CS 581

Assignment 2

Julia Briden

February 27, 2021

1 Introduction

Urban Air Mobility (UAM) addresses current challenges in ground transportation for highly-populated areas by providing manned and unmanned vehicles for city transportation services. To ensure safety during adverse weather conditions and reduce travel time, optimal trajectories must be developed for the fleet of aircraft. To address this challenge, solutions to the urban air mobility (UAM) trajectory planning version of the vehicle route planning problem were generated using simulated annealing. Optimal trajectories for multiple aircraft visiting a set of locations with weather-based constraints were found using the simulated annealing algorithm and compared to the solutions produced by the hill climbing algorithm.

2 Methods

2.1 State Representation

The problem of UAM aircraft optimal trajectory planning is represented as nodes on an nxn grid (Figure 1). Note, this implementation can be generalized to a nxnxm grid in three-dimensional space. The start position for each aircraft in the fleet is labeled as 'O' and the end position is labeled as 'X'. Each cell, inside the grid airspace, is assigned a color corresponding to the weather's risk value. In the aviation sector, the risk levels are:

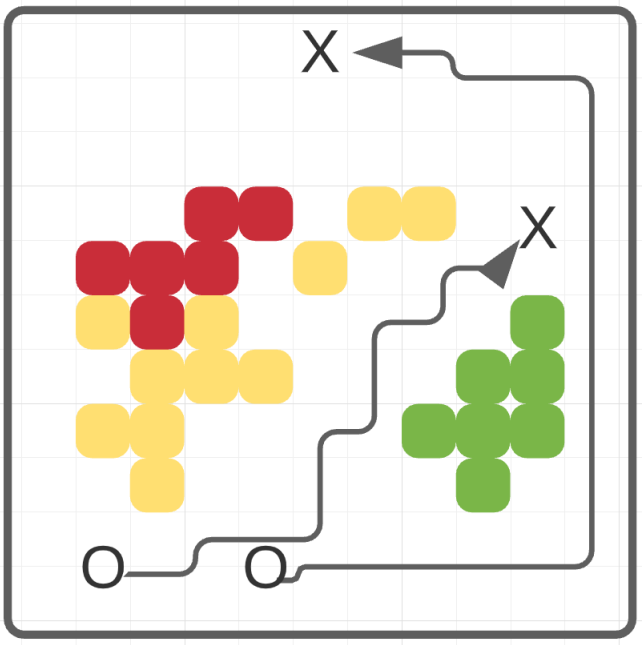

Figure 1: State representation of the optimal trajectory planning problem for multiple aircraft.

level 1 (green or low risk), level 2 (yellow or medium risk), and level 3 (red or high risk). Each node on the grid is connected to $8 * K$ neighboring nodes, where K is the connectivity constant.

At each state, the aircraft checks if it has reached its final destination. If the destination has been reached, the path is returned. Otherwise, the aircraft can move to any of the K neighboring nodes that are not occupied by another aircraft.

2.2 State Generation

Initial states for the simulations are generated randomly using two lists, one for the trajectories and one for the weather map, in Python. An input of grid dimension, n, connectivity constant, K, and number of aircraft are used to initialize the airspace grid and action space for each node. The start and end regions for each trajectory are randomly generated so that no aircraft starts or ends on the same node. All aircraft are initialized once using the state generation function (no new aircraft are generated at a later time step). Weather region risk values are randomly generated and do not change with time.

2.3 Simulated Annealing

2.3.1 Why Hill Climbing is Not Optimal

This form of the optimal trajectory planning problem is non-convex, there may be multiple feasible regions and multiple locally optimal points within each reason. For example, a local cost function minimum may be reached by achieving paths with minimum distance but another local minimum exists when a minimum weather risk is reached. Therefore, the hill climbing algorithm would most likely not generate the globally optimal solution to the problem. Since simulated annealing uses probabilistic techniques to approximate the global optimum, it will likely provide a better optimal path when compared to hill climbing. This section describes the implementation of the simulated annealing algorithm for optimal trajectory planning for multiple aircraft.

