Amazon Prime Air Traffic Control

Behkish Nassirzadeh
ECE 457A - Group 12

Amish Patel
ECE 457A - Group 12

Chang Bok Lee
ECE 457A - Group 12

David Hong
ECE 457A - Group 12

Abstract — **This report focuses on the comparative study of different optimization techniques to solve the Amazon air traffic control problem. The optimization techniques that are examined are genetic algorithm and ant colony optimization. For each technique, the problem is defined and is executed based on the same initial conditions. The algorithms are evaluated on a 3D coordinate search space of a fixed sized. The performance of each technique is compared by executing the algorithm while adjusting algorithm specific parameters. Based on the results, an efficient solution is determined.**

I. Introduction

As a leader in Internet retail, Amazon is constantly coming up with new ways to improve its service for their customers as online shopping becomes more popular. One area that Amazon has focused on is its shipping service. Currently, the fastest possible shipping time is one day shipping. However, most customers prefer to have their packages delivered immediately. In order to fulfill this desire, Amazon introduced a service called Amazon Prime Air. Amazon Prime Air allows customers to receive packages in 30 minutes. This is acheived by the use of multiple quadrotors delivering packages from the Amazon warehouse to the customers. Although this can significantly improve delivery time, a major problem is controlling possible collisions between quadrotors in 3-dimensional space. The goal is to find the shortest path to its destination, while not interfering with other quadrotors. Unlike the traditional surface logistic services, the flying quadrotors operates in 3-dimensional space which allow the quadrotors to be highly efficient. One of the main challenges for operating multiple quadrotors is to avoid collisions between each other and other obstacles. The path planning requires metaheuristic data that can estimate the distance between a start point and an endpoint, the average altitude, the average fuel consumption, etc.

II. Literature Review

One solution, as presented in Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning [1], is to use an optimization algorithm with a 2D matrix which contains a grid representation of search space. Two 2D matrices are used in their implementation. The first matrix uses an approximate cell decomposition of the terrain using a 2D grid where each element of the matrix represents the elevation of the terrain. The second matrix uses the trajectories generated by the optimization algorithm that are composed of line segments and encoded in a matrix where each row represents the coordinates of the ith waypoint. All trajectories are flown at constant speed. Our proposed solution uses a similar representation of the given search space, but differs by using a 3D matrix instead. The proposed solution divides the search space into small cubes, each cube assigned with a single (x, y, z) coordinate. It constraints m number of drones to occupy a single cube trajectory space for collision avoidance.

III. Problem Formulation and Proposed Solution

A. State

The state represents the path of each quadrotor, given in (xi, yi, zi)d coordinate system. The path will contain all the coordinates in an incremental step that will take the quadrotor from the start position to the goal position. The subscript i in the coordinate system represents the current step for the given quadrotor, d. Figure 1 defines the matrix representation of the state and the coordinate taken by the given quadrotor.

$$S = \begin{bmatrix} d_1 & d_2 & d_3 & \dots & d_n \end{bmatrix}$$

$$\mathrm{d}_i = \begin{bmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_k & y_k & z_k \end{bmatrix}$$

Figure 1. Matrix representation of states

B. Initial State

The initial coordinates are generated to randomly without checking for their feasibility.

C. Goal

All quadrotors have paths from a start point to an end point assigned, with no collision and minimum cost.

D. Actions

To move from one grid to another, we move from one coordinate to another via increment and decrement of coordinate component. Each transition between the grid boxes can only be made to an adjacent grid box that is 1 box away from the current grid box.

- Increment - One of the component values within the current position coordinate can be incremented by 1. ex) {1,5,1} -> {1,4,1}
- Decrement - One of the component values within the current position coordinate can be decremented by 1. ex) {6,3,4} -> {5,3,4}

E. Cost

The optimized path will use the minimum cost to deliver a package. The cost function includes CLength, representing the total distance travelled by a quadrotor.

$$\text{Cost} = C_{Length}$$

C_{Length} is defined as the total distance a quadrotor must travel.

