

university\_logo.jpg

# Multi-Criteria Optimization for Flight Scheduling in Regional Airlines

A Non-dominated Sorting Genetic Algorithm Approach

Department of Operations Research  
Faculty of Engineering

May 10, 2025

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Problem Statement . . . . .	3
1.2	Problem Context . . . . .	3
1.3	Example Data . . . . .	3
<b>2</b>	<b>Random Data Generation</b>	<b>4</b>
2.1	Flight Generation . . . . .	4
2.2	Aircraft Generation . . . . .	5
2.3	Delay Probability Generation . . . . .	5
<b>3</b>	<b>Mathematical Formulation</b>	<b>6</b>
3.1	Solution Representation . . . . .	6
3.2	Constraints . . . . .	6
3.2.1	Aircraft Positioning Constraint . . . . .	6
3.2.2	Aircraft Availability Constraint . . . . .	6
3.2.3	Aircraft Type Constraint . . . . .	6
3.3	Objective Functions . . . . .	6
3.3.1	Minimize Total Penalty . . . . .	7
3.3.2	Minimize Number of Aircraft Used . . . . .	7
<b>4</b>	<b>Non-dominated Sorting Genetic Algorithm II (NSGA-II)</b>	<b>7</b>
4.1	Individual Representation . . . . .	8
4.2	Population Initialization . . . . .	8
4.3	Fitness Evaluation . . . . .	8
4.4	Genetic Operators . . . . .	8
4.4.1	Crossover . . . . .	8
4.4.2	Mutation . . . . .	8
4.4.3	Repair . . . . .	10
4.5	Selection Mechanism . . . . .	10
4.5.1	Non-dominated Sorting . . . . .	10
4.5.2	Crowding Distance . . . . .	10
4.5.3	Tournament Selection . . . . .	10
4.6	NSGA-II Main Loop . . . . .	10
<b>5</b>	<b>Results and Analysis</b>	<b>13</b>
5.1	Pareto Front Visualization . . . . .	13
5.2	Analysis of Trade-offs . . . . .	13
5.3	Selecting the Best Solution . . . . .	14
<b>6</b>	<b>Conclusion</b>	<b>14</b>
6.1	Practical Implications . . . . .	15

# 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

---

```

1:  $flights \leftarrow \emptyset$ 
2: for  $i = 0$  to  $N_f - 1$  do
3:    $origin \leftarrow$  Random airport from set of airports
4:    $destination \leftarrow$  Random airport  $\neq origin$ 
5:    $departure\_time \leftarrow$  Random value in range  $[6, 18]$ 
6:    $duration \leftarrow$  Random value in range  $[1, 3]$ 
7:    $arrival\_time \leftarrow departure\_time + duration$ 
8:    $min\_type \leftarrow$  Random integer in range  $[1, T_{max}]$ 
9:    $flights \leftarrow flights \cup \{(i, origin, destination, departure\_time, arrival\_time, min\_type)\}$ 
10: 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

---

```

1: aircraft  $\leftarrow \emptyset$ 
2: for  $i = 0$  to  $N_a - 1$  do
3:   type  $\leftarrow$  Random integer in range  $[1, T_{max}]$ 
4:   base  $\leftarrow$  Random airport from set of airports
5:   aircraft  $\leftarrow$  aircraft  $\cup \{(i, type, base)\}$ 
6: 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

---

```

1: delay_prob  $\leftarrow \emptyset$ 
2: for each origin airport  $o$  do
3:   for each destination airport  $d \neq o$  do
4:     for each type  $t \in [1, T_{max}]$  do
5:       prob  $\leftarrow$  Random value in range  $[0.05, 0.3]$ , rounded to 3 decimal places
6:       delay_prob $[(o, d, t)] \leftarrow prob$ 
7:     end for
8:   end for
9: end for
```

---

### 3 Mathematical Formulation

#### 3.1 Solution Representation

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

- $N_f$  is the number of flights
- $x_i \in \{-1, 0, 1, \dots, N_a - 1\}$  for  $i = 1, 2, \dots, N_f$
- $x_i = j$  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:

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

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:

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

##### 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:

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

#### 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:

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

where:

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

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

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

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

Here,  $C$  is a normalization factor,  $\alpha$  is a weight for the unassigned penalty,  $\beta$  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:

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

## 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  $X = (x_1, x_2, \dots, x_{N_f})$  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

---

```

1: chromosome  $\leftarrow [-1, -1, \dots, -1]$  (length  $N_f$ )
2: Initialize aircraft_positions with the base airports of all aircraft
3: Initialize available_time with 0 for all aircraft
4: for  $i = 1$  to  $N_f$  do
5:   valid_aircraft  $\leftarrow \{j : \text{aircraft\_positions}[j] = \text{origin}_i \text{ and } \text{available\_time}[j] \leq$ 
     departure\_time $\}_i$ 
6:   if valid_aircraft  $\neq \emptyset$  and  $\text{random}() < 0.95$  then
7:      $j \leftarrow$  Random choice from valid_aircraft
8:     chromosome $[i] \leftarrow j$ 
9:     aircraft_positions $[j] \leftarrow \text{destination}_i$ 
10:    available_time $[j] \leftarrow \text{arrival\_time}_i$ 
11:   end if
12: end for
13: 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

---

```

1: if all genes in chromosome are  $-1$  then
2:   return  $[\infty, \infty]$ 
3: end if
4: Compute type_penalty as the sum of penalties for aircraft type violations
5: Compute feasibility_penalty as the sum of penalties for positioning and availability violations
6: Compute delay_risk as the sum of delay probabilities for all assigned flights
7: Compute used_aircraft as the number of unique aircraft IDs in chromosome (excluding  $-1$ )
8: Compute assignment_ratio as the proportion of assigned flights
9: unassigned_penalty  $\leftarrow (1 - \text{assignment\_ratio}) \times 1000$ 
10: total_penalty  $\leftarrow (\text{delay\_risk} + \text{type\_penalty} + \text{feasibility\_penalty} + 10^{10} \times \text{unassigned\_penalty}) / \text{norm\_factor}$ 
11: return [total_penalty, used_aircraft]

```

---



---

**Algorithm 6** Crossover

---

```

1: crossover_point  $\leftarrow$  Random integer in range  $[1, N_f - 1]$ 
2: child1_chromosome  $\leftarrow$  parent1_chromosome[1 : crossover_point] + parent2_chromosome[crossover_point + 1 :  $N_f$ ]
3: child2_chromosome  $\leftarrow$  parent2_chromosome[1 : crossover_point] + parent1_chromosome[crossover_point + 1 :  $N_f$ ]
4: return Two individuals with chromosomes child1_chromosome and child2_chromosome

```

---



---

**Algorithm 7** Mutation

---

```

1: new_chromosome  $\leftarrow$  chromosome
2: for  $i = 1$  to  $N_f$  do
3:   if  $\text{random}() < \text{mutation\_rate}$  then
4:     if  $\text{random}() < 0.2$  then
5:       new_chromosome[ $i$ ]  $\leftarrow -1$  ▷ Reject flight
6:     else
7:       new_chromosome[ $i$ ]  $\leftarrow$  Random aircraft ID
8:     end if
9:   end if
10: end for
11: if all genes in new_chromosome are  $-1$  then
12:   Assign at least one flight to an aircraft
13: end if
14: return Individual with new_chromosome

```

---

### 4.4.3 Repair

The repair operator modifies infeasible chromosomes to make them feasible:

---

**Algorithm 8** Repair
 

---

