Problem Chosen	2022 ShuWei Cup Summary Sheet	Team Control Number
B		2022100412715

Strategize and win thousands of miles in modern wars

Abstract

With the development of the technology and its application in area of strategy, the weapon-target assignment problem has consequently received a great deal of attention in recent years. How to reduce casualties and maximize resource utilization in war is a significant problem.This paper emphasizes the knowledge of deploying troops and optimizing the allocation of resources in war by simulating the offensive and defensive of red and blue sides in combat.

Several models are established: Model I: Multi-Objective Optimization Model; Model II: K-means Clustering Optimization Model; Model III: Game theory model of complete static information, etc.

For problem I, We are required to consider the attack difficulty, march distance and other factors of each node for military deployment. Inspired by **Multi-Objective Optimization algorithms**, We use **Genetic algorithm** and **Floyd algorithm** to determine the specific distribution of arms, built **Multi-Objective Optimization Model** to find the optimal command positions and several alternative positions for red and blue.The results are shown in Figure 4-6.

For problem II,We adopt **K-means Clustering Analysis Method**. Firstly, we divide the theater into 30 regions, and then determine the location of logistics points according to the shortest path of each region. At the same time, considering the potential threat of the enemy, we set up two logistics points in large regions to avoid the enemy's concentrated fire attack; We made assumptions about the demand of logistics materials for different arms, and the final results are shown in **Figure 7 and Table 5.**

For problem III,The **Game Theory Model** is used to establish the attack defense model. For the red side's attack, the blue side adopts the passive mode, thus establishing the blue side's static game theory model, so as to determine the better attack route of the red side.When the blue side retreats, establish the fastest evacuation model, and get the evacuation strategy when the blue side's information is interrupted by solving. In the case of good information, establish three evacuation channels, **372 →37, 98 →140,276 →378,** and then establish the priority evacuation model of frontier nodes to finally get the best evacuation strategy.

Finally, sensitivity analysis of the mathematical expectation shows that our model is not sensitive to change the key variables.It turns out that the changes have little effect on the final result and the model is **relatively stable**.

Keywords: WTA; Multi-Objective Optimization; K-means Clustering; Game theory; Genetic Algorithm; Floyd Algorithm;

Content

1. Introduction

1.1 Problem Background

In modern war, both offensive and defensive sides need to introduce efficient war strategies to increase war threats and reduce losses. Only by forming a relatively stable and balanced war dynamics can the ultimate goal of reaching consensus be realized as soon as possible.

In view of the above war problems, consider the following simplification of the red and blue war: suppose that the red and blue sides can only conduct initial platoons in positions with the same color, and each node has its own attack difficulty. The more difficult the attack is, you need to provide each side with the optimal operational strategy according to the actual number and characteristics of military weapons on both sides. The main fighting units of both sides are infantry, and the main weapons are light tanks, medium tanks, heavy tanks, self-propelled guns, strategic bombers and anti-aircraft guns that can be emptied.

1.2 Restatement of the Problem

The weapon resources of the red and blue sides are determined separately. Please solve the following three problems through appropriate simplifying assumptions and mathematical modeling methods:

(1) Design a mathematical model to determine the assigned positions and quantity scale of infantry, tanks, self-propelled artillery and air defense artillery of both sides, as well as the optimal command positions and several alternative positions of both sides according to the Annex.

(2) Based on the optimization results of question 1,build the optimization model of medical supplies, military supplies and daily supplies distribution and supply for both the Red and Blue. Provide key information such as the total number of workers and vehicles required under non supply mode. Provide the optimal supply plan of the Red and the Blue in the form of tables or graphs in the text.

(3) Combined with the first two questions, please put forward a better attack plan for the red team and a better retreat plan for the blue team when the red team is attacking and the blue team is defending. Given that the retreat node of the blue side is [371,403,78], what are the differences in the overall retreat scheme of the blue side when communication is good or interrupted.

1.3 Our work

This problem requires us to use the multi-objective optimization method to establish the WTA problem model to deduce the assumptions of the red and blue wars, simulate the attack and defense of the two sides, and provide the optimal plan for the strategic deployment and command of the red and blue camps.

For problem one,We use the Multi-objective Programming Method to establish the optimization model, and use the genetic algorithm and Freud algorithm to obtain the best deployment plan and determine the best command location and alternative command location of the red and blue sides;

For problem two,We use K-means Clustering Method to cluster the nodes on the map according to certain standards to establish an optimization model, calculate the γ-matrix, and finally determine the best supply scheme;

For problem three,We use the idea of game theory to make reasonable assumptions and finally achieve an excellent retreat programme.

The flow chart of the whole paper is as follows:

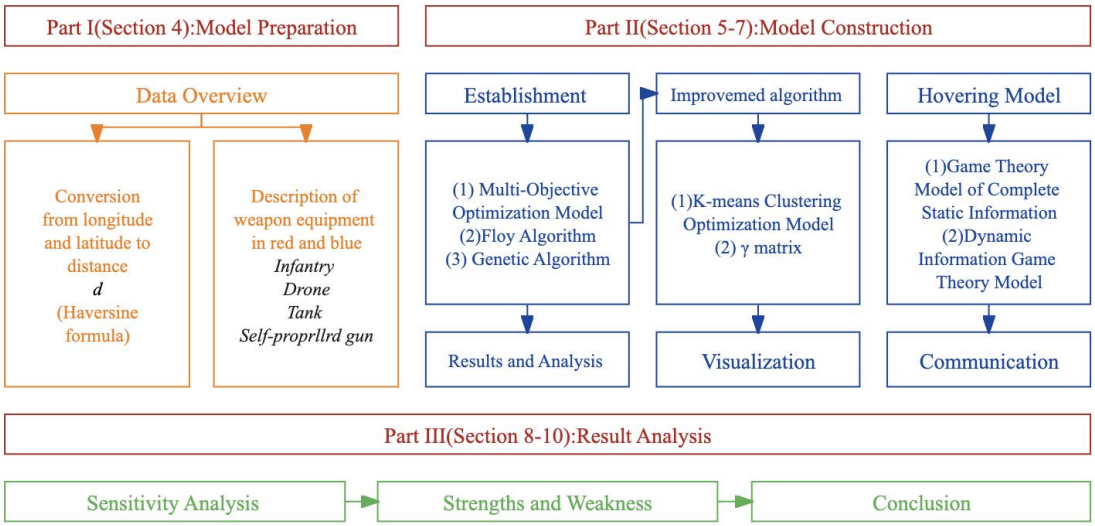

Figure 3: Flow Chart of Our Work

2. Assumptions and Explanations

Considering that practical problems always contain many complex factors, first of all, we need to make reasonable assumptions to simplify the model, and each hypothesis is closely followed by its corresponding explanation:

Assumption 1: Each node only deploys one type of weapon equipment.

Explanation: Different weapons will affect each other's play at the same node but in order to simplify the model ,we ignore the interactions between these factors.

Assumption 2: Infantry and tanks can only walk along the road.

Explanation: The longitude and latitude given by the question, the distance calculated is not proportional to the distance given by the question, so the distance of highway section cannot be determined.

Assumption 3: Medical supplies, military supplies and daily supplies are all in the same node.

Explanation: The three logistics materials are all at the same node, making the model simpler and easier to calculate.

Assumption 4: Each military transport vehicle needs 2 soldiers to drive, and each transport vehicle can transport 1000 units of logistics materials.

Explanation: The question needs to be given the key information such as the total number of workers and vehicles required in the non-supply mode is provided during the modeling.

Assumption 5: The blue side can only retreat to the node without army, and the speed of all arms is the same when retreating.

Explanation: In order to make the retreat of the blue side orderly, avoid a large number of arms gathering at one node when retreating; The same speed of all arms can simplify the model and reduce the complexity of the algorithm.