2.3.2 Cost Function

The state cost function will is the input state for the simulated annealing algorithm. For each aircraft, the total distance traveled and the risk value of the weather regions the aircraft flew through are used to quantify the path's quality. The cost function, for a single aircraft, after action step, i, is shown in Equation 1.

$$c_i = c_{i-1} + a_1 * d + a_2 * r \tag{1}$$

Where c_i is the total cost at the current state, c_{i-1} is the previous state's cost, d is the distance traveled in the action step, r is the sum of the four surrounding risk values, and a_1 and a_2 are the weights for distance and weather risk costs. The weights are set to 1 by default. At any action step, i, the state cost function is the sum of all cost functions for n aircraft (Equation 2).

$$x_i = \sum_{j=1}^{n} c_j \tag{2}$$

Where x_i is the state cost function and input to the simulated annealing algorithm, j is the index of a single aircraft, n is the total number of aircraft, and c_j is the cost function value of aircraft j at the current action step, i.

Based on the performance of the simulated annealing algorithm, the cost function weights may be adjusted to attribute more weight to risk value or distance travelled. To minimize the distance travelled and risk associated with each trajectory, the simulated annealing algorithm seeks to minimize the total path cost (Equation 2).

2.3.3 Input State

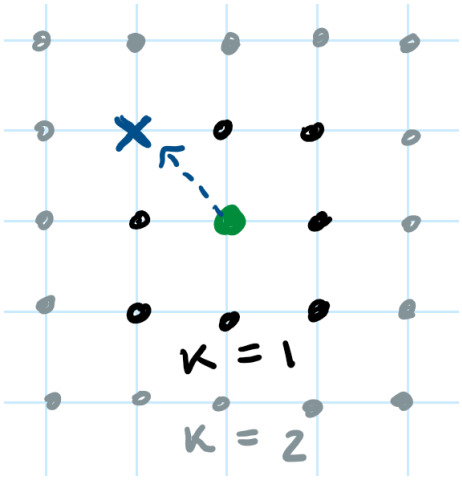

Figure 2: The action space for a single aircraft with connectivity constant, k. X is the new aircraft position after the randomly chosen action is taken.

The input state for the simulated annealing algorithm is the set of aircraft trajectories, which are randomly defined by the state initialization function.

When calculating the initial state, the state initialization function computes a set of random trajectories for each aircraft for a series of time steps. The set of trajectories is calculated as follows: the cost function for the initial aircraft positions is evaluated. In Equation 1, c_{i-1} would be 0, d would be 0, and r would be the sum of the four neighboring weather risk values. Then Equation 2 would be used to sum across all aircraft. For each step in the trajectory decision, the following values are used to determine the next action: a set of all possible actions, determined by the connectivity constant, K, if the aircraft has reached its final destination, and if another aircraft is occupying one or more of the potential movement locations (Figure 2). If the aircraft has reached its final destination, it cannot take any actions and the function moves to the next aircraft. Any actions that lead to a

position occupied by another aircraft are removed from the list. Then the random.randint() Python function is used to select a random action from the available action space. This decision cycle is repeated for each aircraft in the airspace and for each time step. If all aircraft have reached their final destination, the total cost (Equation 2) and input state are returned.

2.3.4 Actions

The next state function for the simulated annealing algorithm randomly selects a time step to change the list of trajectories from. The length of time from the current time step in which the new time step can be generated is designated by the user (similar to next_state_left_right from Assignment 1). Starting at this time step, the trajectories for each aircraft are recomputed, using the method described above. The time step used to randomly compute new trajectories is saved and reused when the next state function is called to calculate a new time step and state. The temperature schedule then uses the total cost function / state value to determine if the next state of trajectories should be accepted as the current state. This process will is repeated until the cost function is minimized.

2.3.5 Temperature Schedule

Based on my results from Assignment 1, I decided to run the simulated annealing algorithm with the exp_temp_schedule_function, with a small decay factor and high initial temperature. An analysis of the effect of the temperature schedule settings was performed.

