Routing Optimization of a Drone Coverage for Post Disaster Inspection

Eren Atsız, Zeynel Batuhan Organ
Industrial Engineering, Özyeğin University, Istanbul, Turkey
{eren.atsiz, batuhan.organ}@ozu.edu.tr

1. Problem Definition

Drones or Aerial Unmanned Vehicles(UAV) technology is rapidly evolving in mostly civil areas. The problem that we are planning to work on is on the upbeat such that time series statistics on the surveyed literature indicate the growing academic interest in the topic.[1] Our main aim is to cover most of the top priority places in a limited time, since time is very crucial for a postdisaster situation. We are planning to solve a routing optimization problem, such as TSP, in order drones to travel from a recharge stations to all possible nodes and cover the area with a limited battery capacity.

Although Drone problems are started to be discussed in the literature, recently, to the best of our knowledge, there is no study that covers a Drone Coverage Route Optimization with a prioritization of places where can be affected mostly in a post-disaster case. The problem is about drones which they leave recharge stations to search through a map based on prioritization to cover the whole possible area. With a drone, highly accurate images of the affected areas can be possible and most importantly, a fast alternative. Due to physical restrictions of Drones, our problem also includes battery consumption of drones and drive them to drop in a recharging areas to recharge their batteries. There are multiple recharge stations and they automatically recharge drone's battery. Also problem considers restricted areas where Drones cannot fly over or no need to, such as a military area, airports or forests.

The problem is basically an extension of TSP. In literature, there are similar articles which are about route for a set of locations, area coverage, searching or disaster response and relief operations. Problem is solved for larger instances by developing a heuristic algorithm since it is an NP-Hard problem which is discussed in the literature, the problem cannot be solved exactly in polynomial time. Therefore, it requires a fast and effective heuristic algorithm.

[1] Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey

● Grid cover-couple approach:

Grid cover-couple approach is used to divide the entire studied region into subregions with slow varying functions. To do this, a mesh of equal-sized squares needs to be designed to cover the entire region. We consider the center of gravity of each grid as the point that drones will stop by within this approach. It is assumed that when the drone reaches the central zone of each grid, it can scan the entire area within the boundaries of the grid. This similar idea is illustrated in Figure 1.

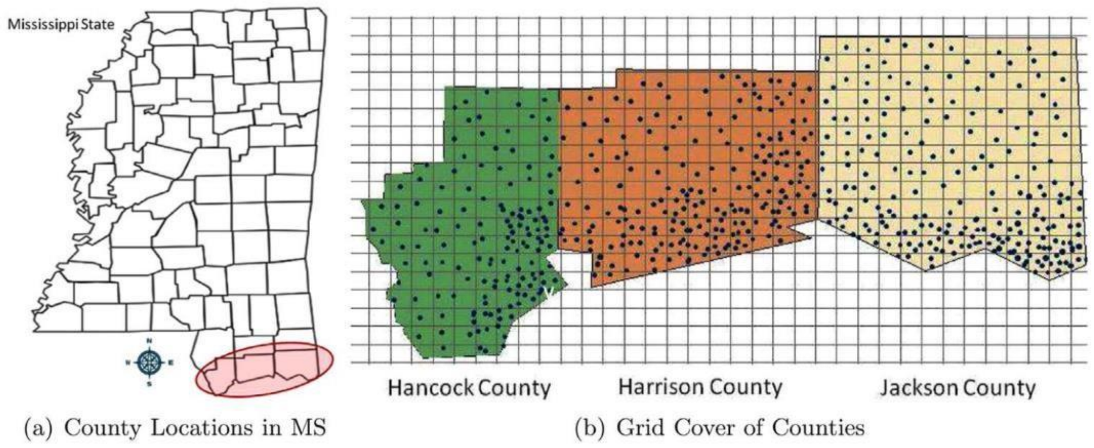


Figure 1. Customer density in grid for Hancock, Harrison, and Jackson counties in MS. (S. Chowdhury et al., 2017)

2. Mathematical Model

The MIP model is adapted from the article Schneider et al. 2014 which is about EVRP with time windows. Since our problem is a bit different, we changed the objective and add some constraints to fit our model. Our objective is to cover more prioritized areas first, which is minimizing the effect of getting lately to a risky place. The objective function calculation basically priority value of the location * completion time of that node. With this objective, model tries to send a drone to all nodes in order to cover all of at the beginning. However, our aim is to do this process with a limited number of drones, called as *D.* That's why we added the constraint (8), basically limits the number of drones leaving the depot. Solving with multiple vehicles forms multiple tours, this requires dummy end nodes, on top of starting depot. In order to limit drones to end at the same depot, we create *D* depots, all located at the same place and push each drone to end different dummy depot by constraint 10 & 11. Since the completion time of a node is important for our objective, we introduce the first contraint that arrival time and scannning time of a node gives us the completion time of that node. EVRP model consider the time windows unlike our problem with capacity constraints. Also, to allow drones to stop by into recharge stations multiple time requires dummy recharge stations.