3. Notations

Some important mathematical notations used in this paper are listed in Table 1.

Table 1: Notations used in this paper

Symbol	Description
S_i	Threat faced by the jth node of the enemy
P_i	The number of nodes within the attack range of our i-th node
w_i	Attack difficulty of the j-th node of the enemy
h_k	The firepower of the k-th soldier
d_i	The strike diameter of the i-th node's deployed arms
$F(i)$	The node set connected by our i-th node
n_{ik}	The number of the k-th soldiers in our i-th node
C_k	Type of nodes containing the k-th soldier
D_{ii}	Distance from node i to node j
K	Threat balance Coefficient
E	Node set of forward position
γ	Demand Matrix
$floyd\ (D_{ii})$	The shortest path from node i to node j
m_{ii}	Whether node i and node j are adjacent
Z_F	Quantity of medical supplies required

4. Model Preparation

4.1 Conversion from longitude and latitude to distance

Haversine formula can determine the distance between the two places. It can well reflect the influence of the curvature of the earth and is a common way to determine the distance. between the two places with given latitude and longitude in geography[1], seen in the equation.

$$d = 2Rarcsin(\sqrt{sin^2(\frac{\varphi_2-\varphi_1}{2}) + cos(\varphi_1)cos(\varphi_2)sin^2(\frac{\lambda_2-\lambda_1}{2})}) \quad (1)$$

Where R represents the radius of the earth, and indicates the latitude of the two places, λ_1 and λ_2 express the longitude of them, d is the distance between the them.

Tip: Using this method, we calculated the distance between the nodes given in Annex1 Sheet1. It is found that the distance between the nodes given in Sheet2 is not proportional to that given in Sheet2, and it is impossible to determine the distance between any two points. Therefore, we give Assumption 2.

4.2 Description of weapon equipment in red and blue

Although both red and blue sides have the same type of weapons, their specific weapon parameters and number of equipment are quite different. We emphatically compared the configuration and quantity of tanks and self-propelled artillery of red and blue sides, and got the following Table 2:

Table 2: Comparison of weapons and equipment between red and blue

type	light tank		medium tank		heavy tank		self-propelled gun	
camp	red	blue	red	blue	red	blue	red	blue
number	800	340	300	570	180	420	7000	1400
speed	85km/h	37km/h	75km/h	46km/h	37km/h	45km/h	665m/s	495m/s
distance	230km	/	400km	/	185km	160km	17.23km	13.25km

5. Multi-Objective Optimization Model

5.1 Establishment of model

The timeliness of weapon-target allocation and the quality of the allocation scheme directly affect the effect of combat. Aiming at the WTA problem[2], firstly, it analyzes that our force distribution has the greatest threat to the enemy's nodes and the greatest safety factor to our nodes, and establishes the optimization model of the WTA problem; Then the model is transformed into a multi-objective programming model; Finally, Genetic algorithm is used to solve the problem.

5.1.1 Decision Variable

N_1 and N_2 are the total number of nodes of our side and the enemy side respectively; n_i indicating the i-th node of our side; m_{ij} indicating whether node i and node j are together; d_{ij} indicating the shortest path distance from node i to node $_j$,K_1 and K_2 represent threat coefficient and safety coefficient respectively.

5.1.2 Objective Function

The two goals can be attributed to the greatest threat to the enemy and the protection node is the most secure. Therefore, it is established as the following

Multi-Objective Optimization Model.

$$max \sum_{j=1}^{N_2} min\{\sum_{i=1}^{N_1} \frac{h_i \bullet n_i \bullet m_{ij}}{K_1 \bullet w_j}, 1\} \tag{2}$$

$$max \sum_{i=1}^{N_1} \sum_{j=1}^{N_1} \frac{h_i \bullet n_i \bullet m_{ij}}{K_2 \bullet d_{ij}} \tag{3}$$

5.1.3 Constraints

✧ To ensure that each node has only one type of weapon equipment:

$$max\{n_{i1}, n_{i1} \cdots n_{i7}\} = \sum_{k=1}^{7} n_{ik} \tag{4}$$

Where i=1,2$\cdots N$, n_{ik} represents the number of the *k*-th soldier in the *i*-th node.

✧ To ensure that the sum of forces at each node should be less than the total force:

$$\sum_{i=1}^{N} n_{ik} \leq max_k \tag{5}$$

Where k=1,2$\cdots 7$, max_krepresents the max number of the *k*-th kind of soldiers.

✧ To ensure that the number of nodes of anti-aircraft guns is less than 10, and the number of UAVs is not concentrated:

$$C(k) \leq 10 \tag{6}$$

Where k=2,7, C(k) indicates the number of nodes with soldier type K.

✧ To ensure that our self-propelled gun will not be attacked directly by the enemy:

$$n_{ik} = 0 \tag{7}$$

Where k=2,6,7, i$\in$ E, E represents a collection of leading edge nodes.

5.2 Research on this problem by Genetic Algorithm

Based on natural selection and biological genetic theory, Genetics Algorithm combines the rule of "natural selection, survival of the fittest" in the process of biological evolution with the random information exchange system of chromosomes within the population. It is an efficient global optimization search algorithm. The specific genetic algorithm process is shown in Figure 2:

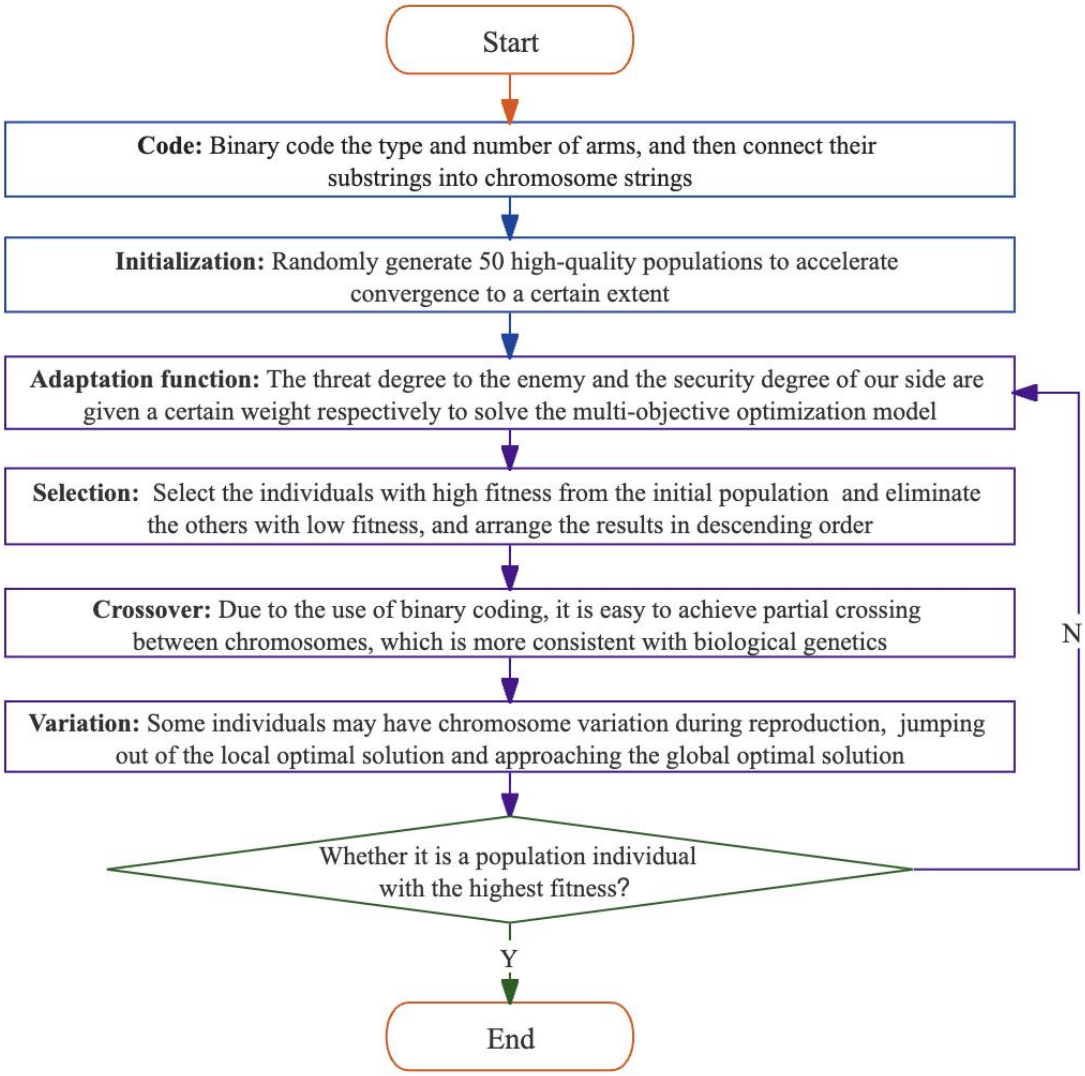

Figure 2: Genetic Algorithm flow chart

5.3 Research on this problem by Floy Algorithm

Floyd algorithm, also known as the interpolation method, is an algorithm that uses the idea of dynamic programming to find the shortest path between multiple source points in a given weighted graph.The specific genetic algorithm process is shown in Figure 3:

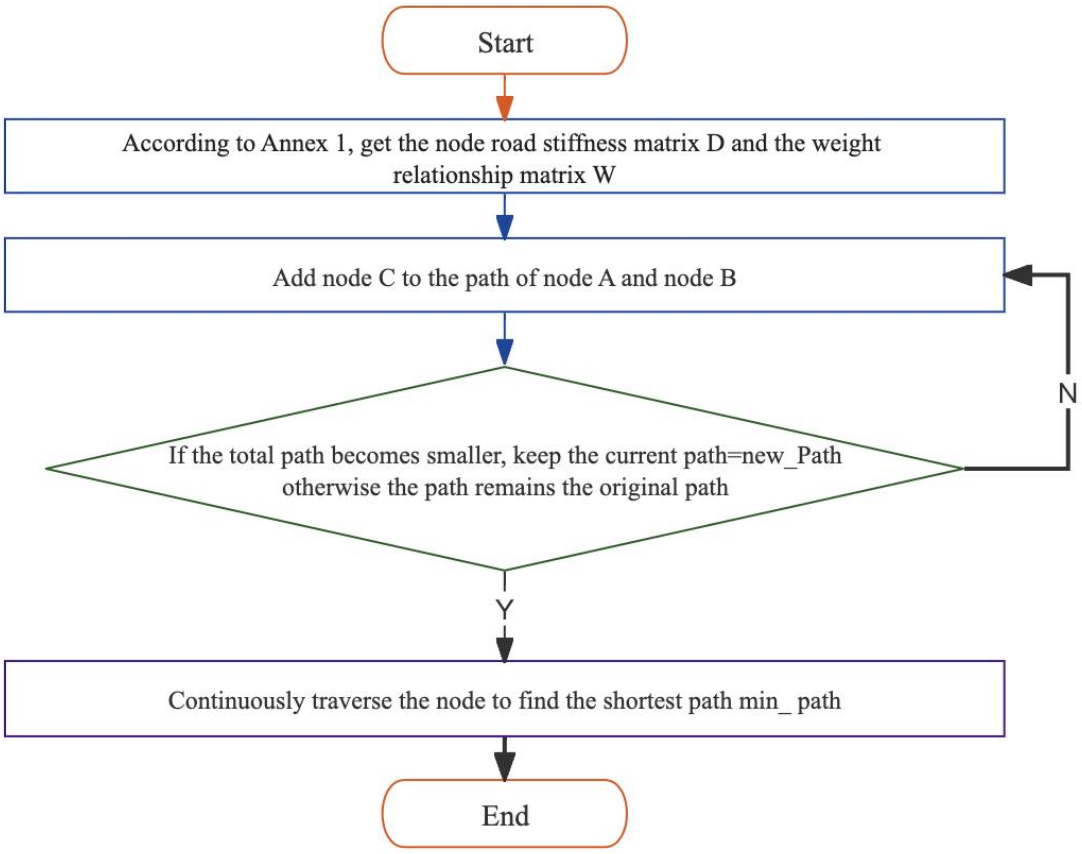

Figure 3: Floyd Algorithm flow chart

5.4 Problem Solving

5.4.1 Time complexity

Time complexity refers to the calculation workload required to execute the algorithm. There are several calculation methods:

①In general, the number of times that the basic operation of the algorithm is a certain function f (n) of the module n is repeated, so the time complexity of the algorithm is recorded: T (n) = O (f (n)).

Analysis: As the value of the module n increases, the growth rate of the time of the algorithm execution is directly proportional to the growth rate of f (n), so the smaller the f (n), the lower the time complexity of the algorithm, and the higher the efficiency of the algorithm.

②When calculating the time complexity, first find out the basic operation of the algorithm, and then determine the number of execution according to the corresponding statement, and then find out the same order of T (n) (its same order of magnitude has the following: 1, log (2) n, n, n log (2) n, n square, cubic of n, 2 n power, n!), After finding out, f (n) = the order of magnitude, if T (n) / f (n), find the limit can get a constant c, then the time complexity T (n) = O (f (n))

5.4.2 Space complexity

The spatial complexity of a program is the size of the memory required to run a program. Using the spatial complexity of the program, you can have a preliminary estimate of the memory required to run the program. In addition to the storage space and the instructions, constants, variables, and input data used by itself, a program also needs some working units to operate on the data and some auxiliary space to store the information needed for realistic computing. The storage space for the program includes the following two parts:

①**Fixed portion.** The size of this part of the space is independent of the number of data input / output. It mainly includes instruction space (i. e., code space), data space (constants, simple variables) and other space occupied. This part belongs to the static space.

②**Variable space.** This part of the space mainly includes the dynamically allocated space, and the space required for the recursive stack. The spatial size of this part is related to the algorithm, and the computational method is basically consistent with the time complexity.

The storage space required by an algorithm is represented by f (n).

S(n) = O(f(n))

Where *n* is the size of the problem, and S(n) represents the spatial complexity. According to the above mentioned, after analyzing the above two algorithms, we obtain the spatial complexity S (n) =O (1) of the single-point timing traffic signal light control algorithm, and S (n) =O (1) of the genetic algorithm.

5.4.4 Solution of Model

Through Python, we can get the military armed deployment plan, the best command position and the alternative command position of the red and blue sides. Tables 3 show the deployment plan of red and blue sides at some nodes; Figures 3 and 4 show the visual pictures of the military armed deployment of the red and blue sides. The numbers 0, 1, 2, 3, 4, and 5 in the table correspond to the blue, green, red, light blue, dark red, and yellow in the figure, which respectively correspond to the industry, heavy tanks, medium tanks, light tanks, drones, and self profiled guns.

