EN.560.650
Operations Research

Team E Final Report

Title

Optimization of the Contactless Drone Food Delivery for Johns Hopkins Hospital Patients

Abstract

Patients staying at home need properly selected food to be delivered on time. When a society is nearly closed due to a crisis, such as the coronavirus pandemic, it becomes much harder for them to get the food. In this project, we designed drone flying paths to promote contactless food delivery for patients affected by the road closure as well as to lower the flying cost. We considered a variety of constraints, such as the payload of drone, the number of grocery stores, the number of patients, food requirement of each patient, and the delivery location. Calculated in Julia JuMP, our model successfully found the optimal paths for each drone in three sample zip code areas. We also tested the robustness of the model, aiming to provide more solutions under different scenarios.

Team members and Task Assignment

Team members: *Yuhang Zhang, Yijia Bai and Ensheng Dong.*

Task assignment and contribution: Yuhang Zhang is mainly responsible for building the model and the algorithm to get the result. Yijia Bai is responsible for data collection and cleaning. Both make contribution to the data visualization and the final report. Ensheng Dong offers useful insights on the final report and the presentation.

Contents

1. Introduction

1.1 Background

Prior to a pandemic, Baltimore County and Johns Hopkins Hospital (JHH) are preparing to deliver food to diet-sensitive and immune-compromised patients by drones, so as to reduce contact times for these patients and lessen the secondary health impact of viral infections.

1.2 Problem Statement

Our goal in this project is to find the best way to match grocery stores and patients in Baltimore County in order to satisfy a once-per-week delivery target and minimize the operational cost.

1.3 Assumptions

- We need to deliver food for 20 patients with chronic kidney disease (CKD) in each aera.
- We regard this problem as a capacitated vehicle routing problem with single depot.
- We have a fixed number of drones to serve patients in each zip code area. The number of drones is decided by the number of patients, the requirement of each patient and the capacity of the drones in each zip code.
- We have only one grocery store in each zip code area and we will choose the most centric grocery store in each zip code as our depot.
- In order to be closer to the reality, we will randomly distribute the locations of patients according to the community distribution in real life.
- Each drone will depart from the grocery store at the same time and eventually return to the grocery store after serving several patients.
- Each patient can be served by only one drone, which means only one drone pass by at each point (patient location).
- Each patient needs 15*kg*-20*kg* of food per week and the value would be randomly chosen for each patient.
- Suppose that the routing path of the drone flying from point *i* to point *j* is a straight line from *i* to *j*.
- Our objective is to find the minimum flying distance of all drones under the condition of satisfying each patient's requirement.

2. Mathematical Model

2.1 Variables

The classic vehicle routing problem can be formally defined as follows. Let $G = (V, arcs)$ be a graph, where $V = \{0,1,2,\dots,n\}$ is a vertex set and $arcs = \{(i,j): i, j \epsilon V\}$ is an arc set, which contains all possible arcs between any two points in V, e.g. (0, 1), (2, 2), (3, 6), etc. We suppose that 0 represents the depot, which is the grocery store in our case, while $patients = \{1,2,\dots,n\}$ represents the patients' locations.

Since our objective is to find the minimum flying distance of all drones, we make our decision variable as follows:

$$x_{ijk} = \begin{cases} 1, & if\ drone\ k\ fly\ from\ i\ to\ j \\ 0, & otherwise \end{cases} \tag{1}$$

2.2 Objective Function

Based on our assumptions, the objective function can be written as follows:

$$Min\ Z = c_1 \sum_{(i,j)\in arcs} \sum_{k=1}^{m} d_{ij} x_{ijk} \tag{2}$$

where c_1 is the cost of unit distance (operational cost), d_{ij} is a $n \times n$ square matrix which contains the distance between patient i and j, $i, j \in V$, m is the number of drones, which is a fixed value based on our assumption. In our project, we suppose $c_1 = 5\$/km$.

