UAV Route Planning to Anticipate the COVID-19 Crowd Clusters with Dynamic Programming

Leonard Matheus Wastupranata
School of Electrical Engineering & Informatics
Institut Teknologi Bandung
leo.matt.547@gmail.com

***Abstract*—Crowds are considered trivial by the community because they feel they have implemented health protocols by wearing masks. Crowds must be minimized so that the spread of the Covid-19 virus does not get higher. This paper aims to plan a UAV shuttle route so that it can approach as many locations as possible with potential crowding while simultaneously leading to the destination of the flight route without having to go around first. The method used is a comparison of Greedy algorithms and Dynamic Programs in determining the most effective route. The flight simulation was carried out using Software in The Loop (SITL) and ArduPilot Mission Planner. The results obtained are that the Dynamic Program can visit 14 locations out of 18 existing location choices, whereas with the Greedy algorithm approach, UAV can only visit 8 locations out of 18 existing location choices. The conclusion is that the Dynamic Program is able to maximize routes so that more locations are visited by UAVs and certainly better than the Greedy Algorithm.**

Keywords—UAV; Dynamic Programming; Greedy Algorithm; COVID-19; crowd

I. INTRODUCTION

The Covid-19 pandemic has brought many changes to world life. Many traditions and cultures have had to face all kinds of adjustments to coexist with the Covid-19 virus, whose spread is never ending. Time goes on and life can't stop moving. Phases in human life will change quickly, especially in the lifeline of Bandung Institute of Technology students who are currently studying. The plan for hybrid lectures continues to be echoed to fill all the deficiencies in taking online lectures.

However, students must follow the health protocols that have been designed by ITB, especially in terms of minimizing crowds. Unfortunately, it is not known whether there was someone in the crowd who tested positive for the Covid-19 virus in their body or not. For this reason, it is necessary to periodically monitor and spray disinfectants during rush hours amidst busy campus activities. Much research has been done on preventing Covid-19, one of which is preventing crowds by spraying disinfectants. Efelina et al. [1] have conducted research on the use of drones to spray disinfectants in rural areas. However, the drone control that is conducted still uses the manual method and is still less effective if it turns out that a large crowd is created. Therefore, an algorithmic strategy is needed that can optimize periodic disinfectant spraying.

In this paper, a comparison of the UAV range strategy to prevent crowds is given by making a comparison between greedy algorithms and dynamic programs. The more locations that are regularly covered by UAVs, the faster crowds can be minimized. However, due to limited UAV power, flight routes will be shuttled from certain locations. The UAV's task is to find a shuttle route that will maximize the location of the crowd to be visited while leading to its destination.

This paper will be divided into five major sections. The Introduction section will explain the background of the problems that arise. Next, the Literature Study section will explain the basics of algorithmic strategies in conducting flight experiments. The part that is no less important, namely the method, will discuss the steps to solve the problem with the greedy algorithm approach and dynamic programming. The Results and Discussion section will provide an explanation of flight simulation and strategy analysis. Finally, the Conclusion section will bring together all the frameworks and experiments that have been conducted in this paper.

II. LITERATURE STUDY

A. Greedy Algorithm

A greedy algorithm is an algorithm that solves a problem step by step in such a way that at each step it takes the best option that can be obtained at that time without regard to future consequences (the principle of "take what you can get now!") and "hope" that by choosing a local optimum at each step will end up with a global optimum.

This algorithm is the most popular and simple method for solving optimization problems. This algorithm puts forward the problem to find the optimal solution. There are two kinds of optimization problems, namely maximization and minimization problems. The general scheme of the greedy algorithm is as shown below. [2]

```
function greedy (C : candidate set)→ solution_set
Declaration
x : candidate
S : solution_set

Algorithm:
S ← {}
while (not SOLUTION(S)) and (C ≠ {}) do
x ← SELECTION(C) { select a candidate from C }
C ← C − {x} {  discard x from C as it has been selected }
if LAYAK(S U {x}) then { x meets eligibility for inclusion
in the set of solutions }
S ← S U {x} { plug x into the solution set }
endif
endwhile
{ SOLUTION(S) or C = {}}
if SOLUTION(S) then { solution is complete }
return S
else
write('no solution')
endif
```

