Modeling Air Pollution Using Environmental Drones

Team 38

Bilal Mufti, Oojas Salunke, Joey Ji, Rishabh Goel, Albert Ko

https://github.gatech.edu/osalunke3/CSE6730-project

I. Abstract

Air pollution can pose significant health risks to human and animals as well as negatively impacting the environment, therefore, it is a crucial first step to accurately measure the air pollution level before finding solutions. This project investigates the usage of small drones to survey air pollution level in an urban environment. This strategy can better measure local air quality comparing to existing spatial interpolation method employed by Air Quality Index measurement. To demonstrate the effectiveness and feasibility of the proposed detection plan, a case study is prepared based on the task of surveying air quality around Georgia Tech campus. The goal of this study is to map drone flight path based on identified potential hotspots on GT campus, the problem can then be transformed to a multi-agent traveling salesman problem. Based on the simulated flight path of each drone, relevant data such as flight time, range, endurance, velocity, cruise altitude are collected and analyzed to assess the performance of different drone fleets. Based on market research, several drones are selected as potential candidates for the surveying task. The flight path for different test cases are also visualized in a Tableau dashboard. Based on several simulation as we very the number and brand of drones, key performance metrics are aggregated such as flight time, flight path, operation cost, and completion indication. In the case of Matrice 600, a solution that uses two drones was sufficient to cover all the nodes, with one drone having a minimum remaining flight time of 5 minutes, providing a safety margin. For the Phantom 4 drone, we varied the number of drones from 2 to 4, and all the solutions using multiple drones were able to fly the required mission with varying minimum remaining flight times. Among these solutions, the one that uses four Phantom 4 drones has the lowest implementation cost of just $6,396, providing a cost saving of approximately 70% compared to the most expensive solution that uses two DJI Matrice 600 drones, which has an implementation cost of $22,000. The final portion of the paper discusses the model validation process. Due to the lack of actual simulation data, we use the results of our simulation to perform face and behavior validation in lieu of using traditional error metrics to validate our model. This project has allowed us to apply modeling and simulation techniques to a real-world problem, and has enabled us to go through the various stages of implementing such a model. Through this process, we have successfully designed and implemented a solution that addresses the problem of measuring air pollution levels over the Georgia Tech campus.

II. Introduction

Measuring air pollution is important because exposure to high levels of pollutants in the air can have severe health and environmental impacts. Air pollution is linked to respiratory and cardiovascular diseases, cancer, and even premature death. In addition to its health effects, air pollution also has ecological impacts, such as acid rain, smog, and damage to crops and forests. Accurate measurement of air pollution levels is essential to understanding the sources of pollution and developing effective strategies to reduce emissions and protect public health and the environment.

A. Literature Review

Historically, air pollution has been modeled and measured using different methods [1, 2]. Air pollution modeling typically uses computational methods to simulate air quality processes, such as the spread of pollutants and their impact on air quality, and assess the effectiveness of air pollution control measures. Two main types of air pollution models exist: dispersion models and photochemical models. Dispersion models have become increasingly preferable due to their fine-tuned calibration, enabling precise simulations [2]. These models account for the air's homogeneous turbulence and represent pollutant movement within it with higher accuracy. They also consider variables such as wind speed, temperature, humidity, and terrain features like structures and buildings while simulating even fleeting concentrations [2]. Additionally, these sophisticated tools can integrate highly variable emission rates when dealing with distribution complexities in simulated scenarios [1]. The air pollution dispersion model will provide a set of initial conditions to feed into the primary model.

To measure air pollution, stationary monitoring stations equipped with sensors detect contaminants such as carbon monoxide, nitrogen dioxide, sulfur dioxide, and particulate matter [3, 4]. These monitoring stations are placed in strategic locations and operate continuously, providing data on air quality over time. This data can be used by regulatory agencies to develop policies and guidelines aimed at improving air quality. However, these stationary monitoring stations have limited spatial coverage, and they cannot capture pollution variations in real-time [5].

Mobile drone sensors could supplement stationary monitoring stations and provide more comprehensive and real-time air pollution data. Drones equipped with sensors can fly over areas of interest and collect data on pollutant

levels at different altitudes and locations, providing a more detailed picture of air quality, especially in areas that are difficult to access or where stationary monitoring stations are not available [6]. Drones can provide data on pollutant sources, such as factories or power plants, that stationary monitoring stations cannot, target emissions reduction efforts, and help regulators and policymakers make more informed decisions [7]. Previous studies have shown the feasibility of using small drones to detect air pollution levels [8, 9]. However, these studies have also highlighted several limitations realized through real-world experiments [8, 9].

One limitation of drone-based air pollution measurement is that drones tend to fly at altitudes that are leveling with nearby buildings, which could result in air quality measurements deviating from actual air quality on the ground [9]. Additionally, this type of system requires additional physical infrastructure on the ground to help guide the fleet based on real-time conditions such as wind direction, population density, etc. [9]. Flying drones in an urban environment can be a complex task due to different flight ceilings and operating restrictions [8].

Some of these issues can be addressed with more sophisticated detection hardware and methods, such as vertical detection to consider air quality variation due to height, or predetermined pollution hot spots in the urban area according to research. A study by [10] developed a drone-based system for monitoring air pollution in urban areas, using a drone equipped with a multi-spectral camera and a gas sensor to collect data on various pollutants. The researchers found that the drone-based system was effective in detecting and mapping air pollution in real time. They concluded that drones have the potential to enhance air quality monitoring and management efforts, particularly in urban areas where traditional monitoring stations may not provide sufficient coverage. A study by [11] provides an overview of the current state of drone technology for monitoring atmospheric pollution. The authors discuss the potential benefits of using drones for air quality monitoring, including their ability to collect data in real time, cover large areas, and reach hard-to-access locations. They also examine the challenges associated with drone-based monitoring, such as the need for accurate and reliable sensors and the development of effective data analysis techniques.

In conclusion, traditional stationary air pollution monitoring stations have been used for years to measure air quality, but they have limitations in terms of coverage and real-time monitoring capabilities. Air pollution modeling has also been used to predict the effects of air pollution on the environment, but it requires initial conditions from real-world measurements. Recent advancements in drone technology have opened up new possibilities for air pollution monitoring, providing more comprehensive and real-time data on pollutant levels and sources, particularly in areas that are difficult to access or where traditional monitoring stations are not available. While there are still challenges to overcome, the use of drones for air pollution monitoring holds great potential for improving air quality management and reducing the negative impact of air pollution on human health and the environment.

