Optimization Techniques for Multi Cooperative Systems MCTR 1021
Mechatronics Engineering
Faculty of Engineering and Materials Science
German University in Cairo

UAV-UGV Cooperative Area Exploration

By

Team 44

Ahmed Daw
Amr Mahgoub
Nabil Hassan
Mohamed Ashraf
Mohamed Yasser
Abdulrahman Manea

Course Team

Assoc. Prof. Dr Omar Shehata

Chapter 1

Literature Review

The field of optimizing cooperative UAV-UGV area exploration has been extensively researched throughout the last decade, in terms of increasing explored area and reaching desired targets in minimal running time.

Xu et al, 2020 explored the field cooperative path planning for multi-UAV system as used in military and civil applications, such as target striking and regional surveillance, in order to increase the probability of task completion and minimizing the risk of the UAVs being captured, by proposing an optimized cooperative path planning for multi-UAV system optimizing a combination of fuel/energy consumption and risk of capture, under the constraints of exploration area and time, establishing a multi constraint objective optimization model, followed by the testing of multiple optimization algorithms, such as an improved version of the grey wolf optimization algorithm, where the algorithm is improved in three main aspects, population initialization, decoy factor updating and individual position updating, simulation results showcase that the algorithm has optimized the generated paths by lowering the path cost, while reaching faster convergence when compared to other tested techniques [1].

In their research, Qiming et al, 2021 have evaluated the use of different meta-heuristic optimization techniques in solving the UAV swarm search task. In their research, the goal was to detect targets as fast as possible, while using the least number of UAVs, in another simulation environment, the goal was to maximize the search area with the UAV swarm. The search area was divided into discrete cells of MxN size and the used UAV model was assumed to be the basic UAV model by treating each UAV as a particle in a 2-Dimensional space. Some of the constraints taken into consideration were the curve angle θ, flight speed v, flight altitude h, the table shown in Table: 1.1 provides more constraints.

Constraint name	**Expression**	**Meaning**
Curve angle constraint	$\theta_{\min} \leq \theta_i \leq \theta_{\max}$	θ_i: turning angle of the UAV at time i $\theta_{\min}$: minimum turning angle $\theta_{\max}$: maximum turning angle
Flight speed constraint	$v_{\min} \leq v_i \leq v_{\max}$	v_i: speed of the UAV at time i $v_{\min}$: minimum speed of UAV $v_{\max}$: maximum speed of UAV
Flight altitude constraint	$h_{\min} \leq h_i \leq h_{\max}$	h_i: altitude of the UAV at time i $h_{\min}$: minimum flight altitude $h_{\max}$: maximum flight altitude
Climb angle constraint	$0 \leq \alpha_i \leq \alpha_{\max}$	α_i: UAV climbing angle at time i $\alpha_{\max}$: maximum climbing angle
Subduction angle constraint	$0 \leq \beta_i \leq \beta_{\max}$	β_i: UAV subduction angle at time i $\beta_{\max}$: maximum subduction angle
Communication constraint	$d_{ab} \leq r$	d_{ab}: distance between two UAVs r: communication radius
Boundary constraint	$l_i \in D$	l_i: UAV location D: restricted area
Collaborative constraint	$d_{\min} \leq d_{ab} \leq d_{\max}$ $t_a \leq T$	d_{ab}: distance between two UAVs $d_{\min}$: shortest distance between UAVs $d_{\max}$: maximum flight range t_a: time of arrival a T: latest time of arrival a

Table 1.1: UAV Constraints Table

In their evaluation, the researchers tested the Genetic Algorithm (GA), Ant Colony (AC), Particle Swarm (PSA), with other optimizing algorithms, in GA, when minimizing the time was taken as the objective function, the total task time was reduced by 65%, however, the algorithm needed a certain number of UAVs and showed linear increase in energy consumption as the number of UAVs increases [2].

In their article, Ramasamy et al, 2022 have explored parameter tuning using genetic algorithm with a set of constraints such as fuel limitations of the UAVs and the speed limitations of the UGVs, reducing the time and/or fuel consumption were important in assessing the validity of the algorithms, taking into consideration the resource and terrain constraints. The paper compared between the Genetic Algorithm and the Bayesian Optimization, despite the fact that the GA provided relatively similar results to BO, it required 3 times more local-search optimization and required 6 times the computation time [3].

