Airline Scheduling with Simulated Annealing

Chi Chun Wan – 2525244
Shabaz Sultan – 2566703
Boudewijn Zwaal – 1897527

Abstract—Abstract text to go here.

I. Introduction

Flight schedules take up a central place in airline businesses. To maximize profit a business has to be as efficient as possible in using its resources and be as effective as possible in meeting marketplace demand. A business wants to plan its flights based on demand, use its airplanes to service these flights as efficiently as possible and assign crews to these flights to minimize expenses. These pose highly complex combinatorial optimization problems that cannot be solved analytically. Instead numerical optimization systems are used and their effectiveness can be influential on the profitability of a passenger focusses airline as a whole.

The Dutch airline KLM flies on 150 destinations with 97 airplanes and needs to produce a flight schedule four times a year that maximizes their profitability [1]. In this paper we will consider this problem in a slightly reduced scenario, with a fictional Mokum Airways that flies on 28 destinations with 6 airplanes. The goal is to maximize profit as measured by revenue passenger kilometers, often used as unit to measure the product being sold in a passenger focussed airline [2]. In order to obtain the best flight schedule, we use heuristics algorithms to analyze all the possible tours and find the optimal one.

II. Background

Constructing airline schedules can be broken down in a number of subproblems, each posing a number of computational and algorithmic challenges. Often first the flights that are provided by an airline are constructed, usually subject to complex regulation and non-deterministic customer demand functions. This is referred to as the Airline Scheduling problem in the literature, where the goal is to maximize profits by meeting said demands [3].

The next step is to assign the flights that are flown to airplanes. This is the so-called Fleet Scheduling Problem [4]. An airplane schedule can be constructed ahead of time, but scheduling methods for real world scenarios need to deal with real-time adjustments as well due to delays on the day itself (e.g. due to weather conditions, technical problems or issues at an airport). Ideally these schedules are repeatable (usually over either a one or seven day period), requiring that if at the start of the day an aircraft is at airport A, there needs to be an aircraft (not necessarily the exact same, just one of the same type) at airport A at the end of the schedule.

Finally crew personnel is assigned to the flights and airplanes. The crew scheduling problem is subject to constraints from flight authority regulation, labour regulation and airline specific regulation and because it represents the largest expenditure for airlines after fuel cost it is an important area of research for the industry as well [5].

This paper tackles the design and optimization of an airline schedule, with airplane assignment being part of the designed schedule. The one consideration given to the constraints from the crew schedule problem is that the schedule needs to have single homebase, each crew member is assumed to live at said homebase and each plane needs to attend that homebase daily for a crew change.

There are various approaches to this problem, such as integer programming [6] or even agent based approaches [7]. One candidate for complex combinatorial optimization problems in general is Simulated Annealing [8]. This meta-heuristic has been applied to related problems such as crew scheduling [9]. It has been specifically used for airline scheduling by [10], which is the main reference for the research described in this paper.

III. Problem

We consider an airline with a number of potential destinations and airplanes. The specific dataset used has 28 destinations and potential flights in both directions between every possible pair of destinations. We will analyse the problem with one and six airplanes. We wish to schedule a number of flights over a one day period. Based on the number of passengers and length of the flights a passenger kilometer score can be calculated for each of these flights. The total passenger kilometer score is the metric that needs to be optimized for. The number of passengers are based on a deterministic demand function, where for each potential flight the total passenger demand for the day is provided.

The flights are subject to a number of restrictions. In reality there are often regulatory flight restrictions only allowing take-offs and landing at certain hours of the day. We assume a 20 hour period of possible flight and that this 20 hour window is the same for every possible airport. This is a simplification that allows us to ignore timezone considerations. During this period there are a number of things that take up time. The flights will have a variable length of time, based on distance between start and end destination and speed of the aircraft (800 km/h in our example scenario). Docking takes up a certain amount of time (one hour in our example).

Beside time the planes' operation is also restricted by fuel consumption. The plane has a certain tank capacity that allows

it a certain range before it has to refuel (3199 km in the example). If a plane wants to fly a route without having enough fuel in its tank it needs to refuel first. We will only consider the scenario where refueling will completely fill up the tank and takes up a constant time (one hour).

Finally we have two airline requirements. There is the requirement to have the schedule be repeatable. To fulfil this requirement each plane will have its starting airport also be its final destination of the day. The airline also has a certain destination airport designated as homebase, which every plane needs to attend during the day for a crew changeover.