2.3.6 Evaluating the Results

To evaluate the solution generated by the simulated annealing algorithm, a visualization of the solution path was generated for each test case and compared to the path generated by hill climbing. Since hill climbing often does not find a global minimum for non-convex problems, it will provide a baseline for the algorithm's performance.

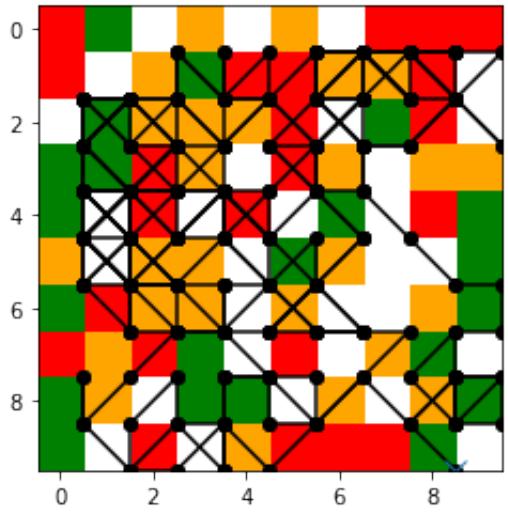

Figure 3: The initial state trajectory for a single aircraft, generated randomly.

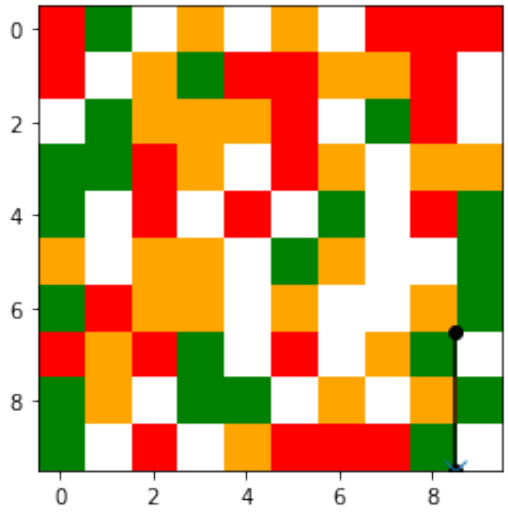

Figure 4: The optimal state trajectory for a single aircraft, generated by the simulated annealing algorithm.

3 Evaluation Methodology

To evaluate the solution generated by simulated annealing, hill climbing was applied to the same problem and the results were compared [1]. Each solution can be plotted to see the path generated by the algorithm. Figure 3 shows an example of the randomly generated initial state. The colors on the map correspond to the degree of weather risk (white = 0, green = 1, yellow = 2, and red = 3).

Figure 4 shows the trajectory after running the simulated annealing algorithm. The total cost function across all aircraft will be used to compare trajectories. The approximate for the globally optimal strategy is calculated by summing the distance from the starting node to the ending node for each aircraft trajectory.

A table of solution costs was generated for four separate initial states: generated by varying the number of aircraft, the side of the grid, and the connectivity constant. For each of these initial states, the simulated annealing algorithm was run 10 times for each setting and the total cost is averaged. This result is compared to the hill climbing algorithm's average total cost after 10 iterations, and the optimal strategy.

4 Results

Table 1 shows the results for an initial problem state with one aircraft, a small grid size, and 8 node connectivity for state transitions. The simulated annealing algorithm was run with settings which vary initial temperature and window size. A solution is considered close to optimal if its cost is less than 100 times the optimal strategy (calculated from the shortest distance between the start and end nodes).

Table 2 shows the results for an initial problem state with a larger number of aircraft, a small grid size, and 8 node connectivity for state transitions. The simulated annealing algorithm was run with settings which vary initial temperature and window size. A solution is considered close to optimal if its cost is less than 100 times the optimal strategy (calculated from the shortest distance between the start and end nodes).