In their research, Lui et al, 2016 used quantum ant colony algorithm (QACA), to optimize the path taken in evacuation practices, while being robust and and high efficiency, comparing their findings with the traditional Ant Colony Algorithm (ACO). In their research, minimizing the time of evacuation evacuation path optimization was critical in assessing the algorithm, as less time equates to less human losses in cases of environmental crisis. The primary objective is to consume as little time as possible to evacuate all evacuees from the danger zones to safe zones, making the time the most significant factor to be considered, another objective was to minimize total density in the paths. There were three benchmark functions used to compare the results of QACA and ACO, where the tests showed that QACA was more efficient in solving the problem, as well as expand the solution as the iterations advance [4].

In their paper, Wu et al 2020 discussed the same topic of cooperative UAV-UGV exploration, but in surveillance applications. In this paper, the path planning problem was formulated to be a 0-1 optimization problem, in which the on-off states of the discrete points are to be optimized. A hybrid algorithm, combining the Estimation of Distribution Algorithm (EDA) and the Genetic Algorithm (GA) was proposed to solve the problem. The goal is to minimize the required time by the vehicles to complete a circular path, where the objective function can be expressed as shown in equation 2.1.

$$J = max(\frac{D_u}{N_u * v_u}, \frac{D_g}{N_g * v_g}) \tag{1.1}$$

Where $D_u \& D_g$ are the lengths of the circular paths, and $N_u \& N_g$ are the number of drones and UGVs and $v_u \& v_g$ are their velocities.

The decision variables are the on-off states of the active points, where each point can either be ”open” (1) or ”closed” (0), and open points must be covered during exploration.

The constraints for UAVs in this paper lied in avoiding collisions with buildings, which are represented as inaccessible grids, also, the vehicles must cover all assigned 2D grids to ensure complete coverage, where the on-off states must satisfy the coverage constraint. In their results, they validated the superiority and rationality of the proposed hybrid algorithm, where the workload can be balanced between the UAVs and UGVs, moreover, the cooperative planning yielded better results than using either systems alone, by significantly reducing the total surveillance time, the suggested algorithm combined the strengths of EDA and GA's strengths, resulting in minimizing the time of task completion, while increasing adaptability since the path is being assessed during runtime rather than before runtime, increasing flexibility in real-world applications [5].

Chapter 2

Methodology

The objective of this project is to develop an efficient algorithm for optimizing the flight paths of multiple Unmanned Aerial Vehicles (UAVs). The UAVs must navigate a defined area while reaching designated targets and avoiding restricted zones. The optimization seeks to minimize the exploration time, maximize coverage efficiency, and ensure UAVs reach their assigned targets while avoiding crossing into prohibited areas, the overall problem aims to minimize exploration time, increase coverage efficiency, the penalty for not reaching designated targets, and the penalty for entering restricted areas, with more importance on time of exploration, in simulation, it can be seen that the UAVs try to converge to the targets as quickly as possible.

2.1 Objective Function

The cost function aims to maximize the explored area while minimizing the time taken, subject to penalties for constraint violations (e.g., communication and collision avoidance).

$$ObjectiveFunction = \alpha * T_{exploration} - \gamma * C_{efficiency} + \delta * Penalty_{targets} + Penalty_{restricted} \tag{2.1}$$

Where:

- α, γ and δ are weight parameters for the time of exploration, Coverage efficiency and designated targets penalty.
- $T_exploration$ is a measure of the total time taken by UAVs to reach their designated targets, represented as a sum of the time taken by each UAV based on their velocity and the distance covered.
- $C_{efficiency}$ is a ratio between the area covered by the UAVs with the total area of the map.

- $Penalty_{targets}$ is a penalty term that accounts for the distance between the UAV positions and their targets, encouraging the UAVs to stay close to the targets.
- $Penalty_{restricted}$ is a high penalty, which is applied to solutions where the solutions overlaps with the restricted area.