IV. ALGORITHM

To deal with the problem, we use two heuristic algorithms, namely hill-climbing and simulated annealing for six airplanes and one algorithm brute force for one airplane.

A. Brute-force

Brute-force is a problem-solving technique that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement. In our case, we check all the possible tours by using a tree structure, where the node represents the destinations. A representation of the tree is given in figure 1.
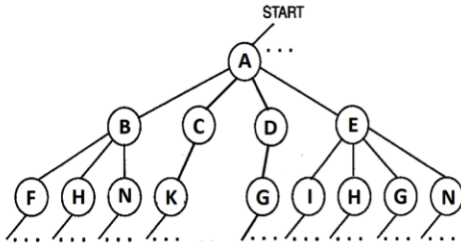

Fig. 1. Tree data structure

An advantage of this algorithm is that the solution found, also is the optimal solution. A disadvantage is that the run time will increases a lot to find a solution, when the state space is big. Therefore, we add branch and bound to the algorithm. The branch and bound algorithm has a upper-and lower bound. The upper bound is the theoretical maximum passengers-kilometer score and the lower bound is the best founded score so far for every step. For every position in the tree, we calculate what the maximum score is that can still be obtained from that position. If the score is lower than the current lower bound. We will not look further anymore from that position. This has as result that it will make the possibilities smaller and therefore a lower run time.

B. Hill-climbing

Hill-climbing is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by changing a single element of the solution.

If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.

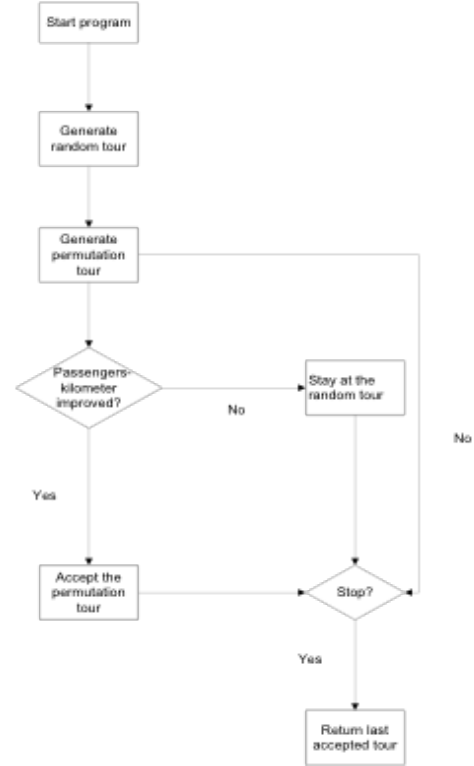In the case of our problem, a flowchart is made in figure 2.

Fig. 2. Flowchart Hill-climbing

1) Tour generation and permutation: To make hill-climbing (and later simulated annealing) work, we need a way to generate tours and to permutate these tours. The way we generate a new tour is as follows. We start in out home-base, and then we add (randomly chosen) cities to the tour until the time is full. Then from the last chosen city we generate a tree-like structure that determines all possible routes from there. If one of those routes lead us back to our home-base within the time window, we accept that one. If such a route does not exist, we cut off the last city of the generated tour and try to reach our home-base again, with the same tree-like structure. If that is not possible again, we cut off the last city one last time and try again. If it is still not possible we try it again with a whole new route. After this we check what would be the most efficient city to start (and end) in, by checking how many times we need to refuel.

When permutating a tour, notice that we first shift it such that our home-base is again the first city in the tour. Then we cut off a random amount of cities from the tour (except the first one, which is our home-base). We then try to create a new tour from the two endpoints created in the route. We use the same algorithm for this as we did for the generated tour, i.e.

we make a tree of all possible routes from one endpoint and check if one of them leads to the other endpoint. If not we cut off a city and try it again, just as when we generate a random tour. We then again check what is the most efficient start-end city.

2) Hill-climbing: We start with an initial random tour generated with the algorithm described above. To find a better solution, we permutate the tour. The score of the passenger-kilometers for the initial tour is stored in a variable best score. If the score of the permutation tour is higher than the score of the arbitrary tour, the score of the permutated tour will be the new best score and the permutation tour will be accepted, otherwise the best score will stay the same and we will stay at the previous tour. This process is repeated until no further improvements can be found and the best tour is the last accepted tour. When applied to multiple planes, first one of the planes in the system is randomly chosen and its flightplan (tour) is permutated. The total score of all airplane tours is then calculated and compared to the previous best score. If it improves on said score, the new tour is accepted. A disadvantage of hill-climbing is that it is good for finding a local optimum, but it is not guaranteed to find the best possible solution (global optimum). To avoid this disadvantage, we use a different algorithm, simulated annealing.

