Comparative Analysis of Optimization Methods for the Airline Tail Assignment Problem

AionOS

November 11, 2025

1 Problem Definition

The problem at hand is a classic Tail Assignment Problem (TAP). Given a set of 5 flights (F1-F5) and 3 available aircraft tails (T1-T3) from `dataset.json`, the goal is to find an optimal assignment of each flight to exactly one tail.

This assignment must be physically possible, respecting all operational constraints:

- **Time Overlap:** A tail cannot be assigned to two flights that are in the air at the same time.
- **Location Mismatch:** A tail cannot fly a sequential route where the first flight's destination does not match the second flight's origin.
- **Turnaround Time:** A minimum turnaround time (45 or 50 minutes) must be respected between a tail's arrival and its next departure from the same airport.

2 Objectives

The optimization process aims to balance three competing objectives based on the cost and flight data:

1. **Minimize Cost:** The total operational cost, calculated as the sum of costs for each specific flight-to-tail assignment.
2. **Maximize Utilization:** Ensure aircraft workloads are balanced. This was measured in two different ways:
   - **PuLP (Imbalance):** Minimize the difference between the most-worked tail and the least-worked tail (Max - Min duration).
   - **Annealing (Variance):** Minimize the statistical variance of all tail workloads from the mean.
3. **Minimize Delay Risk:** Ensure schedules are robust. This was also measured differently:
   - **PuLP (Buffer):** Maximize the total buffer time (time $>$ min turnaround) across all connections. A high score is good.
   - **Annealing (Penalty):** Minimize a penalty score for "tight" connections (time $<$ min turnaround + buffer). A low score is good.

Two distinct computational methods were used to solve this problem.

3 Solution 1: Simulated Annealing (Heuristic)

This approach, from `annealing.ipynb`, used a quantum-inspired heuristic, Simulated Annealing, to find a low-energy (i.e., high-quality) solution for a QUBO (Quadratic Unconstrained Binary Optimization) model.

3.1 QUBO Formulation

The problem was mapped to an energy function H to be minimized:

$$H = (w_{\text{cost}} \cdot H_{\text{cost}}) + (w_{\text{util}} \cdot H_{\text{util}}) + (w_{\text{delay}} \cdot H_{\text{delay}}) + (\gamma_1 \cdot P_{\text{coverage}}) + (\gamma_2 \cdot P_{\text{incompatible}})$$

- H_{cost}, H_{util}, and H_{delay} represent the three objectives.
- P_{coverage} and $P_{\text{incompatible}}$ are high-energy penalties (γ) that punish invalid solutions (e.g., a flight not being assigned, or two flights conflicting).

3.2 Methodology: Weighted-Sum

This model used a flexible **Weighted-Sum** methodology. To find different solutions, the weights (w_{cost}, w_{util}, w_{delay}) were changed to prioritize different objectives in each scenario. This allows the solver to find a good *compromise* between all objectives simultaneously.

3.3 Annealing Results

The Simulated Annealer successfully found a **valid, feasible solution for all four scenarios**. The results are summarized from the `annealing_result.json` file. All solutions were confirmed as valid by the `check_incompatible` function.

Table 1: Valid Solutions from Simulated Annealing

Scenario	Cost	Util. Variance	Delay Penalty	Workloads (T1, T2, T3)
Cost_Focus	**585**	5550.0	225	(180, 225, 120)
Util_Focus	595	5550.0	225	(225, 180, 120)
Delay_Focus	620	5550.0	**0**	(120, 180, 225)
Balanced	595	5550.0	**0**	(225, 180, 120)

4 Solution 2: PuLP (Classical MIP)

This approach, from `pulp.ipynb`, used a classical exact solver, PuLP, to find a mathematically optimal solution for a Mixed Integer Program (MIP).

4.1 Model

The model was corrected (`pulp.ipynb` Cell 2) by adding a new constraint, `C_Incompatible`. This constraint explicitly forbids the solver from making any physically impossible assignments by using the `check_if_incompatible` helper function, making it a valid and correct model.

4.2 Methodology: Epsilon-Constraint

This model used a rigid **Epsilon-Constraint** methodology. It minimizes one primary objective (Cost) subject to *hard limits* (epsilons, ϵ) on the others.

$$\min(\text{Cost}) \quad \text{s.t.} \quad \text{Imbalance} \leq \epsilon_{\text{util}} \quad \text{and} \quad \text{Buffer} \geq \epsilon_{\text{delay}}$$

4.3 PuLP Results

The corrected PuLP model (Cell 2) was run. Unlike the annealer, it was **not** able to find a solution for all scenarios.

- **Cost_Focus:** Found a valid solution with a cost of 585, an imbalance of 105, and a buffer of 0.
- **Delay_Focus:** Found a valid solution with a cost of 585, an imbalance of 105, and a buffer of 45.
- **Util_Focus:** Returned **Infeasible**. The constraint Imbalance ≤ 90 was impossible to meet, as the true optimal imbalance is 105.
- **Balanced:** Returned **Infeasible**. The constraint Imbalance ≤ 100 was also impossible to meet.

5 Final Comparison and Conclusion

The final comparison includes only the **valid** solutions found by both corrected models.

Table 2: Final Comparison of Valid Solutions

Scenario	Solver	Status	Cost	Imbalance (Max-Min)	Delay Score
Cost_Focus	PuLP (Cell 2)	Valid	**585**	105	Buffer = 0
	Annealing	Valid	**585**	105	(Penalty = 225)
Util_Focus	PuLP (Cell 2)	**Infeasible**	—	(Limit was ≤ 90)	—
	Annealing	Valid	595	105	(Penalty = 225)
Delay_Focus	PuLP (Cell 2)	Valid	**585**	105	Buffer = 45
	Annealing	Valid	620	105	**(Penalty = 0)**
Balanced	PuLP (Cell 2)	**Infeasible**	—	(Limit was ≤ 100)	—
	Annealing	Valid	595	105	**(Penalty = 0)**

Note: PuLP's Delay Score (Buffer) and Annealing's (Penalty) are inverse. A Buffer of 45 (PuLP) is good, but a Penalty of 0 (Annealing) is perfect.

5.1 Conclusion: The Winner

The Annealing methodology is the clear winner for this problem.

While the fixed PuLP model is mathematically correct, its rigid "Epsilon-Constraint" method is brittle. By setting hard limits (e.g., Imbalance ≤ 90), it correctly reported "Infeasible" but failed to return any usable solution for the `Util_Focus` or `Balanced` scenarios.

The Annealing model's flexible "Weighted-Sum" approach was more robust. It was not locked into an impossible constraint and was therefore able to find a high-quality, valid solution for **all four scenarios**.

The best overall solution found by either method was the **Annealing 'Balanced'** solution:

- **Cost:** 595 (only 1.7% more than the absolute minimum of 585)
- **Imbalance:** 105 (the best possible imbalance)
- **Delay:** 0 (a perfect delay score)

This demonstrates the advantage of a flexible heuristic approach for exploring and finding a "good-enough" real-world compromise, especially when an exact, perfect solution may not exist under all rigid constraints.