Table: Variable and parameter definitions for the drone routing problem(DRP) model

Sets	
$V_{0'(N+1)}$	Set of all nodes
V'	Set of all nodes except for the dummy start and end nodes
V_F	Set of nodes to be processed
V_{F0}	Set including V_F and the dummy start node
$V_{0'}$	Set of nodes with outflow
$V_{N'+1}$	Set of nodes with inflow
$V_{R'}$	Set of RS nodes including the dummy ones
$V_{R'0}$	Set consisting of the dummy start node and the RS nodes
Parameters	
b_{ij}	Battery consumption required to move from i to j
t_{ij}	Time to move from i to j
k_i	Priority coefficient for node i
p_i	Battery consumption of the photographing process for node i
s_i	Time elapsed during the photographing process of node i
RT	Recharge time
B	Battery level
D	Number of Drones
Decision Variables	
x_{ij}	Binary variable indicating if node i follows j in the route
c_i	Completion time for node i
y_j	Representing the battery level in node j
a_i	Representing the arrival time to node i
M	Summation of all t_{ij}
e_i	Binary variable for conditional constraints

$$\min \sum_{i \in V'_{0,(N+1)}} k_i c_i$$

$$c_j \geq a_j + s_j \qquad \forall_j \in V'_{0\,(N+1)} \qquad (1)$$

$$\sum_{j \in V'_{N+1}} x_{ij} = 1 \qquad \forall_i \in V_F, i \neq j \qquad (2)$$

$$\sum_{j \in V'_{N+1}} x_{ij} \leq 1 \qquad \forall_i \in V'_R, i \neq j \qquad (3)$$

$$\sum_{j \in V'_{N+1}} x_{ji} - \sum_{j \in V'_0} x_{ij} = 0 \qquad \forall_j \in V', i \neq j \qquad (4)$$

$$a_i + (t_{ij} + s_i)x_{ij} - M(1 - x_{ij}) \leq a_j \qquad \forall_i \in V_0, \forall_j \in V'_{N+1}, i \neq j \qquad (5)$$

$$a_i + \left(\left(t_{ij} * x_{ij}\right) + RT * (B - y_i)\right) - M(1 - x_{ij}) \leq a_j \qquad \forall_i \in V'_R, \forall_j \in V'_{N+1}, i \neq j \qquad (6)$$

$$y_i - (b_{ij} + p_i)x_{ij} + B(1 - x_{ij}) \geq y_j \qquad \forall_i \in V'_R, \forall_j \in V'_{N+1}, i \neq j \qquad (7)$$

$$y_j + b_{ij}x_{ij} \leq B \qquad \forall_i \in V'_{R0}, \forall_j \in V'_{N+1}, i \neq j \qquad (8)$$

$$\sum_{j \in V'_{N+1}} x_{0j} \leq D \qquad (9)$$

$$\sum_{j \in V'_R, j \neq N+1} x_{j,N+1} \leq e_i \qquad \forall_i \in \{1, .. D\} \qquad (10)$$

$$\sum_{j \in V'_F, j \neq N+1} x_{j,N+1} \leq (1 - e_i) \qquad \forall_i \in \{1, .. D\} \qquad (11)$$

$$y_j \geq 0 \qquad \forall_j \in V'_{0(N+1)} \qquad (12)$$

$$a_i \geq 0 \qquad \forall_i \in V'_{0(N+1)} \qquad (13)$$

$$c_i \geq 0 \qquad \forall_i \in V'_{0(N+1)} \qquad (14)$$

$$x_{ij} \in \{0, 1\} \qquad \forall_i \in V'_0, \forall_j \in V'_{N+1}, i \neq j \qquad (15)$$

$$e_i \in \{0, 1\} \qquad \forall_i \in \{1, .. D\} \qquad (16)$$

3. Assumptions

Our main assumptions are that about Drone movements. Since Drone flight process can be affected by many conditions such as wind, battery, altitude etc., We are keeping out of the context of this project. Assumptions are listed below:

- Drones will be flying in an optimum height since the battery consumption vs image quality vs coverage area is prespecified.
- Drones will consume constant energy related with the length of the arc with a constant speed.
- Energy consumption while hovering and weighting will be negligible.
- Drone coverage period will be same for all the nodes.

4. Data Generation

We generated dataset which consists of multiple nodes and recharge situations in order to analyze scenarios. There are 5 different instances, which are made up from Solomon Benchmark Instances. All the instances contains a depot, 4 Recharge Stations and 20 Nodes with their randomized priorities. Priority numbers are assigned by the uniform distribution, *U(0,5)*. Also, as a future work, our aim is to try on a real dataset such as a high risk earthquake zone like Istanbul, to test our algorithm in real life case. We basically turn an area or a map to a directed graph for computational easiness. As mentioned above, we divide an area to grids via grid cover-couple approach and assign a node for each grid. The nodes are prioritized by a scoring algorithm which considers population density and risk level of the area in terms of disaster. After assigning a priority score to nodes, we generated a prespecified location where recharge station is placed. Places like airports, or forests or sea are not required to cover since they can be illegal to search and cover the area for airports, can be unnecessary to search forests and sea because of no risk level.

To start with a solution, we generated instances dataset as can be seen below. We started with a depot, 20 nodes to cover and 4 recharge stations to let drones to recharge whenever they need. Also their priority scores are listed. Remember that the depot and recharge stations do not require any prioritization.