Decision Variables

The decision variables consist of the velocities of the UAVs in the x and y directions, in m/s, as well as their positions in x and y in m.

$$x_i = [v_{xi}, v_{yi}, p_{xi}, p_{yi}] \tag{2.2}$$

where:

- v_{xi}, v_{yi} are the velocities in x and y directions (in m)
- p_{xi}, p_{yi} are the positions in x and y (in m)

Constraints

The constraints are handled through a nonlinear constraint function, which is defined as follows:

$$c = d_{safe} - distance(UAV_i, UAV_j) \tag{2.3}$$

where:

- d_safe is a predefined safe distance in the code
- $distance(UAV_i, UAV_j)$ is the euclidean distance between UAV_i and UAV_j

2.1.1 Feasibility Checks

The feasibility check is done for two things:

- **Distance checks:** where the function checks if the distance between the UAVs is within the safe region, to avoid collisions.
- **Restricted Area Checks:** where the function checks whether a UAV is planning to go to the restricted area, if it plans to, a penalty is applied to the solution.

2.2 Simulated Annealing

2.2.1 Test Cases & Evaluation

Overall, the simulated annealing succeeded in minimizing the objective function, reaching convergence where all UAVs reach a target, and we tested the effect of using different weights on the performance of the algorithm.

The initial positions of the UAVs, the targets and the restricted area locations are randomized at every run, however, running the code multiple times provides very similar results.

In all tests, there was 1 obstacle, with 3 UAVs and 3 targets, linear cooling rate was 1, and α of the cooling is 0.99.

Case 1: Constant max velocity, T_{max}, unchanged tolerance

In this case, which is our benchmark in this analysis, the maximum number of iterations in the large loop was 1000, and in the small loop was 100, the weights were $\alpha = 1, \gamma = 0.5 and \delta = 2$, in this run, the objective function decreased from around 970 to reach a final value of 21.27, where all UAVs reached the targets in the accepted tolerance, Figure 2.1 shows the objective function curve, the initial and final positions of the UAVs along with the restricted area, in each iteration, these parameters are randomized, however, the cost function would converge to the same value

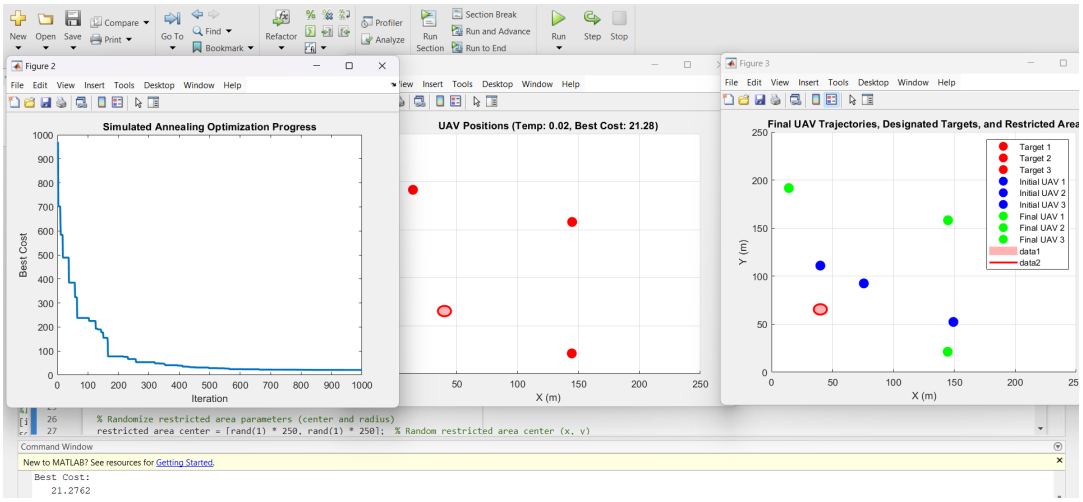

Figure 2.1: The benchmark test

In the second test, $\gamma = 0.5, \delta = 1.5, \alpha = 3$, the objective function reached a minimum of 63.7141, starting from 790, showing a slightly lower convergence, but the UAVs still reached the targets, as shown in Figure: 2.2.