B. Current System for Air Pollution Detection

Air quality in the United States is currently measured using a combination of static monitoring equipment and a method called spatial interpolation. The Air Quality Index (AQI) can be considered as a yardstick of the quality of air and pollution level in the air. The value of AQI varies from 0-500, with 0 being the best quality. AQI is calculated based on the levels of five major air pollutants: ground-level ozone, particulate matter, carbon monoxide, sulfur dioxide, and nitrogen dioxide. The AQI levels in areas between monitors are estimated using the inverse-distance weighted (IDW) method, which assumes that sites that are close to one another are more alike than sites that are farther apart. AQI values in areas without monitors are calculated using a weighted average of the values available at surrounding sites. The maps are interpolated to a resolution of 0.045 decimal degrees, and the uncertainty of the interpolated AQI is measured. AQI for Atlanta can be obtained from open sources of information like IQAir [12] and AirNow [13]. AirNow's current air quality (NowCast AQI) data updates hourly using readings from thousands of monitors across the country.

The current methodology for measuring air quality though effective in implementation fails to give a localized reading of air pollution. The AQI index readings are measured using sensors statically located at different positions across the city. The IDW method used for interpolation is a linear method that assigns weightage based on the inverse of the distance. The current methodology is not well suited (1) If we need AQI readings at specific locations which are sensitive areas for air pollution at different times of the day and the static sensor is not located at these locations. (2) Instead of readings at a point we are more interested in readings along a path or a pre-defined area. (3) We need a higher frequency of AQI readings per day.

III. Air Pollution Detection Over Georgia Tech Campus

The purpose of this study is to develop a system that utilizes environmental drones to monitor air pollution over the Georgia Tech campus. The current air pollution detection system is designed for the city of Atlanta and not optimized

for the campus, which has an enrollment of around 25,000 students and important areas for outdoor activities. By providing accurate real-time AQI measurements for members of the Georgia community, they can make informed decisions and take precautionary measures in case of low air quality. The system will identify potential hotspots based on the population density across campus for every hour of the day. A graph will be created with identified areas as nodes and flight time between them as the length of edges. A path-planning algorithm will determine the minimum flight time across the graph that covers all the areas. Simulations will be conducted for different drone launch and recovery locations, numbers of operational drones, and types of drones. A cost-vs-efficiency trade-off will also determine the optimal number of drones and their flight path. This study will provide a complete solution for monitoring air pollution over campus, with the ultimate goal of helping the Georgia Tech community make informed decisions about their health and well-being.

In the rest of this section, we will identify potential hotspots on the GT campus that will be selected for AQI measurements. We will then do a market survey to find suitable options for e-drones that will be utilized in our study. Different E-drones will be selected to run the simulation mode. The selection will be based on range, endurance, velocity, cruise altitude, and cost of operation. Section IV would give a detailed explanation of the conceptual model of the system. Details of mathematical models and techniques used to solve the system will also be explained in this section. Section V would give a brief introduction to the Python implementation of the model. Section VI will give an overview of the dashboard that is created to visualize various trade-offs and select an optimal design for the system. A detailed explanation of different experiments and trade-offs that will be run will be given in section VII followed by a discussion of the results of the experiments. We will then validate and verify our model followed by summarizing the key takeaways of the project in the conclusion section.

A. Identifying Potential Hot-spots for AQI Measurements

To identify potential hotspots for AQI measurement across campus we selected different locations in which population density is higher for some parts of the day. The selected locations are also used by the campus population for outdoor activities such as running, cycling, etc. Table 1 gives a description of locations selected for AQI measurement and provides justification for selection, whereas Fig 1 gives a visual representation of these points and coordinates of all selected points.

Table 1 Selected Locations for AQI Measurements

S.No	Location	Justification
0	Tech Green	Most crowded outdoor location of the campus
1	Transit Hub	All campus transport transit through this spot so pollution readings can be higher
2	Student Center	Popular Outdoor Seating spot on campus
3	Roe Stamps Field	Used for outdoor physical activities
4	Burger Bowl Field	Used for Outdoor activities and has GT Leadership course
5	Tech Square	Densely populated area, next to regions of high traffic density
6	North Avenue Apartments	Houses a large campus population
7	West Village	Includes a dining commons and residence halls
8	Eco Commons	Outdoor resting places for the campus population
9	Ken Bayers Tennis Complex	Tennis courts routinely used by the campus population

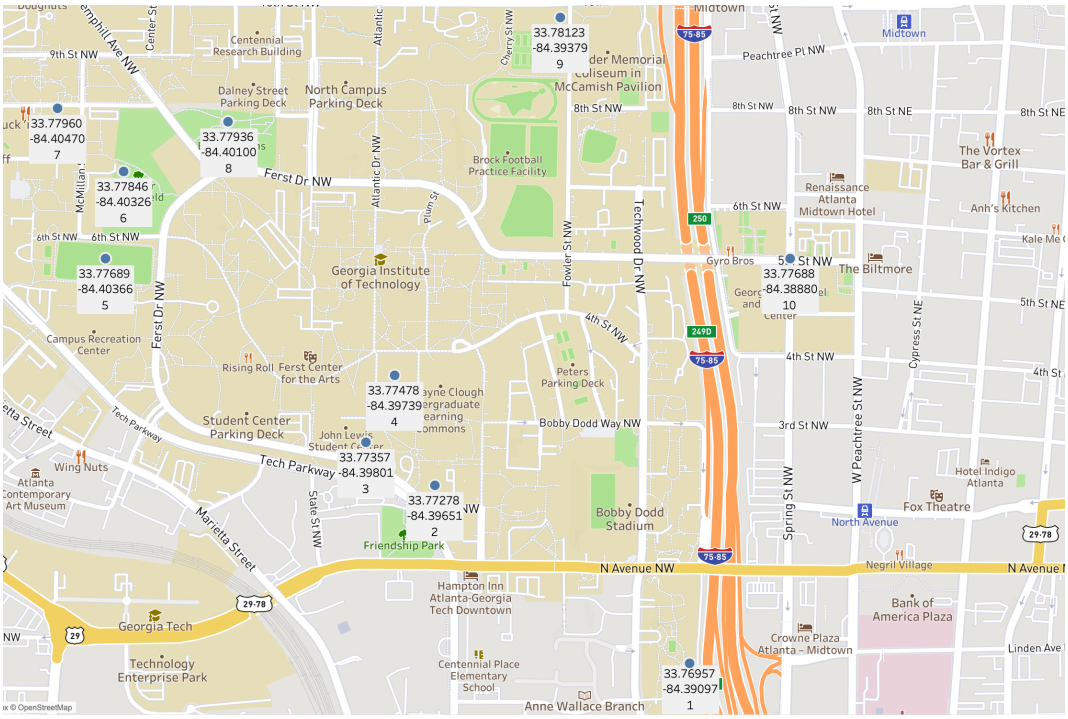

Fig. 1 Mapped Selected Locations on Tableau Dashboard

B. Market Research to Select E-Drones for Simulation

The most prevalent way to acquire air quality data using drones is to attach an air quality monitoring sensor payload to commercially available drones available on the market. the Scentroid DR1000 is a drone payload-based air monitoring sensor system. This system can be used to sample and analyze ambient air at heights of up to 125 meters above ground level. While in flight, every 2 seconds the drone will record GPS positions, altitude, temperature, humidity, H2S, VOC, SO2, Methane, and any chemical that is being monitored. Data can be used to create a map of the emission plume in real-time. It also has a variable sampling rate from once per second to once per minute. It can be attached to various drones a few of which we have compared in Table 2