Nodes	Priority Values of Instances				
	1	**2**	**3**	**4**	**5**
1	2.2	1.91	4.2	4.08	1.37
2	3.84	1.23	1.56	1.66	3.8
3	4.41	3	2.85	3.58	4.84
4	1.25	1.7	3.39	4.06	3.46
5	4.39	2.81	0.96	3.21	3.47
6	2.35	4.83	4.02	1.26	2.23
7	3.16	1.43	2.43	4.94	2.3
8	0.58	4.71	1.48	2.67	2.69
9	1.42	0.49	2.71	3.17	1.53
10	4.92	1.37	1.96	3.2	1.16
11	2.51	2.19	4.11	3.79	1.87
12	3.9	4.32	0.86	3.24	3.54
13	1.52	2.16	4.55	3.73	2.3
14	4.51	0.74	3.3	0.72	3.67
15	4.57	3.9	3.83	1.62	2.81
16	4.85	0.88	3.82	2.29	1.82
17	3.1	1.56	3.01	3.56	2.09
18	4.85	3.4	4.37	0.84	1.59
19	2.02	4.57	1.94	0.04	2.14
20	3.99	1.8	3.29	2.02	1.38

Table 1: Priority Values of All Nodes for all 5 Instances

Points	X	Y	Priority
Depot	50	50	0
Node 1	42	7	2.2
Node 2	40	69	3.84
Node 3	40	65	4.41
Node 4	55	12	1.25
Node 5	71	22	4.39
Node 6	21	44	2.35
Node 7	44	22	3.16
Node 8	98	47	0.58
Node 9	74	64	1.42
Node 10	20	37	4.92
Node 11	42	38	2.51
Node 12	2	67	3.9
Node 13	8	73	1.52
Node 14	81	84	4.51
Node 15	77	37	4.57
Node 16	81	87	4.85
Node 17	76	3	3.1
Node 18	90	90	4.85
Node 19	8	71	2.02
Node 20	37	2	3.99
Recharge Station 1	25	25	0
Recharge Station 2	75	75	0
Recharge Station 3	25	75	0
Recharge Station 4	75	25	0

Table 2: Example Dataset

5. Benchmark Datasets and Solutions

Although drone routing problems have been studied in the literature, in order to understand the post-disaster situation, we define the problem of scanning a whole region as a new problem.

As the problem is new, there is no published dataset and results to compare the metaheuristics/heuristics approach we developed at the first. However, In order to compare our heuristics approach, we tried to solve the problem for small instances by using our mathematical model with the help of commercial solvers, Gurobi and to compare these results with those of our heuristic method.

The parameters while solving the problem can be found below:

- 2 Vehicles
- 4 Recharge Stations
- 1 Depot
- Arc length is calculated by the euclidian distance
- Battery Level = 300
- Battery Consumption on travelling = 2 * distance matrix
- Time elapsed on travelling = 2 * distance matrix
- Battery Consumption during scanning = 10
- Time elapsed during scanning = 10
- Recharge Time = 0 (Drone Battery will recharged immediately)

First of all, to understand how the model behaves, we gave same priority, 1, for all nodes and ran the model for an hour. The results show us that the model behaves like a classical CVRP problem and visits the node respectively as can be seen in the following graph. Note that blue box is the depot, green circles are the recharge stations, and the rest is nodes.

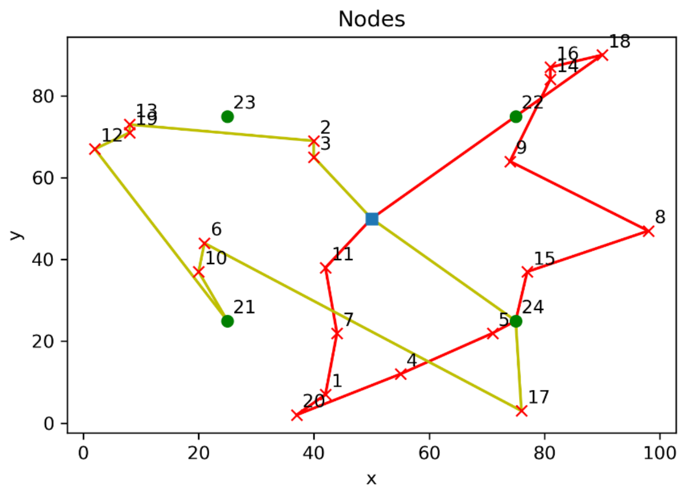


Figure 1: Exact solution(with optimality gap>0) for the same prioritized nodes

We ran the model on python and try to solve via Gurobi Solver on a i7-45100 CPU 8 GB RAM computer. All 5 of the instances ran for 1 hour (3600 s), following table will summarize the results obtained.

Instances	Objective Value	Opt. Gap (%)	Run Time (s)
1	15,023.65	81.30	3600
2	6,119.54	75.62	3600
3	9,681.74	78.33	3600
4	10,769.02	78.20	3600
5	9,521.44	79.70	3600

Table 1: Exact Results for 5 instances

Although the problem instances can be considered as small instances, the average optimality gap of the solutions are around 78.6%, which is huge. However, this shows that the necessity of an heuristic is important. Objective value comparison between exact solutions and greedy & Simulated Annealing will be discussed in following sections.