Table 3 shows the results for an initial problem state with a larger number of aircraft, a larger grid size, and 8 node connectivity for state transitions. The simulated annealing

Strategy	Average Solution Cost	Percent Optimal (less than 100 times optimal strategy)
Optimal Strategy	1.732	100%
Hill Climbing	222.5	40%
SA: T = 1000, wind_size = 10, a1 = 1, a2 = 1	163.6	50%
SA: T = 10000, wind_size = 10, a1 = 1, a2 = 1	124.5	90%
SA: T = 10000, wind_size = 50, a1 = 1, a2 = 1	350.7	30%

Table 1: Average solution cost for initial state: n_aircraft = 1, n = 5, and k = 1.

Strategy	Average Solution Cost	Percent Optimal (less than 100 times optimal strategy)
Optimal Strategy	10.60	100%
Hill Climbing	1068	60%
SA: T = 100, wind_size = 10, a1 = 1, a2 = 1	993.1	70%
SA: T = 1000, wind_size = 10, a1 = 1, a2 = 1	960.3	60%
SA: T = 1000, wind_size = 50, a1 = 1, a2 = 1	949.9	70%

Table 2: Average solution cost for initial state: n_aircraft = 5, n = 5, and k = 1.

Strategy	Average Solution Cost	Percent Optimal (less than 200 times optimal strategy)
Optimal Strategy	14.81	100%
Hill Climbing	6160	0%
SA: T = 100, wind_size = 10, a1 = 1, a2 = 1	6086	20%

Table 3: Average solution cost for initial state: n_aircraft = 5, n = 10, and k = 1.

Strategy	Average Solution Cost	Percent Optimal (less than 100 times optimal strategy)
Optimal Strategy	2.449	100%
Hill Climbing	112.3	80%
SA: T = 1000, wind_size = 10, a1 = 1, a2 = 1	48.27	100%

Table 4: Average solution cost for initial state: n_aircraft = 1, n = 5, and k = 5.

algorithm was run with a low enough initial temperature and window size to be computationally efficient. A solution is considered close to optimal if its cost is less than 200 times the optimal strategy (calculated from the shortest distance between the start and end nodes).

Table 4 shows the results for an initial problem state with one aircraft, a small grid size, and 40 node connectivity for state transitions. The simulated annealing algorithm was run with a low enough initial temperature and window size to be computationally efficient. A solution is considered close to optimal if its cost is less than 100 times the optimal strategy (calculated from the shortest distance between the start and end nodes).

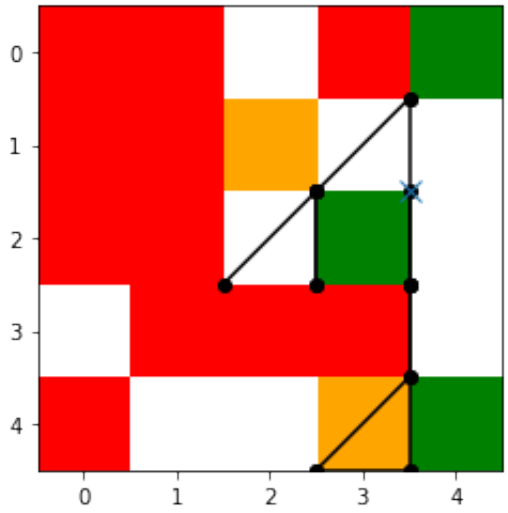

Figure 5: The trajectory calculated by the simulated annealing algorithm with a cost function only dependant upon weather.

Figures 5 and 6 show the effect of changing the cost function weighting parameters. The initial problem state was defined with one aircraft, n = 5, and k = 1 and the simulated annealing algorithm was run with an initial temperature of 100 and a window size of 10. In Figure 5, a_1, the distance weighting parameter is set to zero, so the cost function only depends on weather. In Figure 6, a_2, the weather risk weighting parameter is set to zero, so the cost function only depends on distance.