Table 3: Schedule of weapon at some nodes of red and blue sides

Red nude	1	2	3	4	5	6	7	8	9	10
Type	5	1	1	3	0	4	2	1	5	5
Number	9	8	4	10	5	9	3	9	9	1
Red nude	11	12	13	14	15	16	17	18	19	20
Type	3	1	3	3	5	2	0	5	2	2
Number	5	6	1	8	8	9	7	4	9	10
Blue nude	11	12	13	14	15	16	17	18	19	20
Type	4	3	4	5	0	3	2	2	5	5
Number	0	6	9	12	14	25	0	1	20	2
Blue nude	35	36	37	38	39	40	41	42	43	44
Type	4	5	4	0	0	4	2	1	4	3
Number	14	12	23	5	25	21	1	1	28	14

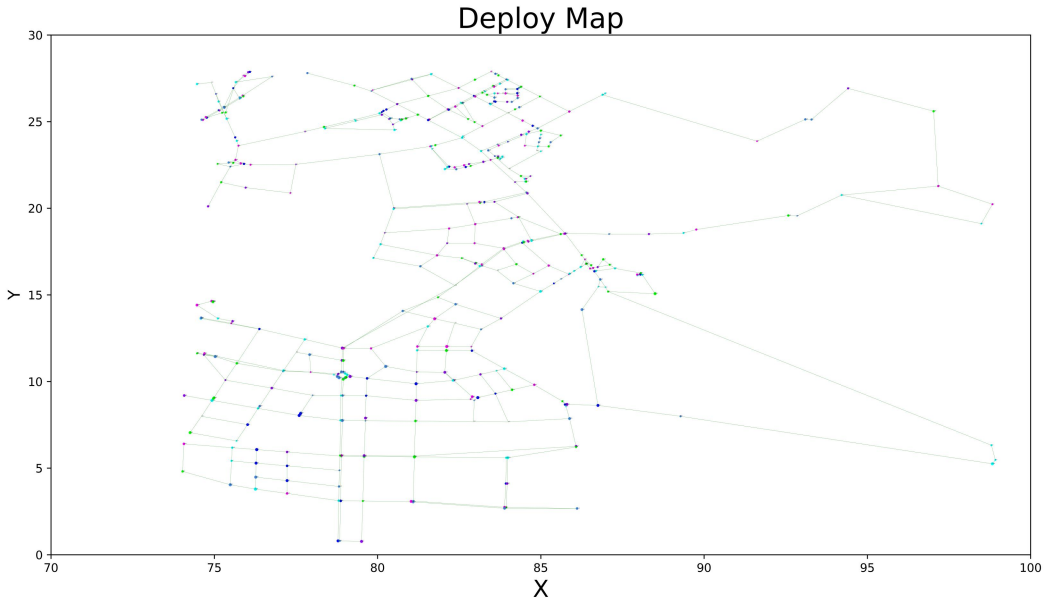

Figure 4: Distribution of arms

Due to the requirements of command location selection, there are many roads around the node, and there are many soldiers at the node, which makes it difficult for the enemy to attack the node. We can get the following expression:

$$argmax\ \frac{\sum_{i=1}^{N} h_i \bullet n_i \bullet m_{ij} \bullet w_j}{floyd\ (D_{ij,i\in E})} \tag{7}$$

We determine the optimal location of the command location by traversing the nodes of the red and blue sides. The results are shown in the Table and Figure below:

Table 4: Command location of red and blue sides

Type name	Node id	Type name	Command location
Optimal place of red	320	Optimal place of blue	76
Alternative place1 of red	147	Alternative place1 of blue	113
Alternative place2 of red	209	Alternative place2 of blue	81
Alternative place3 of red	213	Alternative place3 of blue	43

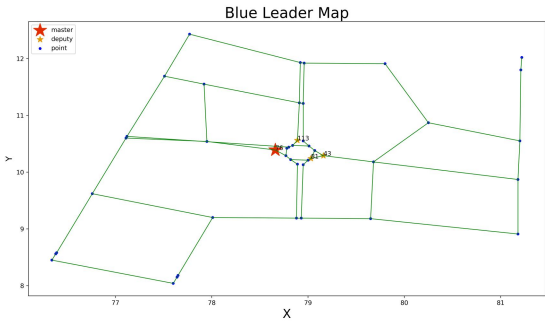

Figure 5: Blue command location

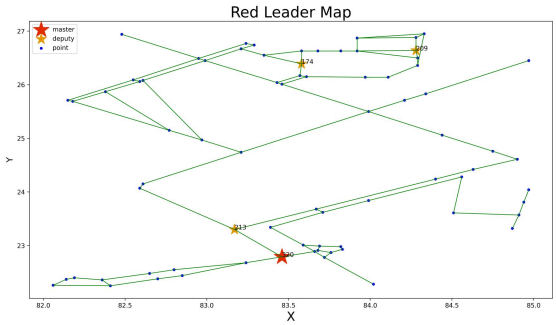

Figure 6: Red command location

6. K-means Clustering Optimization Model

6.1 Establishment of model

This problem requires us to establish the logistics supply plan configuration for the red and blue sides according to the battlefield situation. First, we use K-means clustering analysis to divide the red and blue sides into g1 and g2 categories. Besides calculate the quantity of medical supplies, military supplies and daily supplies required for each category. And then calculate the best logistics location according to the Floyd Algorithm. It is worth noting that due to the threat of enemy artillery and unmanned aerial vehicles, when we depart from the logistics location, military vehicles should try to avoid enemy fire covering the route. Finally use path (Z) to get the driving route.

6.1.1 Cluster Analysis

①Whether each node can be selected as our logistics location depends on the distance between the node and other nodes, as well as the number of nodes in the cluster.

②Sum the distances of each node to find the minimum value, so as to determine the logistics node.

③In order to consider the potential attack of the enemy, the first two scoring nodes are selected as the logistics points for node dense classification points.

6.1.2 γ matrix

Expression for determining logistics location

$$Z_F = argmin \sum_{i \in F} floyd(D_{ij}) \tag{7}$$

Where F represents a single node set under the same classification

The quantity of medical supplies, military supplies and daily supplies at each node can be obtained from the γ matrix as follows:

$$\text{requirement}_{1i} = \gamma_{k1} \bullet n_i \tag{7}$$

$$\text{requirement}_{2i} = \gamma_{k2} \bullet n_i \tag{8}$$

$$\text{requirement}_{3i} = \gamma_{k3} \bullet n_i \tag{9}$$

6.2 Problem Solving

We use Folyd Algorithm to get the clustering graph of red and blue according to 30 clustering points.

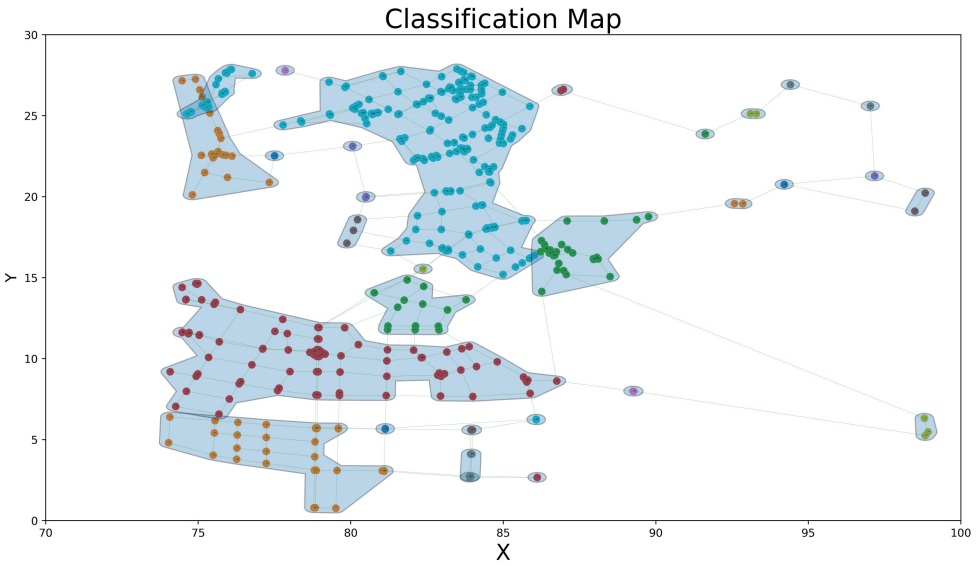

Figure 7: Clustering area

It is worth mentioning that we found that there is an "isolated area" in the red side, and the coordinate range is (73.5,25) ~ (76.5,28), that is, there is no road to the outside world, as shown in Figure.Therefore, there should be a logistics point in the area to distribute materials