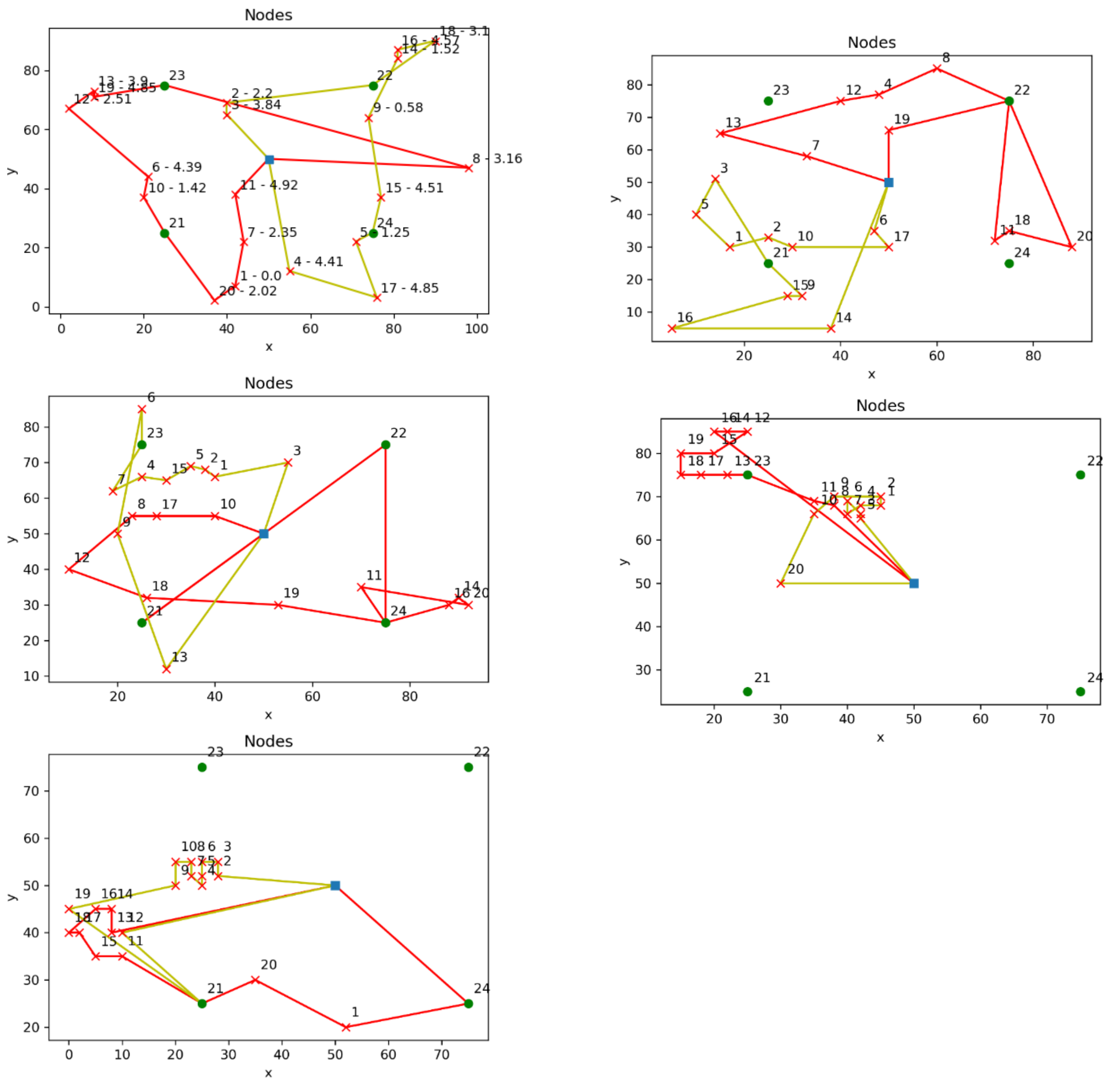


Figure 2: Instance 1 – 2 – 3 – 4 – 5 (left to right respectively)

One of the reason that the model took so long is the dummy variables. Since all nodes must be visited exactly one time, conversly drones can stop by into recharge stations multiple times to recharge their batteries. So, as mentioned in the 2nd section, $V_{R'}$ set includes the dummy recharge station variables. Dummy Recharge stations are generated by copying each RS node multiple times (as many as a pre-determined upper bound on the number of needed rechargings, 5 for each recharge stations for the problem). This means that the number of nodes are almost duplicated.

Another issue with the model is that, as can be seen in Figure of Instance 5, one of the vehicle, red tour, travelled all the assigned nodes and before going back to depot, it travelled between several recharge stations. This is because that the recharge stations and depot do not have priority score, which means their value is 0. So, for a drone, going back to depot lately, or stopping by into recharge stations at the end does not affect the objective function, since the formula is; completion time * priority score which is 0. We can assign a small priority value for the depot and recharge stations in order to prevent this situation.

