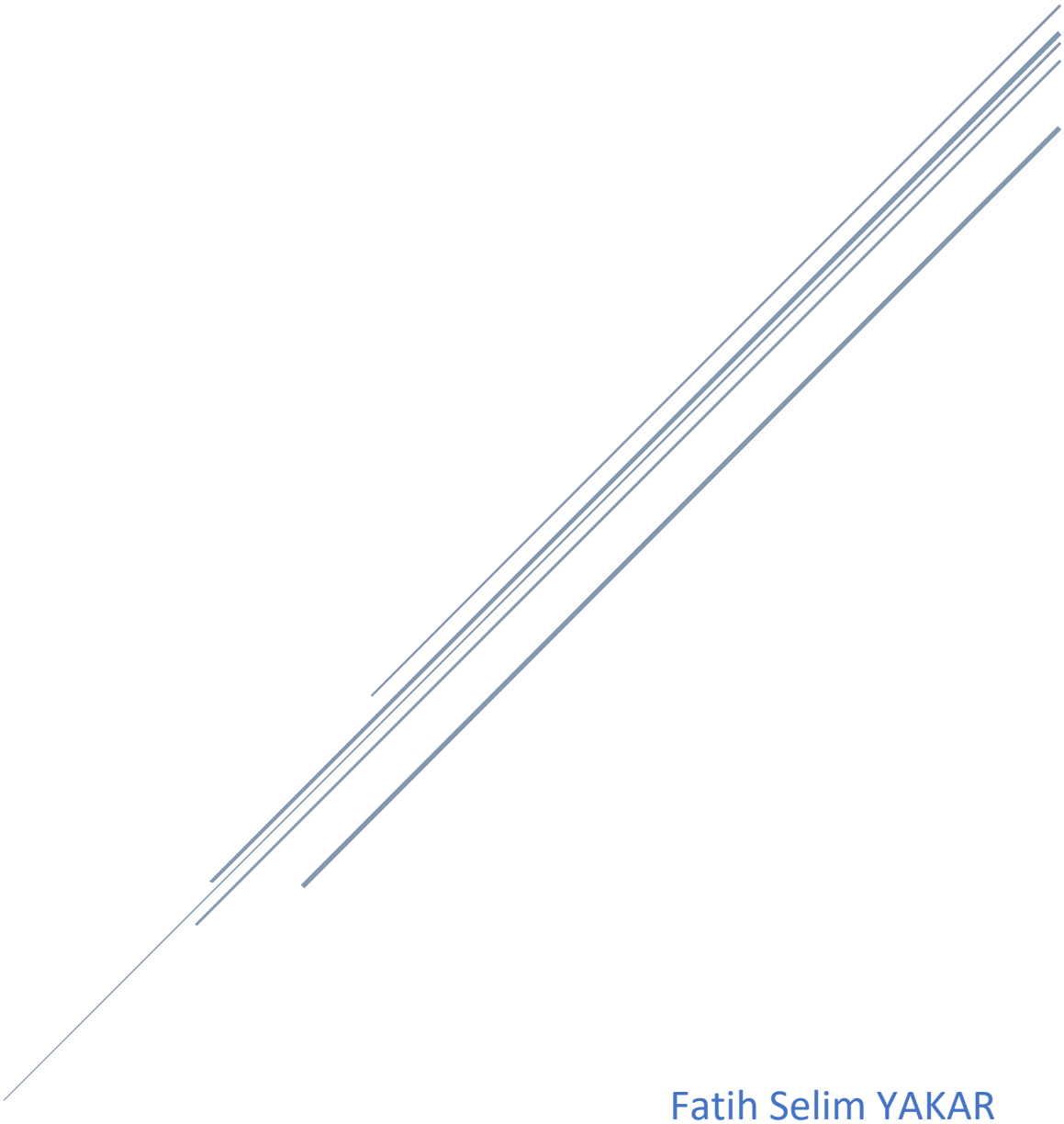
CSE 424 OPTIMIZATION

Aircraft Taxi Route Scheduling Project Report

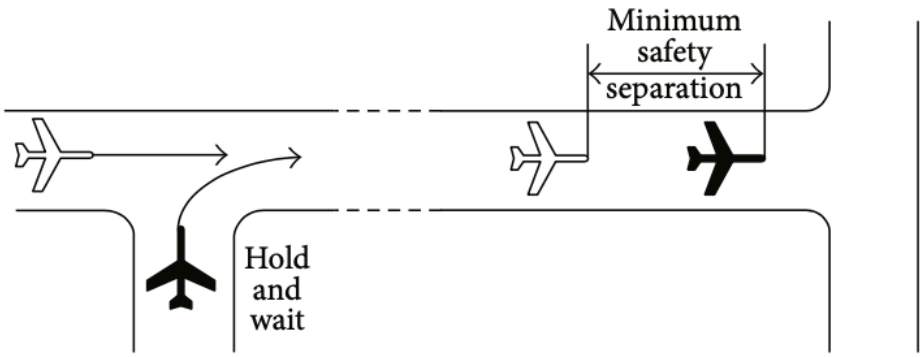


Fatih Selim YAKAR
1601044054

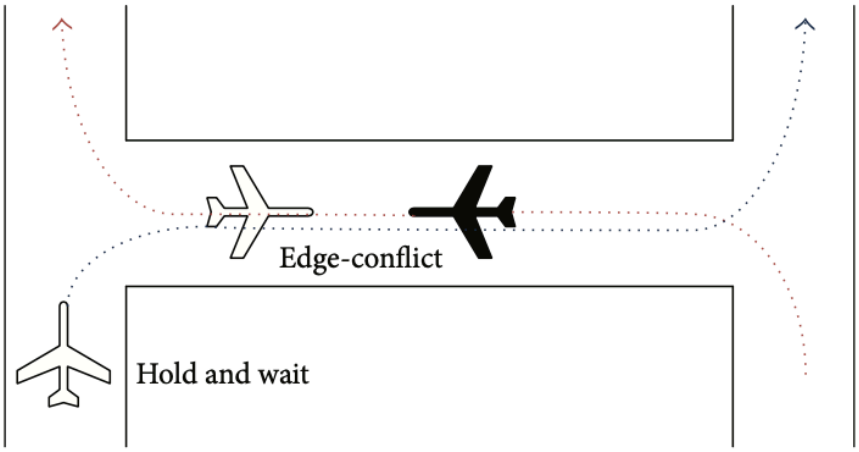
- **Description**

There are connection roads called taxiways at airports to allow the passage of aircraft between main roads or landmarks. This problem aims to find the most efficient and non-overlapping routes between these connection roads and to find a taxiway route in order to take the shortest route in the least amount of time. Airport ground movement is a combined routing and scheduling problem. Aircraft must be guided in a time-efficient manner along the taxiways, meeting assigned times at the runway and respecting safety constraints on the proximity of other aircraft. The objective of path choice is to make the total taxi path length minimum, but the conflicts are also considered during taxiing.