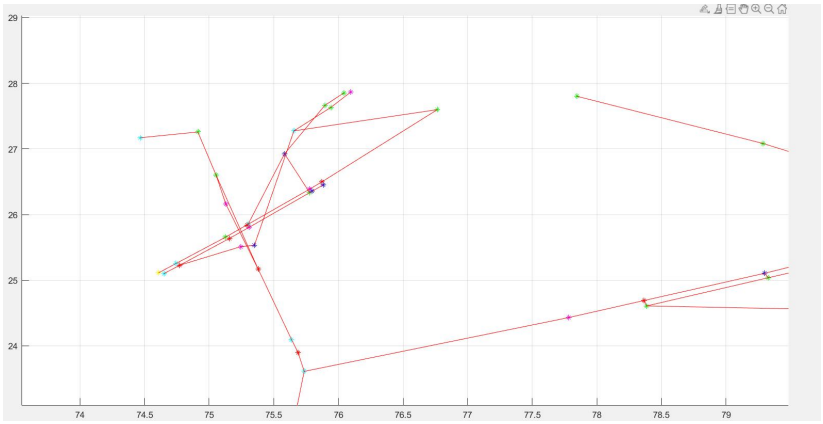

Figure 8: Isolated area

In order to facilitate the solution and meet the actual situation, we make the following assumption after looking up the data: 5000 units of medical supplies, 10,000 units of military supplies and 5000 units of daily supplies are required for every 10,000 infantry. We can draw a conclusion that 30608 military vehicles and 61216 soldiers need to be deployed to support logisticsThe specific distribution is shown in the following table:nodes of the red and blue sides. The results are shown in the Table and Figure below:

Table 4: Command location of red and blue sides

	Medical supplies	Military supplies	Daily supplies
Infantry needs(10^4)	5000	10000	5000
Tanks needs	50	200	50
Drones needs	10	500	10
self-propelled artillery needs	10	10	10

Since there are more than 400 nodes in this article, which cannot be listed one by one, the following 50 node data are provided:

Table 5: Best Supply Plan of Red and Blue

Node number	Medical supplies	Military supplies	Daily supplies	Node number	Medical supplies	Military supplies	Daily supplies
1	7060	353000	7060	26	40	40	40
2	450	1800	450	27	2150	2150	2150
3	400	1600	400	28	7550	30200	7550
4	200	800	200	29	100	400	100
5	50000	100000	50000	30	100	400	100
6	50	50	50	31	20	20	20
7	450	1800	450	32	350	1400	350
8	150	600	150	33	250	1000	250
9	90	4500	90	34	40	40	40
10	90	4500	90	35	20	20	20
11	50	200	50	36	1090	1090	1090
12	250	1000	250	37	90	90	90
13	300	1200	300	38	950	3800	950
14	50	200	50	39	100	400	100
15	80	4000	80	40	160	160	160
16	400	1600	400	41	30450	121800	30450
17	45000	90000	45000	42	0	0	0
18	70	3500	70	43	550	2200	550
19	200	800	200	44	20	20	20
20	450	1800	450	45	12450	49800	12450
21	500	2000	500	46	110	110	110
22	25000	50000	25000	47	950	3800	950
23	300	1200	300	48	110	110	110
24	20	1000	20	49	250	250	250
25	46	186	46	50	1000	4000	1000

7. Game Theory Model

7.1 Game theory model of complete static information

In this model, the blue side will not take the initiative to attack and support.

①Use artillery to bombard the enemy's forward position.When $S_j \geq 1$,it means that you are sure to destroy the military forces of the node and occupy the node ($j \in E$).

②Close to the enemy base by the shortest path. The blue base here is nude 76.

$$path = argmin\ floyd\,(d_{ij}),j=76 \tag{8}$$

③When the infantry is about to lead the way, give priority to infantry: when $d_{ij} < D_{ij}$,k=0.

④When there are multiple strike targets, select the target with the highest material value,$\max\{U_{i,i\in F}\}$.

The computer aided calculation results show that the best attack strategy of the Red Party is: 111→112→131→90→91→96→76.

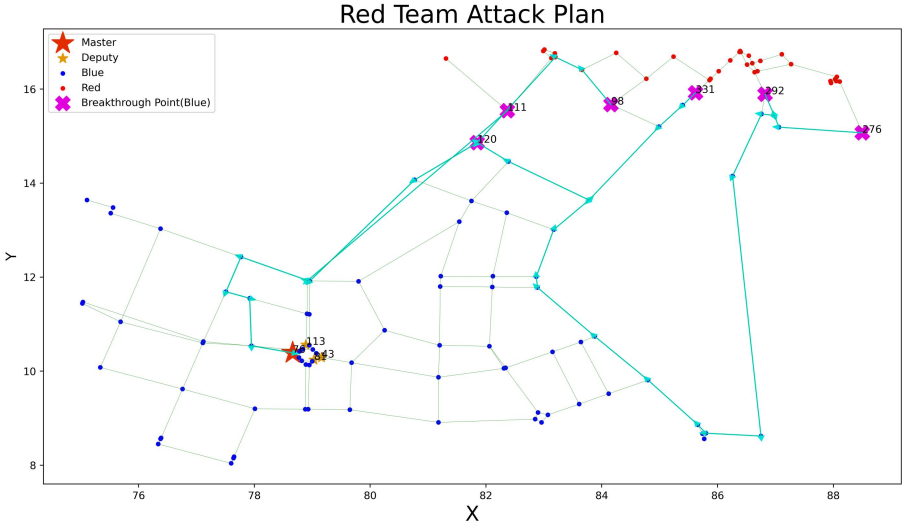

Figure 9:Red attack route

7.2 Communication interruption

When communication is interrupted, follow the fastest retreat method according to the shortest path of $floyd\,(D_{ij})$ and v_i of each node in the blue square,we can get $t_{ij} = \frac{floyd\,(D_{ij})}{v_i}$ .By comparing j=140 and 37,378 respectively, the matrix $T_{3\times168}$ is obtained, and the minimum value is selected from them. [row, col] = argmin T. Then the node to which this value refers will withdraw from the blue theater, T = T − c,

and record the col value. Finally, the **Python** show that:

Table 5: Blue retreat route

		Order	1	2	3	4	5	...	final
Information static retreat	Route	37	434	433	38	55	73		372
		140	139	165	135	305	134		276
		378	379	380	382	388	383		355
		Order	1	2	3	4	5	...	final
Information dynamic retreat	Route	37	434	55	73	387	386		433
		140	115	309	250	106	153		139
		378	379	388	377	329	300		380