6. Solution

First of all, to understand the concept of the problem deeply, we tried to solve it with constructive heuristics at the beginning. Then, in order to have comparable results gained by solution approaches, an additional implementation is decided to be applied. For this purpose, simulated annealing is selected since it shows parallelism with the requirements of the problem.

The span between the new point and the old one is distributed with a probability function that is proportional to the temperature. The algorithm keeps the points that are better than the previous ones but also doesn't neglect the points that are worse; it has certain probability to accept what is worse. Therefore, algorithm avoids being trapped in local minimum and check for more possible solutions. Temperature is decreased in the process of applying the algorithm, which is interpreted as a search mechanism.

The more temperature decreases, the narrower searching range for solution is achieved. That means getting closer to the optimal solution.

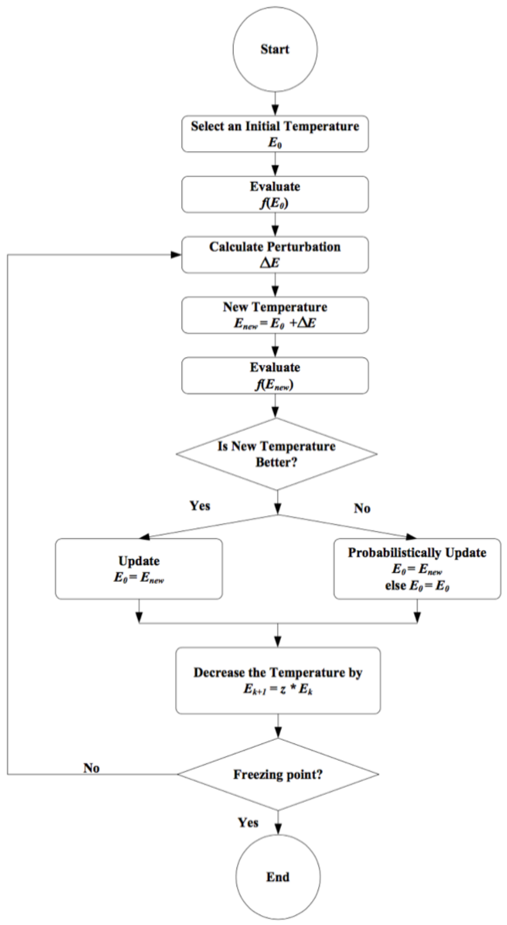

Start
Select an Initial Temperature E_0
Evaluate $f(E_0)$
Calculate Perturbation ΔE
New Temperature $E_{new} = E_0 + \Delta E$
Evaluate $f(E_{new})$
Is New Temperature Better?
Yes
No
Update $E_0 = E_{new}$
Probabilistically Update $E_0 = E_{new}$ else $E_0 = E_0$
Decrease the Temperature by $E_{k+1} = z * E_k$
Freezing point?
No
Yes
End

Figure 2. Overall procedure of the SA algorithm (Zhong & Pan, 2007)

A. Solution Representation

To construct the Simulated Annealing for solving the given problem, a fitting representation scheme that displays the solution characteristics is needed. The solution is consist of multiple nodes to cover and represent a rotue. For each route, there must be a depot at the beginning and at the end, the recharge points may or may not be located within the solution interval depending on the state of the solution.

B. Initial Solution

The initial X_0 has been generated by developing a constructive heuristic. The algorithm steps are listed below:

Greedy Herustic

- **Step 1.** All the nodes are sorted according to their priority from small to large (descending).
- **Step 2.** Add the nodes to drones one by one. If the drone is not empty, calculate the energy consumption required between the last node in the drone and the one that is

wanted to be added. If this value is less than half of the average battery consumption in the system, add this new node to the vehicle. If not, add it to the next empty vehicle. As operating the adding process, control the vehicle's battery required from the added node to the recharge station. Consider whether it reaches to the station or not. If it does, add the node. If it does not reach, do not add the node. Repeat this process untill at least one node is added for all vehicles.

- **Step 3.** After each point has a point, add the new point to the nearest drone. While doing this, check the status of reaching the recharge point after the added point, if it can reach the add, if not, check the other nearby point, continue this process until the point is added.
- **Step 4.** Continue the third step until all points are added to the drones. Stop after all points have been added.

After generating initial solution, we passed it to Simulated Annealing on improving. Initial solution and Simulated Annealing results will be compared in further sections.

C. Neighborhood Generation

Neighborhood search approach is utilized as the algorithm. In order to find a neighbor solution of the current solution during each iteration a narrowed neighborhood generation mechanism is composed. The solution is accepted if it improved, otherwise it accepted with a probability which is calculated by $P_T(\delta) = e^{\frac{-(f_S - f_0)}{T}}$

<u>Swap operator:</u>

We used a swap method as the first operator. The aim is to swap through the nodes inside the routes, in order to improve our results. First of all, the recharge stations are removed from the tours on swapping operation, when two nodes swapped inside the route, we will assign a new recharge station if the batery level after the operation is not enough. The reason we are removing and reassigning the recharge stations that to avoid long distance of recharge edges. After reconstructing the route with new recharge stations, note that recharge stations may stay the same, may increase or decrease in terms of number of routes, in a route. If the solution is improved, the algorithm takes it as the best strategy. On the other hand, it accepts the solution with a probability which is mentioned above. After the swapping all the nodes inside a route, the algorithm saves the best strategic solution to pass into next operator, which is replace.