Node-conflict happens when two or more than two aircrafts taxi through a common node without keeping the minimum safety separation



Edge-conflict happens when two or more than two aircrafts taxi through the common segment but with opposite direction. One aircraft must hold and wait at the entrance node if the minimum safety separation is not satisfied



Three assumptions are made for the model:

1. Generally, all the aircrafts taxi at the same maximum speed.
2. When it is likely to conflict, aircrafts can adjust speed rapidly. So the acceleration is ignored.
3. When node-conflict happens, hold and wait strategy is always efficient whether the aircraft is large or small.

- **Symbols**

$G(V, E)$: Taxi network structure
V: Set of all nodes
E: Set of all edges
n^i_{start}: The start node of taxi i
n^i_{end}: The destination node of aircraft i
R: Set of feasible taxi path for all aircrafts, $R_i \in R$
n_i: Node in taxi network, $n_i \in V$, $R_i = \{n^i_{\text{start}}, n^i_2, \ldots, n^i_{\text{end}}\}$
F: Set of all aircrafts, $i \in F$
P: Set of aircraft priorities, P_i is the priority of aircraft i
T_i: The release time of aircraft i
v_i: The taxi speed of aircraft i
S_{mn}: The edge length between node m and node n
t_{mi}: The time of arrival at node m
t_0: The minimum safety time interval
$T_{f_{ni}}$: The hold and wait time at node n for node-conflict
$T_{c_{mni}}$: The hold and wait time at edge (m, n) for edge-conflict
f_{ijn}: Node-conflict detection 0-1 variables
w_{ij}: Priority comparison 0-1 variables
x_{ijn}: Arrival sequence detection 0-1 variables.

- **Objective Function**

Generally,more than two aircrafts need an assigned taxi path at the same time. The path scheduling can be evaluated by the length of path, the type of conflicts, the time of conflict, and the degree of conflict. The total time cost of all aircrafts reflects partly the path scheduling. So the optimization objective in this project is to make the total time cost of all aircrafts minimum.

$$\text{Min } T = \sum \left(\frac{S_{n^i_{\text{start}} n^i_{\text{end}}}}{v_i} + \sum_{n=\text{start}}^{\text{end}} T_{f_{ni}} + \sum_{n_x=\text{start}}^{\text{end}-1} T_{c_{n_x n_{x+1}}} \right)$$

- **Constraints**

1. f_{ijn} is used to detect node conflict:

$$f_{ijn} = \begin{cases} 1 & \left|t_{ni} - t_{nj}\right| \le t_0 \\ 0 & \text{others} \end{cases} \quad \forall n \in R_i \cap R_j, i \neq j.$$

2. w_{ij} is used to compare priority between aircraft i and j:

$$w_{ij} = \begin{cases} 1 & p_i \ge p_j \\ 0 & \text{others} \end{cases} \quad i \neq j.$$

3. x_{ijn} is used to detect the sequencing of aircraft i and j:

$$x_{ijn} = \begin{cases} 1 & t_{ni} \le t_{nj} \\ 0 & \text{others} \end{cases} \quad \forall n \in R_i \cap R_j, i \neq j.$$

4. When a node-conflict occurs at n node, the aircraft with low priority must wait for a certain amount of other aircraft. This is called hold and wait time ($T_{f_{ni}}$). And it creates the following constraint:

$$T_{f_{ni}} = f_{ijn}\left(1 - w_{ij}\right)\left(t_0 + t_{nj} - t_{ni}\right), \quad \forall n \in R_i \cap R_j, i \neq j.$$

5. When a edge-conflict occurs at n node, the aircraft with low priority must wait for a certain amount of other aircraft. This is called hold and wait time ($T_{c_{n_x n_{x+1} i}}$). And it creates the following constraint:

$$T_{c_{n_x n_{x+1} i}} = \left(1 - w_{ij}\right)\left(\frac{S_{n_x n_{x+1}}}{v_j} + t_{n_x j} - t_{n_{x+1} i}\right),$$

$$\forall \left(n_x, n_{x+1}\right) \in R_i \cap R_j, i \neq j.$$

6. S_{mn}, the path length from node m to node n:

$$S_{mn} = \sum_{x=m}^{n} S_{n_x n_{x+1}}.$$

7. The time of arrival at node n_x, $t_{n_x i}$:

$$t_{n_x i} = \frac{S_{n_1 n_x}}{v_i} + \sum_{\text{start}}^{x} T_{f_{n_x i}} + \sum_{\text{start}}^{x-1} T_{c_{n_x n_{x+1} i}} + T_i.$$

8. The minimum safety time constraint:

$$t_{n_x i} - t_{n_x j} \geq t_0, \quad \forall n_x \in R_i \cap R_j, i \neq j.$$

Genetic Algorithm (Population Based Algorithm)

For the Genetic Algorithm solution, I applied the following target frame:

Algorithm 1 The pseudocode of a GA

```
Set parameters
Choose encode method
Generate the initial population
while i < MaxIteration and Bestfitness < MaxFitness do
    Fitness calculation
    Selection
    Crossover
    Mutation
end while
Decode the individual with maximum fitness
return the best solution
```

Solution Representation For This Problem:

Unlike normal, there are more than one taxiway within the chromosome. These taxiways are separated by the -1 pad character. The numbers on each taxiway represent a node. In other words, if there are 2 aircrafts that want to go from 5th node to 1.node and 2 aircrafts that want to go from 1.node to 5th node, these aircrafts will be displayed as 1 chromosome and 4 flights will be exhibited at once. And it can be shown as:

5	2	1	-1	5	3	4	1	-1	1	4	2	5	-1	1	3	4	5	-1

Thus, the path of 4 aircrafts, which is variable according to the total number of aircrafts, can be shown as a chromosome in the above figure.