C. Simulated Annealing

In case of simulated annealing, its almost the same as hill climbing except when the solution is not improved, this solution will be accepted with a probability P. The probability is based on the number of iterations and the temperature. As the number of iterations increases, the temperature goes down and the probability will decrease.
In figure 3, the flowchart of hill-climbing is adjusted in simulated annealing.

In the case when the passengers-kilometer score is not improved, you will accept the permutation tour with probability $P = e^{\frac{\Delta C}{T}}$, where C = score of the permutation tour score of the current best tour and $T = 0,999^i * T_0$ with i is the number of iterations and T_0 is the start temperature 50.000. This probability will decrease with the number of iterations. Thus when the number of iterations increases, the temperature goes down. When temperature is high, the system will choose new states more or less at random, but as the temperature lowers this algorithm will go to hill-climbing.

V. RESULT

To validate our method, demand and flight distances for a fictional airline are used. Mokum Airways, a newly created Dutch airline based in Amsterdam, has landing rights for 28 destinations around Europe.

The airline has a fleet of six Airbus A321 aircrafts with speed of 800km/h, capacity of 199 and range of 3199 km. The take-off and landing take place between 02:00 and 06:00 and
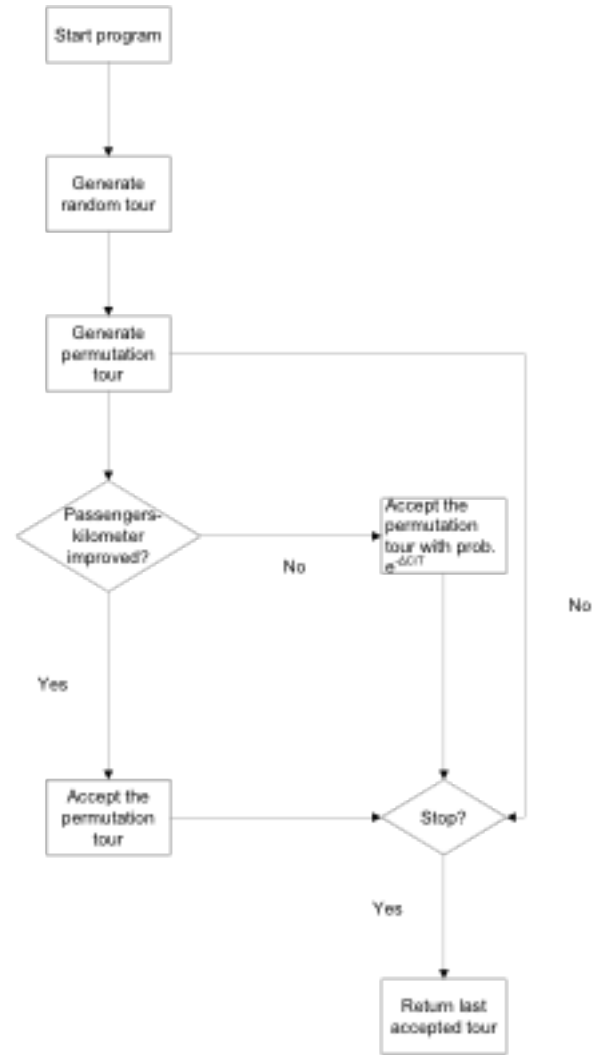

Fig. 3. Flowchart Simulated Annealing
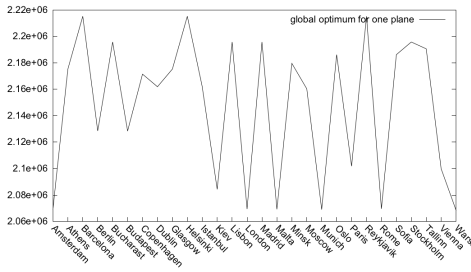

Fig. 4. Mokum Airways destinations

Fig. 5.   Global optimum for different homebases with one plane

docking time and refuel time will take one hour. Moreover, once per day the plane needs to land in the home-base for the crew-change. The flight schedule has to be a cycle, so the start point and the end point of the route has to be the same. We first tested with scheduling one airplane, to set our parameters. For we could compare these results with the global optimum very early (obtained by bruteforcing), we could set the parameters such that we found that optimum the most. These settings are used in scheduling for six airplanes. The cooling rate of the simulated annealing was set at 0.99999. The initial temperature is set at 50.000 per plane, and thus in a simulation of six airplanes the initial temperature was 6*50.000. Finally we terminated the simulation when the accepted schedule had not changed for more than 1000 iterations. The obtained results was accurate, but the runtime was very high.