<u>Replace operator:</u>

Second operator is replace, which basically swaps through the routes (Inter-routes). The removal of recharge stations is same as the swap operator. After that, the algorithm searches for the best strategy by changing the nodes between the routes. If the solution is improved, it accepts, otherwise it accepts with a probablity, again. At the end, it saves the best solution.

<u>Remove-Add operator:</u>

Third operator is remove-add. This operator first removes the recharge station from the given solution represetantaion array as in the other operators, then removes the node from one vehicle tour and adds it to another vehicle tour. Thanks to this operator, the length of the routes can be changed and we can reach the results that previous operators cannot reach.

D. Objective Function

The objective of the model is to minimize the sum product of nodes' completion time and priority for all nodes, hence whenever a neighbor solution is generated during the problem, the swap and replace move operator is used to calculate the new feasible route to find the new objective. So the algorithm tries to generate a solution which covers more prioritized areas first.

E. The Pseudocode of Simulated Annealing Algorithm

```
Step 1. Initialization:
    Step 1.1. Generate an initial solution with constructive herustic ($X_0$)
    Step 1.2. Set $X_b$=$X_c$=$X_0$
    Step 1.3. Set f($X_b$)=f($X_c$)=f($X_0$)
    Step 1.4. Initiate the initial temperature $T_0$, the final temperature $T_F$, and the cooling rate λ Set $T_c$=$T_0$
Step 2. While $T_c$> $T_F$ do:
Step 2.1. Set counter to 1
Step 2.2. While counter <= epoch_length do:
    Step 2.2.1 For each move operator:
            Step 2.2.1.1. Apply neighbor generation mechanism on the current solution ($X_c$) to generate a neighbor solution ($X_n$)
            Step 2.2.1.2. Calculate f( $X_n$ ) in mathematical model for objective function
            Step 2.2.1.3. Calculate $\Delta$f = f($X_n$)-f($X_c$)
            Step 2.2.1.4. If $\Delta$f ≤ 0, then set $X_c$=$X_n$; if f($X_n$)-f($X_b$) < 0  set $X_b$=$X_n$
            Step 2.2.1.5. If $\Delta$f > 0, then generate a random number R ~ [0,1]; if $e^{-\Delta f/T_c}$ > R, then set $X_c$=$X_n$.
            Step 2.2.1.6. Increase counter by 1
    Step 2.3. Set $T_c$ = λ $T_c$

Step 3. Report $X_b$
```

7. Results

I. Greedy Herustic

First of all, we compared the results of the mathematical model mentioned in chapter 5 and the greedy herustic results for the 5 instances we created. We made this comparison to see the performance of our greedy herustic approach as well as to evaluate our results according to mathematical model results. As a result, on average exact solutions are 37% better in terms of objective.

Exact vs Greedy			
Instances	Exact Solutions	Greedy Solutions	% Difference
1	15,023.65	25,721	-42%
2	6,119.54	8,576	-29%
3	9,681.74	15,937	-39%
4	10,769.02	16,399	-34%
5	9,521.44	16,808	-43%
Average			-37%

Table 2: Exact and Greedy Results for 5 instances

II. Classic Simulated Annealing

The results that we generate with constructive heuristics and Simulated Annealing is quite promising. In this simulated annealing implementation, we received the epoch length of 2 and the cooling rate of 90%. Also, we set the initial temperature, which is one of the most important parameters for SA, to be accepted as a result that is twice the greedy heuristic result.

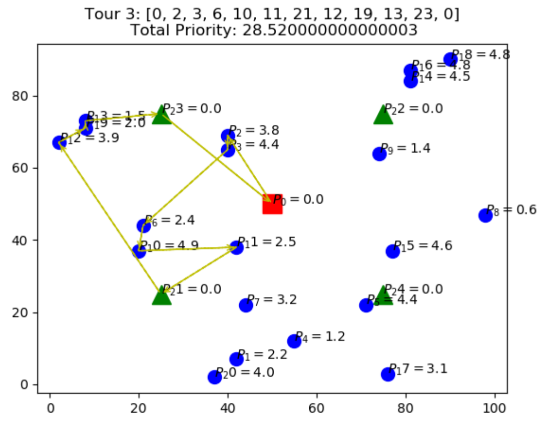
Constructive Heuristic			Simulated Annealing		
Nodes	Completion Time	Score * Completion	Nodes	Completion Time	Score * Completion
10	75.39	370.92	2	52.94	203.30
3	154.21	680.06	3	70.94	312.85
12	248.97	970.98	6	137.58	323.32
2	335.07	1,286.68	10	161.72	795.68
11	407.20	1,022.08	11	215.77	541.58
6	460.88	1,083.08	12	364.34	1,420.93
19	530.82	1,072.25	19	388.76	785.30
13	544.82	828.12	13	402.76	612.20
18	123.14	597.21	18	123.14	597.21
16	152.11	737.74	16	152.11	737.74
15	265.05	1,211.27	15	265.05	1,211.27
14	369.39	1,665.94	14	369.39	1,665.94
5	504.99	2,216.91	5	504.99	2,216.91
20	593.88	2,369.59	20	593.88	2,369.59
7	646.26	2,042.19	7	646.26	2,042.19
17	730.69	2,265.15	17	730.69	2,265.15
1	809.16	1,780.16	1	809.16	1,780.16
9	993.93	1,411.38	9	993.93	1,411.38
4	1,114.66	1,393.32	4	1,114.66	1,393.32
8	1,235.54	716.62	8	1,235.54	716.62
Total	**10,296.17**	**25,721.67**	**Total**	**9,333.64**	**23,402.65**