Cost Of A Solution:

The program calculates the time taken for all taxiways in the chromosome, taking into account edge conflicts and node conflicts. So it calculates the objective function.

$$\sum \left(\frac{S_{n^i_{\text{start}} n^i_{\text{end}}}}{v_i} + \sum_{n=\text{start}}^{\text{end}} T_{f_{ni}} + \sum_{n_x=\text{start}}^{\text{end}-1} T_{c_{n_x n_{x+1}}} \right)$$

Initial Solution Generation:

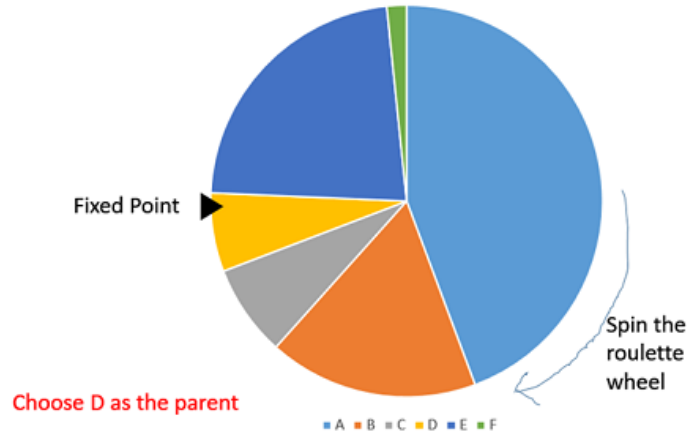
To create the initial solution, a taxiway pool consisting of the taxiways between the starting nodes and the end nodes of the aircraft was created at the beginning of the program. For example, a pool was created containing taxiways starting from 1 to 5 and taxiways starting from 5 to 1 for the aircraft waiting to taxi between 1-5 and 5-1 given in the previous example. Then a chromosome was created by randomly selecting taxiways from these pools.

Population Generation:

The chromosome was created and added together with the method used to create the initial solution in random order as much as the population size considered as a hyperparameter.

Mating Pool/Population Selection:

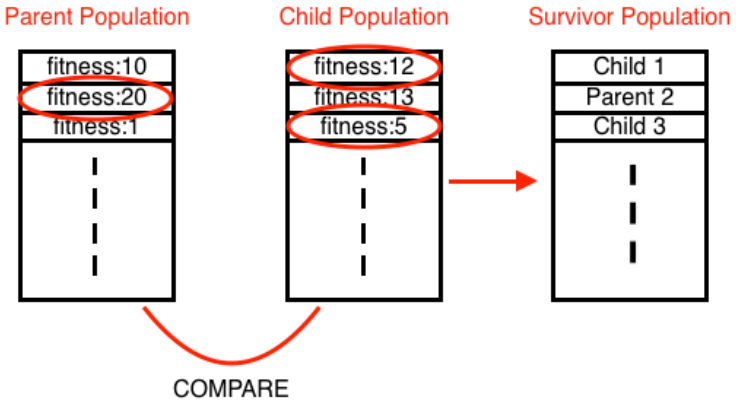
The roulette wheel selection system was used to create a good mating pool from the randomly initialized or ancestral population. That is, if the objective function was the smallest, a random selection was made by dividing it into numerators in a way to provide the greatest probability of being selected.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

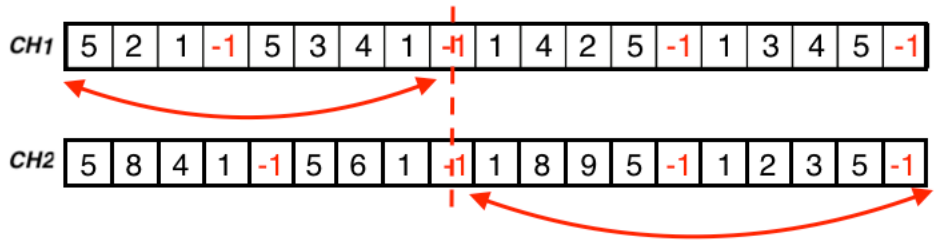
Survivor selection:

Chromosomes in all parent population and chromosomes in all child population were compared with each other according to objective function values. Survivors were selected by choosing the best choice for each chromosome value as a result of the comparisons.



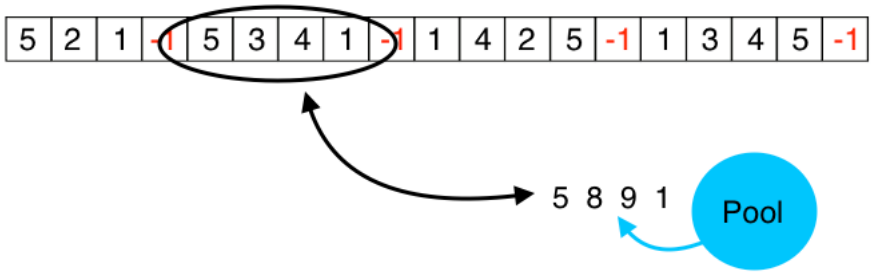
Crossover:

Inside a chromosome there are taxiways separated by -1. And the request of the aircraft corresponding to all population taxiways was the same. In this context, each taxiway was considered as a gene and a 1 point crossover was applied.



Mutation:

Here, for the purpose of applying mutation, a similar process was performed when creating the previous initial solution. First, a random taxiway was chosen within the chromosome. Then, a random new taxiway was selected from the solution pool used to create the Initial solution, the taxiway in the previous solution was deleted and the selected taxiway was added.



Termination Criteria:

I have specified the maximum number of loops as termination criteria. This number of loops is ?.

Pseudocodes of the GA algorithm:

```
algorithm GA(populationSize,xoverProb,loopSize):
        bestValue=MAXINT
        bestPath=[]
        genes=randomPopulationGeneration(populationSize)
        for i in range(loopSize):
            matingPool=matingPoolGeneration(genes)
            crossOver(matingPool,xoverProb)
            mutation(matingPool)
            localBest=self.findBestInPopulation(matingPool)
            if(localBest[0]<bestValue):
                bestValue=localBest[0]
                bestPath=localBest[1]
            survivorSelection(genes,matingPool)

        return bestValue
```

Simulated Annealing Algorithm (Trajectory Based Algorithm)

For the Simulated Annealing solution, I applied the following target frame:

```
procedure simulated annealing
begin
   t ← 0
   initialize T
   select a current point v_c at random
   evaluate v_c
   repeat
      repeat
         select a new point v_n
            in the neighborhood of v_c
         if eval(v_c) < eval(v_n)
            then v_c ← v_n
            else if random[0, 1) < e^((eval(v_n)−eval(v_c))/T)
                 then v_c ← v_n
      until (termination-condition)
      T ← g(T, t)
      t ← t + 1
   until (halting-criterion)
end
```

Solution Representation For This Problem:

Unlike normal, there are more than one taxiway within the solution. These taxiways are separated by the -1 pad character. The numbers on each taxiway represent a node. In other words, if there are 2 aircrafts that want to go from 5th node to 1st node and 2 aircrafts that want to go from 1st node to 5th node, these aircrafts will be displayed as 1 solution and 4 flights will be exhibited at once. And it can be shown as:

5	2	1	-1	5	3	4	1	-1	1	4	2	5	-1	1	3	4	5	-1

Thus, the path of 4 aircrafts, which is variable according to the total number of aircrafts, can be shown as a solution in the above figure.

Cost Of A Solution:

The program calculates the time taken for all taxiways in the solution, taking into account edge conflicts and node conflicts. So it calculates the objective function.