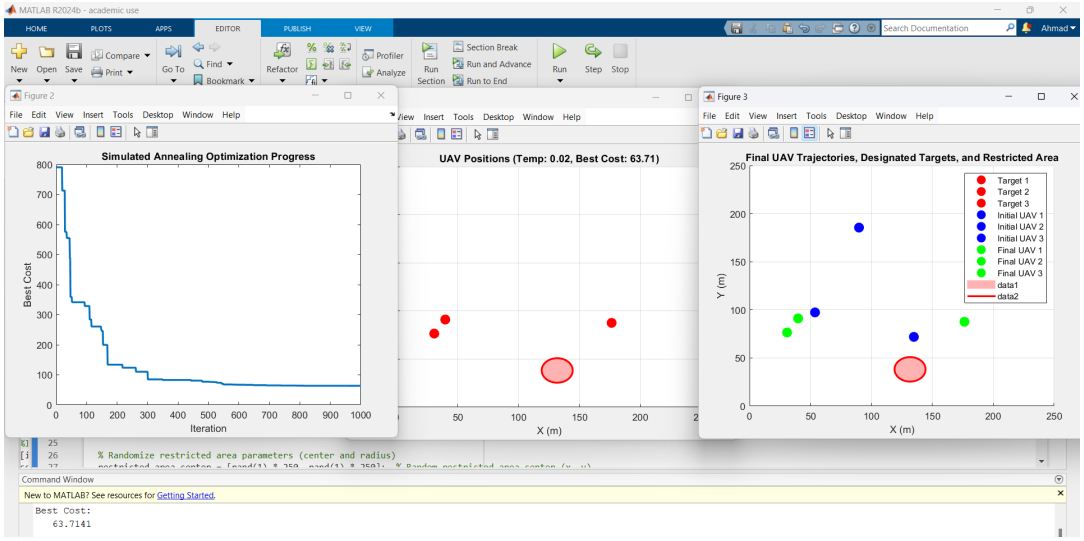

Figure 2.2: Test 2

In the third test, $\gamma = 0.5, \delta = 4.5, \alpha = 3$, the objective function reached a minimum of 63.74, starting from 1600, showing a greater decrease and convergence, the higher initial value is due to the increased value of the weight δ, which is a weight to one of the terms that we want to maximize, the UAVs still reached the targets, as shown in Figure: 2.3.

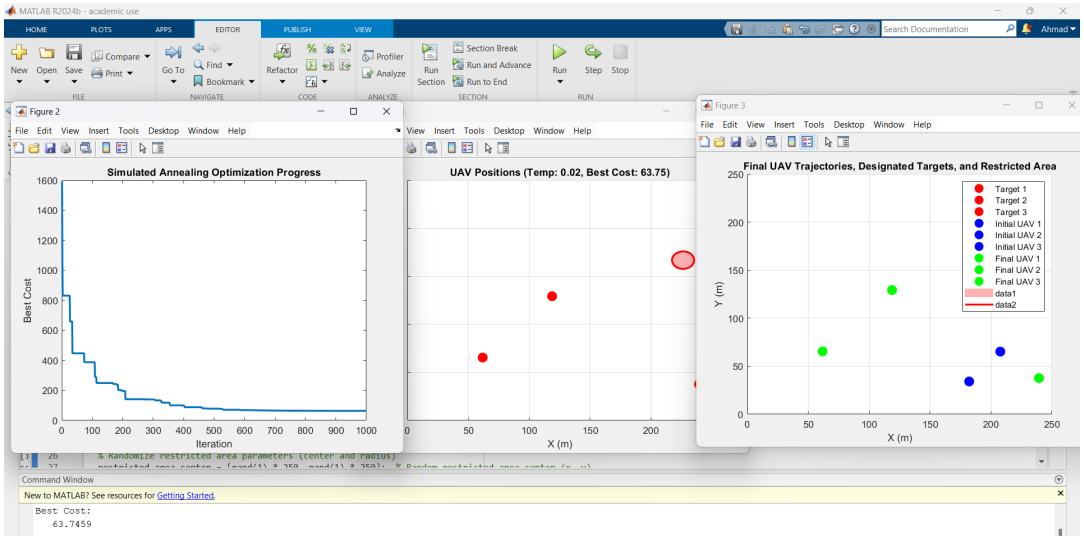

Figure 2.3: Test 3

In the third test, $\gamma = 0.5, \delta = 9, \alpha = 1$, the objective function reached a minimum of 21.4629, starting from 1700, showing a greater decrease and convergence, the higher initial value is due to the increased value of the weight γ, giving more importance to target reaching,

showing very similar convergence, but from a much higher starting value for the objective function, the UAVs still reached the targets, as shown in Figure: 2.4.