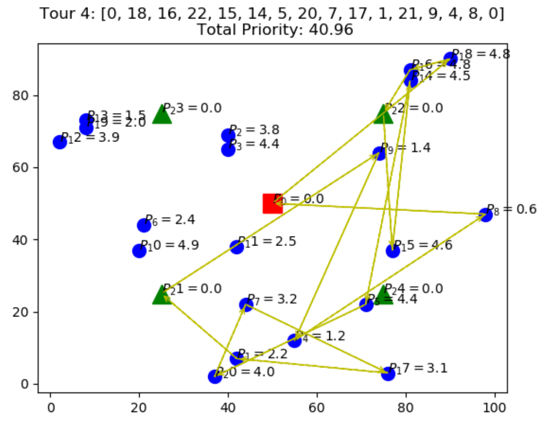
Table 3. Results

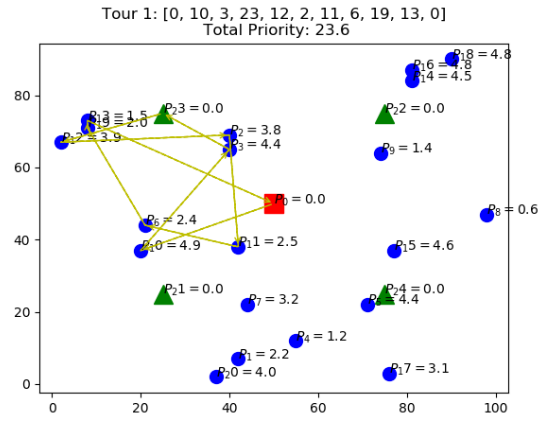
As you can see from the table 2, initial solution is improved about 9% in terms of objective function, which is multiplication of priority score and completion time. Also, the number of stop by into recharge stations is also changed in route 1. However, there is no change in second route. Intra route swap operations improved the solution. Note that 21, 22, 23 and 24 are the recharge stations. Since priorities of the recharge stations are 0, it is excluded from the Table 2. We can run the algorithm with different instances and different parameters in order to improve the constructive solution more.

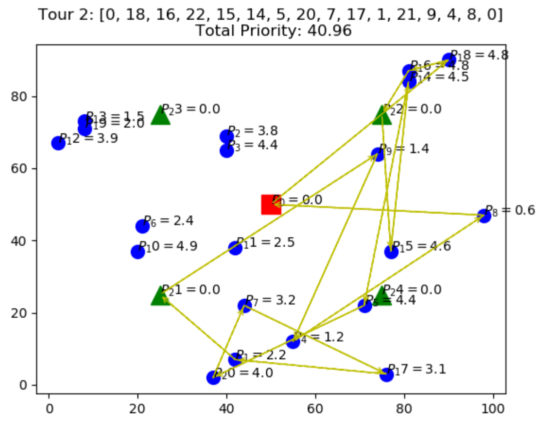
Algorithm	**Tours**
Constructive Heuristic	Drone 1 Route: [0, 10, 3, 23, 12, 2, 11, 6 , 19, 13, 0]
	Drone 2 Route: [0, 18, 16, 22, 15, 14, 5, 20, 7, 17, 1, 21, 9, 4, 8, 0]
Simulated Annealing	Drone 1 Route: [0, 2, 3, 6, 10, 11, 21, 12, 19, 13, 23, 0]
	Drone 2 Route: [0, 18, 16, 22, 15, 14, 5, 20, 7, 17, 1, 21, 9, 4, 8, 0]

Table 4. Tours









As we can see from the figures above, note that tour 1 is related with tour 3, tour 2 is related with tour 4, drone 1 in tour 1 traveled like a star shape which is ineffectively, however, applying simulated annealing with swap operations inside the route, it is improved in terms of less completion time and gained more priority score. Our main aim is to try different parameters to improve the route 2, to get a better solution.

III. Adaptive Simulated Annealing

It was observed that Classic SA algorithm is better than greedy approach but worse than Mathematical model results. In this case, an adaptive strategy has been developed to improve the results of the SA algorithm, to approach the results of the mathematical model, and to give better results than the classical SA algorithm for large instances that cannot be solved with the mathematical model.