Table 2 Comparison of compatible drones

Drone Name	Flight time(mins)	Charging time(hrs)	Cost ($)	Velocity (m/s)	Altitude
AltaX	35	2	18200	26.3	8000
DJI Matrice 300	40	0.7	14700	23	7000
DJI Matrice 600 PRO	25	1.2	11000	18	4500
DJI Phantom 4 Pro V2	20	0.8	1599	14	6000

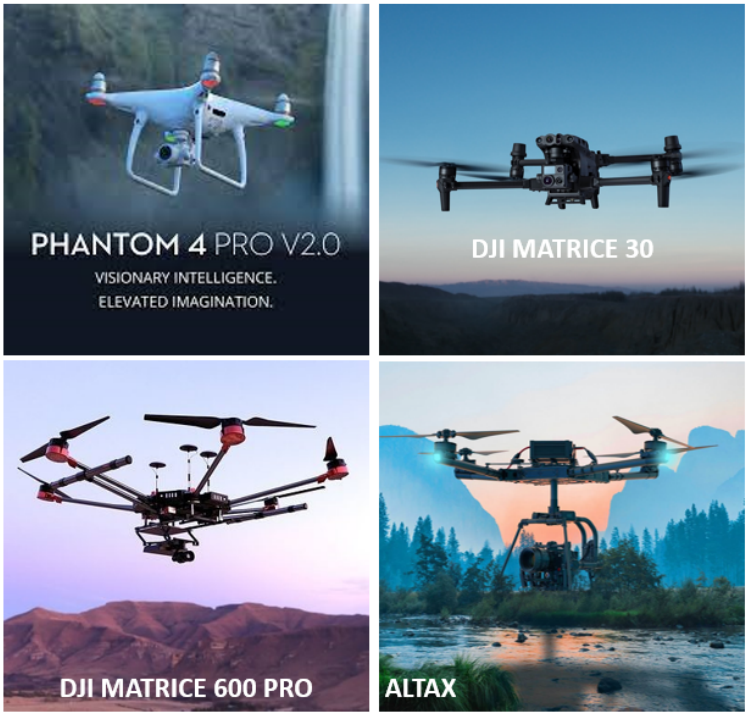

Fig. 2 Selected Drones

IV. Conceptual Model of the System

A. Algorithm

The multi-agent traveling salesman problem (MATSP), a variant of the classic traveling salesman problem (TSP), was selected for this project. It is a problem that deals with multiple salespeople who each need to visit a set of cities and return to their starting location. The objective of the problem is to minimize the total distance traveled by all salespeople. The multi-agent traveling salesman problem (MATSP) is one of the most complex and intricate problems in the realm of optimization. The intricacy stems from the need to consider the variations that arise between salespeople. These variations present a unique challenge that must be tackled.

To tackle this challenge, researchers have employed a diverse range of algorithms. However, the genetic algorithm has emerged as the most suitable algorithm for this task. Its flexibility, adaptability, and efficiency in exploring the problem space make it the go-to algorithm for such complex problems. The genetic algorithm uses an evolutionary approach, similar to that of natural selection processes, such as crossover and mutation, to evolve a population of solutions. These solutions are then continuously evaluated, and the fittest ones are selected. This process is repeated multiple times, leading to the evolution of a diverse range of solutions. The algorithm's ability to maintain diversity is particularly important, as it prevents premature convergence to a local optimum. This ensures that the algorithm can effectively navigate the complex problem space and find high-quality solutions that minimize the total distance traveled by all salespeople in the MATSP. To simulate the MATSP using a genetic algorithm, several steps must be followed [14].

The first step is to generate an initial population of feasible solutions that represent a sequence of locations that each drone will visit. The population should be randomly generated to meet the problem's constraints. Next, the fitness of each solution must be evaluated using a fitness function that considers the total distance traveled by all drones in each solution and the maximum distance travelled by a single agent. This step helps to identify the quality of each solution and the potential for improvement. After evaluating the fitness of each solution, parents must be selected for crossover using a selection operator. The operator should choose parents that are most likely to produce offspring that meet the constraints of the problem. Once parents are selected, the crossover operator combines them to create new offspring solutions. The operator should ensure that the offspring solutions are feasible and meet the constraints of the problem. To introduce diversity into the population, random changes are introduced to the offspring solutions using

a mutation operator. Specifically we use a swap mutation which swaps the location of 2 randomly selected location from the route. This step helps prevent the algorithm from converging to a local optimum and promotes exploration of the search space. After introducing mutations, the fitness of the offspring solutions must be evaluated to identify the potential for improvement. The best solutions from the current and offspring generations are then combined using a replacement operator to form the next generation. This process is repeated until a stopping criterion is met, such as a maximum number of iterations or a convergence threshold. Once the algorithm has converged, the best solution found by the algorithm is returned.

B. Single vs Multi-agent Model

There are a number of real-world problems that require several agents to visit areas of interest, complete tasks, and travel between them. These typically include problems such as surveillance, exploration, or search and rescue. Often in these use cases, the problem involves path planning for an agent to most efficiently travel through several locations without repeating routes or visiting the same point of interest more than once. This type of problem can be formulated using the Traveling Salesman Problem (TSP). Where through linear constraints, we can specify the routing requirement so that the agent can traverse through several waypoints in the shortest way possible. Furthermore, in order to survey a large area, several agents are deployed at the same time to scan the area in parallel. This increases the complexity of the problem. Specifically, the issue arises as to how should we route each agent so that the paths are not overlapping and waste resources.

