For office use only	Team Control Number	For office use only	
T1 ________	**1911507**	F1 ________	
T2 ________		F2 ________	
T3 ________	Problem Chosen	F3 ________	
T4 ________	**B**	F4 ________	

2019
MCM/ICM
Summary Sheet

Send in the Drones: Developing an Aerial Disaster Relief Response System

Summary

This article gives a modified integer programming model based on a combination of traditional branch-and-bound method and simulated annealing bin-packing algorithm. It aims at addressing the complicated resource allocation problem, which naturally occurs in designing disaster relief response system, step by step from the perspective of a overall and comprehensive analysis of the interaction within a complex system.

The DroneGo disaster response system is dispatched to Puerto Rico for two major missons: medical supply delivery and video reconnaissance of road networks, none of which is dispensable. Since focusing on the former solely helps attain a clear insight into this sophisticated problem, we first construct an integer programming model as a basis to be perfected gradually. This model is capable of deciding the cargo container packing configuration, the drone payload packing configurations and the destination of each drones for the demand of the five selected location. Then we extend our analysis to an integrated model for double objectives. In order to specify the appropriate locations of containers and the schedule of drone fleet, we abstract a network from the real map first with the help of geographic information technology. Then we implement the Dijkstra's algorithm to solve for the shortest path as a preparation for drone route planning. Finally we manage to connect our integer programming model with a comlpex network and make a complete model which can solves all the variables which is necessary to establish the DroneGo disaster response system.

To develop more accurate and efficient model, we utilize the simulated annealing algorithm to address the bin-packing problem which are integrated into the traditional branch-and-bound method for integer programming. The last model improves both accuracy and efficiency considerably, epitomizing what we have been dedicated to in this article.

Keywords: integer programming; simulated annealing algorithm; bin-packing problem

Contents

1 Introduction

1.1 Problem Background

Natural disasters often bring huge casualties and economic losses on human society. In 2017, the worst hurricane in nearly a century landed in Puerto Rico, and left almost the entire island without power, and many without running water or cell phone service. Many highways and roads were blocked because of widespread flooding, which brought great difficulties to rescue route planning. Demand for medical supplies was also soaring for some time.

HELP, Inc., one of non-governmental organizations, is planning to design a transportable disaster response system called DroneGo to conduct medical supply delivery and video reconnaissance.

1.2 Restatement of Problem

Our task is to design a DroneGo disaster response system to support potential future disaster scenario similar as the Puerto Rico hurricane. Based on the 2017 situation in Puerto Rico, we will:

- Provide the packing configuration for each of no more than three ISO cargo containers to transport the system.
- Find the best locations on Puerto Rico to position these cargo containers to realize both medical supply delivery and video reconnaissance.
- Design the drone payload packing configurations, delivery routes and schedule to conduct medical supply delivery, and a drone flight plan to realize video reconnaissance of road networks.

2 General Assumptions

Assumption I: DroneGo disaster response system consists of up to three ISO cargo containers, a DroneGo fleet and some emergency medical packages.

The three cargo containers are identical with standard size, designated as Container A, Container B and Container C respectively. The DroneGo fleet is a combination of drones selected from eight types of potential candidates, namely Drone A to Drone H. Only three types of emergency medical packages are available, referred to as MED 1, MED 2, and MED 3.

Assumption II: DroneGo disaster response system is designed for possible Puerto Rico hurricane disaster.

When disasters occurs, a DroneGo fleet and some emergency medical packages will be packed in up to three cargo containers first of all.

Next, each cargo container will be transported to one of the 32 populated places, which are represented by yellow square in Attachment 1, to ensure that DroneGo disaster response system can be timely discovered and well operated.

Then both drones and medical packages will be taken out of the containers, and afterwards the latter will be packed into the drone cargo bay that is in fixed connection with the drone.

Finally, all drones will depart from the up to three container locations and fly along the main roads on schedule. As is shown in Attachment 1, these main roads connect 32 populated places and make a road network. It is worth pointing out that there are two possible situations for these drones. If the drone carries a cargo bay with medical packages in it, the drone must fly via at least one of five locations in need of medical assistance, referred to as five medical nodes: Jajardo, San Pablo, San Juan, Bayamon and Arecibo, for the purpose of offloading its cargo. However, if the drone carries no cargo, any route without deviation of the road network is allowed.

Assumption III: To capture the essence of the problem and get rid of unnecessary complexity, some simplifications have been made. We assume that each drone can deliver medical supplies to at most one location demanding assistance. If two drones from the same container have the same types, they also have the same payload packing configuration.

3 Symbols and definitions

Symbol	Definition
X	The number of the containers X.
Y	The number of the containers Y.
Z	The number of the containers Z.
X_α	The number of the drones α packed in the container X.
Y_α	The number of the drones α packed in the container Y.
Z_α	The number of the drones α packed in the container Z.

Symbol	Definition
$X_{\alpha ij}$	The number of the MED i transported by drone α from container X's location to the demand location j, where $\alpha = A, \cdots, H$, $i = 1, 2, 3$, $j = 1, 2, \cdots, 5$.
$Y_{\alpha ij}$	The number of the MED i transported by drone α from container Y's location to the demand location j, where $\alpha = A, \cdots, H$, $i = 1, 2, 3$, $j = 1, 2, \cdots, 5$.
$Z_{\alpha ij}$	The number of the MED i transported by drone α from container Z's location to the demand location j, where $\alpha = A, \cdots, H$, $i = 1, 2, 3$, $j = 1, 2, \cdots, 5$.
c_i	The cost of MED i, where $i = 1, 2, 3$.
C_α	The cost of Drone α, where $\alpha = A, B, \cdots, H$.
W	The cost of the container.
V_α	The volume of the drone α, where $\alpha = A, B, \cdots, H$.
v_i	The volume of the medical package i, where $i = 1, 2, 3$.
P_α	The max payload capability of the drone α, where $\alpha = A, B, \cdots, H$.
d	The supporting days for the medical package demand of all 5 locations.
M	A sufficiently large number for big-M method in integer programming.
λ	A parameter indicating the significance of the video reconnaissance of road networks compared to medical supply delivery.
l_α	The max flight distance of the drone α.
Q	The total flight distance
T	The number of the nodes in the network of roads.
U_{Xm}	$U_{Xm} = 1$ if container X is located at node m, where $m = 1, 2, \cdots, T$. Otherwise $U_{Xm} = 0$.
U_{Ym}	$U_{Ym} = 1$ if container Y is located at node m, where $m = 1, 2, \cdots, T$. Otherwise $U_{Ym} = 0$.
U_{Xm}	$U_{Zm} = 1$ if container Z is located at node m, where $m = 1, 2, \cdots, T$. Otherwise $U_{Zm} = 0$.
$f(m, j)$	The length of the shortest path from node m to node j.