This Greedy algorithm is almost the same as the exhaustive search and brute force methods, where Exhaustive search is a brute force search technique for solutions to

problems that involve searching for elements with special properties, usually among combinatoric objects such as permutations, combinations, or subsets of a set. Based on this definition, exhaustive search is also brute force. Therefore, exhaustive search is one implementation of brute force in the search case. [3]

B. *Dynamic Programming*

Dynamic programming is a method of solving problems by decomposing a solution into a set of stages so that the solution to a problem can be viewed as a series of interrelated decisions. Dynamic programming is used to solve optimization problems (maximization or minimization).

In dynamic programming, the optimal set of decisions is made using the Optimality Principle. According to the Optimality Principle, if the total solution is optimal, then the part of the solution up to the *k* stage is also optimal. The principle of optimality means that if we work from stage k to stage k + 1, we can use the optimal result from stage k without having to go back to the initial stage.

The characteristics of the problem with dynamic programming are as follows:

1. The problem can be divided into several stages, in which at each stage only one decision is taken.
2. Each stage consists of several states associated with that stage. In general, statuses are the various possible inputs that exist at a stage.
3. The results of the decisions taken at each stage are transformed from the status concerned to the next status in the next stage.
4. Cost at a stage increases steadily with increasing number of stages.
5. The cost at a stage depends on the cost of the stages that are already running and the cost from that stage to the next.
6. There is a recursive relationship that identifies the best decision for each status at stage k to provide the best decision for each state at stage k + 1.
7. The principle of optimality applies to this problem.[4]

C. *Coin-collecting Problem*

In this problem, coins are placed in n×m cells of the board, not more than one coin per cell. The robot starts from the top left of the cell board. The mission of this problem is that the robot must collect as many coins as possible and finish at the bottom right of the cell board.

At each step, the robot can move either one cell to the right or down from its current location. When the robot visits a cell containing coins, the robot will always take the coins. The robot must find the maximum number of coins it can pick and determine the path it must follow to be efficient. [5]

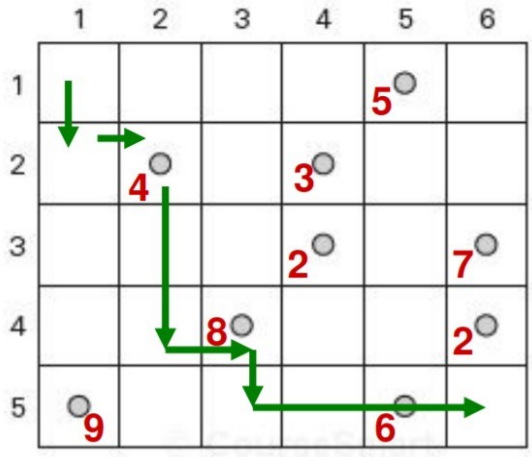

Fig. 1. Coin-collecting Problem [5]

III. Proposed Method

A. *Mission Planner*

There are several potential crowd points that are projected to emerge on the ITB Ganesha campus, the locations of which can be seen in the table below.

TABLE I. Locations Prone to Crowds in Campus Environment

Number	Specific Location		
	Location Name	***Latitude***	***Longitude***
HOME	Saraga Parking	-6.885423	107.608179
1	Saraga Digital Clock	-6.8856415	107.6101238
2	Saraga Canteen	-6.8869463	107.6100782
3	ITB Ganesha North Gate	-6.8878970	107.6103544
4	Octagon Roundabout-TVST	-6.8892657	107.6103705
5	Labtek Lobby 1	-6.8892151	107.6114380
6	East GKU	-6.8903628	107.6117063
7	DPR	-6.8899660	107.6103625
8	West GKU	-6.8903920	107.6091984
9	Intel Pool	-6.8903521	107.6103678
10	Soekarno Monument	-6.8909166	107.6103759
11	CTim-CBar	-6.8912335	107.6103839
12	Basketball Court	-6.8916649	107.6100540
13	Love Square	-6.8916702	107.6106602
14	East Hall	-6.8923972	107.6106924
15	West hall	-6.8924291	107.6102981
16	Southwest Gate ITB	-6.8931294	107.6102981
17	Southeast Gate ITB	-6.8930895	107.6106495
18	Public East Parking	-6.8931294	107.6116392