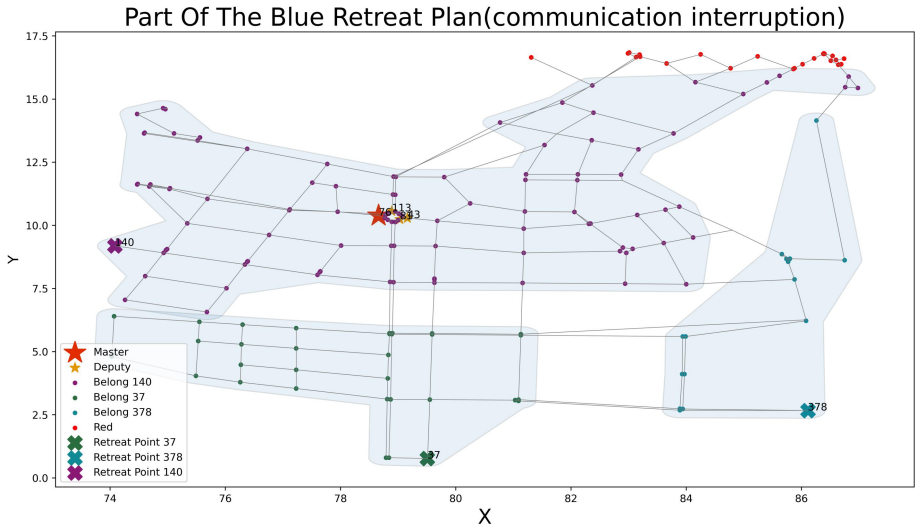

Figure 10: Blue retreat route (communication interruption)

7.3 Good communication

When the communication is not interrupted, according to the theater priority rule, we can know the offensive trend of the red side from the above game theory, so we can make a retreat deployment in advance, try to avoid fighting with the red side, and let the $i \in E$ node give priority to the fastest retreat method, so other forces on the way need to evacuate first, so as to make room for the theater forces. When there are troops withdrawing, we can update the theater data, re acquire the collection of frontier nodes E, and repeat the operations continuously, Finally, it is calculated by computer that:

Set up three special channels for three evacuation points to provide priority evacuation capability for each forward position. Through comparison of information interruption n_i, we found that the evacuation points are the same regardless of whether the information is interrupted, but the evacuation sequence is different. Under the conditions of the middle section, select the farthest node, that is, the forward node,

respectively. To this end, establish three channels, **372→37,98→140,276→378**. The first node on this channel exits, and the subsequent nodes retreat in order of Max (floyd (D_{ij})).

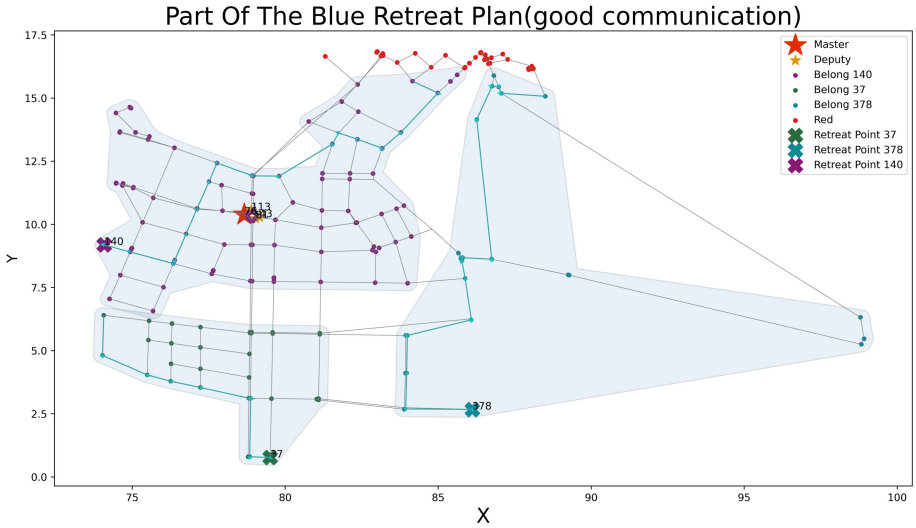

Figure 11: Blue retreat route (good communication)

8. Sensitivity Analysis

Sensitivity analysis is often used in optimization methods to study the stability of the optimal solution when the original data is inaccurate or changed.

To assess the stability of our model,we changed the key variables.It turns out that the changes have little effect on the final result and the model is relatively stable.

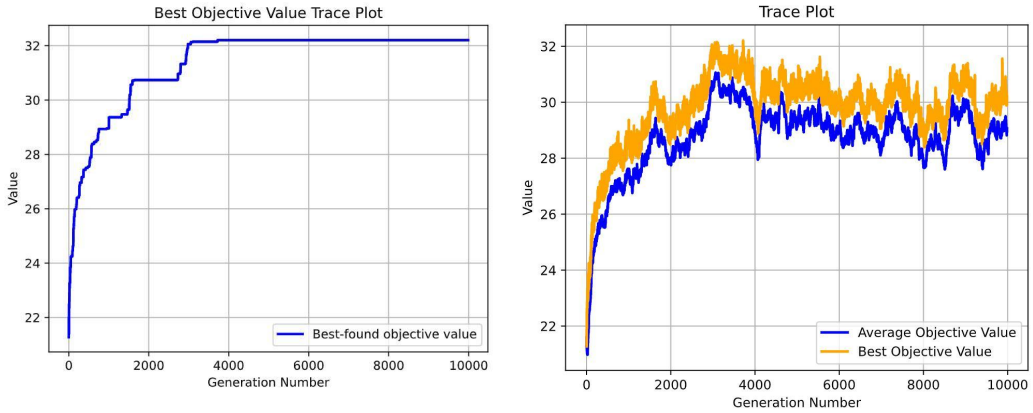

Figure 12:Sensitivity analysis of genetic algorithm

9. Strengths and Weakness

9.1 Strengths

Our model offers the following strengths:

- The main strength is its enormous extensible and including all factors into asingle, robust framework.

- The Multi-Objective Optimization Model is scientific and reasonable,
- The visualization work is done very well by us.
- Benefit by intelligent optimization algorithm, our model effectively achieved all of the goals. It was not only fast and could handle large quantities of data.
- Effectiveness of the model can be demonstrated under different parameter of the model by sensitivity analysis. So the model can be applied to much more events.

9.2 Weaknesses and Further Improvements

Our model has the following limitations and related improvements:

- During the military deployment of the red and blue sides, we only consider deploying one type of arms for each node. Although the model is greatly simplified, it is contrary to the actual situation.
- Applying some approximate analysis methods to modeling in other places may lead to suboptimal situations.

10. Conclusion

10.1 Conclusions of problem 1

Based on the Multi-Objective Optimization Model,We draw the following conclusions:

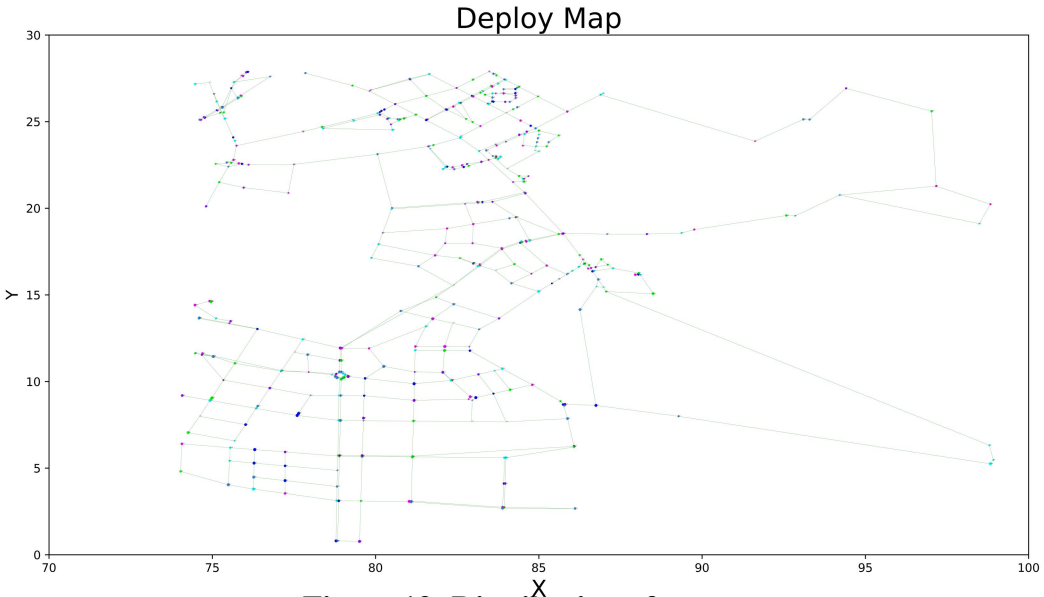

Figure 13: Distribution of arms

The complete answer is in

Schedule of weapon at some nodes of red and blue sides.xls

Table 6: Command location of red and blue sides

Type name	Node id	Type name	Command location
Optimal place of red	320	Optimal place of blue	76
Alternative place1 of red	147	Alternative place1 of blue	113
Alternative place2 of red	209	Alternative place2 of blue	81
Alternative place3 of red	213	Alternative place3 of blue	43