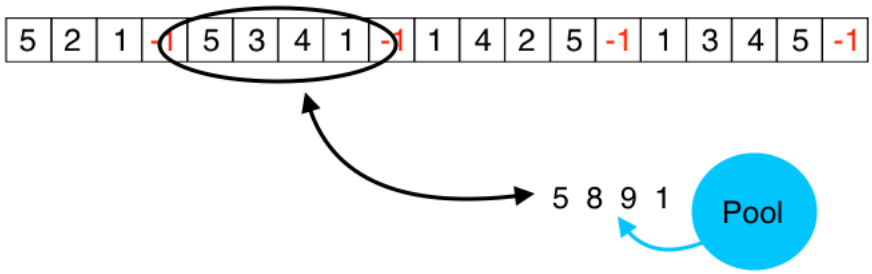
$$\sum \left(\frac{S_{n^i_{\text{start}} n^i_{\text{end}}}}{v_i} + \sum_{n=\text{start}}^{\text{end}} T_{f_{ni}} + \sum_{n_x=\text{start}}^{\text{end}-1} T_{c_{n_x n_{x+1}}} \right)$$

Initial Solution Generation:

To create the initial solution, a taxiway pool consisting of the taxiways between the starting nodes and the end nodes of the aircraft was created at the beginning of the program. For example, a pool was created containing taxiways starting from 1 to 5 and taxiways starting from 5 to 1 for the aircraft waiting to taxi between 1-5 and 5-1 given in the previous example. Then a solution was created by randomly selecting taxiways from these pools.

Neighborhood:

I applied similar process was performed when creating the previous initial solution. First, a random taxiway was chosen within the chromosome. Then, a random new taxiway was selected from the solution pool used to create the Initial solution, the taxiway in the previous solution was deleted and the selected taxiway was added.



Pseudocodes of the SA algorithm:

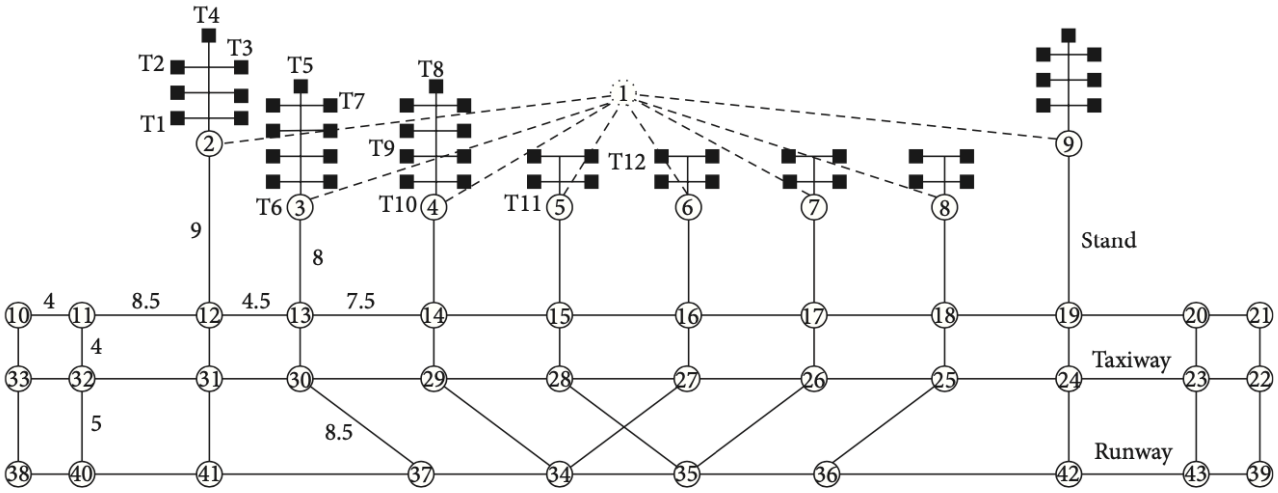
```
algorithm SA(self,temperatureCooling,innerLoop,outerLoop):
        bestValue=MAXINT
        bestSolution=[]
        initialSolution=self.createInitialSolution()
        temperature=objectiveFunc(initialSolution)
        currentVal=temperature
        counter=0
        while(counter<outerLoop):
```

```
        i=0
        while(i<innerLoop):
            newSolution=initialSolution
            newSolution=neighbourhoodSA(newSolution)
            newSolutionVal=objectiveFunc(newSolution)
           if((newSolutionVal-currentVal<=0)
           or(random.random()<=
           math.exp(-((newSolutionVal
           currentVal)/temperature)))):
                currentVal=newSolutionVal
                initialSolution=newSolution
            if(currentVal<bestValue):
                bestValue = currentVal
                bestSolution=initialSolution
            i+=1
        temperature*=temperatureCooling
        counter+=1

    return bestTeta,bestSolution
```

Hyperparameters and Tests

Tests for hyperparameter optimization were performed in GA and SA for the test case specified in the reference article. The reference article contained the following graph. The edge values shown on it were on unit basis and each unit corresponded to 75m. The minimum safety time interval is set as 4 units (the conversion of 300 meters). The average speed of aircraft is set as 36 Km/h, which means that the minimum safety time interval is 30 seconds. According to the flight scheduling in this airport, 9 flights need to be scheduled at same time.



The table for the scheduling of the aircraft was as follows:

	Flight no.	ETD	Type	Priority	Stand	D/A
1	MU5178	10.35	320	4	T1	D
2	CZ3118	10.35	330	4.5	T2	D
3	CZ6218	10.35	330	4.5	T7	D
4	MU2078	10.35	320	4	T8	D
5	CA1605	10.35	737	5	T5	D
6	CA1802	10.35	738	3.5	T4	A
7	MF8115	10.35	737	3	T9	A
8	HU7196	10.35	734	2.5	T12	A
9	GS6574	10.35	319	2	T11	A

In addition, the routes these aircrafts wanted to go were as follows:

Flight no.	**Source**	**Destination**	**Priority**
MU5178	2	38	4
CZ3118	2	38	4.5
CZ6218	3	38	4.5
MU2078	4	38	4
CA1605	3	38	5
CA1802	37	2	3.5
MF8115	37	4	3
HU7196	37	5	2.5
GS6574	37	5	2

SA Hyperparameter Tests

Initial Temperature

The objective function value of the solution determined as the initial solution was used as the Initial temperature.

Reheat Cooling Coefficient

The constant was set at 0.9.