C. Mathematical Model Formulation

The following mathematical model formally formulates the problem:

- Given a list of tasks T with size N and a list of agents A with size M.
- Each task has a node potential u_i.
- Let x_{ija} indicate whether agent a will go from task i to j.
- Let c_{ija} be the cost for agent a to travel from task i to j.

$$x_{ija} = \begin{cases} 1 & \text{if agent } a \text{ visits task } j \text{ after } i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\begin{aligned} \min_{x_{ija}} \quad & \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{a=1}^{N} x_{ija} c_{ija} \\ \text{s.t.} \quad & \sum_{i=1}^{N} x_{ipa} - \sum_{j=1}^{N} x_{pja} = 0, \quad a \in A, p \in T \\ & \sum_{j=1}^{N} x_{1ja} = 1, \quad \forall a \in A \\ & u_i - u_j + N \sum_{a=1}^{M} x_{ija} \leq N - 1, \quad \forall i \neq j \neq 1 \\ & x_{ija} \in 0, 1 \forall i, j, a \end{aligned} \tag{2}$$

The objective here is to minimize the cost for all agents traveling between the assigned task. The constraints ensure that each node is only visited once by one agent. Each node can also only get passed through once. Also, every agent is only used once.

V. Model Implementation

Currently, we are using Jupyter Notebook to implement our genetic algorithm to solve MATSP at the GitHub repository. Our algorithm is a modification of the genetic algorithm as seen in [15], which is a genetic algorithm used to solve the singular-agent traveling salesman problem. We created a Hotspot class to keep track of locations coordinates as well as calculating the distance between 2 points.

```
class Hotspot:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def distance(self, hotspot):
```

```
        xDis = abs(self.x - hotspot.x)
        yDis = abs(self.y - hotspot.y)
        distance = np.sqrt((xDis ** 2) + (yDis ** 2))
        return distance

    def distance_haversine(self, hotspot):
        return hs.haversine((self.x, self.y), (hotspot.x, hotspot.y), unit=hs.Unit.METERS)

    def get_coords(self):
        return (self.x, self.y)

    def __repr__(self):
        return "(" + str(self.x) + "," + str(self.y) + ")"
```

We also have a fitness class to calculate the fitness of a route. The fitness takes into account the total distance of the route as well as the maximum distance travelled by a single agent. Adding the maximum distance travelled by an agent as a parameter of the fitness function rewards solutions with evenly distributed work amongst the agents. We also add a penalty if the route of an agent does not have more than 1 spot to heavily punish agents with no routes. Since we want to minimize the total distance we then take the inverse of the sum of the above mentioned values to get our final fitness value.

```
class Fitness:
    def __init__(self, routes):
        self.routes = routes
        self.distance = 0.0
        self.fitness= 0.0
        self.max_dist = 0.0

    def routeDistance(self):
        if self.distance ==0.0:
            pathDistance = 0.0
            for i in range(0, len(self.routes)):
                for j in range(0, len(self.routes[i])):
                    fromHotspot = self.routes[i][j]
                    toHotspot = None
                    if j + 1 < len(self.routes[i]):
                        toHotspot = self.routes[i][j + 1]
                    else:
                        toHotspot = self.routes[i][0]
                    pathDistance += fromHotspot.distance_haversine(toHotspot)
            self.distance = pathDistance
        return self.distance

    def max_agent_route_dist(self):

        if self.max_dist == 0.0:
            for agent in self.routes:
                dist = route_dist(agent)
                if dist > self.max_dist:
                    self.max_dist = dist
        return self.max_dist

    def routeFitness(self):
        if self.fitness == 0:
            penalty = 0
```

```
            for agent in range(0, len(self.routes)):
                if len(self.routes[agent]) <= 1 :
                    penalty = float('inf')
                    continue

                if route_dist(self.routes[agent]) > 925:
                    penalty += 1000
                    continue


            self.fitness = 1 / (float(self.routeDistance()) + penalty + self.max_agent_route_dist()*10)
        return self.fitness
```

Currently, the most major changes were to the mutation and breeding functions. Each individual in the genetic algorithm represents an k-sized array of arrays, each of which represents the path that one of the k drones will take. In order to properly breed between two parent individuals, we interpret each individual by appending the k arrays together into one large array and then use the crossover operation seen in the singular agent travelling salesman problem to breed the two parents together. We show the implementation of such a breeding operation below.

```
def breed(parent1, parent2):
    child = []
    childP1 = []
    childP2 = []

    totalChild = []

    routeLengthsParent1 = [0]
    routeLengthsParent2 = [0]
    routeLengthsParent = []

    totalParent1 = []
    totalParent2 = []

    for i in range(0, len(parent1)):
        routeLengthsParent1.append(len(parent1[i]) + routeLengthsParent1[i])
        routeLengthsParent2.append(len(parent2[i]) + routeLengthsParent2[i])
        totalParent1 = totalParent1 + parent1[i]
        totalParent2 = totalParent2 + parent2[i]
    geneA = int(random.random() * len(totalParent1))
    geneB = int(random.random() * len(totalParent1))

    startGene = min(geneA, geneB)
    endGene = max(geneA, geneB)

    for i in range(startGene, endGene):
        childP1.append(totalParent1[i])

    childP2 = [item for item in totalParent2 if item not in childP1]

    totalChild = childP1 + childP2
    for i in range(0, len(parent1)):
        routeLengthsParent.append((routeLengthsParent1[i] + routeLengthsParent2[i]) // 2)

    routeLengthsParent.append(len(totalChild))

    for i in range(0, len(parent1)):
```

```
        child.append(totalChild[routeLengthsParent[i]: routeLengthsParent[i + 1]])

    return child
```

Similarly, we also implement the mutation operation by swapping two random elements likewise, as seen in the implementation below:

```
def mutate(individual, mutationRate):
    totalIndividual = []
    routeLengths = [0]
    for i in range(0, len(individual)):
        totalIndividual = totalIndividual + individual[i]
        routeLengths.append(routeLengths[i] + len(individual[i]))

    for swapped in range(len(totalIndividual)):
        if(random.random() < mutationRate):
            swapWith = int(random.random() * len(totalIndividual))

            hotspot1 = totalIndividual[swapped]
            hotspot2 = totalIndividual[swapWith]

            totalIndividual[swapped] = hotspot2
            totalIndividual[swapWith] = hotspot1
    newIndividual = []
    for i in range(0, len(individual)):
        newIndividual.append(totalIndividual[routeLengths[i]: routeLengths[i + 1]])

    return newIndividual
```

We have implemented the hotspots listed in Table 1, using those locations' real-life latitudes and longitudes as sseen in Table 3 to calculate distance. We have done so in order to use in the calculation of optimal drone paths, and in order to generate some data on the convergence of such a path produced by the genetic algorithm. In order to measure the distance between the hotspot locations based on their coordinates, we use the Haversine formula using any two hotspots' longitudes and latitudes to find their distance in meters. Furthermore, for our fitness calculation, we are measuring the total distance traveled by every drone in the system. This should allow us to reach a minimum average distance traveled per drone. Finally, we also utilize the velocity and flight times for each drone as seen in 2 to determine which drones are able to make the distances in these paths.