There are eighteen locations that can be visited to distribute disinfectants to help sterilize when there are crowds. However, because the spraying has been conducted thoroughly at night and in the morning, the UAV will fly from HOME to the last point, namely the East Public Parking with a relative altitude of 100 meters from the ground. A flight route will be determined so that the UAV can visit as many

potential crowds as possible, with the weight of each location having the same value (in this case it is symbolized by the number 1).

The flight simulation will be conducted using Software in the Loop (SITL) with ArduPilot Library and Mission Planner version 1.3.74. As shown in **Figure 2**, the potential crowd locations are scattered in **Table 1**. Since the UAV will fly from the HOME point which is in the northwest and headed for the ITB East Public Parking Destination Point which is in the southeast, the UAV will incline past ITB Ganesha in a diagonal direction.

Fig. 2. Locations Prone to Crowds in the Campus Environment

B. Element of Greedy Algorithm

In solving this problem, the solution will be abstracted into elements in the Greedy algorithm as follows.

1) Candidate Set of C

Contains the node candidates to be selected at each step, containing at least one node candidate. The selected node can only be to the right or bottom of the previous node. In this case, the node to be selected will be a representation of the location that has the potential to cause crowding.

2) Solution Set S

Pre-selected nodes. In this case, S is a location that has been visited and disinfectant spraying has been conducted.

3) Solution Function

This function will check whether the selected node is a goal node (ITB East Parking) or not.

4) Selection Function

The selected node is the node that is in the vicinity (both on the right and bottom of the previous node) with the closest distance.

5) Feasible Function

Checks whether the newly selected node makes the UAV movement inclined to move southeast (because of site selection that is only to the right or below the previous location).

6) Objective Function

Get a UAV flight route with as many locations to visit as possible.

C. Development Elements in Dynamic Programs

1) Characterize the structure of the optimal solution.

Suppose F(i, j) is the largest number of the problem of abstraction of the location to be visited in the cell(i,j) in the i-th row and the j-th column. The cell can be reached from the cell (i-1, j) to above it or from the cell (i, j-1) to around its left.

Of course, in this case, there is no other row above it in the cell in the first row and no other column to the left of it in the column in the first cell. It is assumed that F(i-1, j) and F(i, j-1) are equal to the number 0 for locations that do not have the potential to cause crowds.

2) Recursively define the optimal solution value.

$$F(0,j) = 0, 1 \leq j \leq m \qquad \textbf{basis}$$

$$F(i,0) = 0, 1 \leq i \leq n \qquad \textbf{basis}$$

$$F(i,j) = max\{F(i-1,j), F(i,j-1)\} + c_{ij}$$

$$for\ 1 \leq i \leq n, 1 \leq j \leq m \qquad \textbf{recurrent}$$

Where,

F(i, j): is the value of the i-th row and the jth column, it could be the cell to be selected next.

n: number of rows of the completion table

m: number of columns of the settlement table

c_{ij}: constant, in this case serves as a Boolean of whether there is a location pin in the table (number 0 if there is no defined location, number 1 if there is a location defined).

3) Calculate the value of the optimal solution in advance.

Before performing calculations regarding the optimal solution, a table will be created containing the division of rows and columns from the extraction of the coordinates of each location. This division is based on approximations for some

specific latitude and longitude scales and maximizes free space for more effective cell decomposition.

Fig. 3. Locations Prone to Crowds in the Campus Environment

As shown in **Figure 3**, there are 5×11 cells joined in a location representation table. One cell is only filled with a maximum of one location pin, so that the calculation of each stage will be right on target and will not experience excessive bias. In addition, the position of HOME and the destination node of the UAV will also be incorporated in the calculation at the time of flying.