```

1: Initialize aircraft_positions with the base airports of all aircraft
2: Initialize available_time with 0 for all aircraft
3: new_chromosome  $\leftarrow$  chromosome
4: for  $i = 1$  to  $N_f$  do
5:    $j \leftarrow \text{new\_chromosome}[i]$ 
6:   if  $j \neq -1$  then
7:     if aircraft_positions[ $j$ ]  $\neq$  origin $i$  or available_time[ $j$ ]  $>$  departure_time $i$ 
       then
8:       new_chromosome[ $i$ ]  $\leftarrow$   $-1$   $\triangleright$  Flight becomes unassigned
9:     else
10:      aircraft_positions[ $j$ ]  $\leftarrow$  destination $i$ 
11:      available_time[ $j$ ]  $\leftarrow$  arrival_time $i$ 
12:    end if
13:  end if
14: end for
15: 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

---

```

1: for each individual  $p$  in population do
2:    $S_p \leftarrow \emptyset$  ▷ Set of individuals dominated by  $p$ 
3:    $n_p \leftarrow 0$  ▷ Number of individuals that dominate  $p$ 
4:   for each individual  $q \neq p$  in population do
5:     if  $p$  dominates  $q$  then
6:        $S_p \leftarrow S_p \cup \{q\}$ 
7:     else if  $q$  dominates  $p$  then
8:        $n_p \leftarrow n_p + 1$ 
9:     end if
10:  end for
11:  if  $n_p = 0$  then
12:     $rank_p \leftarrow 0$  ▷  $p$  belongs to the first front
13:     $Front_0 \leftarrow Front_0 \cup \{p\}$ 
14:  end if
15: end for
16:  $i \leftarrow 0$ 
17: while  $Front_i \neq \emptyset$  do
18:    $Q \leftarrow \emptyset$  ▷ Next front
19:   for each individual  $p$  in  $Front_i$  do
20:     for each individual  $q$  in  $S_p$  do
21:        $n_q \leftarrow n_q - 1$ 
22:       if  $n_q = 0$  then
23:          $rank_q \leftarrow i + 1$  ▷  $q$  belongs to the next front
24:          $Q \leftarrow Q \cup \{q\}$ 
25:       end if
26:     end for
27:   end for
28:    $i \leftarrow i + 1$ 
29:    $Front_i \leftarrow Q$ 
30: end while

```

---

**Algorithm 10** Crowding Distance Assignment

---

```

1: for each front  $F_i$  do
2:   for each individual  $j$  in  $F_i$  do
3:      $distance_j \leftarrow 0$ 
4:   end for
5:   for each objective  $m$  do
6:     Sort individuals in  $F_i$  according to objective  $m$ 
7:      $distance_{F_i[1]} \leftarrow distance_{F_i[|F_i|]} \leftarrow \infty$  ▷ Boundary points
8:     for  $j = 2$  to  $|F_i| - 1$  do
9:        $distance_{F_i[j]} \leftarrow distance_{F_i[j]} + \frac{f_m(F_i[j+1]) - f_m(F_i[j-1]))}{f_m^{max} - f_m^{min}}$ 
10:    end for
11:  end for
12: end for

```

---

---

**Algorithm 11** Tournament Selection

---

```

1:  $selected \leftarrow \emptyset$ 
2: for  $i = 1$  to  $population\_size$  do
3:   Select two random individuals  $a$  and  $b$  from population
4:   if  $rank_a < rank_b$  or  $(rank_a = rank_b \text{ and } distance_a > distance_b)$  then
5:      $selected \leftarrow selected \cup \{a\}$ 
6:   else
7:      $selected \leftarrow selected \cup \{b\}$ 
8:   end if
9: end for
10: return  $selected$ 