In figure(6), the behavior of simulated annealing for the score is shown during the course of the simulation. In the beginning, when the temperature is high, the score is very random and as the temperature drops to lower values the score becomes less random and will look more like a hill climber. Finally, it converges to what the simulation believes is the global optimum (although this is not always the case).
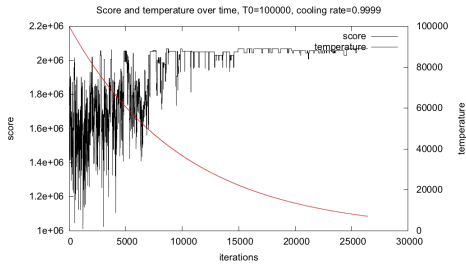

Fig. 6.   Score over time for Simulated Annealing

The results of the best score with one airplane for different home-bases is plotted in figure(6). The home-bases with the highest score (2.195.766) are Barcelona, Helsinki and Reyk-javik with the tours shown in figure **??** and for the current home-base Amsterdam, the score would be 2.069.202 with the tour shown in figure (8). Acquiring these results took about two-three minutes per simulation.

Fig. 7.   Best tour for one plane with any homebase, passenger kilometer score 2.215.268.

The result for the main problem with six airplanes is shown in figure(10) for different home-bases. The results are shown both for the brute-force algorithm as for the simulated annealing. Moreover, the score with six identical airplanes that fly the best route for one airplane is given to make an analysis of the algorithms. See Analysis section.

Fig. 8.   Best tour for one plane with homebase Amsterdam, passenger kilometer score 2.069.202.

The best score with simulated annealing is 12.388.745 with as home-base Amsterdam. The best home-base to choose is Lisbon with the highest score of 12.908.931. Acquiring these results now took about 10-30 minutes. This has a large variance, because of the random nature of simulated annealing. What can clearly be seen is that most of the times the scores of the simulated annealer is not far off from our bruteforce approach. This indicates that our simulated annealer works quite well, further analysed in the analysis section.

To evaluate our simulated annealing, the average number of iterations to find the optimum for the simulated annealing is given in the table below.

```
Average number of iterations: 328075.885714
```
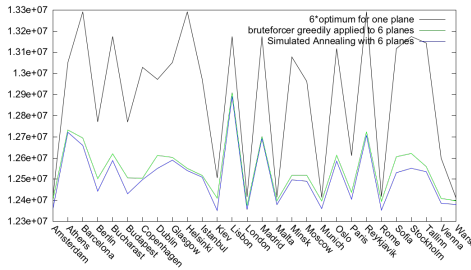
Fig. 9. Results for different homebases with six planes, Simulated Annealing line based on best of 5 runs for each scenario, cooling rate of 0.99999.

```
Variance of the n.o.i.: 2025145743.66
Standard deviation of the n.o.i.: 45001.6193448
```

The simulation needed quite a lot iterations before terminating. For this data, we have done five runs per home-base for 28 different home-bases, that's a total of 5*28 runs. What's remarkable is that both the algorithms have about the same peaks as the same peaks as the theoretical six-times-one-plane situation. This indicates a strong correlation between these two and thus let us estimate the behavior of six planes for the behavior of one plane. See Analysis section.

VI. ANALYSIS

To analyze the solution of simulated annealing. We use the result of the brute force for one airplane as reference, since the result is the global maximum. In table (1), the percentage of the global optimum found for simulated annealing is given for different cooling rates. When the decimals for the cooling rate increases, the percentage global optimum found will increase strongly, because more iterations will be run and therefore the probability to find the global optimum will be higher.