Through trial and error, we were able to find a suitable mutation rate, elite size, and population size to have our algorithm always converge to an optimal solution as shown in the graph below. The Y-axis is the inverse of the fitness function which will be proportional to the distance of the best route for the population.

Table 3 Location Matrix

Location	Latitude	Longitude
Tech Green	33.7747760996229°*N*	84.397389045613°*W*
Transit Hub	33.772779872748515°*N*	84.39651065684585°*W*
Student Center	33.77357055668125°*N*	84.398009214924°*W*
Roe Stamps Field	33.77688902399273°*N*	84.40366325910463°*W*
Burger Bowl Field	33.77845950552504°*N*	84.40326275302543°*W*
Tech Square	33.776877666423005°*N*	84.38880262066144°*W*
North Avenue Apartments	33.769571690695614°*N*	84.39096887259626°*W*
West Village	33.779599557496205°*N*	84.40469972898788°*W*
Eco Commons	33.77935960869498°*N*	84.40099513026848°*W*
Ken Bayers Tennis Complex	33.78123012543277°*N*	84.39378517172695°*W*

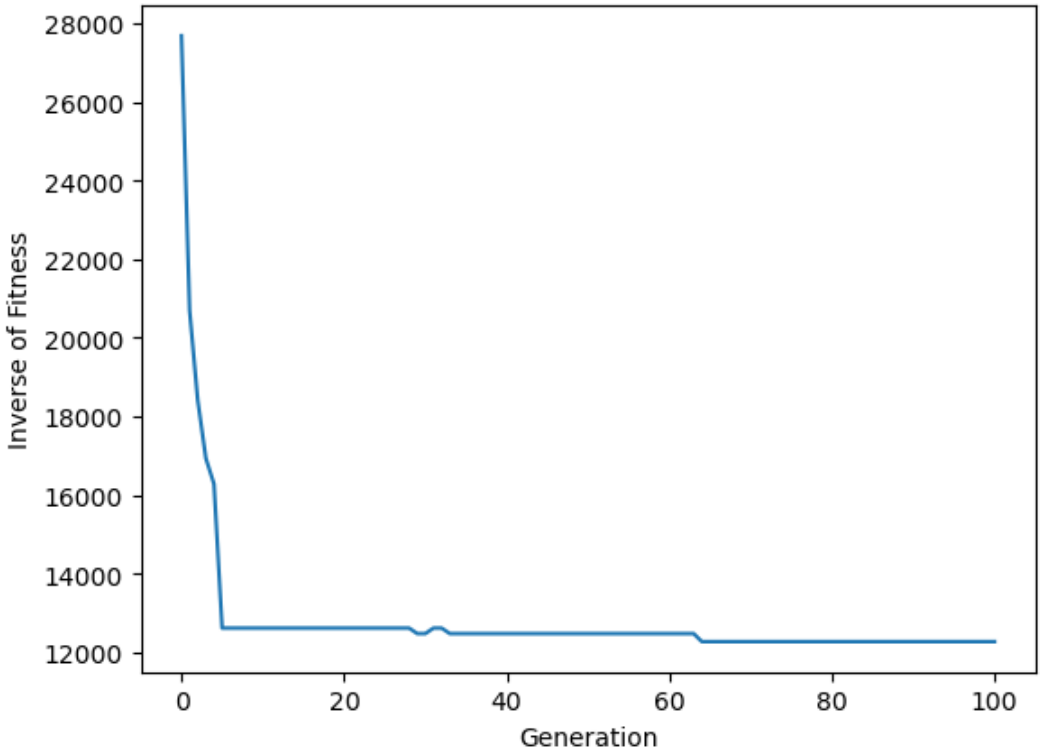

Fig. 3 Progress of the Genetic Algorithm

VI. Visualization Dashboard

This project will leverage Tableau (a data visualization tool) and the capabilities of Tableau's mapping feature, to generate an intricate visualization of the air quality index (AQI) measurements and the potential hotspots that may exist on the Georgia Tech campus. With the aid of geospatial data collected from the selected locations, we can construct an interactive and dynamic map that exhibits the AQI levels. By utilizing color-coded schemes based on air quality, we can enable users to recognize regions with inferior air quality and make decisions pertaining to their outdoor activities. Moreover, the map is flexible for customization that enables us to display population density at different intervals during the day.

In addition to this, Tableau's mapping feature enables visualizing drone flight paths and simulating diverse launch and recovery locations, drone types, and operational parameters. By superimposing the flight paths on the map, we can assess the efficiency and coverage of various scenarios and evaluate the trade-offs between cost and performance. This level of analysis can facilitate stakeholders to arrive at informed decisions regarding the optimal number of drones and their flight paths, culminating in a more effective air pollution monitoring system. Furthermore, through integrating the dashboard, we can implement an interactive platform that enables users to explore various trade-offs and visualize the consequences of different choices on the overall system performance and the level of awareness regarding air quality for the Georgia Tech community. We were unable to plot the final return leg of our optimal path in the dashboard due to a Tableau glitch. Hence, all the images shown in the results section would not have the final return leg.

The dashboard has been uploaded to the GitHub repository and images from the dashboard will be used in the rest of the sections of this report.

VII. Experiments and Results

We assumed that each drone would take a certain amount of time to obtain an AQI reading once it reaches a node location. The actual time required may vary, as different sensors have different AQI sensing times. Additionally, the time required to set up the drone for takeoff and landing, as well as the time required for it to climb to its cruise altitude and return to the ground, were all factors we considered in designing and executing our experiments.

The maximum velocities listed in Table 2 are the highest speeds achievable by each drone type. In reality, drones are not expected to maintain their maximum velocity at all times, as wind speed and direction can affect their speed. To account for these factors, we imposed the following constraints in our simulation model:

1) **AQI Sensing Time:** Each drone would spend approximately 60 seconds at each node location to obtain an AQI reading.
2) **Velocity:** The drones would fly at approximately 70% of their maximum velocity.
3) **Takeoff and Landing Setup Times:** Each drone would require approximately 5 minutes for setup, which includes warming up and climbing to the selected cruise altitude.

These constraints were carefully considered to ensure that our simulation model accurately reflected the capabilities and limitations of each drone type.

A. Single Agent Model Experiments

To evaluate the feasibility of using a single drone to cover all required locations in a timely manner, we designed experiments to simulate different scenarios, including the drone type, starting location, and flight path. Our goal was to determine whether using a single drone would be sufficient for our needs. We optimized our model to ensure that each drone took off and landed from the same node point, which helped reduce operational and maintenance costs.