2.3 Constraints

$$\sum_{i=0}^{n} \sum_{k=1}^{m} x_{ijk} = 1 \quad \forall j \neq 0 \tag{3}$$

$$\sum_{j=0}^{n} \sum_{k=1}^{m} x_{ijk} = 1 \quad \forall i \neq 0 \tag{4}$$

$$\sum_{i=0}^{n} x_{ihk} - \sum_{j=0}^{n} x_{hjk} = 0 \ \ \forall h \neq 0, \forall k \tag{5}$$

$$\sum_{j=1}^{n} x_{0jk} = 1 \quad \forall k \tag{6}$$

$$\sum_{i=1}^{n} x_{i0k} = 1 \quad \forall k \tag{7}$$

$$\sum_{i\epsilon patients, j\in V} q_i * x_{ijk} \leq Q \quad \forall k \tag{8}$$

$$\sum_{i,j\in arcs} d_{ij} x_{ijk} \leq D \quad \forall k \tag{9}$$

$$u_{0k} = 1 \quad \forall k \tag{10.1}$$

$$2 \leq u_{ik} \leq n + 1 \quad \forall i \neq 0, \forall k \tag{10.2}$$

$$u_{ik} - u_{jk} + 1 \leq n * \left(1 - x_{ijk}\right) \quad \forall i \neq 0, \forall j \neq 0, \forall k \tag{10.3}$$

Constraints (3) (4) ensure that each patient is visited exactly once. Constraint (5) ensures the same drone k entering and leaving each patient's location. Constraints (6) and (7) ensure all drones departing from and eventually returning to the same grocery store.

Constraints (8) and (9) are capacity constraint and distance constraint, where q_i is the requirement of patient i, Q is the capacity of each drone, D is the maximum flying distance of each drone. Constraints (10.1) (10.2) (10.3) are subtour elimination constraints (MTZ formulation). Finally, we should notice that decision variables are binary.

3. Implementation and Analysis

3.1 Data Collection

3.1.1 Data of Drones

Our project choose gyroplane sold at Alibaba as the drone prototype, which has a max payload of $100kg$. More details are shown in the following table [1].

Table 1: Details about Selected Drone

Price($)	*Max Payload Capacity(kg)*	*Drop Precision*(m^2)	*Max Flying Distance*(km)
$48{,}000$	100	< 10	10

3.1.2 Data of Requirements

We assume each patient needs $15\text{kg} - 20\text{kg}$ of food per week and the value would be randomly chosen for each patient [2].

3.1.3 Locations of Grocery Stores

Based on our assumptions, there is only one grocery store serving patients in each zip code area [3]. The information of selected grocery stores is shown as follows.

Table 2: Details about Grocery Stores

Name	*ZIP_Code*	*Point_X*	*Point_Y*
Mars Super Markets 20	21239	-76.58307192	39.36690241
9 Petro Inc	21206	-76.54115594	39.34519304
Win Rich	21214	-76.56167204	39.3524232

3.1.4 Locations of Patients

We generated 20 random points in each zip code area in ArcGIS Pro [4]. Coordinates of each patient’s location are listed in these files: *“patients21239_TableToExcel.csv”*, *“patients21214_TableToExcel.csv”*, *“patients21206_TableToExcel.csv”*.

3.2 Implementation in Julia JuMP

3.2.1 Using 3 drones

We use Julia JuMP and Gurobi to build and solve the model. We first import necessary packages. Then we define some basic variables and containers in Julia.

Step 1: Define variables

```
Random.seed!(1)
n = 20                              # The number of patients
c_1 = 5                             # Unit distance cost
patients = [2:n+1;]                 # Container of patients
V = [1:n+1;]                        # Vertex set, 1 denotes the grocery store
q = [rand(12:15,n+1);]              # Requirement of each patient
arcs = []                           # Container of all arcs
for i in V
   for j in V
      push!(arcs,(i,j))
   end
end
Q = 100                             # Capacity of each drone
D = 10;                             # Max flying distance
```

Next, we clean and import data to Julia, and calculate the distance matrix based on Haversine Formula.