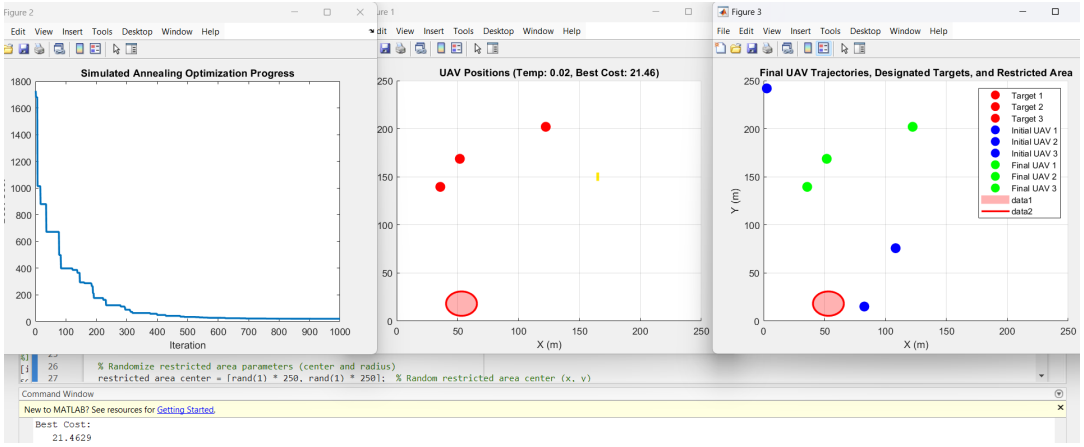

Figure 2.4: Test 4

2.3 Genetic Algorithm

The Genetic Algorithm (GA) follows Charles Darwin's theory of evolution, through survival of the most fit individuals, allowing them to pass their traits to the next generations, in our problem, we followed a similar manner, in initializing a random population to start our GA run, and as iterations progress, the fitness value decreases showcasing the evolution.

2.3.1 Code Flow and Criteria

Our problem required us to run the GA n times per point generated, where n is the number of generations, so in order to generate a single point in the path, we ran the GA algorithm n times.

Each generation consisted of m number of chromosomes, where m was a 4x3 matrix, with the columns representing all the information per UAV, so column 1 holds the information for UAV 1 and so on, and the rows are ordered as follows:[v_x,v_y,p_x,p_y]

Before the very first run of the GA, a random population is generated to fill the first generation, and their fitness values are calculated, until all generations are completed, the UAV positions and velocities are then updated with the best fit chromosomes in that run, to enter the new run and generate another point in the path, until all UAVs reach their desired targets, from each generation, the fitness values are calculated, and the population is sorted based on their fitness value, from best to worst, the chromosomes with the highest fitness, get passed to the next generation, without undergoing any change, the crossover parents are selected from the elite population, where 2 random elite chromosomes are selected to produce 2 offsprings, and the fitness value and feasibility are calculated and checked, if the offspring is feasible, it gets

passed to the next generation, the number of produced offsprings is determined by a variable called GA.CrossoverRatio, which could be changed in the code

Since our problem is an arithmetic problem, the crossover is performed using a form arithmetic recombination as follows:

$$child_1 = \alpha * parent_1 + (1 - \alpha) * parent_2 \tag{2.4}$$

$$child_2 = \alpha * parent_2 + (1 - \alpha) * parent_1 \tag{2.5}$$

where α is an interpolation factor, which is a constant initialized at the start of the code.

Mutation is done in two different locations, to introduce more exploration and avoid early convergence to suboptimal solutions too quickly, firstly, the worst individuals in a generation are mutated by adding a noise to the chromosome's genes, also, mutation occurs to some of the offsprings produced from the crossover, based on the variable value GA.MutationRatio, random children are selected for mutation, the same way the worst chromosomes are mutated, the noise added can be altered using the variable GA.NoiseScale.