We ran our single-agent model for all drone types and monitored the flight path and flight time for each drone. Table 4 shows the results of our experiments with the single-agent drone. For each drone case, the simulation model found North Avenue apartments to be the most ideal take-off point and attempted to make sure that the drone landed back at the same location. The DJI Matrice 300 drone covered all selected nodes with ease, with a remaining flight time of 14 minutes as shown in Fig 4. It's high speed and large flight time ensured that it met all simulation requirements. The Alta X drone was also able to complete the mission with 11 minutes remaining flight time.

On the other hand, the Phantom 4 drone had the least amount of flight time and was only able to fly over 8 node locations. It failed to take pollution readings over Ken Byer's tennis complex and Tech Square, and it was unable to fly back to its starting location. Although the DJI Matrice 600 drone was able to fly the complete mission, it landed back at its starting point with just 3 minutes of remaining flight time. Overall, our simulation results suggest that using a single drone may not be sufficient for our scenario, as some drone types were unable to complete the mission.

Table 4 Single Agent Results for all drones

S.No	Drone Type	Takeoff Location	All Nodes Covered	Remaining Flt Time (min)	Flight Path	Cost($)
1	DJI Matrice 300	North Avenue Apartments	Yes	14	6-1-2-0-3-4-7-8-9-5-6	14700
2	DJI Matrice 600	North Avenue Apartments	Yes	3	6-1-2-0-3-4-7-8-9-5-6	11000
3	DJI Phantom 4	North Avenue Apartments	NO	0	6-1-2-0-3-4-7-8	1599
4	Alta X	North Avenue Apartments	Yes	11	6-1-2-0-3-4-7-8-9-5-6	18200

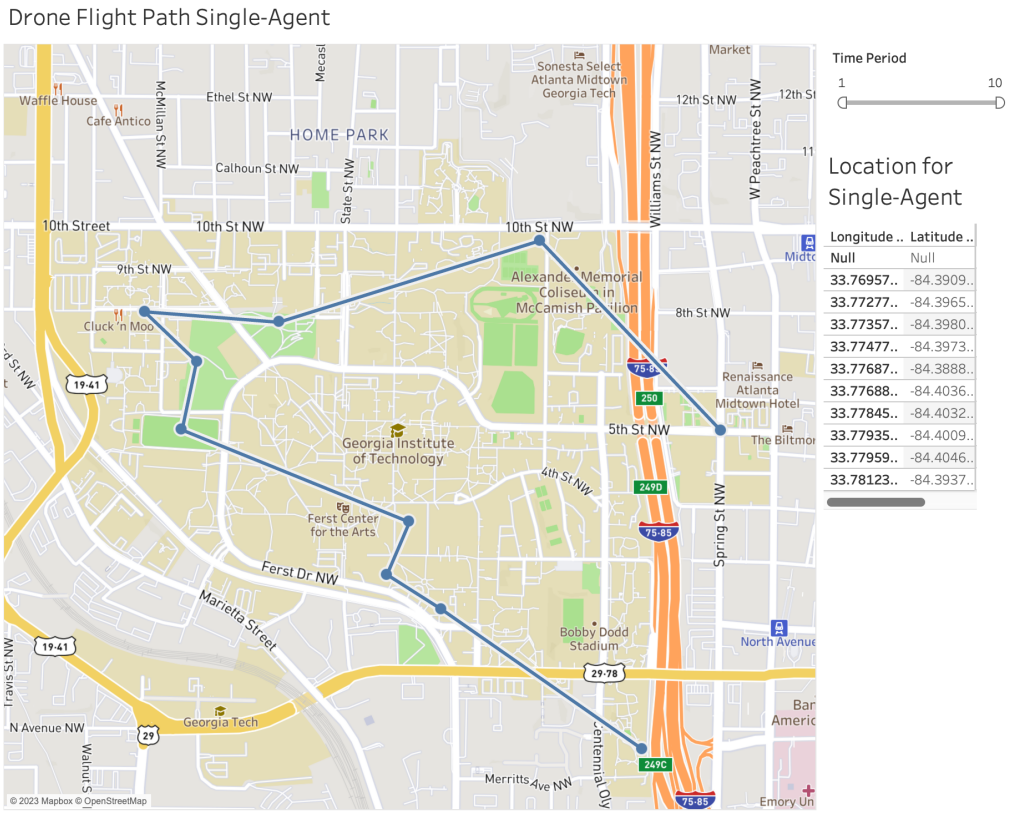

Fig. 4 Tableau Dashboard: Single Agent Results [DJI Matrice 300]

B. Multi-agent Model Experiments

Our single-agent experiments showed that the DJI Matrice 300 and Alta X drones were able to cover the entire domain with sufficient flight time remaining, indicating that they could be used in single-agent settings. However, these drones are expensive, and relying on a single drone can lead to serviceability issues, where any problems or delays in maintenance could result in the pollution measuring capability being stopped. On the other hand, the DJI Matrice 600 drone was able to complete the mission but returned to its take-off point with only 3 minutes of flight time remaining. This is not ideal as some AQI sensors require more than 60 seconds to record a reading, and using such sensors would make it impossible for the drone to complete our mission requirements. We also recorded if our solution was able to fly over all the selected nodes or not.

To address these issues, we conducted multi-agent experiments using DJI Phantom 4 and Matrice 300 drones. We varied the number of each drone to find the best path and the remaining flight time for each solution. We used the same drone types for multi-agent experiments to avoid complexity in maintaining the system. Using different drone types would require spare parts and charging stations for each type, and the operators and maintainers would have to be trained to handle multiple drone types.

Table 5 Multi-Agent Results for optimal number of drones for selected drone

Drone Type	Num. of Drones	All Nodes Covered	Remaining Flight Time(min)
DJI Matrice 600	2	YES	5
DJI Phantom 4	2	YES	1
DJI Phantom 4	3	YES	2
DJI Phantom 4	4	YES	3

Table 5 shows the results of our multi-agent simulations for the DJI Matrice 600 drone and Phantom 4 drones. The minimum remaining flight time recorded was the lowest flight time left among all the drones. In the case of Matrice 600, a solution that uses two drones was sufficient to cover all the nodes, with one drone having a minimum remaining flight time of 5 minutes, providing a safety margin. For the Phantom 4 drone, we varied the number of drones from 2 to 4, and all the solutions using multiple drones were able to fly the required mission with varying minimum remaining flight times. Fig 5 shows the path followed by drones in the case of 2 and 4 Phantom drones. From the visualization dashboard image, we can see the multi-agents follow a different path. Increasing the number of drones from 2 to 4 makes each drone follow a separate optimal path.