Step 2: Import data and calculate

```
data = CSV.read("Patients21239_TableToExcel.csv")                    # Import coordinates of all points
x_loc = data[:,3];                                                    # Get x-coordinate of all points
y_loc = data[:,4];                                                    # Get y-coordinate of all points
d = zeros(n+1,n+1)                                                    # Initialize distance matrix
for i = 1: length(V)
   for j = 1: length(V)
      d[i,j] = haversine((x_loc[i],y_loc[i]),(x_loc[j],y_loc[j]),6372.8)   # Calculate distance matrix
   end
end
```

Finally, we build our model by using JuMP.

Step 3: Modeling and solving

```
using JuMP, Gurobi
v = 1:3                                                               # Number of drones
m = Model(Gurobi.Optimizer);
@variable(m,x[arcs,v],Bin)
@variable(m,u[1:n+1,v])
# Each patient will be visited exactly once
@constraint(m,single1[j in patients],sum(x[(i,j),k] for i in V for k in v)==1)
@constraint(m,single2[i in patients],sum(x[(i,j),k] for j in V for k in v)==1)
# Drone starts from depot
@constraint(m,depot1[k in v],sum(x[(1,j),k] for j in patients)==1)
# Drone ends at depot
```

```
@constraint(m,depot2[k in v],sum(x[(i,1),k] for i in patients)==1)
# Capacity constraint
@constraint(m,capacity[k in v],sum(q[i]*x[(i,j),k] for i in patients for j in V)<=Q)
# Balance constraint
@constraint(m,balance[h in patients,k in v],sum(x[(i,h),k] for i in V)-sum(x[(h,j),k] for j in V)==0)
# Subtour Elimination (MTZ)
@constraint(m,subtour1[k in v], u[1,k]==1)
@constraint(m,subtour2[k in v,i in 2:n+1], u[i,k]>=2)
@constraint(m,subtour3[k in v,i in 2:n+1], u[i,k]<=n)
@constraint(m,subtour4[k in v,i in 2:n+1, j in 2:n+1], (u[i,k]-u[j,k]+1)<=(n-1)*(1-x[(i,j),k]))
# Maximum flying distance constraint
@constraint(m,maxdis[k in v], sum(d[i,j]*x[(i,j),k] for i in V for j in V)<=D)
# Objective Function
@objective(m,Min,c_1*sum(d[i,j]*x[(i,j),k] for i in V for j in V for k in v))
optimize!(m)
```

After running the code above, we get the objective value. However, we still need to know the exact decision variables to verify if this solution is feasible or not. The follow snippet generates the arcs contributing to the minimal cost.

Step 4: Get the route of each drone

```
val = value.(x)                          # Get the value of each decision variable
for k in v
   for i in V
      for j in V
         if val[(i,j),k]==1              # Find the decision variable with value of one
            println(x[(i,j),k])          # Print these variables
         end
      end
   end
end
```

3.2.2 Using 4 or 5 drones

If the number of drone changes, what we need to do is to change the variable v in step 3, which represents the number of drones. In this way can we easily get the objective value in different scenarios.

3.2.3 Certain order needed

In order to satisfy certain patients who need to receive their food first, we can add some constraints on the variables to predefine some fixed routes manually. For example, suppose we need to deliver food to patient 3, patient 7 and patient 11 first, we can add constraints like this:

```
@constraint(m,x[(1,4),1]==1)
@constraint(m,x[(1,8),2]==1)
@constraint(m,x[(1,12),3]==1)
```

It is worth noting that the index of Julia's array starts from 1. Therefore, the grocery store is represented as 0 in the mathematical model while it shows as 1 in the Julia script. The same index number increase is also applied to patients' locations.

3.3 Analysis and Results

3.3.1 Question 1

How many drones are used based on your recommendation? What happens if the number of drones is increased (you get a higher budget)? What if it decreases (budget cut or some of the drones are not in working conditions)?