Constraints handling and feasibility checks were done in a similar fashion to what was previously explained in earlier sections.

2.3.2 Case Studies

Overall, the GA succeeded in reaching an minimum value, achieving the required functionality, but in some runs, the algorithm gets stuck in a minima, where 2 UAVs reach their designated targets, but the 3rd UAV gets stuck in the obstacle boundary, failing to reach the target, since the algorithm is unable to find a solution with a fitness value better than what was already provided, in this first run, which we will use as our benchmark for testing the effect of changing different parameters on the performance of the algorithm, the number of generations per point was 60, with each population having a population of 20 chromosomes, the elitism ratio was 0.1, the crossover ratio was 0.6 and the mutation ratio was 1-(crossover ratio + elite ratio), with $\alpha = 0.4$ and the mutation noise factor to be 0.35, the step size was 3, all UAVs started from the same position (50,50), and the targets were kept in the same position at every run (200,50), (50,200), (200,200), the obstacle was kept in the same place at every run with location (125,125) and a radius of 10.

For this first run, $\alpha_{fitness}$ = 1, γ = 0.5 and δ = 9, giving greater weight for target exploration, the graphs show that the UAVs reached their targets successfully, reaching convergence in around 6000 generations, as it got stuck in a local minima after around 900 generations, which can be shown in the path coloured in blue, the UAV got stuck trying to find a better solution to get out from the danger zone, until it managed to after 5500 generations , with a best cost value of around 14.0204, as shown in Figure 2.5:

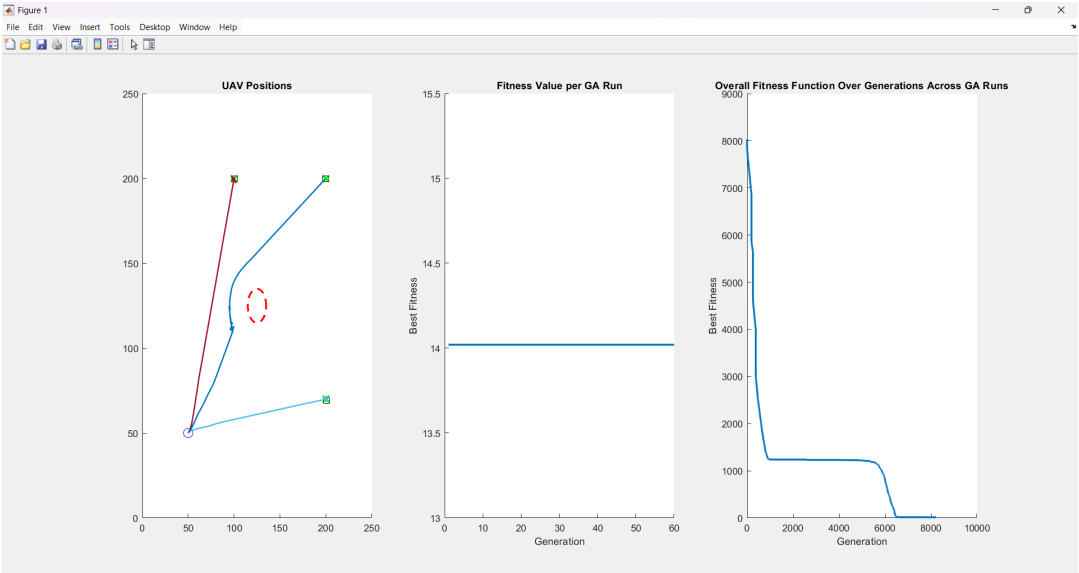

Figure 2.5: The map with UAVs in their target positions and their path plotted (Left), Fitness value per GA run (Middle), the overall convergence curve (Right).