The above decomposition can be translated in the form of a binary table representing the location to be visited as much as possible. In this case, each pin will be represented by the number 0 or 1 to represent the presence of a crowd location in the table.

TABLE II. BINARY TABLE REPRESENTATION OF LOCATIONS TO VISIT

(i, j)	*0*	*1*	*2*	*3*	*4*
0	1	0	1	0	0
1	0	0	1	0	0
2	0	0	1	0	0
3	0	0	1	0	1
4	0	0	1	0	0
5	0	1	1	0	1
6	0	0	1	0	0
7	0	0	1	0	0
8	0	0	1	1	0
9	0	0	1	1	0
10	0	0	1	1	1

4) Reconstruction of the optimal solution.

The solutions found in the completion table will be used to reconstruct the route that the UAV must travel to obtain the optimal route.

D. Waypoint Drafting

1) Drafting with the Greedy Algorithm

Since the principle of the Greedy algorithm is to find the shortest route first that prioritizes its right or bottom direction, the movement pattern will look like this.

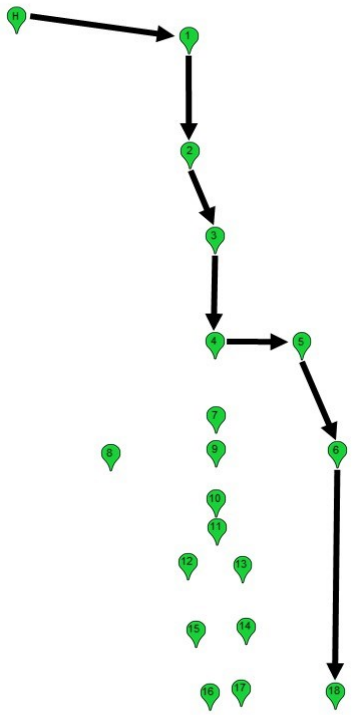

Fig. 4. Waypoint Preparation with Greedy Algorithm

Thus, the route will be generated in waypoint numbers as follows: HOME → 1 → 2 → 3 → 4 → 5 → 6 → 18

2) Drafting with Dynamic Programs

The calculation is conducted to get a place that will spray as much disinfectant as possible. The table of calculation of the optimal solution can be seen below.

TABLE III. PROCESSING TABLE OF LOCATIONS TO VISIT

(i, j)	*0*	*1*	*2*	*3*	*4*
0	1	1	2	2	2
1	1	1	3	3	3
2	1	1	4	4	4
3	1	1	5	5	5
4	1	1	6	6	6
5	1	2	7	7	7
6	1	2	8	8	8
7	1	2	9	9	9
8	1	2	10	10	11
9	1	2	11	12	12
10	1	2	12	13	14

With this processing table, the solution will be reconstructed from the destination node by looking for numbers around the node after which there is a dispute of one, where a node whose number is smaller than itself will be searched. Since movement is prioritized to the right first, if the above and left nodes are of equal value, the construction of the node on the left of itself will be searched first.

Thus, the route will be generated in waypoint numbers as follows: HOME → 1 → 2 → 3 → 4 → 7 → 9 → 10 → 11 → 12 → 15 → 14 → 17 → 18

E. Simulation Environment

The program will run on Lubuntu Operating System version 18.04 as most applications will run on the Linux environment. For Visualization, a Gazebo application (**Figure 5)** is used so that the UAV's flying behavior can be seen. The UAV will fly following the distance on the GPS so that the visualization of the waypoint passed can be seen properly. [6]