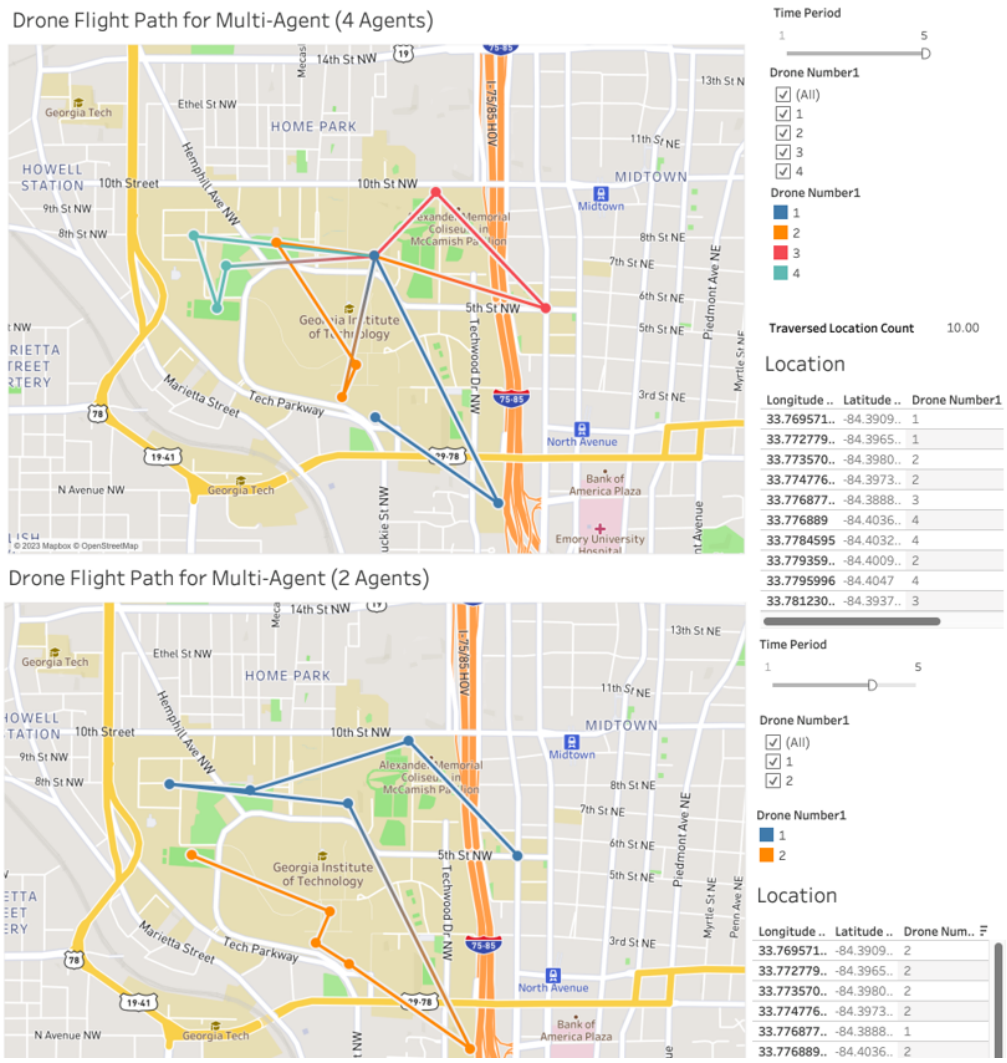

Fig. 5 Tableau Dashboard output of Multi-agent Results [DJI Phantom 4 for 2 and 4 agents]

C. Cost Analysis

In this section, we will discuss the trade-off between cost and meeting our objectives for the drone-based air pollution measurement system. To achieve this, we have selected four solutions from our single-agent and multi-agent experiments.

The single-agent experiments showed that DJI Matrice 300 and AltaX drones were able to cover the entire domain with sufficient flight time remaining. However, these drones are quite expensive and would result in a costly implementation of the experiment. Additionally, relying on a single drone could lead to serviceability issues, as any problem with the drone or delay in maintenance could result in the pollution measuring capability being compromised.

On the other hand, the DJI Matrice 600 drone was able to complete the mission, but with only 3 minutes of flight time remaining, which would not be enough time for an AQI sensor that takes slightly over 60 seconds to record a reading.

Moving on to multi-agent experiments, we selected DJI Phantom 4 and Matrice 300 drones for our simulations, using the same drone types to avoid adding complexity to the system. Table 5 shows the results for multi-agent simulations for DJI Matrice 600 and Phantom 4 drones, where the minimum flight time remaining among all drones was recorded.

After analyzing the results, we selected four solutions that meet our objectives while also being cost-effective as shown in table 6. Among these solutions, the one that uses four Phantom 4 drones has the lowest implementation cost of just $6,396, providing a cost saving of approximately 70% compared to the most expensive solution that uses two

DJI Matrice 600 drones, which has an implementation cost of $22,000. Therefore, we conclude that implementing the solution that uses four Phantom 4 drones is the most optimal and cost-effective approach for our problem.

Table 6 Cost analysis for different selected solutions

Solution No.	Solution Type	Drone Type	No. of Drones	Total Cost	% Diff
1	Single-agent	DJI Matrice 300	1	14700	33.18 %
2	Single-agent	Alta X	1	18200	17.72%
3	Multi-agent	DJI Matrice 600	2	22000	-
4	Multi-agent	DJI Phantom 4	4	6396	70.92 %

VIII. Validation and Verification

We want to validate and verify our model to establish that we have selected the correct model and the way we have implemented is right.

A. Model Verification

Model verification is the process of establishing if the model has been constructed correctly or not. In order to establish that our model works correctly we used the single-agent results. From the results of the single-agent shown in table 4 we can see that for all of our drone types, our model finds the same optimal starting point. The genetic algorithm also proceeds to find the same optimal path, but the path is scaled based on the flight time of the drone. This shows that our model tends to find the optimal path for all cases. The simple single-agent results can be considered as a unit test of our code and establish that our implemented model matches the conceptual model we developed based on the traveling salesman problem. The convergence of our model fitness function shown in fig 3 also shows that our model converges to a solution and there is no problem with the implementation of the model.

B. Model Validation

Model validation is the process of determining if the model captures the true behavior of the system. Due to the lack of actual simulation data, we cannot use traditional error metrics to validate our model. However, we can still use the results of our simulation to perform face and behavior validation.

1. Face Validation

Face validation is the simplest form of validation that we can perform. It involves comparing the results of our simulation to the expected behavior of the system. In our case, we can use the graphs we have plotted to check if the observed behaviors match the expected behaviors. For example, we can see from the multi-agent graphs in figure 5 that the drones are covering different groups of nodes based on their distances. This matches our expectations since we know that drones can cover more ground if they fly shorter distances between nodes. Therefore, face validation gives us confidence that our model is working as expected.

