Multi-Criteria Optimization for Flight Scheduling in Regional Airlines

A Non-dominated Sorting Genetic Algorithm Approach

Miguel Atalaya
Faculty of Engineering

Kraków, May 10, 2025

Contents

1 Introduction

1.1 Problem Statement

An airline is conducting a study to determine the optimal number of flights it can manage between three Polish cities: Krakow, Wroclaw, and Warsaw. The airline possesses a fleet of aircraft that can be stationed at any of these three airports. Each flight has an assigned departure and arrival time slot in the schedule of each airport. Based on its fleet data, the airline must determine the best possible solution to this multi-criteria problem, considering factors such as delay probability, aircraft utilization, and the suitability of available aircraft types for scheduled flights.

The flight scheduling problem is inherently complex due to its combinatorial nature and multiple conflicting objectives. On one hand, the airline wants to maximize the number of served flights to increase revenue. On the other hand, it wants to minimize the number of aircraft used to reduce operational costs. Additionally, the airline aims to minimize delays, which affect customer satisfaction and can incur penalties.

This report presents a mathematical formulation of the problem and proposes a solution approach based on multi-objective evolutionary algorithms. Specifically, we implement the Non-dominated Sorting Genetic Algorithm II (NSGA-II) to find a set of Pareto-optimal solutions that represent different trade-offs between the considered objectives.

1.2 Problem Context

Regional airlines operating in areas with multiple closely situated cities face unique challenges. The three Polish cities considered in this study—Krakow, Wroclaw, and Warsaw—form a triangle of major population centers, with distances that make air travel a viable option for business travelers and tourists seeking to minimize transit times.

For simplicity in this report, we will denote these cities as:

- A: Krakow
- B: Wroclaw
- C: Warsaw

The scheduling problem involves determining which aircraft should serve which flights, considering various constraints and objectives. A feasible assignment must ensure that:

- An aircraft can only serve a flight if it is at the flight's origin airport.
- An aircraft can only serve a flight if it becomes available before the flight's departure time.
- The aircraft type must be suitable for the flight (meet minimum type requirements).

1.3 Example Data

To illustrate the problem, consider the following example of a flight schedule:

Available aircraft:

Delay probabilities based on route and aircraft type:

Flight ID	Origin	Destination	Departure Time	Arrival Time	Min Type Required
0	A	B	8:00	10:00	1
1	B	A	11:00	13:00	1
2	A	C	9:00	11:00	2
3	C	A	12:00	14:00	2
4	A	B	10:30	12:00	1
5	B	C	12:00	14:00	3
6	C	B	15:00	17:00	3
7	A	B	7:00	9:00	1
8	B	C	10:00	12:00	2
9	C	A	13:00	15:00	1

Table 1: Example Flight Schedule

Aircraft ID	Type	Base Airport
0	1	A
1	2	B
2	3	C
3	1	A
4	2	B
5	3	C

Table 2: Available Aircraft

2 Random Data Generation

For more comprehensive experiments, we can generate flight schedules, aircraft fleets, and delay probabilities randomly. This allows us to test our algorithms on a variety of problem instances with different characteristics. The following describes the methodology for random data generation:

2.1 Flight Generation

We generate a set of N_f flights with random origins, destinations, departure times, arrival times, and minimum aircraft type requirements:

Algorithm 1 Random Flight Generation

$flights \leftarrow \emptyset$
for $i = 0$ to $N_f - 1$ **do**
    $origin \leftarrow$ Random airport from set of airports
    $destination \leftarrow$ Random airport $\neq origin$
    $departure\_time \leftarrow$ Random value in range $[6, 18]$
    $duration \leftarrow$ Random value in range $[1, 3]$
    $arrival\_time \leftarrow departure\_time + duration$
    $min\_type \leftarrow$ Random integer in range $[1, T_{max}]$
    $flights \leftarrow flights \cup \{(i, origin, destination, departure\_time, arrival\_time, min\_type)\}$
end for