- Based on our model, in case of 20 patients in each area, we recommend 3 drones for each area. The minimum objective value is ***81.59*** for area 21239, ***70.60*** for area 21214 and ***102.45*** for area 21206.
- We can easily figure out that there is no feasible solution if the number of drones is fewer than 3 because of the capacity constraints. However, we could have fewer drones if multiple trips for each drone are allowed.
- If the number of drones is increased, we only need to change the variable v which is the number of drones, and the optimal solution of the minimal cost will be also increased. Specifically, the table below shows the relationship between the number of drones and the objective value.

Table 3: Objective value of different number of drones

# of drones	*21239*	*21214*	*21206*
2	Infeasible	Infeasible	Infeasible
3	81.59	70.60	102.45
4	83.95	73.06	105.39
5	86.75	75.69	109.85

- *Figure 1 (left) and Figure 2 (left)* is the optimal solution with 3 drones in each zip code area, while *Figure 1 (right) and Figure 2 (right)* are the solutions when there are 4 or 5 drones in each zip code area respectively. Compared with only 3 drones in each area, we could see more shorter arcs when there are more drones. That also means one drone may serve only 2 patients while the other may serve more than 10 patients in one task. However, that is unnecessary and a waste of cost.

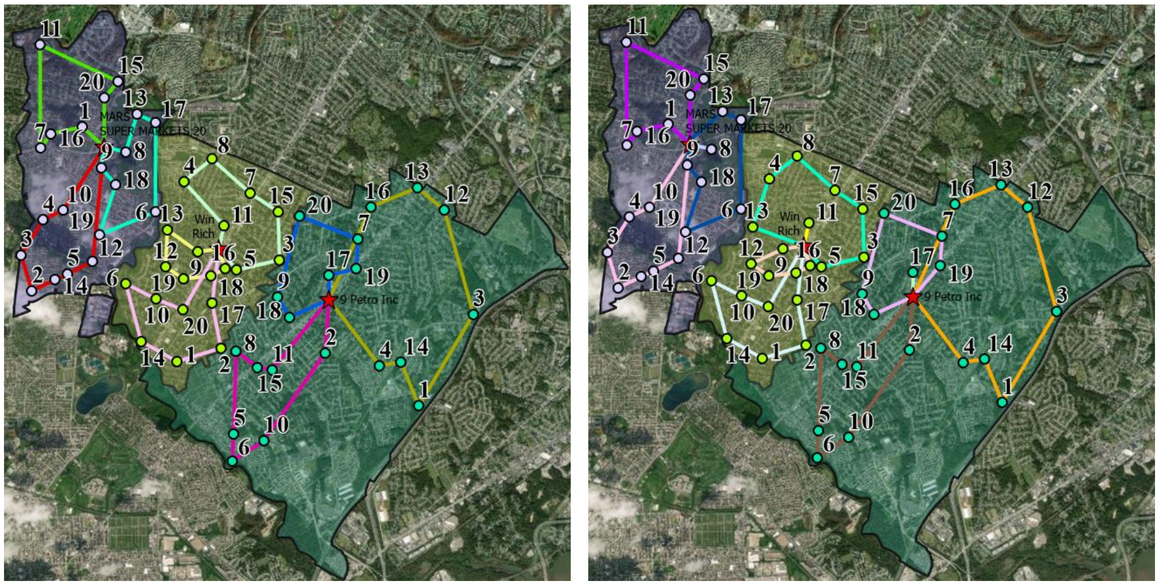

Figure 1: Delivery Routes of 3 Drones(left) and 4 Drones(right)