Symbol	Definition
L	The length of the bin
W	The width of the bin
H	The Height of the bin of the bin
x_i	LBB coordinate of item i in x axis
y_i	LBB coordinate of item i in y axis
z_i	LBB coordinate of item i in z axis
s_{ij}	item is in the left side of item or not
u_{ij}	item i is under item j or not
b_{ij}	item i is in the back of item j or not
δ_{i1}	orientation of item i is front-up or not
δ_{i2}	orientation of item i is front-down or not
δ_{i3}	orientation of item i is side-up or not
δ_{i4}	orientation of item i is side-down or not
δ_{i5}	orientation of item i is buttom-up or not
δ_{i6}	orientation of item i is buttom-down or not

4 Integer programming model for single objective

The DroneGo disaster response system is sent to Puerto Rico for two major missons: medical supply delivery and video reconnaissance of road networks, none of which is dispensable. Nevertheless, to start with, focusing on the former solely helps attain a clear insight into this sophisticated problem. That leads to our integer programming model.

4.1 Analysis and assumptions

Now assume that the response system are only for medical supply delivery and that the location of each container is so close to its destination that each drone can reach it before the power supply runs out. According to the requirement, the response system must meet the daily medical package demand of the five selected locations in Puerto Rico. Therefore, a natural objective is to maximize the so-called supporting days d. Suppose that the amount of the medical packages provided by the Drone fleet can serve location j $(j = 1, 2, \cdots, 5)$ for d_j days. Then the supporting days is defined as

$$d = \min\{d_1, d_2, d_3, d_4, d_5\}.$$

For the sake of the optimization of d, a lot of independent variables need to be specified appropriately. Firstly, we need to specify the number of the containers transported to the disaster area. We introduce binary variables X, Y, Z to indicate whether the corresponding cargo container is utilized respectively. Secondly, we have to decide the packing configuration for each container in use. Given the definition of X_α and $X_{\alpha ij}$ in section 3, the number of each item to be packed into container X (8 types of drones: Drone A to Drone H; 3 types of medical packages: MED1, MED2, MED3) can be expressed as

$$X_A, \cdots, X_H, \sum_{\alpha=A}^{H}\sum_{j=1}^{5} X_{\alpha 1j}, \sum_{\alpha=A}^{H}\sum_{j=1}^{5} X_{\alpha 2j}, \sum_{\alpha=A}^{H}\sum_{j=1}^{5} X_{\alpha 3j}$$

respectively. Likewise, we can choose values for $Y_\alpha, Y_{\alpha ij}$ and $Z_\alpha, Z_{\alpha ij}$ to give the packing configuration for the container Y and the container Z respectively. Thirdly, we have to decide the drone payload packing configurations, or alternatively how the medical packages are packed into the drone cargo bay. The number of each type of medical package to be packed into the drone α can be expressed as

$$\sum_{j=1}^{5} X_{\alpha 1j}, \sum_{j=1}^{5} X_{\alpha 2j}, \sum_{j=1}^{5} X_{\alpha 3j}$$

Finally, the destination of each drone should be given. That is exactly what the variables $X_{\alpha ij}, Y_{\alpha ij}, Z_{\alpha ij}$ indicate.

In a word, we need to adjust the 387 variables

$$X, Y, Z, X_\alpha, Y_\alpha, Z_\alpha, X_{\alpha ij}, Y_{\alpha ij}, Z_{\alpha ij}$$

$$(\alpha = A, \cdots, H; i = 1, 2, 3; j = 1, \cdots, 5)$$

for the maximization of d.

4.2 Formalization

By imposing proper constraints we obtain the following integer programming problem

$$\max \quad d$$

$$
\text{s.t.}\begin{cases}
\sum_{i=1}^{3}\sum_{j=1}^{5} v_i X_{\alpha ij} \le 8\times 10\times 14 \ \text{ if } \alpha = A, B, D & (1)\\
\sum_{i=1}^{3}\sum_{j=1}^{5} v_i X_{\alpha ij} \le 24\times 20\times 20 \ \text{ if } \alpha = C, E, F, G & (2)\\
\sum_{i=1}^{3}\sum_{j=1}^{5} v_i Y_{\alpha ij} \le 8\times 10\times 14 \ \text{ if } \alpha = A, B, D & (3)\\
\sum_{i=1}^{3}\sum_{j=1}^{5} v_i Y_{\alpha ij} \le 24\times 20\times 20 \ \text{ if } \alpha = C, E, F, G & (4)\\
\sum_{i=1}^{3}\sum_{j=1}^{5} v_i Z_{\alpha ij} \le 8\times 10\times 14 \ \text{ if } \alpha = A, B, D & (5)\\
\sum_{i=1}^{3}\sum_{j=1}^{5} v_i Z_{\alpha ij} \le 24\times 20\times 20 \ \text{ if } \alpha = C, E, F, G & (6)\\
\sum_{\alpha=A}^{H} V_\alpha X_\alpha + \sum_{\alpha=A}^{H}\sum_{i=1}^{3}\sum_{j=1}^{5} v_i X_{\alpha ij} \le 231\times 92\times 94 & (7)\\
\sum_{\alpha=A}^{H} V_\alpha Y_\alpha + \sum_{\alpha=A}^{H}\sum_{i=1}^{3}\sum_{j=1}^{5} v_i Y_{\alpha ij} \le 231\times 92\times 94 & (8)\\
\sum_{\alpha=A}^{H} V_\alpha Z_\alpha + \sum_{\alpha=A}^{H}\sum_{i=1}^{3}\sum_{j=1}^{5} v_i Z_{\alpha ij} \le 231\times 92\times 94 & (9)\\
\sum_{i=1}^{3}\sum_{j=1}^{5} X_{\alpha ij} \le M X_\alpha \ \text{ for } \alpha = A, B, \cdots, G & (10)\\
\sum_{i=1}^{3}\sum_{j=1}^{5} Y_{\alpha ij} \le M Y_\alpha \ \text{ for } \alpha = A, B, \cdots, G & (11)\\
\sum_{i=1}^{3}\sum_{j=1}^{5} Z_{\alpha ij} \le M Z_\alpha \ \text{ for } \alpha = A, B, \cdots, G & (12)\\
\sum_{\alpha=A}^{H} X_\alpha \le M X & (13)\\
\sum_{\alpha=A}^{H} Y_\alpha \le M Y & (14)\\
\sum_{\alpha=A}^{H} Z_\alpha \le M Z & (15)\\
X \le 1 & (16)\\
Y \le 1 & (17)\\
Z \le 1 & (18)\\
\sum_{j=1}^{5} (2X_{\alpha 1j} + 2X_{\alpha 2j} + 3X_{\alpha 3j}) \le P_\alpha \text{ for } \alpha = A, B, \cdots, G & (19)\\
\sum_{j=1}^{5} (2Y_{\alpha 1j} + 2Y_{\alpha 2j} + 3Y_{\alpha 3j}) \le P_\alpha \text{ for } \alpha = A, B, \cdots, G & (20)\\
\sum_{j=1}^{5} (2Z_{\alpha 1j} + 2Z_{\alpha 2j} + 3Z_{\alpha 3j}) \le P_\alpha \text{ for } \alpha = A, B, \cdots, G & (21)
\end{cases}
$$