Coldest Temperature

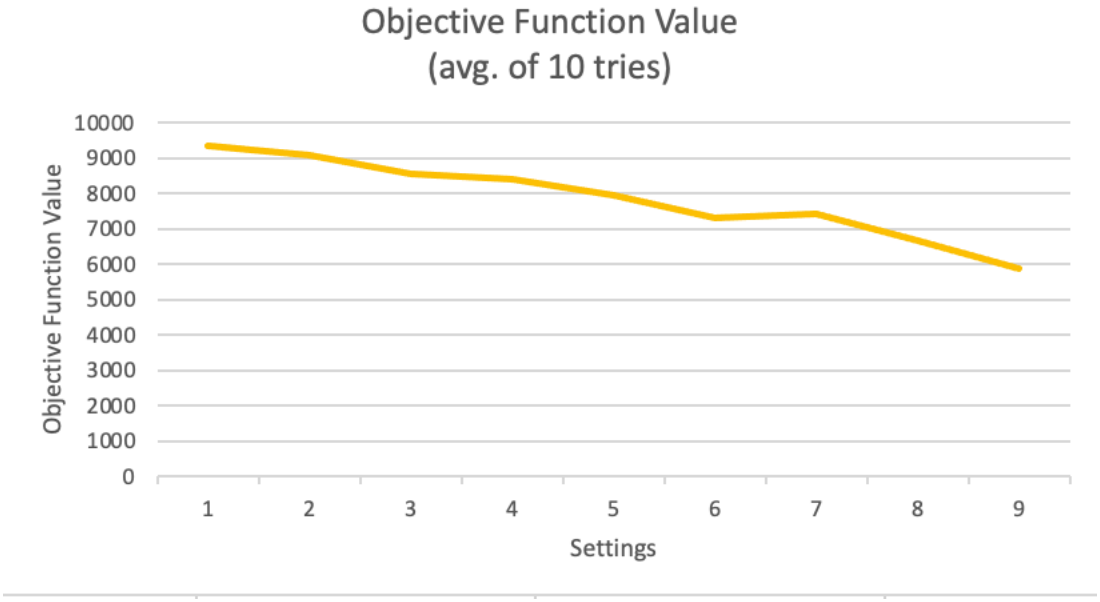
0.05 was used as the coldest temperature. At this temperature and at temperatures lower than that, it did not make sense that it was even lower, as the likelihood of accepting a different bad solution was very low. The reason why this coldest temperature is lower than normal is to make a little more intensification.

Cooling Coefficient

First, by keeping the number of outer loops and the number of inner loops constant, experiments up to 0.1-0.9 were performed on the cooling coefficient. The following table emerged as a result of these experiments. In this part, there was a period of injustice for larger cooling coefficients due to more outer loop turns(time). The reheat number was used to compensate for this injustice.

Settings	Cooling Ratio	Coolest Temperature	Inner Loop Size	Reheat Size	Objective Function Value (avg. of 10 tries)
1	0,1	0,05	20	21	9341,59
2	0,2	0,05	20	14	9091,27
3	0,3	0,05	20	12	8553,81
4	0,4	0,05	20	9	8402,61
5	0,5	0,05	20	6	7962,57
6	0,6	0,05	20	5	7312,43
7	0,7	0,05	20	4	7437,66
8	0,8	0,05	20	2	6689,29
9	0,9	0,05	20	1	5865,53

In addition, the graph below was obtained from this table according to the objective function distribution:

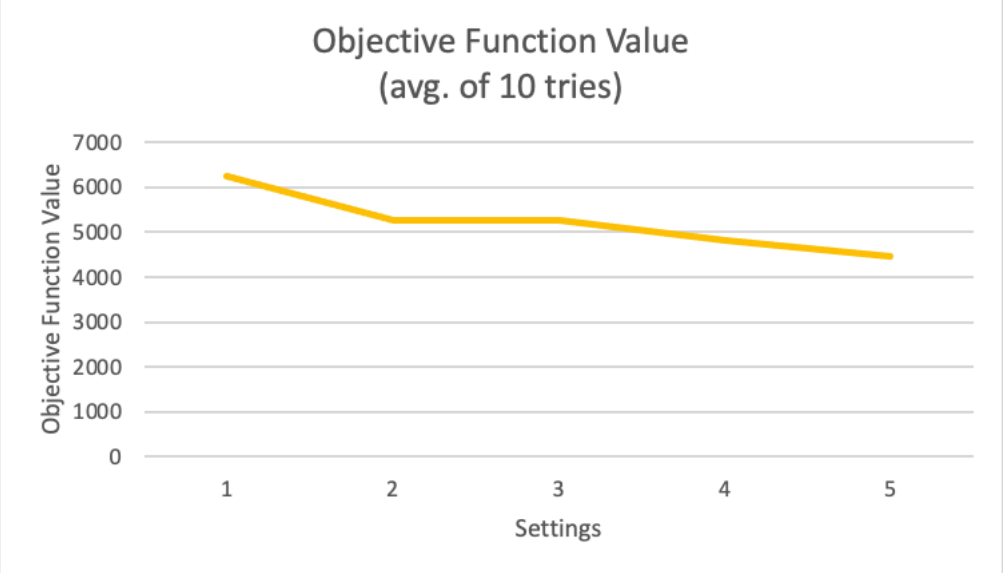


Inner Loop

In this section, certain tests were performed for inner loop value. Here, inner loop values are balanced with reheat sizes in order to protect fairness for the time. The 0.9 value found in the previous test was used as the cooling coefficient. Inner loop tests were performed between 10-50 and the following table came out as a result of the test.

Settings	Inner Loop Size	Cooling Ratio	Reheat Size	Coolest Temperature	Objective Function Value (avg. of 10 trie
1	10	0,9	10	0,05	6262,35
2	20	0,9	5	0,05	5258,32
3	30	0,9	4	0,05	5270,55
4	40	0,9	3	0,05	4813,05
5	50	0,9	2	0,05	4464,3

Based on this table, the following objective function graph has emerged.