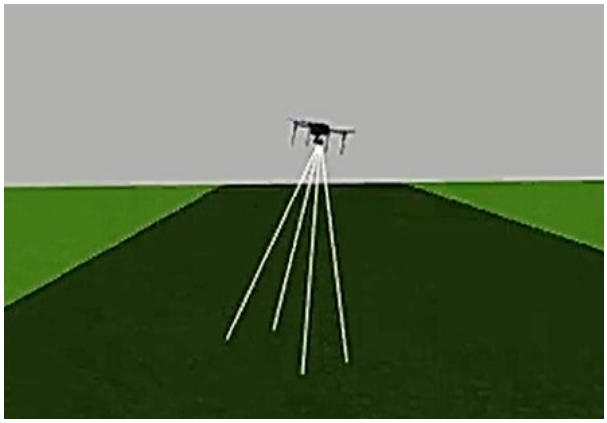

Fig. 5. Drone View via Gazebo application in *real time*

IV. RESULTS AND DISCUSSION

A. Results

A simulation of UAV flying behavior has been conducted designed to approach the location of the swarming site. The Flight Route Plan will be arranged in such a way by the application that later the UAV will fly according to the algorithm that has been prepared before.

The experiment of the flight route planner with the Greedy Algorithm resulted in 8 waypoints that included HOME itself, the specific route arrangement can be seen in **Figure 6.**

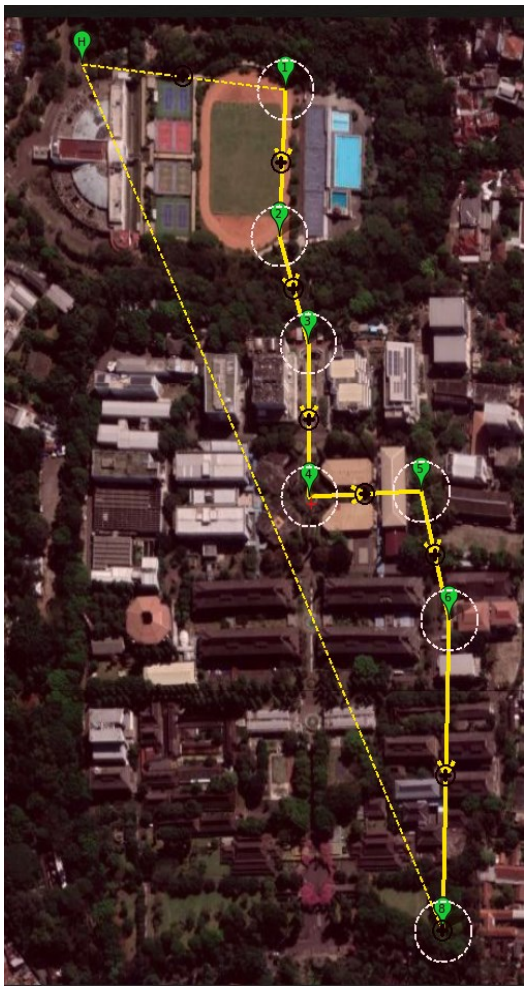

Fig. 6. Routes with the Greedy Algorithm approach

With this approach, UAVs can visit 6 locations that have the potential to cause crowds, namely Saraga Digital Clock, Saraga Canteen, ITB Ganesha North Door, Oktagon-TVST Roundabout, Labtek 1 Hall, and East GKU.

The total time the UAV spends flying from Take Off all the way to the public East parking lot is 4 minutes 30 seconds. The battery used to perform one flight route is 9.2644 Volts from the battery capacity of 12.19 Volts. The distance traveled by the UAV starts from HOME to reach the ITB Public Park as far as 1,205 meters.

In the next experiment, an experiment is conducted using the Dynamic Program approach. This experiment resulted in 14 waypoints that included HOME itself, the specific route arrangement can be seen in **Figure 7.**