In this second test, we will decrease the population size from 20 to 10, to see the effect on convergence, as shown in the Figure: 2.6

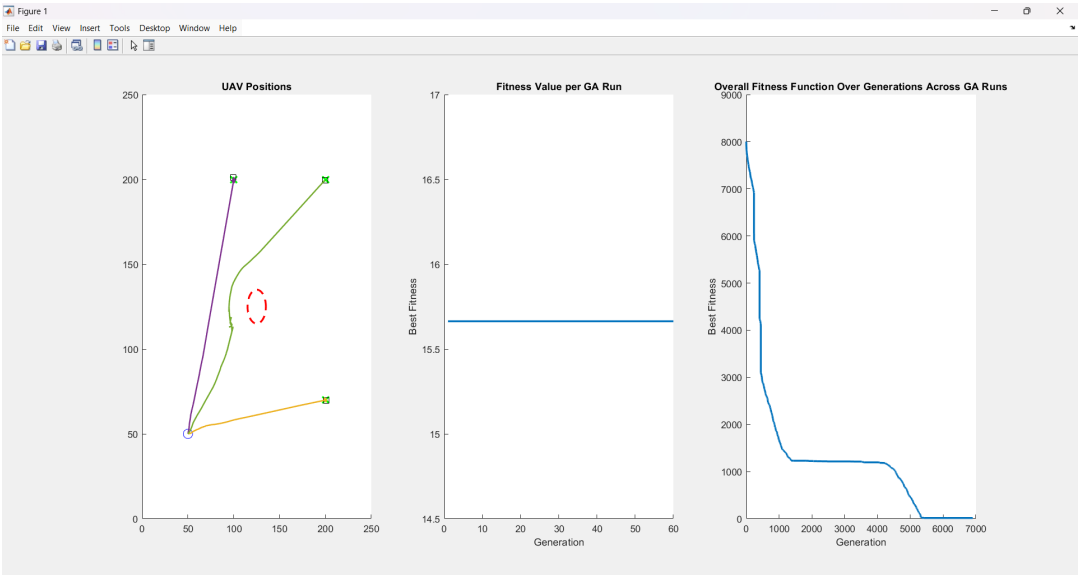

Figure 2.6: The map with UAVs in their target positions and their path plotted (Left), Fitness value per GA run (Middle), the overall convergence curve (Right).

It can be seen that the convergence curve showed very little variation between both runs.

In the 3rd test, we changed the generation size to be 40, instead of 60, and the graphs can be

shown in Figures 2.8 & 2.8:

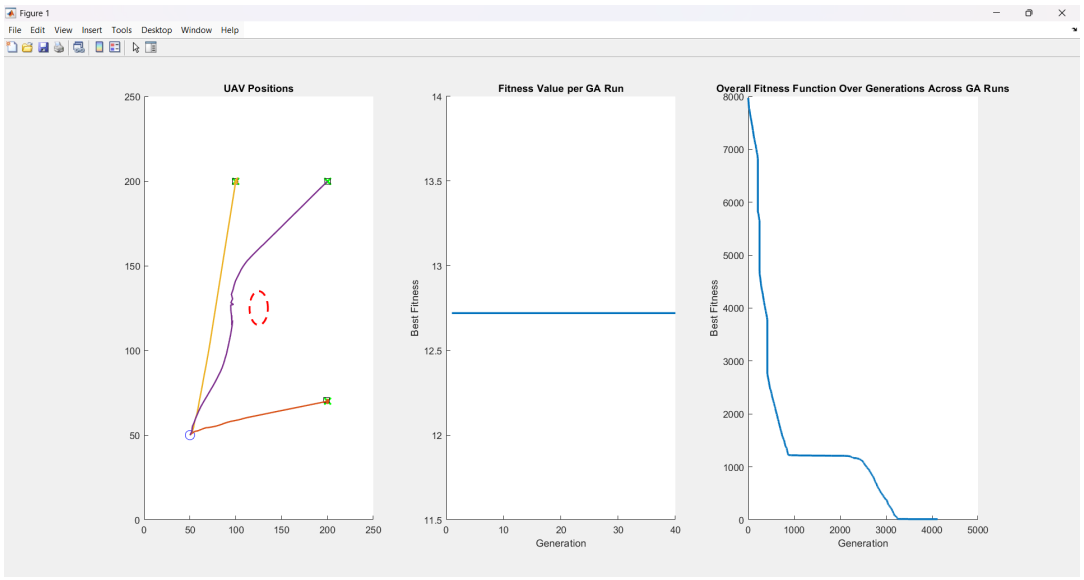

Figure 2.7: The map with UAVs in their target positions and their path plotted (Left), Fitness value per GA run (Middle), the overall convergence curve (Right).

