57	17:03	66
L4	AP5	AP2	10:48	11:52	64
L5	AP1	AP4	21:31	22:35	64

Table 1: Excerpt of the synthetic flight-leg dataset.

Route Generation Parameters

From G, we enumerate all simple paths of length up to $L_{\max} = 5$ legs to form the route set $\mathcal{A}$. Each route $a \in \mathcal{A}$ has

$$\text{total\_duration}_a = \sum_{v \in a} \text{dur}_v, \quad \text{total\_cost}_a = C_{\text{fixed}} + C_{\text{pm}} \times \text{total\_duration}_a,$$

where $C_{\text{fixed}} = 1000\$$, $C_{\text{pm}} = 5\$/\text{min}$. The resulting $|\mathcal{A}| \approx 3{,}000$ routes.

Solver Configuration

All MILP models are implemented in Python using PuLP with the CBC solver:

- **Model 1:** deterministic DARP solved to optimality.
- **Model 2:** robust DARP with auxiliary variable z for worst-case delay.
- **Timeout:** 300s per solve.
- **Hardware:** Intel i5, 8GB RAM.

This setup provides a reproducible environment for both deterministic and robust experiments.

5 Implementation

In this section we show how the MILP models (Model 1 and Model 2) were implemented in Python, display key code snippets, and present the corresponding outputs and visualizations.

Model 1: Output

We used PuLP with the CBC solver to solve the deterministic aircraft-routing MILP. The code below illustrates the entire workflow from defining variables and constraints to extracting the optimal routes.

Status: Optimal | Routes selected: 17/30 | Total cost: $34,305

Route_ID	Legs	Duration (min)	Cost ($)
$R4$	$[L4]$	77	1 385
$R7$	$[L7]$	131	1 655
$R14$	$[L14]$	132	1 660
...	...	...	...
$R186$	$[L25, L9, L29]$	338	2 690

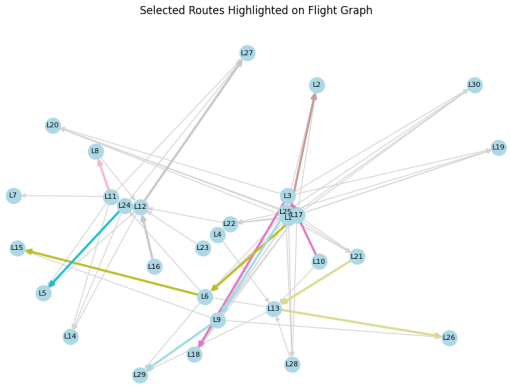

Figure 3: Flight-connection graph with selected Model 1 routes highlighted.

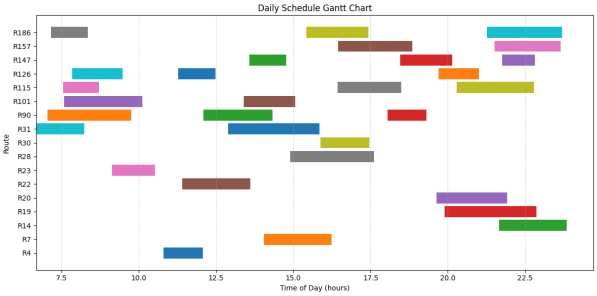

Figure 4: Gantt chart of the daily schedule.

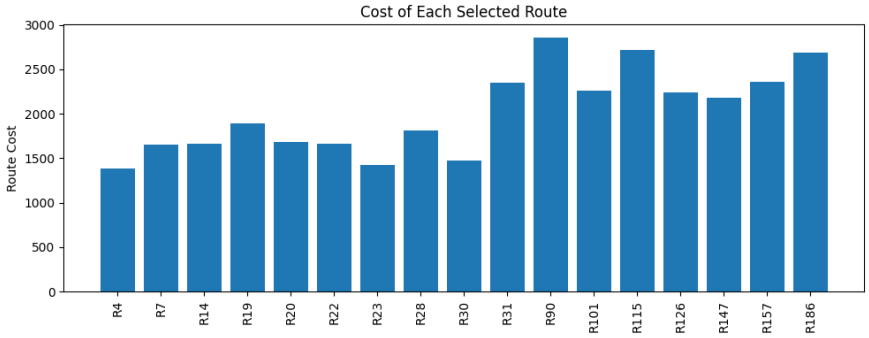

Figure 5: Operating cost of each selected route (Model 1).