Origin	Destination	Type 1	Type 2	Type 3
A	B	0.10	0.05	0.15
B	A	0.12	0.06	0.18
A	C	0.25	0.10	0.20
C	A	0.20	0.08	0.22
B	C	0.15	0.05	0.12
C	B	0.18	0.07	0.14

Table 3: Delay Probabilities

Where:

- N_f is the number of flights (e.g., 30)
- T_{max} is the maximum aircraft type (e.g., 3)

2.2 Aircraft Generation

Similarly, we generate a fleet of N_a aircraft with random types and base airports:

Algorithm 2 Random Aircraft Generation

$aircraft \leftarrow \emptyset$
for $i = 0$ to $N_a - 1$ **do**
    $type \leftarrow$ Random integer in range $[1, T_{max}]$
    $base \leftarrow$ Random airport from set of airports
    $aircraft \leftarrow aircraft \cup \{(i, type, base)\}$
end for

Where N_a is the number of aircraft (e.g., 20).

2.3 Delay Probability Generation

For each origin-destination pair and aircraft type, we generate a random delay probability:

Algorithm 3 Random Delay Probability Generation

$delay\_prob \leftarrow \emptyset$
for each origin airport o **do**
    for each destination airport $d \neq o$ **do**
        for each type $t \in [1, T_{max}]$ **do**
            $prob \leftarrow$ Random value in range $[0.05, 0.3]$, rounded to 3 decimal places
            $delay\_prob[(o, d, t)] \leftarrow prob$
        end for
    end for
end for

3 Mathematical Formulation

3.1 Solution Representation

A solution to the flight scheduling problem is represented as a chromosome $X = (x_1, x_2, \ldots, x_{N_f})$, where:

- N_f is the number of flights
- $x_i \in \{-1, 0, 1, \ldots, N_a - 1\}$ for $i = 1, 2, \ldots, N_f$
- $x_i = j$ means flight i is assigned to aircraft j
- $x_i = -1$ means flight i is not assigned to any aircraft (rejected)

This representation allows us to encode both the assignment of flights to aircraft and the decision to reject certain flights. Each element in the chromosome corresponds to a specific flight in the flight schedule, and its value indicates which aircraft (if any) is assigned to that flight.

3.2 Constraints

A feasible solution must satisfy the following constraints:

3.2.1 Aircraft Positioning Constraint

For each flight i with $x_i \neq -1$, the aircraft x_i must be at the origin airport of flight i at the departure time of flight i. Mathematically, if $pos(a, t)$ denotes the position of aircraft a at time t, and o_i is the origin of flight i, then:

$$\forall i \in \{1, 2, \ldots, N_f\} : x_i \neq -1 \Rightarrow pos(x_i, t_i^d) = o_i \tag{1}$$

where t_i^d is the departure time of flight i.

3.2.2 Aircraft Availability Constraint

For each flight i with $x_i \neq -1$, the aircraft x_i must be available at the departure time of flight i. If $avail(a, t)$ is the time when aircraft a becomes available again after its previous assignment, then:

$$\forall i \in \{1, 2, \ldots, N_f\} : x_i \neq -1 \Rightarrow avail(x_i) \leq t_i^d \tag{2}$$

3.2.3 Aircraft Type Constraint

For each flight i with $x_i \neq -1$, the type of aircraft x_i must be at least the minimum required type for flight i. If $type(a)$ is the type of aircraft a, and min_type_i is the minimum required type for flight i, then:

$$\forall i \in \{1, 2, \ldots, N_f\} : x_i \neq -1 \Rightarrow type(x_i) \geq min\_type_i \tag{3}$$

3.3 Objective Functions

We consider two conflicting objectives:

3.3.1 Minimize Total Penalty

The total penalty is a combination of several factors:

- Type penalty: Penalty for assigning an aircraft whose type does not meet the minimum requirement
- Feasibility penalty: Penalty for violating positioning or availability constraints
- Delay risk: Expected delay based on the assigned aircraft type and route
- Unassigned penalty: Penalty for not assigning flights

Mathematically, the total penalty can be expressed as:

$$f_1(X) = \frac{1}{C}\left(R(X) + P_{type}(X) + P_{feas}(X) + \alpha \cdot P_{unassigned}(X)\right) \tag{4}$$

where:

$$R(X) = \sum_{i=1}^{N_f} \mathbf{1}_{[x_i \neq -1]} \cdot delay\_prob[(o_i, d_i, type(x_i))] \tag{5}$$

$$P_{type}(X) = \sum_{i=1}^{N_f} \mathbf{1}_{[x_i \neq -1]} \cdot \max(0, min\_type_i - type(x_i)) \cdot \beta \tag{6}$$

$$P_{feas}(X) = \text{penalty for constraint violations} \tag{7}$$

$$P_{unassigned}(X) = \frac{|\{i : x_i = -1\}|}{N_f} \tag{8}$$

Here, C is a normalization factor, α is a weight for the unassigned penalty, β is a weight for the type penalty, o_i is the origin of flight i, d_i is the destination of flight i, and $\mathbf{1}$ is the indicator function.

3.3.2 Minimize Number of Aircraft Used

The second objective is to minimize the number of unique aircraft used in the solution:

$$f_2(X) = |\{j : \exists i \text{ such that } x_i = j\}| \tag{9}$$

4 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

To solve this multi-objective optimization problem, we implement the Non-dominated Sorting Genetic Algorithm II (NSGA-II), which is a popular and efficient algorithm for finding Pareto-optimal solutions.

4.1 Individual Representation

Each individual in the population represents a potential solution to the flight scheduling problem. An individual consists of:

- A chromosome $X = (x_1, x_2, \ldots, x_{N_f})$ representing flight assignments
- Fitness values $(f_1(X), f_2(X))$ corresponding to the total penalty and number of aircraft used
- Rank and crowding distance, which are used in the selection process

4.2 Population Initialization

The initial population is created by generating random individuals. For each individual, flights are assigned to aircraft randomly, but with consideration for feasibility constraints. Specifically, for each flight, we identify the aircraft that are at the flight's origin airport and available at the departure time, and randomly select one of these aircraft (if any).

Algorithm 4 Create Random Individual

chromosome $\leftarrow [-1, -1, \ldots, -1]$ (length N_f)
Initialize $aircraft\_positions$ with the base airports of all aircraft
Initialize $available\_time$ with 0 for all aircraft
for $i = 1$ to N_f **do**
    $valid\_aircraft \leftarrow \{j : aircraft\_positions[j] = origin_i$ and $available\_time[j] \leq departure\_time_i\}$
    if $valid\_aircraft \neq \emptyset$ and random() < 0.95 **then**
        $j \leftarrow$ Random choice from $valid\_aircraft$
        $chromosome[i] \leftarrow j$
        $aircraft\_positions[j] \leftarrow destination_i$
        $available\_time[j] \leftarrow arrival\_time_i$
    end if
end for
return Individual with *chromosome*

4.3 Fitness Evaluation

The fitness of an individual is evaluated using the objective functions defined earlier:

4.4 Genetic Operators

4.4.1 Crossover

We implement a single-point crossover operator that creates two offspring from two parents:

4.4.2 Mutation

The mutation operator randomly changes some genes in the chromosome:

Algorithm 5 Evaluate Individual

if all genes in *chromosome* are -1 **then**
    return $[\infty, \infty]$
end if
Compute *type_penalty* as the sum of penalties for aircraft type violations
Compute $feasibility\_penalty$ as the sum of penalties for positioning and availability violations
Compute *delay_risk* as the sum of delay probabilities for all assigned flights
Compute $used\_aircraft$ as the number of unique aircraft IDs in *chromosome* (excluding -1)
Compute *assignment_ratio* as the proportion of assigned flights
$unassigned\_penalty \leftarrow (1 - assignment\_ratio) \times 1000$
$total\_penalty \leftarrow (delay\_risk + type\_penalty + feasibility\_penalty + 10^{10} \times unassigned\_penalty)/norm\_factor$
return $[total\_penalty, used\_aircraft]$