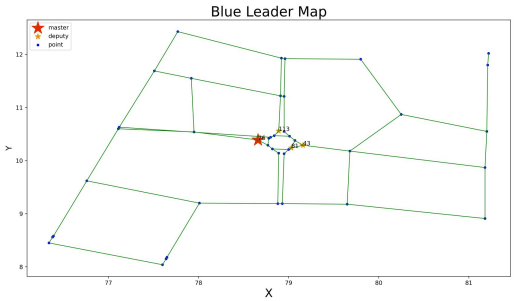

Figure 14: Blue command location

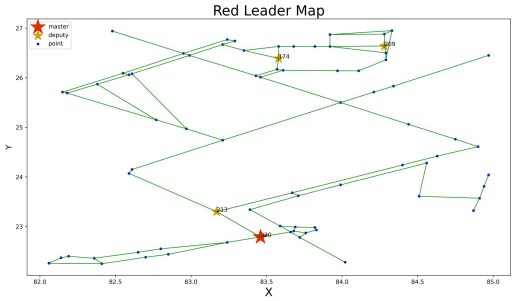

Figure 15: Red command location

10.2 Conclusions of problem 2

Based on the K-means Clustering Optimization Model,We draw the following conclusions:

Table 7: Best Supply Plan of Red and Blue

Node number	Medical supplies	Military supplies	Daily supplies	Node number	Medical supplies	Military supplies	Daily supplies
1	7060	353000	7060	26	40	40	40
2	450	1800	450	27	2150	2150	2150
3	400	1600	400	28	7550	30200	7550
4	200	800	200	29	100	400	100
5	50000	100000	50000	30	100	400	100
6	50	50	50	31	20	20	20
7	450	1800	450	32	350	1400	350
8	150	600	150	33	250	1000	250
9	90	4500	90	34	40	40	40
10	90	4500	90	35	20	20	20
11	50	200	50	36	1090	1090	1090
12	250	1000	250	37	90	90	90
13	300	1200	300	38	950	3800	950
14	50	200	50	39	100	400	100

15	80	4000	80	40	160	160	160
16	400	1600	400	41	30450	121800	30450
17	45000	90000	45000	42	0	0	0
18	70	3500	70	43	550	2200	550
19	200	800	200	44	20	20	20
20	450	1800	450	45	12450	49800	12450
21	500	2000	500	46	110	110	110
22	25000	50000	25000	47	950	3800	950
23	300	1200	300	48	110	110	110
24	20	1000	20	49	250	250	250
25	46	186	46	50	1000	4000	1000

The complete answer is in **Best Supply Plan of Red and Blue(Full version).xls**

10.3 Conclusions of problem 3

Based on the Game theory model of complete static information,We draw the following conclusions:

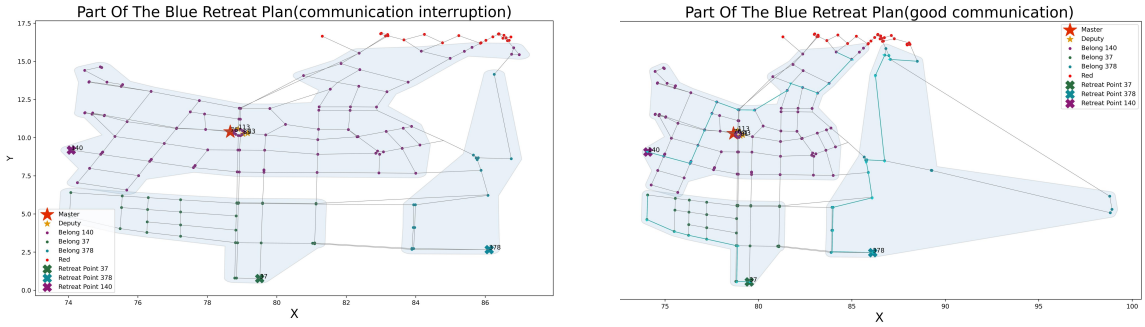

Figure 16: Blue retreat route

The complete answer is in **Blue retreat route (communication interruption).xls**

References

[1] G T S Lee, Lee G T S, Arisandi D, et al. Travel App - showing nearest tourism site using Haversine formula and directions with Google Maps. 2020,852(1):012161-.

[2] Li Jinjun, Cong Rong, Xiong Jiguang. Dynamic WTA optimization dodel of air defense operation of warships' formation [J]. Journal of System Engineering and Electronics, 2006, 17(1):126 – 131

[3] Chen Xia, Zhao Mingming. Multi-UAV Air-Combat Strategy Reseatch Base on the IncompleteInformation Dynamic Game in Uncertain Environment[C], 2013 International Conference on SystemsControl, Simulation and Modeling (ICSCSM'13).2013:1755-1759.

[4] Wang Xianyu, Xiao Yuming. Game Theory and Its Application [M]. Beijing: Science Press, 2011

Appendix

List of supporting materials:

(1) Blue retreat route (communication interruption).xls
(2) Best Supply Plan of Red and Blue(Full version).xls
(3) Schedule of weapon at some nodes of red and blue sides.xls
(4) Pythoncode.txt