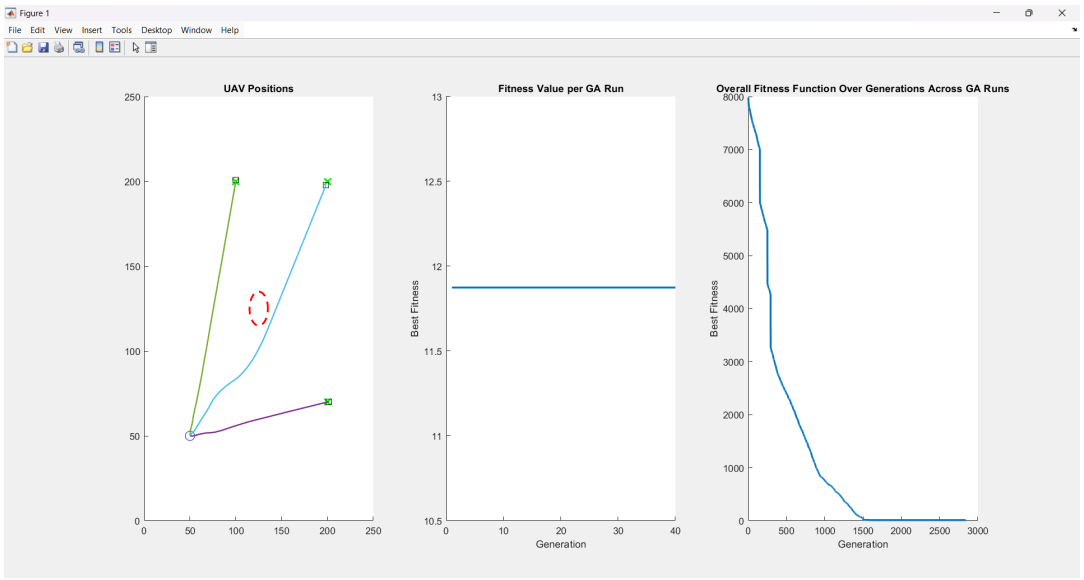

Figure 2.8: The map with UAVs in their target positions and their path plotted (Left), Fitness value per GA run (Middle), the overall convergence curve (Right).

Performance Indices

Standard Deviation, Mean Fitness & Best Overall Fitness:

For the GA, I ran the algorithm 7 times, with the following GA parameters:

- Population size = 20
- Generation size = 60
- Elite Ratio = 0.1
- Crossover Ratio = 0.7
- Mutation Ratio = 0.2
- Noise Scale = 0.35
- α = 0.7

After running the GA code 7 times, the standard deviation value was equal to 1.7420481219483, with a mean of 14.748371428571, the best overall fitness value was 12.034.

Bibliography

[1] Cheng Xu, Ming Xu, and Chanjuan Yin. Optimized multi-uav cooperative path planning under the complex confrontation environment. *Computer Communications*, 162:196–203, 2020.

[2] Zhu Qiming, Wu Husheng, and Fu Zhaowang. A review of intelligent optimization algorithm applied to unmanned aerial vehicle swarm search task. In *2021 11th International Conference on Information Science and Technology (ICIST)*, pages 383–393. IEEE, 2021.

[3] Subramanian Ramasamy, Md Safwan Mondal, Jean-Paul Reddinger, James Dotterweich, James Humann, Marshal Childers, and Pranav Bhounsule. Heterogenous vehicle routing: comparing parameter tuning using genetic algorithm and bayesian optimization. pages 104–113, 06 2022.

[4] Min Liu, Feng Zhang, Yunlong Ma, Hemanshu Roy Pota, and Weiming Shen. Evacuation path optimization based on quantum ant colony algorithm. *Advanced Engineering Informatics*, 30(3):259–267, 2016.

[5] Yu Wu, Shaobo Wu, and Xinting Hu. Cooperative path planning of uavs & ugvs for a persistent surveillance task in urban environments. *IEEE Internet of Things Journal*, 8(6):4906–4919, 2020.