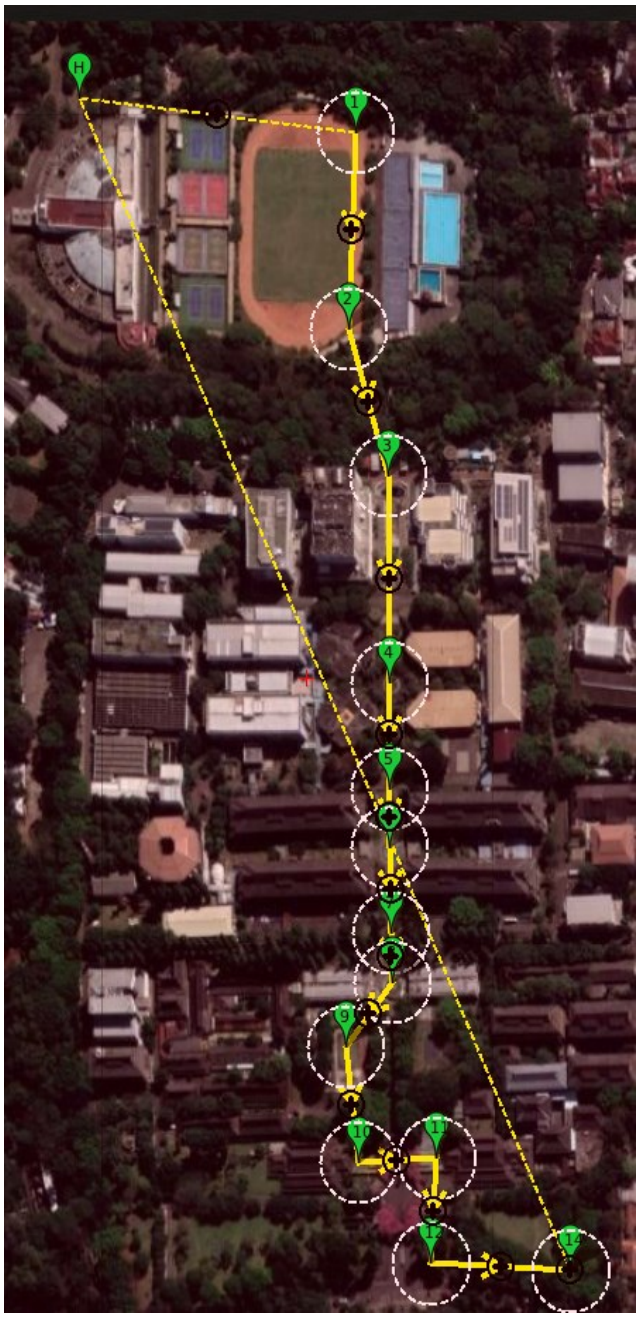

Fig. 7. Routes with a Dynamic Program Approach

With this approach, UAVs can visit 12 locations that have the potential to cause crowds, namely Saraga Digital Clock, Saraga Canteen, ITB Ganesha North Door, Oktagon-TVST Roundabout, DPR, Intel Pond, Soekarno Monument, CTim-CBar, Basketball Court, West Hall, East Hall, and ITB South Door east side.

The total time the UAV spends flying from Take Off all the way to the public East parking lot is 4 minutes 30 seconds. The battery used to perform a single flight route is 9,752 Volts from the battery capacity of 12.19 Volts. The distance traveled by the UAV starts from HOME to reach the ITB Public Park as far as 1,280 meters.

The time, distance, and size of the battery used are calculated only for the trip, not counting the time to spray disinfectant in the crowd. More details about the experimental results can be found in **Table 4**.

TABLE IV. UAV FLIGHT ROUTE EXPERIMENT RESULTS

	Parameter	**Algorithm**	
		Greedy	**Dynamic Programming**
1.	Number of Locations visited	6	14
2.	Flying Time	4’30’’	4’30’’
3.	Flying Distance	1.205 m	1.280 m
4.	Battery Used	9.2644 V	9.752 V

B. Discussions

Experiments in determining an Effective Route to visit potentially crowded locations have been conducted. The interesting thing is the number of locations approached by the UAV for different algorithms.

In the Greedy Algorithm, a node around it will be searched for that has the closest distance, no matter whether at the stage ahead it turns out that the result is not optimal. As a result, in waypoint number 4, this decision-making error also reduces the location that UAVs can approach. On the contrary, in the Dynamic Program Algorithm, node optimization will be sought at a certain stage of maximum value. Of course, this consideration is based not only on the closest distance, but the optimum solution for the whole case.