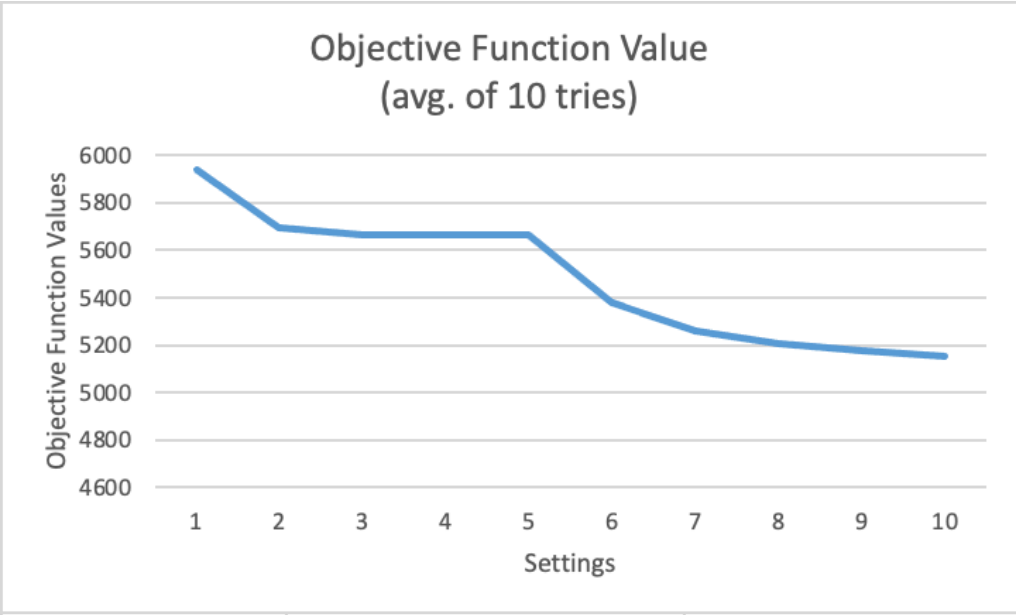


Reheat Size

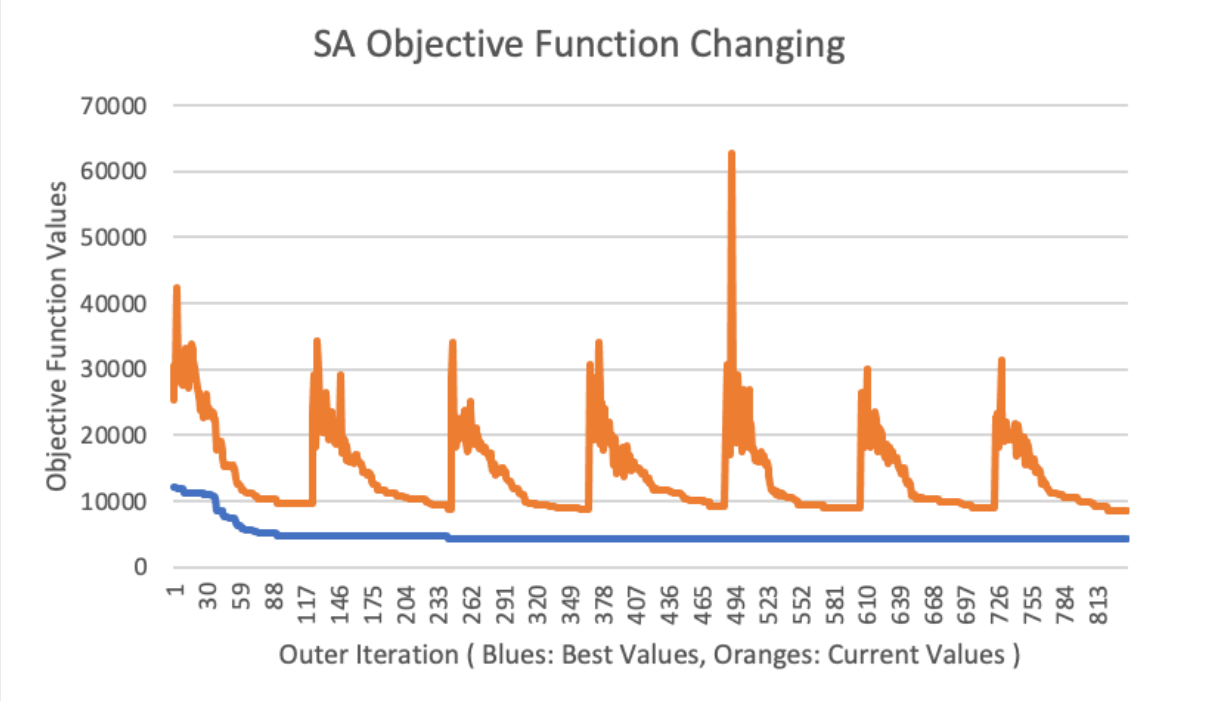
In this part, a test was made for the amount of rehat size in SA. The range of 1-10 was determined as the rehat size range. Within this specified range, the values found in previous tests were used as the number of internal loop size and the cooling coefficient. The table below was obtained with these trials.

Settings	Reheat Size	Cooling Ratio	Inner Loop Size	Coolest Temperature	Objective Function Value (avg. of 10 tries)
1	1	0,9	50	0,05	5938,56
2	2	0,9	50	0,05	5692,48
3	3	0,9	50	0,05	5667,97
4	4	0,9	50	0,05	5662,97
5	5	0,9	50	0,05	5662,97
6	6	0,9	50	0,05	5376,18
7	7	0,9	50	0,05	5258,34
8	8	0,9	50	0,05	5203,88
9	9	0,9	50	0,05	5174,2
10	10	0,9	50	0,05	5152,93

Based on this table, the following graph has emerged for the objective function change.



The best result was not chosen in this test. Because after a while, the development was decreasing. In this context, 6 reheat iterations were selected as the best according to the process time spent.The parameters obtained as a result of all these tests were Cooling coefficient: 0.9, Inner loop size: 50, coldest temperature: 0.05, reheat size: 6. When we run these parameters for the solution, the best objective function change is as follows.



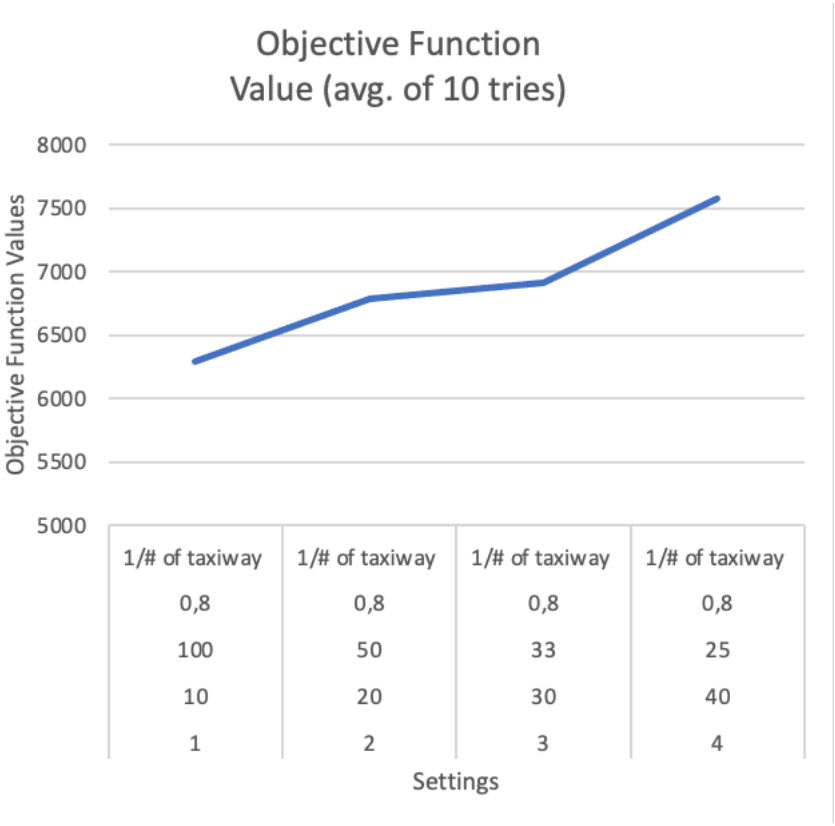
GA Hyperparameter Tests

Population Size

In this section, tests were carried out in a way that the population amount varied between 10-40. The number of iterations was used to compensate for the time injustice of the population size. A constant 0.8 was used as the crossover probability and *1/number of taxiway* was used as the mutation probability. As a result of these tests, the results in the table below were obtained.