Cooling Rate	Percentage global optimum found
0.99	4%
0.999	6%
0.9999	38%
0.99999	83%

TABLE I.   TABLE 1, SCORE ONE AIRPLANE SA, 100 RUNS, T0 = 50.000

Moreover, the quality of our heuristics is tested. In figure (10), the RMSE of the score for simulated annealing is plotted as a black line. The number of iterations depends on the cooling rate, a lower cooling rate will increase the number of iterations and thereby the run time. However, the RMSE decreases with the number of iterations. Thus, the result will be more accurate. To analyze the two heuristics, hill climbing and simulated annealing, we can look at the same figure (10) for both heuristics for the same iterations against the RMSE. The RMSE for the same iteration for the hill climber is higher than the simulated annealing. Thus, the simulated annealing is in this case better for finding the optimal solution than the hill climbing, since the risk to get into a local optimum is limited.

An interesting analysis is whether the solution for one airplane can predict the solution for six airplanes. In figure (10), the score times 6 for one airplane follows almost the same pattern as the score of the simulated annealing for six airplanes for different home bases. We ran a correlation test. This resulted in a correlation coefficient of 0.820 for simulated annealing. Therefore, we can conclude there is a correlation between the prediction for the score of six airplanes when the score for one airplane is known.
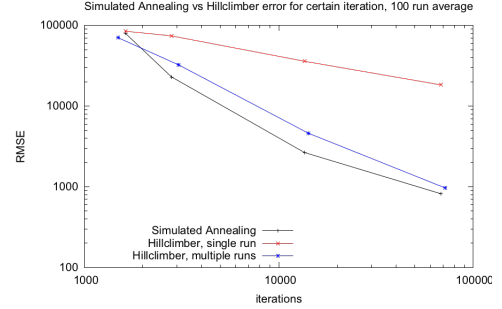

Fig. 10. Error for Simulated Annealing and Hill-climbing.

VII. DISCUSSION

VIII. CONCLUSIONS

In conclusion we can say that our simulated annealer works very well. After analyzing the various results we see that compared to the bruteforcer our simulated annealer performs very close to it. Also with a runtime of between 10 and 30 minutes to find an optimal schedule for six airplanes, it is not all that slow. Of course one can do the simulation faster, but we showed that the accuracy of the results then will decline. We can conclude as well that the simulated annealer works better than the hill climber, as discussed in the analysis section. We also showed that the outcome of scheduling six airplanes can be predicted with the output of the simulation with one plane. We found a high correlation between the global optimum of one airplane and the global optimum of six airplanes, and this can be seen also in figure (9), as the peaks agree of both sources. [IMPROVEMENTS?]

REFERENCES

[1] F. Bian, E. Burke, S. Jain, and G. Kendall, "Solution Methodologies for generating robust Airline Schedules," *European Study Group ...*, no. February, pp. 9–19, 2003. [Online]. Available: http://www.wiskgenoot. nl/swi/Proceedings/Leiden2003/proceedingsSWI2003.pdf#page=15

[2] M. Schefczyk, "Operational performance of airlines: an extension of traditional measurement paradigms," *Strategic Management Journal*, vol. 14, no. 4, pp. 301–317, 1993. [Online]. Available: http: //www.jstor.org/stable/2486797

[3] M. Etschmaier and D. Mathaisel, "Airline scheduling: An overview," *Transportation Science*, vol. 19, no. 2, pp. 127–138, 1985.

[4] R. a. Rushmeier and S. a. Kontogiorgis, "Advances in the Optimization of Airline Fleet Assignment," *Transportation Science*, vol. 31, no. 2, pp. 159–169, 1997.

[5] B. Gopalakrishnan and E. L. Johnson, "Airline crew scheduling: State-of-the-art," *Annals of Operations Research*, vol. 140, no. 1991, pp. 305–337, 2005.

[6] S. Raff, "Routing and scheduling of vehicles and crews," *Computers & Operations Research*, vol. 10, no. 2, pp. 63–211, 1983.

[7] J. Langerman and E. Ehlers, "Agent-based airline scheduling," *Computers & industrial engineering*, vol. 33, pp. 849–852, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360835297002647

[8] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi *et al.*, "Optimization by simmulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[9] D. Sosnowska and J. Rolim, "Fleet scheduling optimization: a simulated annealing approach," *Practice and theory of automated timetabling III*, 2001. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-44629-X_14

[10] J. Mashford and B. Marksjö, "Airline base schedule optimisation by flight network annealing," *Annals of Operations Research*, pp. 293–313, 2001. [Online]. Available: http://link.springer.com/article/10.1023/A:1016027516013