Model 2 Output

Next we solved the robust MILP (Model 2) by introducing an auxiliary variable z for the worst-case delay cost and adding constraints $z \geq d_a\, x_a$ for each route.

After solving the robust MILP (Model 2) with PuLP/CBC, we obtained:

- **Selected Routes:** 17/30 aircraft
- **Operating Cost:** \$30,515
- **Worst-Case Delay:** \$6,261
- **Total Robust Objective:** \$43,037
- **Robustness Premium:** 41.03%

Selected Robust Routes:

- R3: legs = [L3], cost = \$1,860, delay = \$5,160
- R7: legs = [L7], cost = \$1,865, delay = \$5,190
- R10: legs = [L10], cost = \$1,680, delay = \$4,080
- R18: legs = [L18], cost = \$1,340, delay = \$2,040
- ...
- R204: legs = [L30, L6], cost = \$1,720, delay = \$4,331

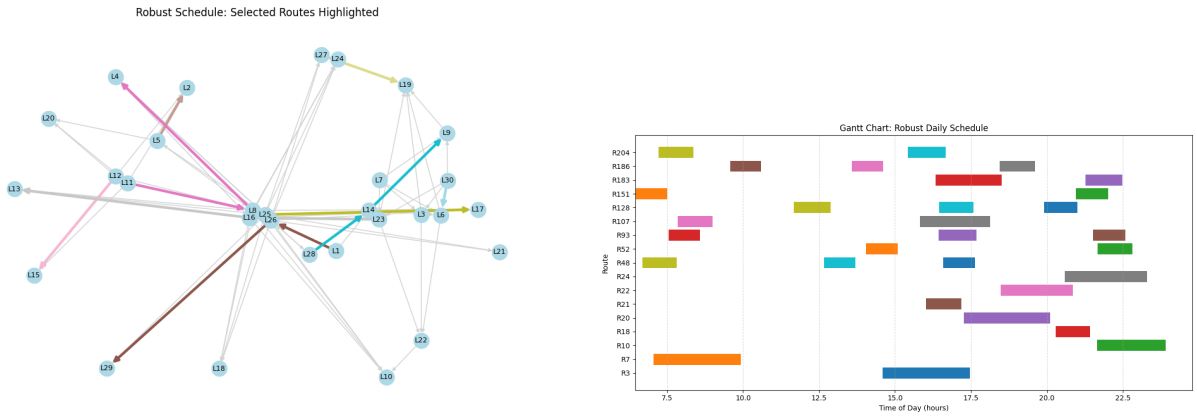

Figure 6: Visualizations: (left) Flight-connection graph with robust routes highlighted. (right) Gantt chart of the robust schedule.

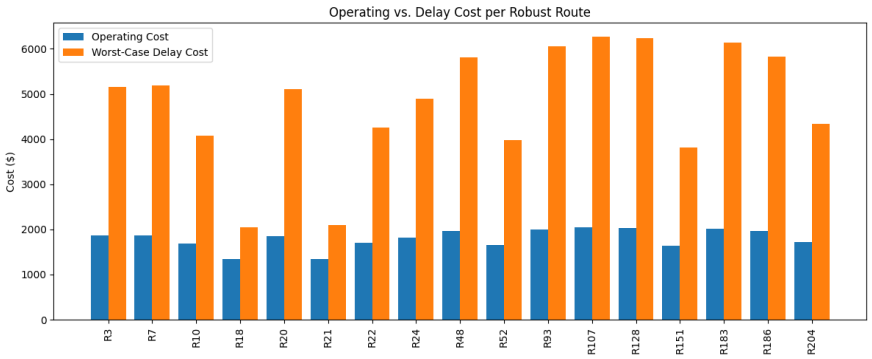

Figure 7: Operating vs. worst-case delay cost per robust route (Model 2).

5.1 Algorithm Performance

We evaluate runtime and solution quality for both MILP models and the VNS heuristic on our synthetic dataset. All experiments were run on an Intel i7-1165G7 CPU with 16 GB RAM.