Settings	Population Size	Iteration	Crossover Probability	Mutation Probability	Objective Function Value (avg. of 10 tries)
1	10	100	0,8	1/# of taxiway	6292,74
2	20	50	0,8	1/# of taxiway	6791,11
3	30	33	0,8	1/# of taxiway	6913,51
4	40	25	0,8	1/# of taxiway	7576,15

Based on the above table, the following graph has been drawn in order to see the change of the objective function.

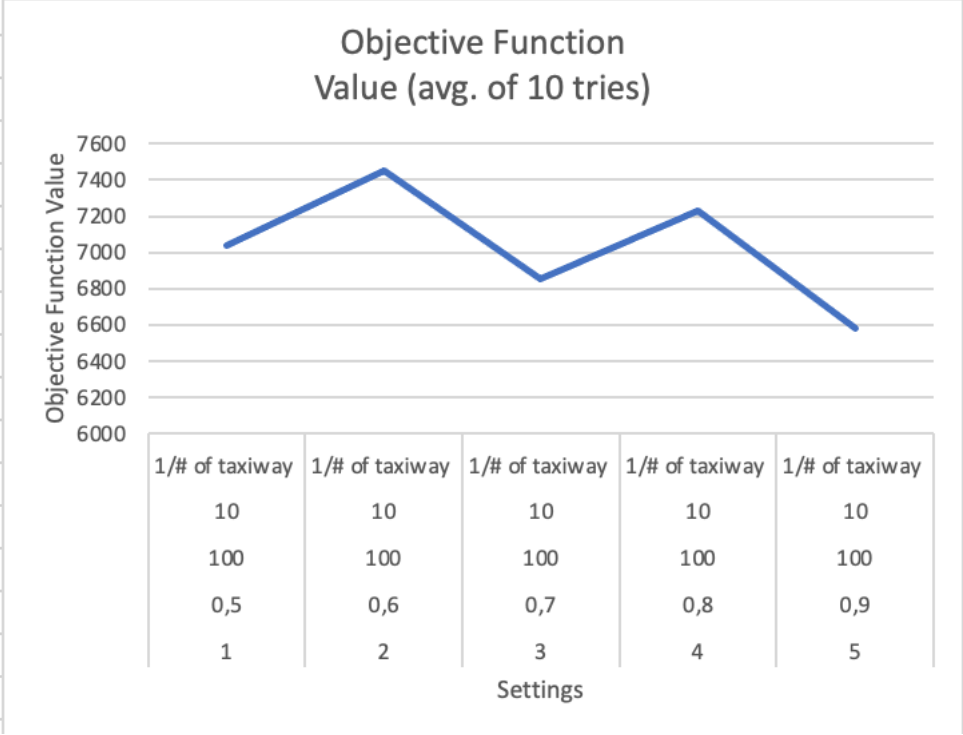


Crossover Probability

In this part, by keeping the value in the population size part 10, the iteration value as 100, and the mutation probability as *1/number of taxiway*, trials between 0.5-0.9 were made for crossover probability. As a result of these trials, the data in the table below were obtained.

Settings	Crossover Prob	Iteration	Population Size	Mutation Proba	Function Value (avg. of 10 trie
1	0,5	100	10	1/# of taxiway	7038,48
2	0,6	100	10	1/# of taxiway	7450,26
3	0,7	100	10	1/# of taxiway	6853,29
4	0,8	100	10	1/# of taxiway	7230,68
5	0,9	100	10	1/# of taxiway	6579,87

In order to see the objective function change in this table better, the following graph is obtained from the table.

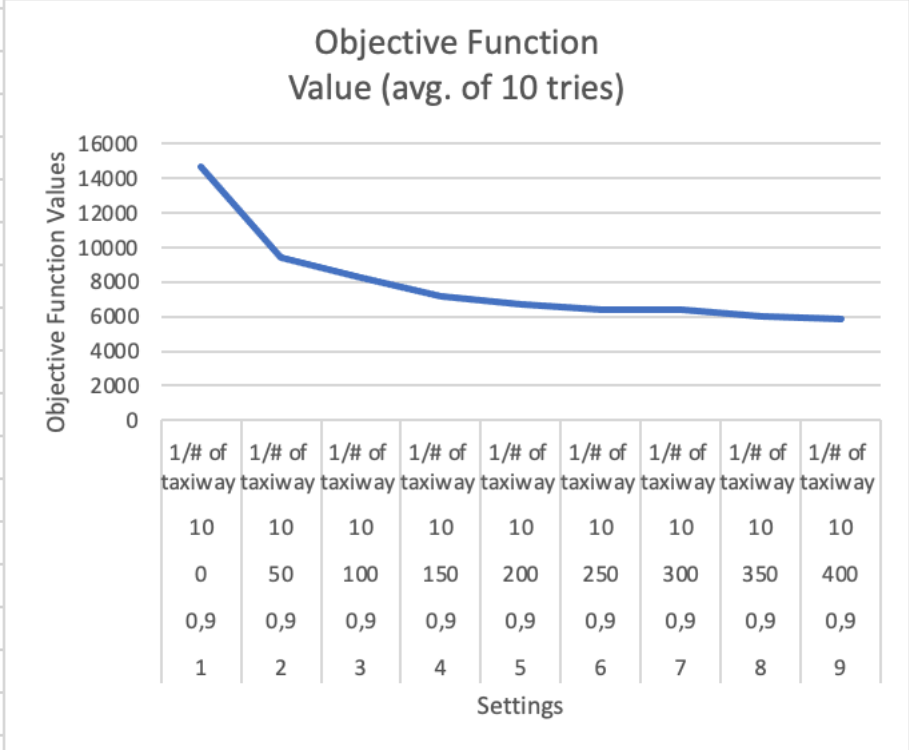


Iteration (Termination Condition)