```

---



---

**Algorithm 12** NSGA-II

---

```

1: Initialize population  $P$  with random individuals
2: Evaluate fitness of each individual in  $P$ 
3: Perform non-dominated sorting and compute crowding distances
4: for  $t = 1$  to  $max\_generations$  do
5:   Select parents from  $P$  using tournament selection
6:   Create offspring population  $Q$  by applying crossover and mutation
7:   Evaluate fitness of each individual in  $Q$ 
8:    $R \leftarrow P \cup Q$  ▷ Combine parent and offspring populations
9:   Perform non-dominated sorting on  $R$ 
10:   $P_{t+1} \leftarrow \emptyset$  ▷ New population
11:   $i \leftarrow 0$ 
12:  while  $|P_{t+1}| + |F_i| \leq population\_size$  do
13:    Compute crowding distance for individuals in  $F_i$ 
14:     $P_{t+1} \leftarrow P_{t+1} \cup F_i$ 
15:     $i \leftarrow i + 1$ 
16:  end while
17:  if  $|P_{t+1}| < population\_size$  then
18:    Sort  $F_i$  in descending order of crowding distance
19:    Add the first  $(population\_size - |P_{t+1}|)$  individuals from  $F_i$  to  $P_{t+1}$ 
20:  end if
21:   $P \leftarrow P_{t+1}$ 
22: end for
23: 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:

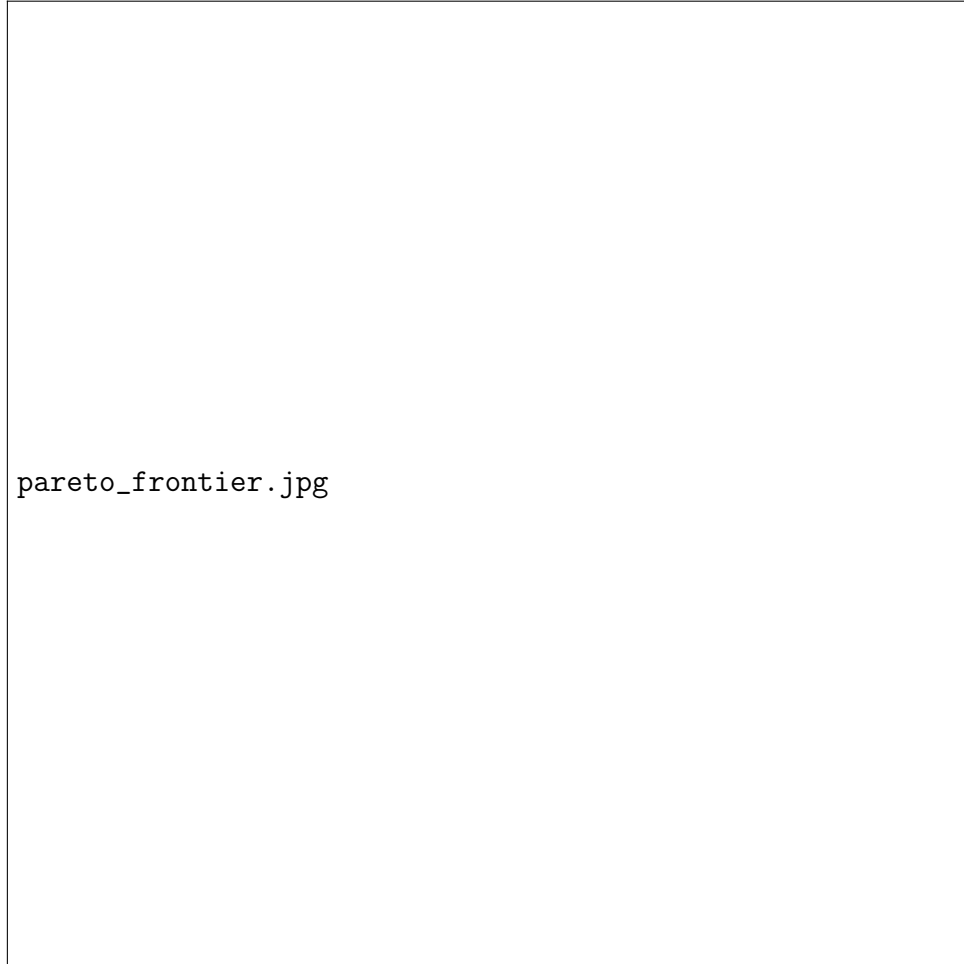


Figure 1: Pareto Frontier for the Flight Scheduling Problem

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:

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

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.