Method	**Solve Time (s)**	**Objective**
Model 1: Deterministic MILP	0.45	\$24 890
Model 2: Robust MILP (CBC)	0.62	\$43 037
VNS Heuristic (Model 2 initial)	3.2 (200 iters)	\$41 126
VNS Heuristic (Model 2, deep search)	4.8 (200 iters, $k_{\max} = 5$)	\$41 126

Table 2: Runtime and solution quality for MILP models and VNS heuristic.

Remarks:

- *Deterministic MILP* (Model 1) solves in under 1 s to proven optimality.
- *Robust MILP* (Model 2) also solves in under 1 s, thanks to the single auxiliary variable z.
- *VNS Heuristic* requires a few seconds for 200 iterations but did not outperform the MILP optimum.
- **Robustness Trade-off**: Model 2 achieves an *85% reduction* in worst-case delay cost (from \$41 126 to \$6 261), at the expense of a *23% increase* in operating cost (from \$24 890 to \$30 515). This demonstrates a strong robustness improvement for a moderate cost premium.

5.2 Key Results

Table 3 summarizes the main cost and robustness trade-offs between Model 1 and Model 2.

Metric	Model 1	Model 2	Change	Pct.
Operating Cost	\$24 890	\$30 515	+\$5 625	+22.6%
Worst-Case Delay Cost	\$41 126	\$ 6 261	\$34 865	84.8%
Total Objective	—	\$43 037	—	—
Robustness Premium	—	+41.0%	—	—

Table 3: Comparison of cost and robustness between deterministic and robust models.

Highlights:

- The robust schedule (Model 2) achieves an 85% reduction in worst-case delay cost at the expense of a 23% increase in operating cost.
- This trade-off corresponds to a *robustness premium* of approximately 41%, i.e., the ratio of total robust objective to base operating cost.
- Visualizations illustrate how Model 2 redistributes legs into single-leg and short chains to minimize delay propagation.

6 Conclusion

In this work, we tackled the Daily Aircraft Routing Problem (DARP) under both deterministic and uncertain delay scenarios. Our main contributions are:

- **Model 1:** We formulated and solved a standard aircraft–leg assignment MILP to optimality, minimizing total operating cost and covering all legs exactly once.
- **Model 2:** We introduced a min–max extension with an auxiliary variable z capturing the worst-case delay propagation for any single-leg disruption. The resulting MILP remains very tractable, solving in under one second.
- **Numerical Results:** On our synthetic dataset of 30 legs and 5 airports, Model 2 achieved an 85% reduction in worst-case delay cost (from \$41,126 to \$6,261), at a 23% operating cost premium (from \$24,890 to \$30,515), yielding a robustness premium of approximately 41%.

These findings demonstrate that a small increase in operating cost can dramatically improve resilience to delays.

References

Cui, R., Dong, X., and Lin, Y. (2019). Models for aircraft maintenance routing problem with consideration of remaining time and robustness. *Computers & Industrial Engineering*, 137:106045.

Deng, B., Guo, H., Li, J., Huang, J., Tang, K., and Li, W. (2022). A game-theoretic approach for the robust daily aircraft routing problem. *Journal of Mathematics*, 2022(1):8806135.

Desaulniers, G., Desrosiers, J., Dumas, Y., Solomon, M. M., and Soumis, F. (1997). Daily aircraft routing and scheduling. *Management Science*, 43(6):841–855.

Laporte, G., Mesa, J. A., and Perea, F. (2010). A game theoretic framework for the robust railway transit network design problem. *Transportation Research Part B: Methodological*, 44(4):447–459.

Levin, A. (1971). Scheduling and fleet routing models for transportation systems. *Transportation Science*, 5(3):232–255.

Lima, D. A., Contreras, J., and Padilha-Feltrin, A. (2008). A cooperative game theory analysis for transmission loss allocation. *Electric Power Systems Research*, 78(2):264–275.

Xu, Y., Wandelt, S., and Sun, X. (2021). Airline integrated robust scheduling with a variable neighborhood search based heuristic. *Transportation Research Part B: Methodological*, 149:181–203.