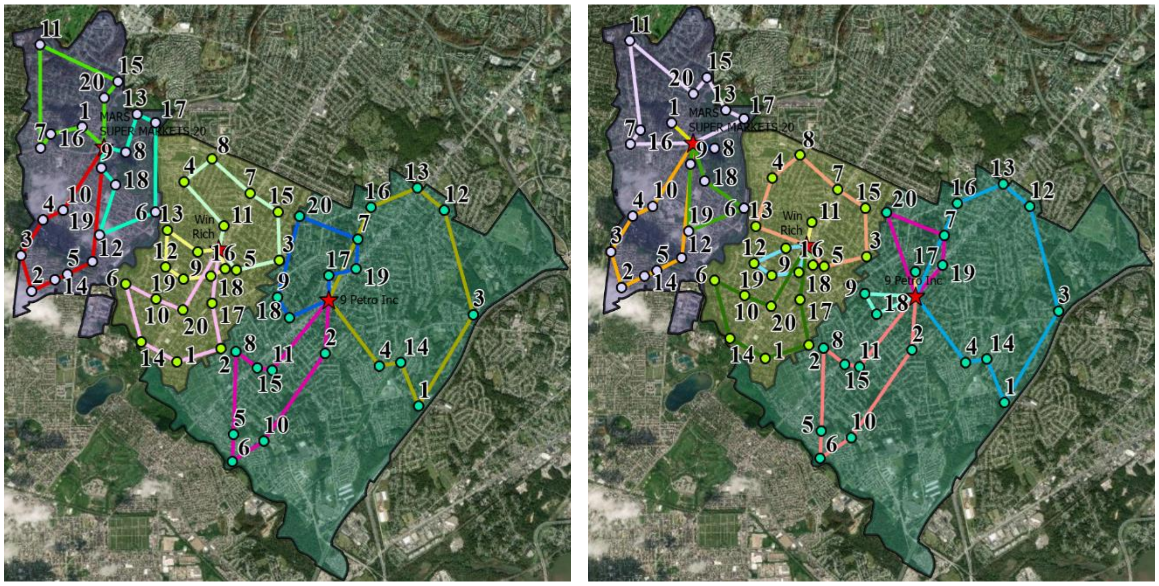

Figure 2: Delivery Routes of 3 Drones(left) and 5 Drones(right)

3.3.2 Question 3

What is the order in which the patients receive food from these drones? Is there a better ordering? What if certain patients had to receive their food first? How would your model and how would the optimal solution change?

- Let’s take three drones as an example, the order of food delivery is like follows:

  Zip code 21239:

  Drone 1: 0→10→4→3→2→14→5→12→0

Drone 2: 0→8→13→17→6→19→18→9→0

Drone 3: 0→1→16→7→11→15→20→0

Zip code 21214:

Drone 1: 0→16→18→19→12→13→9→0

Drone 2: 0→17→2→1→14→6→10→20→0

Drone 3: 0→5→3→15→7→8→4→11→0

Zip code 21206:

Drone 1: 0→2→10→6→5→8→15→11→0

Drone 2: 0→4→14→1→3→12→13→16→0

Drone 3: 0→17→19→7→20→9→18→0

Since this is optimal solution, we consider it as the best ordering.

- If we need to prioritize the delivery for certain locations, patient 3, 7 and 11 in area 21239 for instance, we only need to add these constraints.

```
@constraint(m,x[(1,4),1]==1)
@constraint(m,x[(1,8),2]==1)
@constraint(m,x[(1,12),3]==1)
```