$$
\text{s.t.}\begin{cases}
\sum_{\alpha=A}^{H}(X_{\alpha11}+Y_{\alpha11}+Z_{\alpha11})\geq d & (22)\\
\sum_{\alpha=A}^{H}(X_{\alpha31}+Y_{\alpha31}+Z_{\alpha31})\geq d & (23)\\
\sum_{\alpha=A}^{H}(X_{\alpha12}+Y_{\alpha12}+Z_{\alpha12})\geq 2d & (24)\\
\sum_{\alpha=A}^{H}(X_{\alpha32}+Y_{\alpha32}+Z_{\alpha32})\geq d & (25)\\
\sum_{\alpha=A}^{H}(X_{\alpha13}+Y_{\alpha13}+Z_{\alpha13})\geq d & (26)\\
\sum_{\alpha=A}^{H}(X_{\alpha23}+Y_{\alpha23}+Z_{\alpha23})\geq d & (27)\\
\sum_{\alpha=A}^{H}(X_{\alpha14}+Y_{\alpha14}+Z_{\alpha14})\geq 2d & (28)\\
\sum_{\alpha=A}^{H}(X_{\alpha24}+Y_{\alpha24}+Z_{\alpha24})\geq d & (29)\\
\sum_{\alpha=A}^{H}(X_{\alpha34}+Y_{\alpha34}+Z_{\alpha34})\geq 2d & (30)\\
\sum_{\alpha=A}^{H}(X_{\alpha15}+Y_{\alpha15}+Z_{\alpha15})\geq d & (31)\\
X,Y,Z,X_\alpha,Y_\alpha,Z_\alpha,X_{\alpha ij},Y_{\alpha ij},Z_{\alpha ij}\text{ are nonnegative integers.} & (32)
\end{cases}
$$

4.3 Interpretation

Then we will explicitly explain the implications of all the constrains. Constrains (1)-(6) which characterize the volume inequalities are necessary but not sufficient conditions for medical packages to be totally packed into the drone cargo bays. We dispose of the bin packing problem for the time being, since the calculation can be considerably simplified in this way. Similarly, Constrains (7)-(9) are necessary but not sufficient conditions for drones and medical packages to be packed into the containers successfully. Since M is a sufficiently large positive number, constrains (10)-(12) will be activated if only if X_α, Y_α or Z_α equal 0. Hence it is impossible for unused drone cargo bays to be loaded any goods in. Likewise, Constrains (13)-(15) can avoid packing items in a unused container. Constrains (16)-(18) indicates that X, Y and Z are binary variables. Constraints (19)-(21) exclude the case that the weight of medical packages exceeds the max payload capability of the drone. Constrains (22)-(31) assures that the medical package demand can be satisfied for the first d days at least. Constrain (32) is just a standard condition in any integer programing problem.

4.4 Variation

If we take the cost of containers, drones and medical packages into consideration and fix the value of d, minimizing the total cost also makes sense. To address the minimization problem, we only need to transform the objective function into the following form while all original constraints (1)-(32) still hold:

$$\min \quad \sum_{\alpha=A}^{H}\sum_{i=1}^{3}\sum_{j=1}^{5} c_i\left(X_{\alpha ij}+Y_{\alpha ij}+Z_{\alpha ij}\right)+\sum_{\alpha=A}^{H} C_\alpha\left(X_\alpha+Y_\alpha+Z_\alpha\right)+W(X+Y+Z).$$

5 Integrated model for double objectives

Now it is time to extend our analysis to the complete case that both medical supply delivery and video reconnaissance of road networks need to be carried out.

5.1 Road Network Detection

To decide the best locations of cargo containers and make a drone flight plan to realize video reconnaissance of road networks, we are supposed to add road network information to the model.

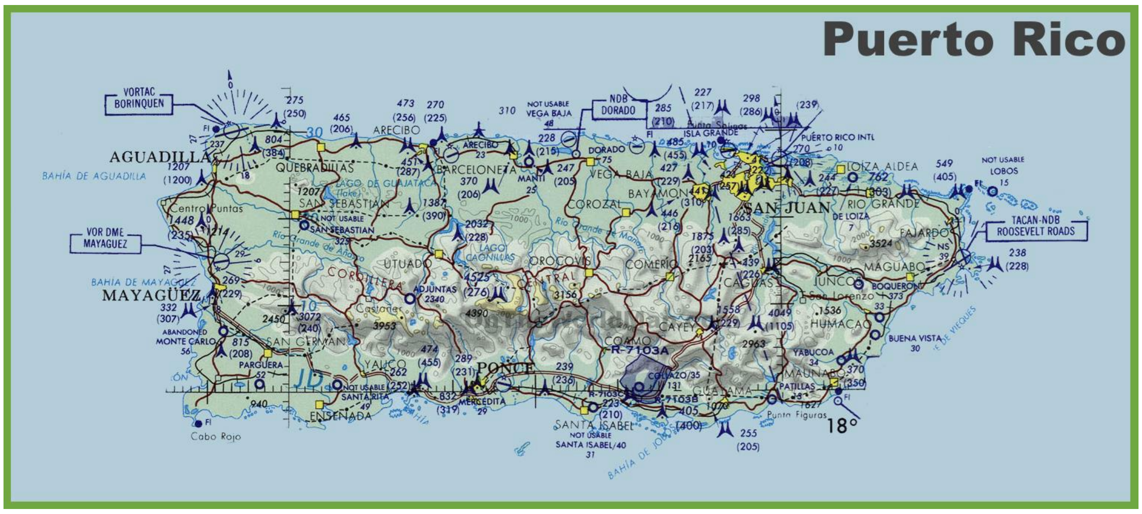

Figure 1: Map of Puerto Rico

Based on the following map (Figure 1) provided in Attachments, we use AutoCAD to extract the regional road network of Puerto Rico. The results of road network extraction are shown in Figure 2.

For simplification, we use matrix to represent the topological structure of road network. As shown in the figure 2 below, a simple road network can be represented as a four-dimensional matrix.