At the decomposition stage of each location using the Dynamic Program Algorithm, the waypoint taken is seen turning left even though the restrictions in this case are not allowed. This happens because the dynamic program will divide each problem into smaller sub-problems so that for tolerances of several degrees, longitude and latitude in the image will be one solid and parallel column.

Furthermore, for the comparison parameter of total flying time, the interesting thing is that the time difference between the two is ridiculously small, so it can be seen that the optimization problem emphasized is how many locations can be approached by UAVs to prevent potential crowds from appearing in the middle of the campus.

The total Flying Distance between the two experiments has a difference of 75 meters. The determination of flight routes using greedy algorithms proves that the problem of distance optimization is highly prioritized for the surrounding location that is closest to the previous location. Experiments with a dynamic program approach resulted in a longer total flight distance due to correspondence with the number of locations already visited.

The batteries used in the UAV for both experiments showed a difference of about 0.5 Volts. Experiments with the Dynamic Program approach consume more battery because the locations visited are also more than using the Greedy algorithm. That is, for battery parameters, the values correspond to the parameters of the visited location as shown in **Table 4.**

V. CONCLUSION

Crowding or gathering is something that must be avoided because the spread of the Covid-19 virus will be higher if it is considered trivial. With this, UAVs can be a solution to monitor and spray disinfectants to sterilize the potential crowd. For this reason, an effective strategy is needed so that

UAVs can reach as many locations as possible with the shuttle method.

The more locations that will become potential swarming points, the more alternative UAV solutions will be to approach as many existing points as possible. For this reason, the strategy of the UAV route determination algorithm to prevent gathering points is seen from the parameters of many locations, distances, time, and battery power used. UAV route determination for Gathering Point Prevention is better using Dynamic Programs than Greedy Algorithms.

VI. Future Works

The next suggestion for researchers is that further UAV implementation can be developed using Robotic Operating System (ROS) and HITL (Hardware In The Loop). In addition, the Dynamic Program decomposition table can be developed by giving weight to each different crowd intensity so that the UAV can prioritize the highest intensity crowds must first be visited and disinfectant spraying conducted.

It is also recommended to do comparisons other than the Greedy Algorithm, for example the Branch & Bound Algorithm, the A* Algorithm, and others. This is necessary because for different cases, it is hoped that the best solution can be found so that the UAV route determination as much as possible reaches the crowd location and conducts prevention effectively and efficiently.

Video Link at Youtube

https://youtu.be/22x-_2e10tm

References

[1] V. Efelina, S. Dampang, and I. Maulana, "Penggunaan Drone untuk Penyemprotan Disinfektan dalam Pencegahan Covid-19 di Masa Pandemi (Studi Kasus di Desa Margasari)," *Selaparang*, vol. 4, no. April, pp. 368–373, 2021.

[2] R. Munir, "Algoritma Greedy," no. Bagian 1, 2021, [Online]. Available: http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Greedy-(2018).pdf

[3] H. Sunandar and Pristiwanto, "Optimalisasi Implementasi Algoritma Greedy dalam Fungsi Penukaran Mata Uang Rupiah," *J. Tek. Inform. Unika St. Thomas*, vol. 04, no. 02, pp. 193–201, 2019.

[4] R. Munir, "Program Dinamis (Dynamic Programming)," *Progr. Stud. Tek. Inform. STEI-ITB*, vol. 2021, pp. 1–57, 2021, [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf

[5] N. Meghanathan, "Module 4 Dynamic Programming," pp. 6–11, 2015, [Online]. Available: http://www.jsums.edu/nmeghanathan/files/2015/04/CSC323-Sp2015-Module-4-DynamicProgramming.pdf?x61976

[6] L. M. Wastupranata, "UAV Waypoint Strategy for COVID-19 Medicine Delivery based on Cheapest Link and Hamilton Circuit Algorithm," *TechRxiv*, 2023, [Online]. Available: https://doi.org/10.36227/techrxiv.22218544