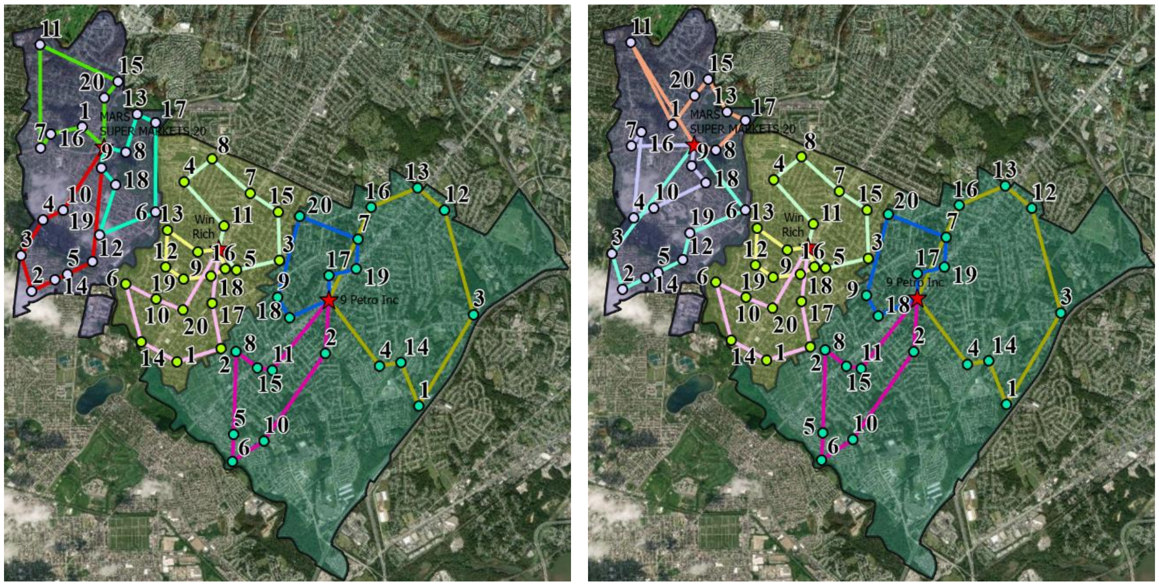

Figure 3: Delivery Routes of Certain Orders (21239) left: before; right: after

Based on assumptions in section 3.2.3, the results are displayed in Figure 3 above and the detailed pathes below.

Zip code 21239:

Drone 1: 0→**3**→2→14→5→12→19→6→0
Drone 2: 0→**7**→16→4→10→18→9→0
Drone 3: 0→**11**→1→20→15→13→17→8→0

- The objective value of certain orders is ***87.29*** while it is ***81.59*** before this change. Generally, we think the objective value would increase since these certain orders are lower likely to be the optimal route.

3.3.3 Question 4

What if the number of patients triples? What if it is 10 times or 100 times the current number? What if you need to deliver food to the entire population?

- It will take a much longer time to run the model as the number of patients increases. For 100 patients, we didn't get a solution even after 10 hours running based on our model. However, this result is not surprising because the number of variables is increasing with an exponential trend as the number of patients(points) increases.
- We can divide large problem into several smaller problems. Specifically, for 100 patents, we can set up 20 receiving stations, each of which is responsible for 5 patients nearby. Thus, the large problem is simplified to problems with smaller number of points.

4. Conclusion

4.1 Recommendation

During this unprecedented period, we recommend to operate at least 3 drones, or gyroplanes, to deliver food for 20 patients of each area. Following the route given in section 3.3.2, we can achieve the goal of minimizing the operation cost.

4.2 Further Considerations

There are a couple of things we need to consider in the future:

- Delivery time will be considered, that is, there is a time window for each patient. The food delivery should be done within certain required period. What if a customer failed to pick up the food on time? What if a customer took too much time to get the food? More scenarios need to be put into consideration.

- The capacity of the grocery store. We need to make sure if each depot stores enough food to meet all its customers' need.
- The flying restricted and altitude zones. We could consider adding more constraints to deal with regulatory restricted zones according to the Federal Aviation Administration (FAA) and local authorities.
- We need to consider the impact of the height of buildings on the routes of each drone.
- It is less possible to have only one grocery store as our depot, so we should consider this problem with multiple depots.

References

[1] Data about drone (c_1, c_2, Q, D): we choose the Gyroplane from Alibaba:

https://www.alibaba.com/product-detail/Large-100kg-payload-drone-heavy-lifting_62149339243.html?spm=a2700.7724857.normalList.1.59767857GdRBrG

[2] Data about food requirement of each patient:

https://www.quora.com/How-many-pounds-of-food-does-the-average-person-eat-each-day

[3] Data about locations of grocery stores:

https://data-clf.hub.arcgis.com/datasets/c4a2bd61eaac4425b3e2e9c40735a7ae218

[4] Data about patients' locations:

Generated randomly from ArcGIS Pro.