F. Initial Solution

$S_{initial}$ = [{0,0,0}, {0,0,1}, {0,1,1}, {0,2,1}, {1,2,1}, {1,2,2}, {1,3,2}, {2,3,2}, {2,2,2}, {3,2,2}, {4,2,2}, {5,2,2}, {5,2,1}, {5,2,0}]

G. *Suitable* Solving Strategy

Two optimization algorithms are considered to solve the presented problem; a Genetic algorithm and an Ant Colony Optimization algorithm. The genetic algorithm is suitable as it is a population based non-deterministic optimization method. The GA simulates how pathing of quadrotors can be improved by applying a series steps similar to biological evolutionary steps taken in nature. The ACO algorithm is another population based probabilistic optimization method. The ACO can find a path to the destination grid with the minimum cost after number of iterations.

H. Parameter Selection

- Genetic Algorithm

Table 1. GA parameter values

Parameter	*Value*
Selection Type	Elitism
Crossover Type	1-Point
Crossover Probability	0.6
Mutation Probability	0.25
# of generations until terminate	25
# of population per generation	20

- Ant Colony Optimization

Table 2. ACO parameter values

Parameter	*Value*
Pheromone Update Type	Ant Density Model
Initial Pheromone	50
M	100
α	1
β	1
q_0	0.5
ρ	0.1

I. Genetic Algorithm

The initial set of solutions is generated randomly and will be tested for its fitness to find the best solution. Each generation of solutions is first tested for its fitness. The fitness test includes collision test to make sure no more than 10 quadrotors are sharing the same 3D space at the same time. Once the solution passes the fitness test, the solution path goes through crossover and/or mutation in attempt to generate a better solution. The better solution is defined as the solution that uses less cost compared to the current cost, while also passing the fitness test.

The initial version of crossover strategy is based on the 1-point crossover using two drones from different parents. First, extract the starting half sub-list of the flying route of a drone from one parent. Then the ending half is extracted from the

other parent. Since the system has a constraint of moving one box at a time, it is necessary to 'glue' those two halves together. Unfortunately, this introduced higher cost as the total length of the flying route almost always increases. The second approach for crossover tries to resolve this problem by considering the set of drones as a gene rather than considering each drone as a gene. The second strategy, crossover2, sort the drones based on the length of the flying route and take the best half from each parent. Since it is now just mixing two sets of different drones into one, gluing is not required for this approach. This seems to work generally as the system produces better result as the generation passes by.

J. Ant Colony Optimization

Similar to the GA, a number of solutions are generated randomly initially. Based on probability, ants can choose to follow the generated solution path or move to a random adjacent grid inside the search space. Whenever ants move from one grid to another, pheromones are left along the path. The pheromones have a pre-defined evaporation rate, and ants will eventually follow the path with the strongest pheromone to reach the destination.

IV. Performance Evaluations

Table 3. Performance Evaluations for the GA and the ACO implementation

Algorithm	*Top Score 1*	*Top Score 2*
GA	9226	10534
ACO	22231	24854

The metric has been determined based on the total length of the path found. Lower values indicate better performance.

V. Conclusion and Recommendation

Based on the performance evaluation, the GA performed better. The ACO wasn't exactly suitable for the problem, as it's better suited to find the shortest path. The project problem involved not only finding the best path from a starting grid to a goal grid, but also had a constraint where only 10 quadrotors can be in the same grid at any given time. It was difficult to introduce such constraint with the ACO. The parameter selection also posed another difficulty with the ACO. A low q0 value resulted in a biased solution to the first randomly generated path. A high q0 value showed an explorative behavior and resulted in a random solution.

The biggest challenge, based on the current problem formulation, was generating a path that does not backtrack. Initially both algorithms were written to not consider a grid that was visited previously. The team soon realized that both algorithms get stuck during runtimes as there were cases where all the neighboring grids were visited and the quadrotor could not make a move that is 1 grid away from its current location. In order to get around this behavior, both algorithms were modified to allow backtracking. One recommendation to improve both algorithms is to use a Tabu list. A straight-line length from the current grid coordinate and the goal grid coordinate can be easily calculated. If the algorithms were allowed to backtrack only if the next move yields a better heuristic value, better performance values would be obtained.

References

[1] Roberge, V., Tarbouchi, M., & Labonte, G. (2013). Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning. IEEE Transactions on Industrial Informatics, 9(1), 132-141. doi:10.1109/tii.2012.2198665.

[2] Kwok, D., & Sheng, F. (n.d.). Genetic algorithm and simulated annealing for optimal robot arm PID control. Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence. doi:10.1109/icec.1994.349971.

[3] A design of self-tuning PID controllers using a genetic algorithm. (1999). Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251). doi:10.1109/acc.1999.783590.