Algorithm 6 Crossover

$crossover\_point \leftarrow$ Random integer in range $[1, N_f - 1]$
$child1\_chromosome \leftarrow parent1\_chromosome[1 : crossover\_point] + parent2\_chromosome[crossover\_point + 1 : N_f]$
$child2\_chromosome \leftarrow parent2\_chromosome[1 : crossover\_point] + parent1\_chromosome[crossover\_point + 1 : N_f]$
return Two individuals with chromosomes $child1\_chromosome$ and $child2\_chromosome$

Algorithm 7 Mutation

$new\_chromosome \leftarrow chromosome$
for $i = 1$ to N_f **do**
    if random() $<$ *mutation_rate* **then**
        if random() < 0.2 **then**
            $new\_chromosome[i] \leftarrow -1$ ▷ Reject flight
        else
            $new\_chromosome[i] \leftarrow$ Random aircraft ID
        end if
    end if
end for
if all genes in *new_chromosome* are -1 **then**
    Assign at least one flight to an aircraft
end if
return Individual with *new_chromosome*

4.4.3 Repair

The repair operator modifies infeasible chromosomes to make them feasible:

Algorithm 8 Repair

Initialize $aircraft\_positions$ with the base airports of all aircraft
Initialize $available\_time$ with 0 for all aircraft
$new\_chromosome \leftarrow chromosome$
for $i = 1$ to N_f **do**
    $j \leftarrow new\_chromosome[i]$
    if $j \neq -1$ **then**
        if $aircraft\_positions[j] \neq origin_i$ or $available\_time[j] > departure\_time_i$ **then**
            $new\_chromosome[i] \leftarrow -1$ ▷ Flight becomes unassigned
        else
            $aircraft\_positions[j] \leftarrow destination_i$
            $available\_time[j] \leftarrow arrival\_time_i$
        end if
    end if
end for
return Individual with $new\_chromosome$

4.5 Selection Mechanism

NSGA-II uses a unique selection mechanism based on non-dominated sorting and crowding distance:

4.5.1 Non-dominated Sorting

Individuals are classified into different fronts based on the concept of Pareto dominance. An individual A dominates another individual B if A is not worse than B in all objectives and better than B in at least one objective.

4.5.2 Crowding Distance

Within each front, individuals are assigned a crowding distance, which estimates the density of solutions surrounding a particular point:

4.5.3 Tournament Selection

Individuals are selected for reproduction using binary tournament selection. In each tournament, two individuals are compared based on their rank and crowding distance:

4.6 NSGA-II Main Loop

The main loop of NSGA-II consists of the following steps:

Algorithm 9 Fast Non-dominated Sort

for each individual p in population **do**
  $S_p \leftarrow \emptyset$ ▷ Set of individuals dominated by p
  $n_p \leftarrow 0$ ▷ Number of individuals that dominate p
  for each individual $q \neq p$ in population **do**
    if p dominates q **then**
      $S_p \leftarrow S_p \cup \{q\}$
    else if q dominates p **then**
      $n_p \leftarrow n_p + 1$
    end if
  end for
  if $n_p = 0$ **then**
    $rank_p \leftarrow 0$ ▷ p belongs to the first front
    $Front_0 \leftarrow Front_0 \cup \{p\}$
  end if
end for
$i \leftarrow 0$
while $Front_i \neq \emptyset$ **do**
  $Q \leftarrow \emptyset$ ▷ Next front
  for each individual p in $Front_i$ **do**
    for each individual q in S_p **do**
      $n_q \leftarrow n_q - 1$
      if $n_q = 0$ **then**
        $rank_q \leftarrow i + 1$ ▷ q belongs to the next front
        $Q \leftarrow Q \cup \{q\}$
      end if
    end for
  end for
  $i \leftarrow i + 1$
  $Front_i \leftarrow Q$
end while