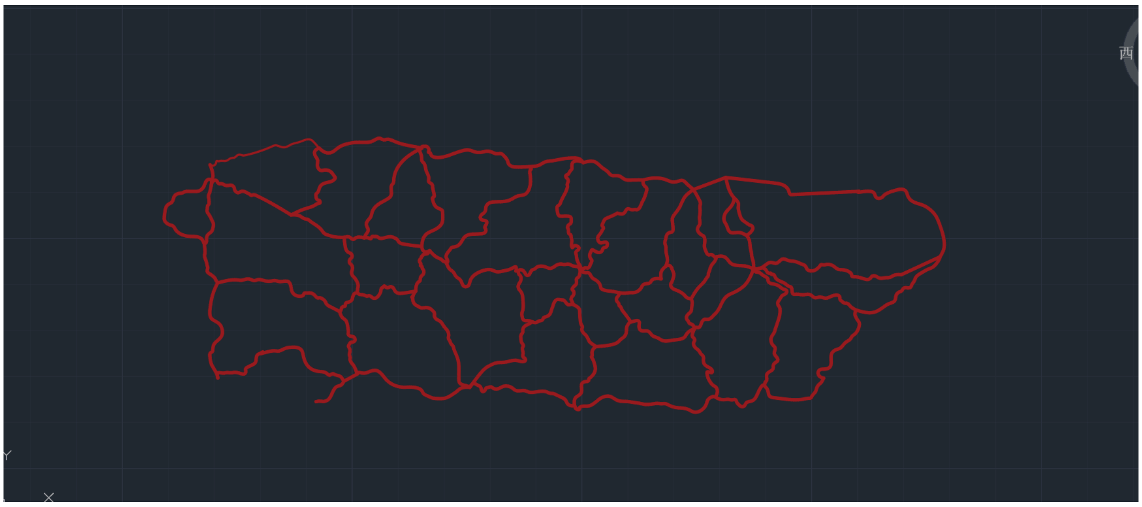

Figure 2: Road Network

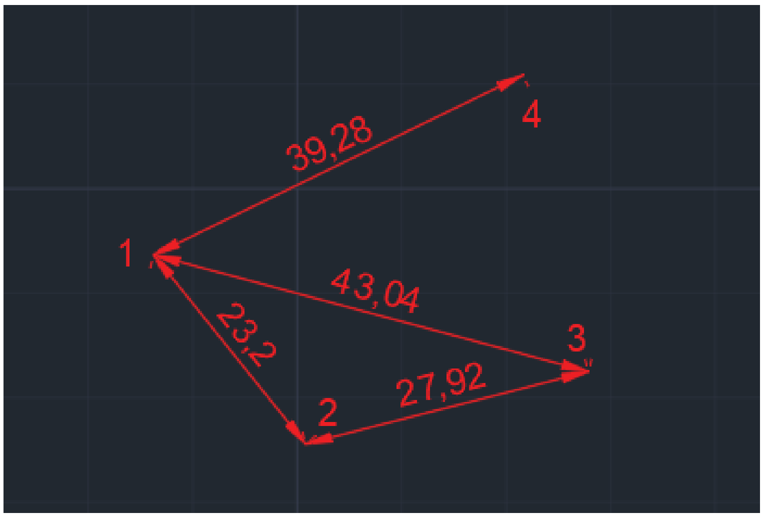

Figure 3: Simple Road Network

$$\begin{pmatrix} 0 & 23.2 & 43.04 & 39.28 \\ 23.2 & 0 & 27.92 & -1 \\ 43.04 & 27.92 & 0 & -1 \\ 39.28 & -1 & -1 & 0 \end{pmatrix}$$

Similarly, the actual road map shown below, can be expressed as a 32-dimensional matrix. Note that nodes numbered from 1 to 5 in the figure 4 represent hospitals, we just call them medical nodes. And other nodes are mainly populated residential areas, which can be named as normal nodes.

Then we implement the Dijkstra's algorithm to solve for the shortest path $f(m, j)$ as a preparation for later drone route planning.

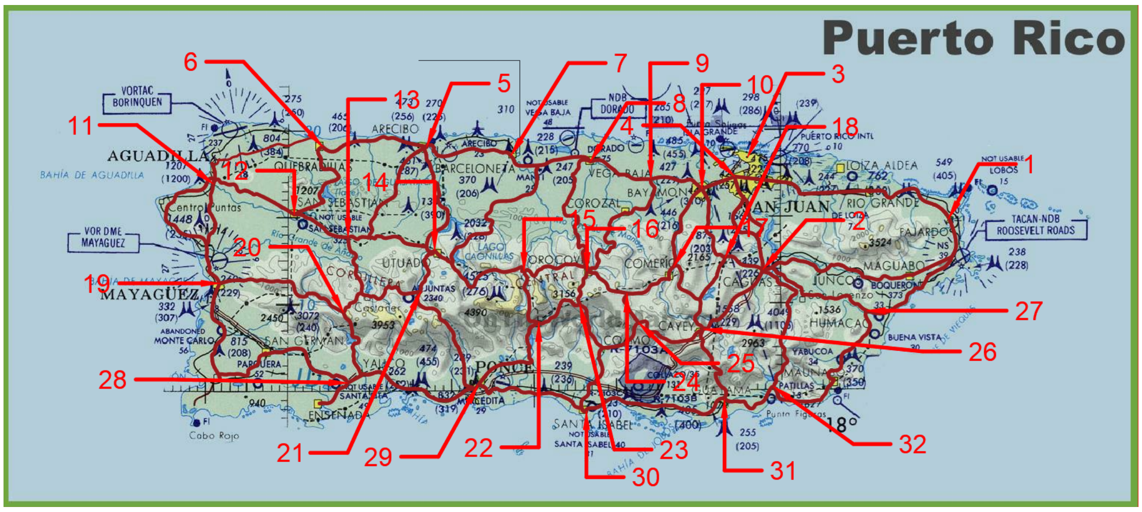

Figure 4: Actual Road Network

5.2 Analysis and assumptions

Since the variable d named supporting days has been shown to be a reasonable measure of the effect of medical supply delivery, developing some appropriate measures of the effect of video reconnaissance is our priority.

Assume that drones cannot be recharged, which implies each drone will fly along the road network until its batter is discharged. Another assumption is the max flight distance of the drone is unrelated to the weight of its carrying cargo. When the number of drones is rather large, it is totally reasonable to assume that under no circumstances will any two drones fly abreast along a same road. Thus the sum of flight distances of all the drones can well measure the effect of video reconnaissance. Note that Drone F is not capable of recording video. We can deduce the total flight distance

$$Q = \sum_{\substack{\alpha=A \\ \alpha\neq F}}^{G} l_\alpha (X_\alpha + Y_\alpha + Z_\alpha).$$

There are several methods of multiobjective optimization. Here we choose a simple linear weighing method in order to obtain a integer programming model. Let λ be the weight indicating the significance of the video reconnaissance of road networks compared to medical supply delivery. Then we need to manipulate variables to maximize the weighted measure

$$d + \lambda L,$$

which can be interpreted as the total effect or total utility. Besides the 9 variables

$$X, Y, Z, X_\alpha, Y_\alpha, Z_\alpha, X_{\alpha ij}, Y_{\alpha ij}, Z_{\alpha ij}$$

considered in the integer programming model for single objective, more variables are required to describe the location of the container X, Y, and Z, or more specifically, to describe which node every container should be sent to given the network of roads.

Therefore, we resort to binary variables U_{Xm}, U_{Ym}, U_{Zm} to indicate whether each of containers X, Y, Z is located at node m respectively. To sum up, we are to specify the following 483 variables

$$X, Y, Z, X_\alpha, Y_\alpha, Z_\alpha, X_{\alpha ij}, Y_{\alpha ij}, Z_{\alpha ij}, U_{Xm}, U_{Ym}, U_{Zm}$$
$$(\alpha = A, \cdots, H; i = 1, 2, 3; j = 1, \cdots, 5; m = 1, \cdots, 32)$$

to maximize $d + \lambda L$.