2. Behaviour Validation

The behavior validation was conducted by looking at the final selected optimal paths. For the single-agent model, our drones always flew to points that were nearest to the current node. This is an expected behavior for most path-planning algorithms. Similarly, for multi-agent problems, we experienced a clear grouping in the nodes for each drone. This grouping revealed that our multi-agent model is trying to find a grouping of nodes for each drone to minimize the time needed to cover all the nodes.

IX. Conclusion

In this project, we have successfully implemented a solution to measure air pollution levels over the Georgia Tech campus using environmental drones. Our approach involved identifying potential hotspots of pollution readings across the campus and calculating the distances between these locations. We then conducted a market review to identify drones that could be used for this purpose, ultimately opting for readily available off-the-shelf drones due to their affordability and accessibility. To solve the problem of finding an optimal path for these drones, we turned to the traveling salesman problem and its multi-agent extension, ultimately utilizing a genetic algorithm to determine the optimal solution.

Through a series of single- and multi-agent experiments, we were able to analyze our proposed solution. Our results showed that, with the exception of the Phantom 4 drone, all other drones were able to fly the required mission, although the Matrice 600 drone had a low remaining flight time. We then ran multi-agent experiments for the two selected drones, in order to determine the optimal number of drones needed, as well as the optimal take-off and landing points and path for each case. Ultimately, we found that using four Phantom 4 drones was the most cost-effective solution that met our mission requirements.

To aid in the visualization of our results, we developed a dashboard, which allowed us to visualize the flight paths for both single- and multi-agent experiments. Additionally, the dashboard proved helpful in validating and verifying our model.

Overall, this project has allowed us to apply modeling and simulation techniques to a real-world problem and has enabled us to go through the various stages of implementing such a model. Through this process, we have successfully designed and implemented a solution that addresses the problem of measuring air pollution levels over the Georgia Tech campus.

Future work utilizing our simulation can expand on including the other properties of the chosen drones from our market review to further understand the viability of each drone to run the paths in our model. For example, there can be future work that implements not just the flight time that each drone has but also take into account how long each drone takes to charge after multiple runs of each drone's path. Furthermore, in the future, researchers can extend the model with an implementation of pollution around Georgia Tech throughout the day, and modify our model's fitness functions to prioritize collecting data from more polluted hotspots within our areas.

References

[1] Daly, A., and Zannetti, P., "Air pollution modeling–An overview," *Ambient air pollution*, 2007, pp. 15–28.

[2] Aggarwal, A., Haritash, A. K., Veera, and Kansal, G., "Air Pollution Modelling - A Review," *International Journal of Advanced Technology in Engineering and Science*, Vol. 2, No. 1, 2014, pp. 355–364.

[3] Rohi, G., Ejofodomi, O., and Ofualagba, G., "Autonomous monitoring, analysis, and countering of air pollution using environmental drones," *Heliyon*, 2020. https://doi.org/10.1016/j.heliyon.2020.e03252.

[4] Narayana, M. V., Jalihal, D., and Nagendra, S., "Establishing A Sustainable Low-Cost Air Quality Monitoring Setup: A Survey of the State-of-the-Art," *Sensors*, Vol. 22, No. 1, 2022, p. 394.

[5] Kanaroglou, P. S., Jerrett, M., Morrison, J., Beckerman, B., Arain, M. A., Gilbert, N. L., and Brook, J. R., "Establishing an air pollution monitoring network for intra-urban population exposure assessment: A location-allocation approach," *Atmospheric Environment*, Vol. 39, No. 13, 2005, pp. 2399–2409.

[6] Xie, X., Semanjski, I., Gautama, S., Tsiligianni, E., Deligiannis, N., Rajan, R. T., Pasveer, F., and Philips, W., "A Review of Urban Air Pollution Monitoring and Exposure Assessment Methods," *ISPRS International Journal of Geo-Information*, Vol. 6, No. 12, 2017. https://doi.org/10.3390/ijgi6120389, URL https://www.mdpi.com/2220-9964/6/12/389.

[7] Lambey, V., and Prasad, A., "A review on air quality measurement using an unmanned aerial vehicle," *Water, Air, & Soil Pollution*, Vol. 232, 2021, pp. 1–32.

[8] Gallacher, D., "Drone Applications for Environmental Management in Urban Spaces: A Review," *International Journal of Sustainable Land Use and Urban Planning*, Vol. 3, No. 4, 2016, pp. 1–14.

[9] Villa, T. F., Gonzalez, F., Miljievic, B., Ristovski, Z. D., and Morawska, L., "An overview of small unmanned aerial vehicles for air quality measurements: Present applications and future prospectives," *Sensors*, Vol. 16, No. 7, 2016, p. 1072.

[10] Wang, T., Han, W., Zhang, M., Yao, X., Zhang, L., Peng, X., Li, C., and Dan, X., "Unmanned aerial vehicle-borne sensor system for atmosphere-particulate-matter measurements: Design and experiments," *Sensors*, Vol. 20, No. 1, 2019, p. 57.

[11] Jońca, J., Pawnuk, M., Bezyk, Y., Arsen, A., and Sówka, I., "Drone-Assisted Monitoring of Atmospheric Pollution—A Comprehensive Review," *Sustainability*, Vol. 14, No. 18, 2022, p. 11516.

[12] "First in Air Quality," , n.d. URL https://www.iqair.com/us.

[13] "ECSS AirNow Customer Service Portal," , n.d. URL https://usepa.servicenowservices.com/airnow.

[14] Yang, C., and Szeto, K. Y., "Solving the traveling salesman problem with a multi-agent system," *2019 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2019, pp. 158–165.

[15] Stoltz, E., "Evolution of a salesman: A complete genetic algorithm tutorial for python," , Jul 2018. URL https://towardsdatascience.com/evolution-of-a-salesman-a-complete-genetic-algorithm-tutorial-for-python-6fe5d2b3ca35.

A. Appendix: Division of Labor

- Bilal Mufti
    1) Define and formulate the problem.
    2) Develop experiments to be performed and trade-offs to be analyzed.
    3) Help in the implementation of the model.
- Oojas Salunke
    1) Develop the conceptual model of the problem.
    2) Develop the visualization dashboard.
    3) Created the summary video/presentation
- Joey Ji
    1) Develop the conceptual model of the problem.
    2) Develop the visualization dashboard.
- Rishabh Goel
    1) Carry out a market survey for e-drones and develop the input graph matrix.
    2) Carry out execution of experiments and generate output files for the dashboard
    3) Help in the implementation of the model and experiment execution
- Albert Ko
    1) Take the lead in the implementation of the model.
    2) Help with the execution of experiments and trade-offs.