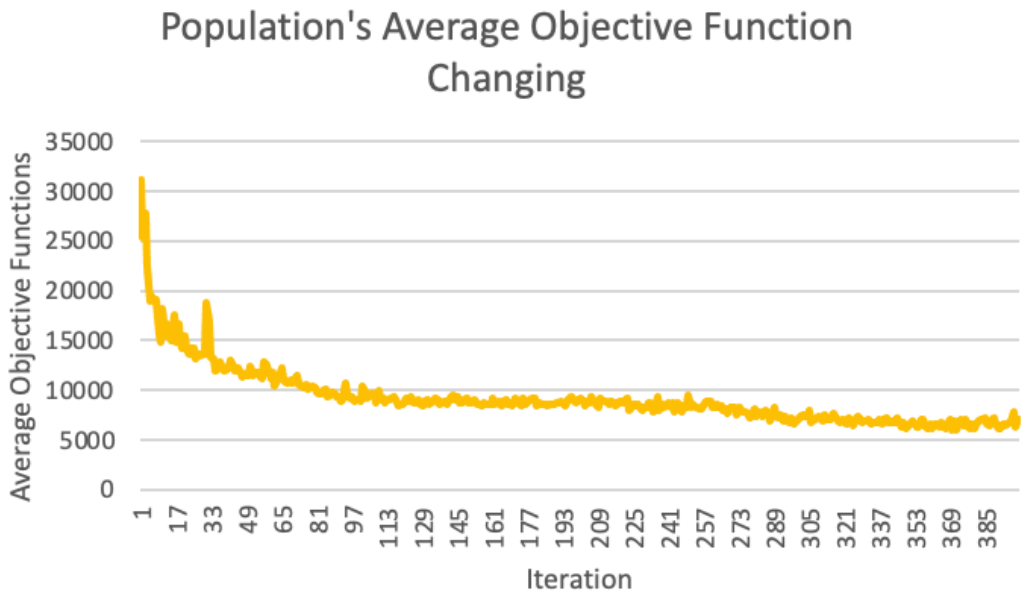
In this section, the best variables found until the iteration experiment were used consistently. In other words, 0.9 was used as crossover probability and 10 was used as the population size. If the iteration value was between 0-400, an experiment was made. As a result of this experiment, the data in the table below was obtained.

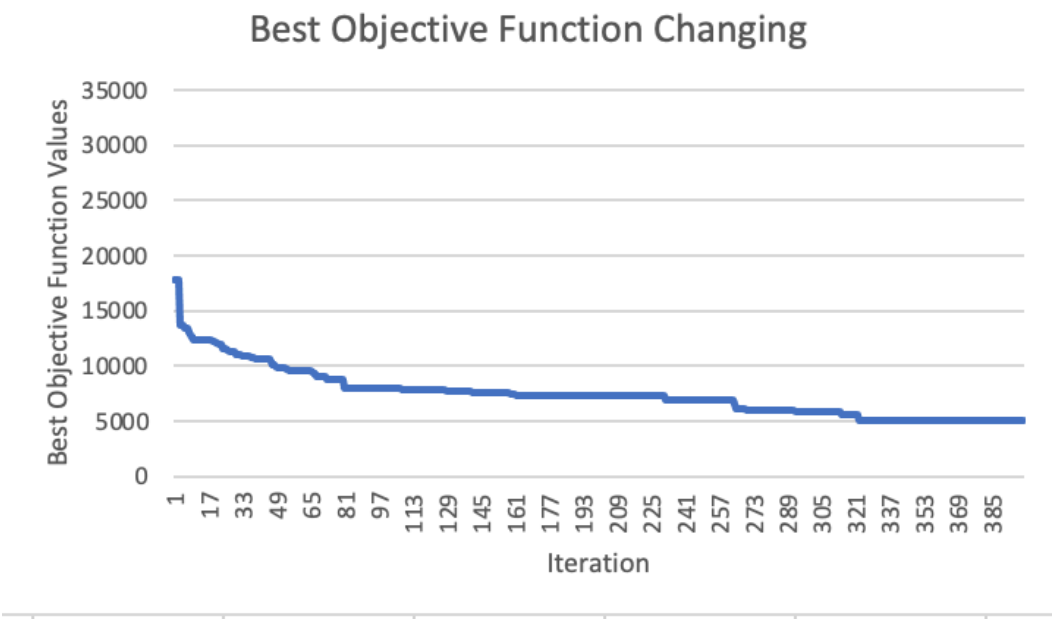
Settings	Crossover Probability	Iteration	Population Size	Mutation Probability	Objective Function Value (avg. of 10 tries
1	0,9	0	10	1/# of taxiway	14649,99
2	0,9	50	10	1/# of taxiway	9416,03
3	0,9	100	10	1/# of taxiway	8276,97
4	0,9	150	10	1/# of taxiway	7167,95
5	0,9	200	10	1/# of taxiway	6676,42
6	0,9	250	10	1/# of taxiway	6428,93
7	0,9	300	10	1/# of taxiway	6408,92
8	0,9	350	10	1/# of taxiway	6021,74
9	0,9	400	10	1/# of taxiway	5850,49

400 was chosen as the solution, as 400 iterations worked best within the specified range.



Based on all these, the best GA working version was determined as follows: Population size: 10, crossover probability: 0.9, iteration size: 400, mutation probability: *1/number of taxiway.*





Compared With The Article's Solution

In the written GA and SA, the problem mentioned in the article and developed was used (the problem specified in the table above). However, since the initial solutions given by the article are much better than the initial solutions we use, they did not achieve good results. But improvements have been achieved in the following table.

Specs	Article's GA	My GA	My SA
Best Solution In Initial Population	4091	17784	13069
Result Solution	2895	5026	4333

References

- Una Benlic ,Alexander E.I. Brownlee , Edmund K. Burke, Heuristic search for the coupled runway sequencing and taxiway routing problem,*Transportation Research Part C,* (2016).
- Xuan Wang, and Qinghai Zuo, Aircraft taxiing route planning based on airport hotspots, *AIP Conference Proceedings 1839,* (2017).
- Yu Jiang, Zhihua Liao, and Honghai Zhang. A Collaborative Optimization Model for Ground Taxi Based on Aircraft Priority , *Hindawi Publishing Corporation Mathematical Problems in Engineering Volume 2013,* (2013).
- Yu Jiang, Xinxing Xu, Honghai Zhang, and Yuxiao Luo. Taxiing Route Scheduling between Taxiway and Runway in Hub Airport, *Hindawi Publishing Corporation Mathematical Problems in Engineering Volume 2015,* (2015).
- Prof.Dr. Fatih Erdoğan Sevilgen. Optimization(CSE424), *Course Presentations,* (2021)