Note that there is no need to introduce any other variables to characterize the flight schedule of the Drone fleet. Instead, as a variety of flight paths lead to the same total flight distance, flight schedule can be specified by simple rules. If a drone carries the medical packages to be transported to node i, it will first leave for node i along the shortest path and then fly along an arbitrary major road. If a drone carries no medical packages, it can choose its path from all main roads freely.

5.3 Formalization

The objective function of this model is shown as follows:

$$\max \quad d + \lambda \sum_{\substack{\alpha=A \\ \alpha \neq F}}^{G} l_\alpha (X_\alpha + Y_\alpha + Z_\alpha)$$

In additional to the constraints (1)-(32) introduced in section 4, new constraints (33)-(39) must hold in the maximization problem.

$$\text{s.t.} \begin{cases} \sum_{m=1}^{T} U_{Xm} = X & (33) \\ \sum_{m=1}^{T} U_{Ym} = Y & (34) \\ \sum_{m=1}^{T} U_{Zm} = Z & (35) \\ \sum_{i=1}^{3} X_{\alpha ij} \leq M(1 - U_{Xm}) \quad \text{if } l_\alpha < f(m, j), & \\ \qquad \text{for } \alpha = A, \cdots, G, m = 1, \cdots, T, j = 1, \cdots, 5 & (36) \\ \sum_{i=1}^{3} Y_{\alpha ij} \leq M(1 - U_{Ym}) \quad \text{if } l_\alpha < f(m, j), & \\ \qquad \text{for } \alpha = A, \cdots, G, m = 1, \cdots, T, j = 1, \cdots, 5 & (37) \\ \sum_{i=1}^{3} Z_{\alpha ij} \leq M(1 - U_{Zm}) \quad \text{if } l_\alpha < f(m, j), & \\ \qquad \text{for } \alpha = A, \cdots, G, m = 1, \cdots, T, j = 1, \cdots, 5 & (38) \\ U_{Xm}, U_{Ym}, U_{Zm} \in \{0, 1\}, \ m = 1, \cdots, T & (39) \end{cases}$$

5.4 Interpretation

What is left to interpret is the implication of the constrains (33)-(39). Constrain (33) ensures that the variables U_{Xm} are consistent with X. In other words, if the container X is utilized, it has to be transported to exact one node of the network. In contrast, if the container X is not in use, it is expected to be placed nowhere. Constrains (34)-(35) are saying the same thing.

The constrains (36)-(38) is designed to avoid such cases where some drones carrying medical packages run out of power before reaching their destinations. If the longest flight distance of a drone l_α is still less than the shortest distance from its own location to the target medical node, it is not allowed to deliver any cargo at all.

The constraint (39) is trivial.

5.5 Variation

Just by replacing the objective function yields the corresponding cost minimization problem

$$\begin{aligned}\min \quad & \sum_{\alpha=A}^{H}\sum_{i=1}^{3}\sum_{j=1}^{5} c_i\left(X_{\alpha ij}+Y_{\alpha ij}+Z_{\alpha ij}\right)+\sum_{\alpha=A}^{H} C_\alpha\left(X_\alpha+Y_\alpha+Z_\alpha\right)+W(X+Y+Z)\\ & -\lambda\sum_{\substack{\alpha=A\\ \alpha\neq F}}^{G} l_\alpha(X_\alpha+Y_\alpha+Z_\alpha),\end{aligned}$$

while all constraints (1)-(39) must hold.

6 Modified model based on bin-packing algorithm

In this section we are to resolve a major defect in our previous models. We have been supposing that various items can be packed into a bin if the total volume of all items is not greater than the interior volume of the bin. However, 3D bin-packing simulation software illustrates that in most cases cuboid goods can only occupy 70%-90% of the interior space of the bin. Thus our modified model based on advanced 3D bin-packing algorithm is brought forward to reduce the inaccuracy caused by the rough volume constraints in integer programming.

6.1 Unpractical integer programming bin-packing algorithm

It seems to be a natural choice of insisting on the integer programming method to dealing with the bin-packing problem as before. And such method exists indeed.

In a typical 3D bin-packing problem, a set of items must be packaged in a designated box in a way that minimizes the number of boxes used. Our goal is to find a box with the smallest surface area that can hold all the items. We define (x_i, y_i, z_i) as the left-bottom-back (LBB) coordinate of item i and define $(0, 0, 0)$ as the left-bottom-back coordinate of the bin. Based on the description of problems and symbols, the mathematical formulas of the new three-dimensional bin-packing problem are given as follows:

$$\min L$$

$$\text{s.t.} \begin{cases} s_{ij} + u_{ij} + b_{ij} = 1 & (40) \\ \delta_{i1} + \delta_{i2} + \delta_{i3} + \delta_{i4} + \delta_{i5} + \delta_{i6} = 1 & (41) \\ x_i - x_j + L \cdot s_{ij} \le L - \hat{l}_i & (42) \\ y_i - y_j + W \cdot u_{ij} \le W - \hat{w}_i & (43) \\ z_i - z_j + H \cdot b_{ij} \le H - \hat{h}_i & (44) \\ 0 \le x_i \le L - \hat{l}_i & (45) \\ 0 \le y_i \le W - \hat{w}_i & (46) \\ 0 \le z_i \le H - \hat{h}_i & (47) \\ \hat{l}_i = \delta_{i1} l_i + \delta_{i2} l_i + \delta_{i3} l_i + \delta_{i4} w_i + \delta_{i5} l_i + \delta_{i6} h_i & (48) \\ \hat{w}_i = \delta_{i1} w_i + \delta_{i2} h_i + \delta_{i3} l_i + \delta_{i4} l_i + \delta_{i5} l_i + \delta_{i6} l_i & (49) \\ \hat{h}_i = \delta_{i1} h_i + \delta_{i2} w_i + \delta_{i3} h_i + \delta_{i4} l_i + \delta_{i5} w_i + \delta_{i6} l_i & (50) \\ s_{ij}, u_{ij}, b_{ij} \in \{0, 1\} & (51) \\ \delta_{i1}, \delta_{i2}, \delta_{i3}, \delta_{i4}, \delta_{i5}, \delta_{i6} \in \{0, 1\} & (52) \end{cases}$$

Constraints (48), (49), (50) denote the length, width and height of Item i after orientation. Constraints (40), (41), (42), (43) are used to ensure that there is no overlap between the two packages, while constraints (45), (46), (47) are used to ensure that the goods will not be placed outside the warehouse.

Unfortuanately it is very difficult to solve in reasonable time limit and this problem has beeb proved to be NP-hard. Thus we need to develop a more effective to solve the bin-packing problem.

6.2 Effective Simulated annealing bin-packing algorithm

6.2.1 Background

Simulated annealing arithmetic was formed in the early 1980s. Its idea originated from the annealing process of solids, heating solids to a sufficiently high temperature and then cooling them slowly. When the temperature was raised, the internal energy of particles in solids became disordered and increased, while when the temperature was slowly cooled, the particles gradually became orderly. In theory, if the cooling process was slow enough, then either of them would be cooled. Solids can achieve thermal equilibrium at temperature, and when cooled to low temperature, they will reach the minimum state of internal energy at this low temperature.