Adaptive strategy is provided by 2 different approaches. These are tabu list that prevent access to existing solutions that have been created recently and cycling. The second is the diversification technique, observed in SA in some cases, which allows us to increase the solution space by eliminating local solutions.

a. Tabu List

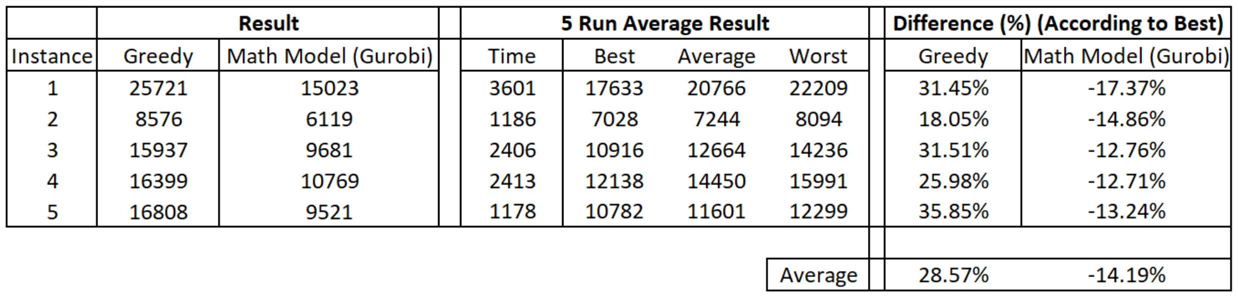
We modified the classic tabu list such that result values were retained instead of the most recent node changes to the tabu list. The purpose of this is to make it easier to implement and get fast solutions, as changes between the same and different routes can be made.

The SA algorithm has two main characteristics; stochastic and memoryless. SA algorithm has become a memory-based algorithm with tabu list implementation.

b. Diversification

It has been observed that the classic SA algorithm is stucked in local solutions and in some scenarios it cannot escape. Diversification has been applied to solve this situation and to reach different solutions. This is best applied if the solution does not change for a certain period of time or if some results are repeated frequently. This approach creates a random solution and is accepted without a condition.

In order to compare two algorithms (i.e., clasic SA and adaptive SA) more fairly, we prefer to use same parameters. The Adaptive SA algorithm was run with 5 instace mentioned in chapter 5. Each instance was run 5 times and examined for best, average and worst solution. The results were compared with the Greedy heuristic and the results of the mathematical model. As shown in Table 4, the best results of the SA algorithm were compared. The Adaptive SA algorithm performed 28.57% better on average than the Greedy approach, but was 14.19% worse than the mathematical model.



	Result		**5 Run Average Result**				**Difference (%) (According to Best)**	
Instance	Greedy	Math Model (Gurobi)	Time	Best	Average	Worst	Greedy	Math Model (Gurobi)
1	25721	15023	3601	17633	20766	22209	31.45%	-17.37%
2	8576	6119	1186	7028	7244	8094	18.05%	-14.86%
3	15937	9681	2406	10916	12664	14236	31.51%	-12.76%
4	16399	10769	2413	12138	14450	15991	25.98%	-12.71%
5	16808	9521	1178	10782	11601	12299	35.85%	-13.24%
						Average	28.57%	-14.19%

Table 4. Comparison Tables

Combining SA with tabu list increases the computational time. Therefore, it is necessary to clearly state the benefits of making tabu lists. For this purpose, the first instance was considered and the adaptive SA algorithm was run 5 times without Tabu list. The average result is 21,998. When tabu list is combined to SA, this result is improved on average by 5.5%.

8. Conclusion

One thing that we learnt from the exact solution is that the drone visits a recharge station although it has a battery left to reach next node. In our algorithm, we did not consider this since our aim is to visit all nodes as soon as possible, when there is not enough battery left to reach a node then a recharge station, our solution first visits the recharge station, then the node. However, exact solution shows that a drone can visit recharge stations even it has enough battery to visit the node.

It can be concluded that although exact solutions are better in terms of objective function, Simulated Annealing is faster than exact methods and can be easily implement into a real-life case, with larger nodes and real parameters, since time is more important in these type of problems.

Exact methods performs better than SA (i.e., the gap is by 14.19%); however, computational time of SA is critically lower than the exact methods. Therefore, SA is more promising for this humanitarian setting.

9. Future Work

As future work, we are planning to solve the problem with larger instances in Istanbul dataset. Also, the parameters of the problem can be more realistic, since we solved the problem with basic parameters. Moreover, we are planning to update our meta heuristic with a different adaptive moves since it performs not very well when comparing with exact solution in small instances.

10. References

- Schneider, M., Stenger, A. and Goeke, D. (2019). *The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations*.

- Otto, A., Agatz, N., Campbell, J., Golden, B., & Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. *Networks*, *72*(4), 411-458. doi: 10.1002/net.21818

- Chauhan, D., Unnikrishnan, A. and Figliozzi, M. (2019). Maximum coverage capacitated facility location problem with range constrained drones. *Transportation Research Part C: Emerging Technologies*, 99, pp.1-18.

- Chowdhury, S., Emelogu, A., Marufuzzaman, M., Nurre, S. and Bian, L. (2017). Drones for disaster response and relief operations: A continuous approximation model. International Journal of Production Economics, 188, pp.167-184.

- Oruc, B. and Kara, B. (2018). Post-disaster assessment routing problem. *Transportation Research Part B: Methodological*, 116, pp.76-102.