5 Discussion

From Tables 1 and 2, the simulated annealing algorithm achieves the closest to globally optimal solutions when the number of aircraft is small. Since the optimal strategy calculation was computed using distance only and, therefore, does not account for aircraft obstructions or weather-related constraints, it is expected that solutions for multiple aircraft will be much greater than the optimal strategy. In addition, both tables show that the hill climbing algorithm consistently calculates sub-optimal solutions for all variations in fleet size.

From varying the simulated annealing algorithm's parameters in Table 1, it is shown

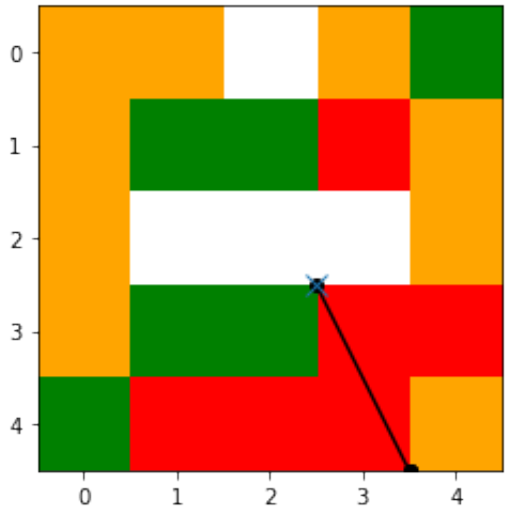

Figure 6: The trajectory calculated by the simulated annealing algorithm with a cost function only dependant upon distance.

that peak performance is achieved with a large initial temperature and medium window size. Smaller window sizes were tested, but exhibited results similar to the window size of 10. The increase in solution optimality with large initial temperatures matches the conclusion drawn from Assignment 1. This optimality is achieved with a decrease in computational efficiency, evident in the run time required to compute the trajectories.

From varying the simulated annealing algorithm's parameters in Table 2, temperature and window size had less of an effect on the solution's optimality, when compared to Table 1. This means an increase in the number of aircraft will likely lead to a solution that is about 60%-70% optimal, regardless of the parameter values. Since hill climbing achieves results close to that of simulated anenaling, problems with small airspace size and a large number of aircraft can achieve comparable optimality and an increase in computational efficiency when hill climbing is used to calculate the solution.

Table 3 shows that a problem with a large number of aircraft and large airspace size leads to a significant decrease in performance for the hill climbing and simulated annealing algorithms. The simulated annealing algorithm still outperforms hill climbing, reaching up to 20% optimal. In addition, very large initial temperatures cannot be used because of the time complexity of the new state calculation function. Since a large number of aircraft are

initialized, changing the parameters of the simulated annealing algorithm will not have a significant effect on its performance, as shown in the previous Table.

Table 4 shows the effect of a large connectivity constant between states. A connectivity constant of 5 allows an action to move the aircraft five states out, into one of 40 possible locations. Increasing the connectivity constant significantly increases the percent optimality of both the hill climbing and simulated annealing algorithms, with the simulated annealing algorithm achieving relative optimality in 100% of the test cases. This result is as expected because an increase in connectivity would increase the probability of finding the end position from the current state.

Figures 5 and 6 show how altering the cost function weights change the path generated by the simulated annealing algorithm. Figure 5 is calculated by setting the distance weight to zero. Therefore, the path achieves optimality when it passes through minimal weather risk values. The figure shows that the path is not the minimum distance but regions of inclement weather are avoided. Figure 6 is calculated by setting the weather risk weight to zero. Therefore, the path achieves optimality when it achieves the minimum distance between start and end points. The figure shows that the path does not avoid high-risk weather regions.

6 Conclusion

In this assignment, the multi-trajectory path planning problem for UAM vehicles was solved using simulated annealing. When compared to the baseline hill climbing algorithm, simulated annealing consistently achieved a higher percent optimality. The simulated annealing algorithm can be improved further by using a large initial temperature and increasing the connectivity between states. The formulated cost function can be adjusted to prioritize weather risk or total distance travelled, which is useful for scheduling flights which have high-risk cargo or strict time constraints.

7 Literature Cited

[1] https://github.com/jujubriden/TrajectoryPlanning