In this process, it is an important step to achieve thermal equilibrium at any constant temperature. When applying simulated annealing to optimization problems, temperature T can generally be regarded as control parameter, objective function value f as internal energy E, and a state of solid at a certain temperature T corresponds to a solution x. Then the algorithm tries to reduce the objective function value f (internal energy E) gradually with the decrease of the control parameter T until it reaches the global minimum (the lowest energy state in low temperature annealing), just like the solid annealing process.

6.2.2 Model Assumptions

- The dimensions of the goods are integers.
- The goods can only be placed along the coordinate axes of the container, that is, parallel to the edges of the container.
- The coordinate system is a three-dimensional Cartesian coordinate system. The length, width and height of the container correspond to the X, Y and Z axes of the Cartesian coordinate system respectively.
- (x, y, z) is the coordinate of the lower-left corner of the front side of the container.
- In the process of loading, the goods put into the container can move downward, forward and left until its bottom, front and left are adjacent to other goods or containers.
- The goods arrive at the same destination.
- The extrusion deformation of goods can be neglected.

6.2.3 Simulated annealing Algorithm

- Set the initial annealing temperature T_0, and generate an initial solution x_0 at random and calculate the corresponding objective function $E(X_0)$.
- Make T equal to the next value in the cooling schedule T_i.
- Pick a random neighbour of the current solution x_i, generate a new solution x_j, calculate the corresponding objective function value $E(x_j)$, and get $\Delta E = E(x_j) - E(x_i)$.
- If $\Delta E > 0$, the new solution x_j is accepted as the new current solution; otherwise, x_j is accepted with a probability of $exp(-\Delta E/T_i)$.

- Repeat steps 3 and 4 L_k times at temperature T_i.
- Transfer the algorithm to step 2 till $T = T_f$.

The algorithm is essentially divided into two layers of cycles, generating new solutions at any temperature random disturbance, and calculating the change of the objective function value to determine whether it is accepted or not. Because the initial temperature of the algorithm is relatively high, the new solution of increasing E may also be accepted at the beginning, so it can jump out of the local minimum, and then by slowly lowering the temperature, the algorithm may eventually converge to the global optimal solution. It is also pointed out that although the acceptance function is very small at low temperature, the possibility of accepting worse solutions is still not excluded. Therefore, the best feasible solution (historical optimal solution) encountered in annealing process is generally recorded and output together with the last accepted solution before terminating the algorithm.

6.3 Modified model simulated annealing algorithm

In this section we will modify our normal integer programming model by combining simulated annealing algorithm. Consider a general integer programming problem.

$$\max Z = \sum_{j=1}^{n} c_j x_j$$

s.t.

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \text{ for } i = 1, 2, \cdots, m$$

and

$$x_j \geq 0, \quad \text{for } j = 1, 2, \cdots, n$$
$$x_j \text{ is integer}, \quad \text{for } j = 1, 2, \cdots, n.$$

A common and efficient algorithm for integer programming is branch-and-bound method, which can be summarized as follows:

0. Initialization: Set Z^*. Apply the bounding step, fathoming step, and optimality test described below to the whole problem. If not fathomed, classify this problem as the one remaining subproblem for performing the first full iteration below. Steps for each iteration:

1. Branching: Among the remaining (unfathomed) subproblems, select the one that was created most recently. (Break ties according to which has the larger bound.) Among the integer-restricted variables that have a noninteger value in the optimal solution for the LP relaxation of the subproblem, choose the first one in the natural ordering of the variables to be the branching variable. Let x_j be this variable and x_j^* its value in this solution. Branch from the node for the subproblem to create two new subproblems by adding the respective constraints $x_j \leq [x_j^*]$ and $x_j \geq [x_j^*] + 1$.

2. Bounding: For each new subproblem, obtain its bound by applying the simplex method (or the dual simplex method when reoptimizing) to its LP relaxation and using the value of Z for the resulting optimal solution.

3. Fathoming: For each new subproblem, apply the three fathoming tests given below, and discard those subproblems that are fathomed by any of the tests.
   Test 1: Its bound Z^*, where Z^* is the value of Z for the current incumbent.
   Test 2: Its LP relaxation has no feasible solutions.
   Test 3: The optimal solution for its LP relaxation has integer values for the integerrestricted variables. (If this solution is better than the incumbent, it becomes the new incumbent and test 1 is reapplied to all unfathomed subproblems with the new larger Z^*.)
   Optimality test: Stop when there are no remaining subproblems that are not fathomed; the current incumbent is optimal. Otherwise, perform another iteration.

Now we change the traditional fathoming step to yield a more accurate solution. For given X_α and $X_{\alpha ij}$, we can use the simulated annealing bin-packing algorithm to decide whether it is a possible packing configuration. Thus the original test 2 should be extended. Even though its LP relaxation has feasible solutions, they can be specified as an impossible packing configuration by bin-packing algorithm, which implies fathoming should be conducted.

After the simulated annealing bin-packing algorithm is involved, we find the optimal solution in normal integer programming was overwhelmingly large than the actual situation.

7 Results and sensitivity analysis

The result of the modified model depends on the parameter λ which indicates the significance of the video reconnaissance of road networks compared to medical supply delivery.

- $\lambda = 1$

Cargo Containers	MED1	MED2	MED3	A	B	C	D	E	F	G	H
X	606			2	9	7	10	8	2	9	1
Y	1818	1212	1212	1	4	3	20	20	5	15	1
Z	1818		1212	2	8	5	8	5	2	7	1

- $\lambda = 2$

Cargo Containers	MED1	MED2	MED3	A	B	C	D	E	F	G	H
X	569			2	8	5	9	9	3	7	2
Y	1680	811	792	1	3	4	12	18	10	13	1
Z	1918		1173	1	7	6	8	4	4	4	1

- $\lambda = 5$

Cargo Containers	MED1	MED2	MED3	A	B	C	D	E	F	G	H
X	269	223		4	9	12	13	3	6	19	1
Y	1080		92	5	14	1	23	10	5	18	6
Z	918		1173	2	12	7	8	15	4	4	3

The variations of the result is not considerable if λ changes a little. As we can expect, the demanding medical packages are reducing when λ rises.

8 Strengths and weaknesses

8.1 Strengths

- **Overall planning**
  Our planning treats the DroneGo disaster response system as a unit, as an organic whole. All variables that interacts with each other are decided simultaneously in a complete and comprehensive framework.
- **Universality**
  The model we have put forward not only works well for developing an aerial disaster relief response system, but also fits a number of analogous scenarios.

8.2 Weaknesses

- We assume that each drone can deliver medical supplies to at most one location demanding assistance and that if two drones from the same container have the same types, they also have the same payload packing configuration. By relaxing these limitations, it is possible to design an intricate flight schedule to reach a more optimized solution.

Memo

To : Chief Operating Officer
From : Team 1911507
Date : 29 January, 2019
Subject : Suggestions for DroneGo Disaster Response System

We are writting to report our main modeling results and suggestions for DroneGo disaster response system.

Modeling Results :

- The model results are shown in the following table.