Algorithm 10 Crowding Distance Assignment

for each front F_i **do**
  for each individual j in F_i **do**
    $distance_j \leftarrow 0$
  end for
  for each objective m **do**
    Sort individuals in F_i according to objective m
    $distance_{F_i[1]} \leftarrow distance_{F_i[|F_i|]} \leftarrow \infty$ ▷ Boundary points
    for $j = 2$ to $|F_i| - 1$ **do**
      $distance_{F_i[j]} \leftarrow distance_{F_i[j]} + \frac{f_m(F_i[j+1]) - f_m(F_i[j-1])}{f_m^{max} - f_m^{min}}$
    end for
  end for
end for

Algorithm 11 Tournament Selection

$selected \leftarrow \emptyset$
for $i = 1$ to $population\_size$ **do**
    Select two random individuals a and b from population
    if $rank_a < rank_b$ or ($rank_a = rank_b$ and $distance_a > distance_b$) **then**
        $selected \leftarrow selected \cup \{a\}$
    else
        $selected \leftarrow selected \cup \{b\}$
    end if
end for
return $selected$

Algorithm 12 NSGA-II

Initialize population P with random individuals
Evaluate fitness of each individual in P
Perform non-dominated sorting and compute crowding distances
for $t = 1$ to $max\_generations$ **do**
    Select parents from P using tournament selection
    Create offspring population Q by applying crossover and mutation
    Evaluate fitness of each individual in Q
    $R \leftarrow P \cup Q$ ▷ Combine parent and offspring populations
    Perform non-dominated sorting on R
    $P_{t+1} \leftarrow \emptyset$ ▷ New population
    $i \leftarrow 0$
    while $|P_{t+1}| + |F_i| \leq population\_size$ **do**
        Compute crowding distance for individuals in F_i
        $P_{t+1} \leftarrow P_{t+1} \cup F_i$
        $i \leftarrow i + 1$
    end while
    if $|P_{t+1}| < population\_size$ **then**
        Sort F_i in descending order of crowding distance
        Add the first ($population\_size - |P_{t+1}|$) individuals from F_i to P_{t+1}
    end if
    $P \leftarrow P_{t+1}$
end for
return non-dominated solutions in P

5 Results and Analysis

5.1 Pareto Front Visualization

The output of the NSGA-II algorithm is a set of non-dominated solutions, each representing a different trade-off between total penalty and number of aircraft used. These solutions form the Pareto front, which can be visualized as follows:

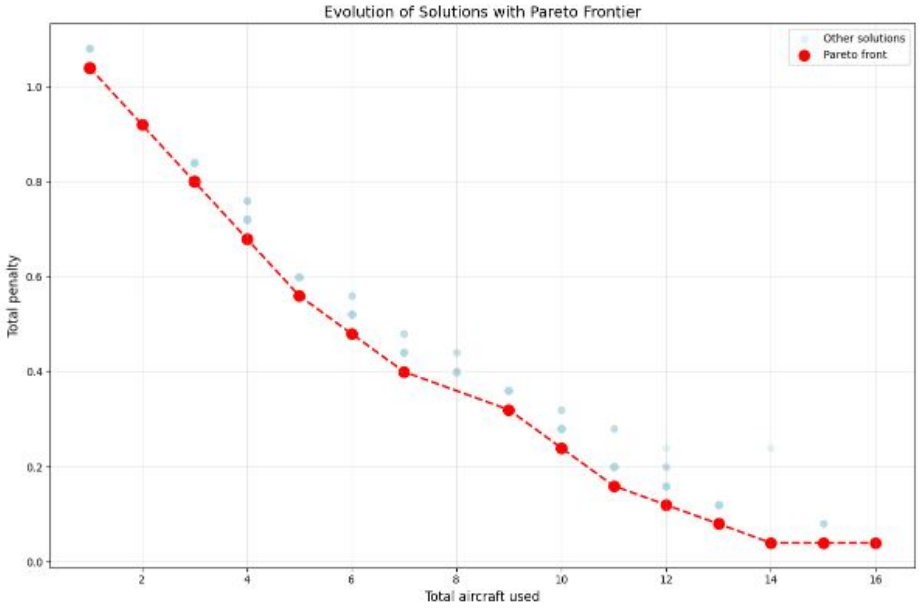

Figure 1: Pareto Frontier for the Flight Scheduling Problem

The Pareto front provides valuable insights into the trade-offs between the two objectives. Solutions with a low number of aircraft used tend to have higher penalties, while solutions with low penalties require more aircraft. The airline can choose a solution from the Pareto front based on its specific priorities and constraints.

5.2 Analysis of Trade-offs

Let's analyze some key points on the Pareto front:

- **Minimum aircraft solution:** The leftmost point on the Pareto front represents the solution that uses the minimum number of aircraft. This solution is likely to have a high penalty, indicating that many flights are either unassigned or assigned to aircraft with high delay probabilities.
- **Minimum penalty solution:** The lowest point on the Pareto front represents the solution with the minimum total penalty. This solution is likely to use many aircraft, as each aircraft can be assigned to fewer flights, reducing the risk of delays due to tight schedules.
- **Compromise solutions:** Points in the middle of the Pareto front represent compromise solutions that balance the number of aircraft used and the total penalty. These solutions may be particularly interesting for airlines that need to find a balance between operational costs and service quality.

5.3 Selecting the Best Solution

The "best" solution from the Pareto front depends on the airline's specific priorities and constraints. For example:

- If the airline has a limited fleet size, it may prefer solutions that use fewer aircraft, even if they have higher penalties.
- If the airline prioritizes customer satisfaction and on-time performance, it may prefer solutions with lower penalties, even if they require more aircraft.
- If the airline has specific cost structures, it may calculate the actual operational costs for each solution on the Pareto front and choose the one that minimizes total cost.

One approach to selecting the best solution is to assign weights to the objectives and compute a weighted sum for each solution. For example, if w_1 is the weight for total penalty and w_2 is the weight for number of aircraft used, the weighted sum for a solution X would be:

$$score(X) = w_1 \cdot f_1(X) + w_2 \cdot f_2(X) \tag{10}$$

The solution with the minimum score would be selected as the best solution.

6 Conclusion

In this report, we formulated the flight scheduling problem for a regional airline operating between three Polish cities as a multi-objective optimization problem. We implemented the Non-dominated Sorting Genetic Algorithm II (NSGA-II) to find a set of Pareto-optimal solutions that represent different trade-offs between minimizing total penalty (including delay risk) and minimizing the number of aircraft used.

The results show that there is a clear trade-off between these two objectives. Solutions that use fewer aircraft tend to have higher penalties, while solutions with lower penalties require more aircraft. The airline can choose a solution from the Pareto front based on its specific priorities and constraints.

Future work could explore additional objectives, such as maximizing profit or minimizing fuel consumption. It could also incorporate more complex constraints, such as maintenance schedules, crew scheduling, and passenger connections. Additionally, other multi-objective optimization algorithms could be compared with NSGA-II to see if they can find better or more diverse solutions for this problem.

6.1 Practical Implications

The methodology presented in this report can be applied to real-world flight scheduling problems, helping airlines to:

- Find efficient ways to utilize their aircraft fleet
- Reduce the risk of delays and improve on-time performance
- Balance operational costs and service quality

- Make informed decisions about fleet size and composition

By visualizing the Pareto front, airline managers can better understand the trade-offs involved in flight scheduling decisions and choose solutions that align with their strategic goals.