Cargo Containers	MED1	MED2	MED3	A	B	C	D	E	F	G	H
X	606			2	9	7	10	8	2	9	1
Y	1818	1212	1212	1	4	3	20	20	5	15	1
Z	1818		1212	2	8	5	8	5	2	7	1

- The best locations on Puerto Rico to position these cargo containers are shown in the following figure.

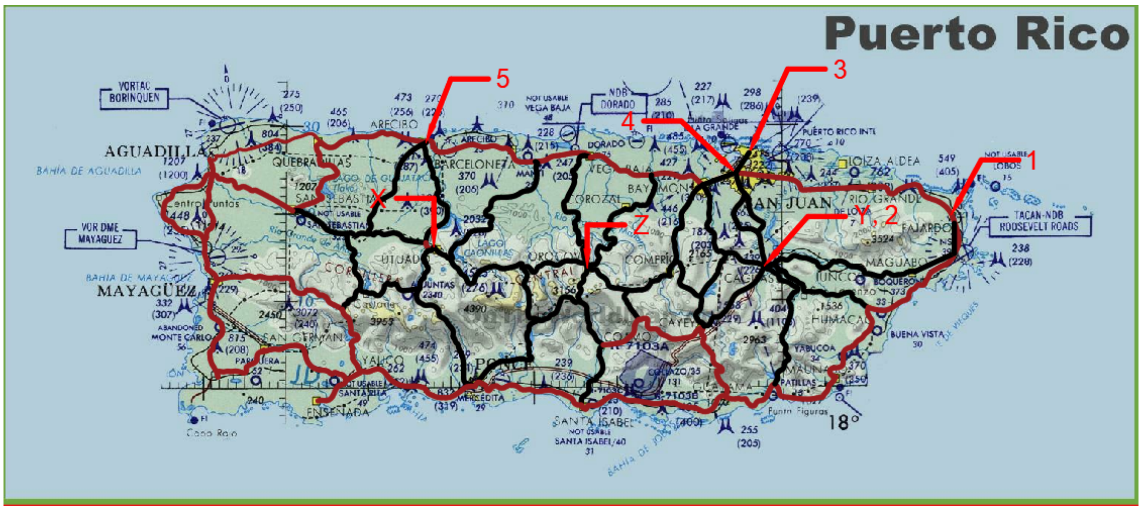

Figure 5: Locations

- Based on the above results, DraneGo may conduct medical supply for 606 days, and realize video reconnaissance of road networks which are marked black in the above figure.

Recommendations :

- Containers in different locations are faced with different demands, and the corresponding packing problem needs to be considered separately.
- The suitable locations to position cargo containers should be as dispersed as possible on the premise of meeting medical supply.
- If the drone is rechargeable, a reasonable flight plan may make it more efficient.

We hope that our suggestions are helpful. Please contact us if you have any questions.

References

[1] Maarouf W F , Barbar A M , Owayjan M J . A New Heuristic Algorithm for the 3D Bin Packing Problem[M]

[2] Hu H, Zhang X, Yan X, et al. Solving a New 3D Bin Packing Problem with Deep Reinforcement Learning Method[J]. 2017.

[3] https://en.wikipedia.org/wiki/Simulated_annealing

[4] https://en.wikipedia.org/wiki/Bin_packing_problem

[5] Cruz-Reyes L, Nieto-YÃąÃśez D M, TomÃąs-Solis P, et al. Solving Bin Packing Problem with a Hybridization of Hard Computing and Soft Computing[M]

[6] Fu Z D , University X , Xiamen, et al. A Hybrid Simulated Annealing Algorithm for the Three-Dimensional Packing Problem[J]. Chinese Journal of Computers, 2009, 32(11):2147-2156.

[7] Faina L . A global optimization algorithm for the three-dimensional packing problem[J]. European Journal of Operational Research, 2000, 126(2):340-354.

Appendices

Appendix A Simulated annealing algorithm

Here are programmes we used in our model as follow.

Matlab code

```
function L_min = Packing_ANN(l,w,h)


a=0.95;
num=size(l,1);
E_current = inf;
E_best = inf;
E_new = inf;
xi_new=ceil(rand(1,num)*924);
yi_new=ceil(rand(1,num)*92);
zi_new=ceil(rand(1,num)*94);
mi_new=ceil(rand(1,num)*6);
si_new=ceil(rand(num,num)+0.5)-1;
ui_new=zeros(num,num);
bi_new=zeros(num,num);
for i=1:num
    for j=1:num
```

```
        if si_new(i,j)==1
            ui_new(i,j)=0;
            bi_new(i,j)=0;
        else
            k=ceil(rand+0.5)-1;
            if k==1
                ui_new(i,j)=1;
                bi_new(i,j)=0;
            else
                ui_new(i,j)=0;
                bi_new(i,j)=1;
            end
        end
    end
end
xi_cur=xi_new;
yi_cur=yi_new;
zi_cur=zi_new;
mi_cur=mi_new;
si_cur=si_new;
ui_cur=ui_new;
bi_cur=bi_new;

xi_best=xi_new;
yi_best=yi_new;
zi_best=zi_new;
mi_best=mi_new;
si_best=si_new;
ui_best=ui_new;
bi_best=bi_new;
to=100;
tf=3;
t=to;




while  t>=tf
    for r = 1:10000
        six=ceil(rand*6);
        tmp=ceil(rand*num);
        switch six
            case 1
                if(E_best<924)
                    xi_new(tmp)=ceil(rand*924);
                else
                    xi_new(tmp)=ceil(rand*E_best);
                end
            case 2
                yi_new(tmp)=ceil(rand*92);
            case 3
                zi_new(tmp)=ceil(rand*94);
            case 4
                mi_new(tmp)=ceil(rand*6);
            case 5
                tmp2=ceil(rand*num);
                while tmp2==tmp
                    tmp2=ceil(rand*num);
                end
                si_new(tmp,tmp2)=ceil(rand+0.5)-1;
                if si_new(tmp,tmp2)==1
                    ui_new(tmp,tmp2)=0;
                    bi_new(tmp,tmp2)=0;
                else
                    bi_new(tmp,tmp2)=1-ui_new(tmp,tmp2);
                end
            case 6
                tmp2=ceil(rand*num);
                while tmp2==tmp
                    tmp2=ceil(rand*num);
                end
                ui_new(tmp,tmp2)=ceil(rand+0.5)-1;
                if ui_new(tmp,tmp2)==1
                    si_new(tmp,tmp2)=0;
                    bi_new(tmp,tmp2)=0;
                else
                    bi_new(tmp,tmp2)=1-si_new(tmp,tmp2);
                end
        end
        [E_new,bool]=packing_st(xi_new,yi_new,zi_new,mi_new,si_new,ui_new,bi_new,l,w,h)
        if bool==1
           if E_new<E_current
               E_current = E_new;
               switch six
                   case 1
                       xi_cur=xi_new;
                   case 2
                       yi_cur=yi_new;
                   case 3
                       zi_cur=zi_new;
                   case 4
                       mi_cur=mi_new;
                   case {5,6}
                       si_cur=si_new;
                       ui_cur=ui_new;
                       bi_cur=bi_new;
               end
               if E_new<E_best
                   E_best=E_new;
                   switch six
                       case 1
                           xi_best=xi_new;
                       case 2
                           yi_best=yi_new;
                       case 3
                           zi_best=zi_new;
                       case 4
                           mi_best=mi_new;
                       case {5,6}
                           si_best=si_new;
                           ui_best=ui_new;
                           bi_best=bi_new;
                   end
               end
           else
               if rand<exp(-(E_new-E_current)/t)
                   E_current=E_new;
               switch six
                   case 1
                       xi_cur=xi_new;
                   case 2
                       yi_cur=yi_new;
                   case 3
                       zi_cur=zi_new;
                   case 4
                       mi_cur=mi_new;
                   case {5,6}
                       si_cur=si_new;
                       ui_cur=ui_new;
                       bi_cur=bi_new;
               end
               else
               switch six
                   case 1
                       xi_new=xi_cur;
                   case 2
                       yi_new=yi_cur;
                   case 3
                       zi_new=zi_cur;
                   case 4
                       mi_new=mi_cur;
                   case {5,6}
                       si_new=si_cur;
                       ui_new=ui_cur;
                       bi_new=bi_cur;
               end
               end
           end
        end
    end
    t = t * a;
end
L_min=E_best;
disp('ŒîÓÅ¡âÎł:');
E_best
disp('Ã£ÿöžÐŒÓţÄŒøśêijřřÚůÅů¡ÏòÎł:');
for i=1:num
    fprintf('\nţÚ%dÿöžÐŒÓţÄÎżÖÃŒøśêžÍÌåżýijřů¡ÏòÎł:',i);
    tmp=[xi(i) yi(i) zi(i) l(i) w(i) h(i) mi(i)]
end

end
```

```
function [E_best,bool]=packing_st(xi,yi,zi,mi,si,ui,bi,l,w,h)
bool=0;
num=size(si,1);
l_=zeros(1,num);
w_=zeros(1,num);
```

```
h_=zeros(1,num);
E_best=0;
    for i=1:num
        switch mi(i)
            case 1
                l_(i)=l(i);
                w_(i)=w(i);
                h_(i)=h(i);
            case 2
                l_(i)=l(i);
                w_(i)=h(i);
                h_(i)=w(i);
            case 3
                l_(i)=w(i);
                w_(i)=l(i);
                h_(i)=h(i);
            case 4
                l_(i)=w(i);
                w_(i)=h(i);
                h_(i)=l(i);
            case 5
                l_(i)=h(i);
                w_(i)=l(i);
                h_(i)=w(i);
            case 6
                l_(i)=h(i);
                w_(i)=w(i);
                h_(i)=l(i);
            otherwise
                bool=0;
                E_best=inf;
                return;
        end
        if yi(i)+w_(i)>92
            bool=0;
            E_best=inf;
            return;
        end
        if zi(i)+h_(i)>94
            bool=0;
            E_best=inf;
            return;
        end
        if E_best<xi(i)+l_(i)
            E_best=xi(i)+l_(i);
        end
    end
    for i=1:num
        for j=1:num
            if j~=i
               if  92*(1-ui(i,j))<(yi(i)-yi(j)+w_(i))
                    bool=0;
                    E_best=inf;
                    return;
               end
               if 94*(1-bi(i,j))<(zi(i)-zi(j)+h_(i))
```

```
                        bool=0;
                        E_best=inf;
                        return;
                    end
                    if si(i,j)==1
                        if xi(i)>xi(j)-l_(i)
                            bool=0;
                            E_best=inf;
                            return;
                        end
                    else
                        if E_best<(xi(i)-xi(j)+l_(i))
                            E_best=(xi(i)-xi(j)+l_(i));
                        end
                    end
                end
            end
    end
bool=1;
end
```

```
l=[75 60 45 40 32 30 27 25 14 12 8];
w=[65 50 45 40 32 30 25 25 7 7 5];
h=[41 30 25 25 17 22 20 20 5 4 5];
L_min= Packing_ANN(l,w,h);
```

Appendix B Dijkstra's shortest path algorithm

some more text **C++ code:**

```
#include<iostream>
#include"Graph.h"
#include"Graphmatrix.h"
#include <fstream>
#include <regex>
#include <string>
#include <vector>
#include <iomanip>
using namespace std;

template<typename Tv, typename Te>
void DisplayMatrix(GraphMatrix<Tv, Te>* graph)
{
    cout << "\t";
    for (int i = 0; i < graph->Graph<Tv, Te>::n; i++)
    {
        cout << graph->vertex(i) << "\t";
    }
    cout << endl;
    for (int i = 0; i < graph->Graph<Tv, Te>::n; i++)
    {
        cout << graph->vertex(i) << "\t";
        for (int j = 0; j < graph->Graph<Tv, Te>::n; j++)
        {
            if (graph->exists(i, j))
            {
                cout << graph->weight(i, j);
            }
            else
            {
                cout << 0;
            }
            cout << "\t";
        }
        cout << endl;
    }
}

int main()
{
    vector<double> temp_line;
    vector<vector<double>> Vec_Dti;
    string line;
    ifstream in("Network.txt");
    regex pat_regex("[[:digit:]][[:digit:]][[:punct:]][[:digit:]]");
    while (getline(in, line))
    {
        for (sregex_iterator it(line.begin(), line.end(), pat_regex), end_it; it != end_
        {
            cout << it->str() << " ";
            temp_line.push_back(stod(it->str()));
        }
        cout << endl;
        Vec_Dti.push_back(temp_line);
        temp_line.clear();
    }
    cout << endl << endl;
    GraphMatrix<int, double> Network;
    for (int i = 1; i <= 32; i++)
    {
        Network.insert(i);
    }
    for (int i = 0; i < 32; i++) {
        for (int j = i; j < 32; j++) {
            Vec_Dti[i][j] = Vec_Dti[j][i];
        }
    }
    for (int i = 0; i < 32; i++) {
        for (int j = 0; j < 32; j++) {
            if ((int)Vec_Dti[i][j] != 0)
                Network.insert(0, Vec_Dti[i][j], i, j);
        }
    }
    cout << endl;
    DisplayMatrix(&Network);
    cout << endl;
    double shortest[32][32] = { 0 };
    cout << "min_len";
    for (int i = 0; i < 32; i++)
    {
        Network.dijkstra(i);
        cout << i << ":\n";
        for (int j = 0; j < Network.Graph<int, double>::n; j++)
        {
            shortest[i][j] = Network.priority(j);
            cout << "\t" << Network.priority(j);
            if (j % 10 == 9)
                cout << endl;
        }
        cout << endl;
    }
    cout << endl;
    ofstream mycout("temp.txt");
    for (int i = 0; i < 32; i++)
    {
        for (int j = 0; j < Network.Graph<int, double>::n; j++)
        {
            cout << setw(5) << left << shortest[i][j];
            mycout << shortest[i][j] << '\t';
        }
        cout << endl;
        mycout << endl;
    }
    cout << endl;
    mycout.close();
    return 0;
}
```