Python code

```
import enum

import math

import pickle

import time

import numba

import numpy as np

from openpyxl import load_workbook

import geatpy as ea

import data.t1_1_data as data

from multiprocessing import Pool as ProcessPool

import multiprocessing as mp

from multiprocessing.dummy import Pool as ThreadPool




@enum.unique

class Arms(enum.Enum):

    infantry = 0

    lightTank = 1

    mediumTank = 2

    heavyTank = 3

    selfPropelledGun = 4

    # 无人机
```

```
    UAV = 5
    # antiaircraft = 6



# 策略
class Ploy:
    """
    Ploy 策略
    """

    '''
    火力映射表
    兵种->单位数量武器火力值
    '''
    fireMap = {
        # 一万步兵 / 4
        Arms.infantry: 4.44 / 4,
        Arms.lightTank: 1.01,
        Arms.mediumTank: 1.14,
        Arms.heavyTank: 1.92,
        Arms.selfPropelledGun: 0.047 * 40,
        Arms.UAV: 1.25,
        # Arms.antiaircraft: 0,
    }

    # 求解的目标阵营
    targetCamp = 'blue'

    @staticmethod
    def getPointMap(camp=None):
        if camp is None:
```

```
            return data.pointMap
        return getattr(data, f'{camp}PointMap')

    @staticmethod
    def getEnemyPointMap(camp):
        """
        获取敌方驻点
        :param camp:
        :return:
        """
        return Ploy.getPointMap('blue' if camp == 'red' else 'red')

    @staticmethod
    def getMyPointMap(camp):
        """
        获取我方驻点
        :param camp:
        :return:
        """
        return Ploy.getPointMap(camp)

    def initId2PointTable(self):
        """
        初始化  id->point 哈希表
        :return:
        """
        self.pointMap = self.getPointMap()
        self.myPointMap = self.getPointMap(self.targetCamp)
        self.enemyPointMap = self.getPointMap('blue' if self.targetCamp == 'red' else 'red')
        return self.pointMap
```

```
        # #  每次重新计算
        # pointMap = {}
        # for row in tuple(self.sheet_point_list.values)[1:]:
        #         id_ = int(row[0])
        #         x = float(row[1])
        #         y = float(row[2])
        #         camp = row[3]
        #         BeAttackDifficulty = float(row[4])
        #         pointMap[id_] = {
        #                 'x': x,
        #                 'y': y,
        #                 'camp': camp,
        #                 'BeAttackDifficulty': BeAttackDifficulty
        #         }
        # self.pointMap = pointMap

    def initXlsx(self):
        """
        加载表格
        :return:
        """
        wb = load_workbook('../1.xlsx')
        sheets = wb.worksheets
        self.sheet_point_list = sheets[0]
        self.sheet_distance = sheets[1]

    def initMatrix(self, camp):
        """
        初始化计算所有矩阵
        :return:
        """
```

```
        # self.adjacencyMatrix = adjacencyMatrix
        # [self.distanceMatrix, self.routeMatrix] = floyd(self.adjacencyMatrix)
        [self.adjacencyMatrix, self.distanceMatrix] = [
            getattr(data, f'{camp}CampAdjacencyMatrix'),
            getattr(data, f'{camp}CampDistanceMatrix')
        ]
        return [self.adjacencyMatrix, self.distanceMatrix]

    def initWeaponParam(self, weaponParam, weaponSizeParam):
        for pointId, weaponType in enumerate(weaponParam):
            self.pointMap[str(pointId + 1)]['weaponType'] = int(weaponParam[pointId])
            self.pointMap[str(pointId + 1)]['weaponSize'] = int(weaponSizeParam[pointId])

    def __init__(self, weaponParam, weaponSizeParam, camp='blue'):
        """
        :param weaponParam:  兵种部署情况
        :param weaponSizeParam:  兵种数量情况
        """
        # self.initXlsx()
        self.initId2PointTable()
        self.initMatrix(camp)
        self.initWeaponParam(weaponParam, weaponSizeParam)

    def getDistance(self, pointId1, pointId2) -> float:
        """
        计算两点之间的距离
        :param pointId1:
        :param pointId2:
        :return:
        """
        return float(self.distanceMatrix[pointId1][pointId2])
```

```
def getComputedFireParam(self, pointId: int) -> float:
    """
    获取计算火力值
    计算火力值 = 单位武器火力 * 武器数量
    :param pointId:
    :return:
    """
    return Ploy.getFireParamValue(self.pointMap[str(pointId + 1)]['weaponType'],
                                  self.pointMap[str(pointId)]['weaponSize'])

def getFireParam(self, pointId: int) -> float:
    """
    获取火力系数
    :param pointId:
    :return:
    """
    weaponTypeIndex = self.pointMap[str(pointId + 1)]['weaponType']
    return Ploy.getFireParamValue(tuple(Arms)[weaponTypeIndex])

@staticmethod
def getFireParamValue(weaponType: Arms, weaponSize: int = 1) -> float:
    """
    获取火力参数
    :param weaponType:  兵种或者武器类型
    :param weaponSize:  武器数量
    :return:  单位数量武器火力值
    """
    return Ploy.fireMap[weaponType] * weaponSize

def getWeaponSizeParam(self, pointId: int) -> int:
```

```
        """
        获取一点的兵力数量
        :param pointId:
        :return:
        """
        return self.pointMap[str(pointId + 1)]['weaponSize']

    def getBeAttackedDifficultyParam(self, pointId: int) -> float:
        """
        获取一点的被攻击难度
        :param pointId:
        :return:
        """
        return self.pointMap[str(pointId + 1)]['BeAttackDifficulty']

    def isConnected(self, pointId1: int, pointId2: int) -> bool:
        """
        判断两点之间是否连通
        :param pointId1:
        :param pointId2:
        :return:
        """
        return self.getDistance(pointId1, pointId2) != np.inf

    def getAroundConnectedCount(self, pointId) -> int:
        """
        获取周边连通点数
        :param pointId:
        :return:
        """
        # TODO: 有点问题，好像会算错
```

```
            return len(np.where(self.adjacencyMatrix[pointId] != np.inf)[0]) - 1



class Config:
    """
    兵力限制
    """
    limit = {
        'red': {
            # 单位：2500
            Arms.infantry: 500,
            # 单位：1
            Arms.lightTank: 420,
            # 单位：1
            Arms.mediumTank: 300,
            # 单位：1
            Arms.heavyTank: 180,
            # 单位：40
            Arms.selfPropelledGun: 175,
            # 单位：1
            Arms.UAV: 500,
            ## 单位：1
            # Arms.antiaircraft: 10,
        },
        'blue': {
            # 单位：2500
            Arms.infantry: 400,
            # 单位：1
            Arms.lightTank: 400,
            # 单位：1
            Arms.mediumTank: 570,
```

```
            # 单位：1
            Arms.heavyTank: 340,
            # 单位：40
            Arms.selfPropelledGun: 350,
            # 单位：1
            Arms.UAV: 300,
            ## 单位：1
            # Arms.antiaircraft: 10,
        }
    }

    """
    当前求解阵营
    """
    targetCamp = 'blue'

    """
    当前阵营驻点数量
    """
    pointSize = 0

    """
    兵种类型数量
    """
    armTypeSize = 0

    callTimes = 0

    @staticmethod
    def getLimit() -> int:
        ratio = {
```

```
            'blue': 20,
            'red': 50
        }
        return math.floor(max(list(Config.limit[Config.targetCamp].values())) / ratio[Config.targetCamp])
        return math.ceil(np.array(list(Config.limit[Config.targetCamp].values())).sum() / (Config.pointSize))

    @staticmethod
    def getPointList() -> dict:
        """
        获取己方阵营驻点数量
        :return:
        """
        return Ploy.getMyPointMap(Config.targetCamp)


class MyProblem2(ea.Problem):  # 继承 Problem 父类
    """
    max f1  威胁系数之和
    max f2  安全系数之和
    s.t.
    兵种数量限制

    各点兵种  a0,a1,a2,a3,a4 ∈ {0,1,2,3,4,5}
    各点兵力数量  b0,b1,b2,b3,b4 ∈ {0,1,2,...,8225}
    各方布置 10 个防空点
    """

    def __init__(self, M=1, PoolType='Thread'):
        """
        :param M:  目标函数个数
        :param PoolType:
```

```
        """
        # 当前阵营驻点数量
        Config.pointSize = len(Config.getPointList())
        # 兵种类型数量
        Config.armTypeSize = len(tuple(Arms))

        name = 'MyProblem2'  # 初始化 name（函数名称，可以随意设置）
        Dim = Config.pointSize * 2  # 初始化 Dim（决策变量维数）
        maxormins = [-1] * M  # 初始化 maxormins（目标最小最大化标记列表，1：最小化该目标；-1：最大化该目标）
        varTypes = [1] * Dim  # 初始化 varTypes（决策变量的类型，0：实数；1：整数）
        lb = [0] * Dim  # 决策变量下界
        ub = [Config.armTypeSize - 1] * Config.pointSize + [Config.getLimit()] * Config.pointSize  # 决策变量上界
        lbin = [1] * Dim  # 决策变量下边界（0 表示不包含该变量的下边界，1 表示包含）
        ubin = [1] * Dim  # 决策变量上边界（0 表示不包含该变量的上边界，1 表示包含）
        # 调用父类构造方法完成实例化
        ea.Problem.__init__(self,
                            name,
                            M,
                            maxormins,
                            Dim,
                            varTypes,
                            lb,
                            ub,
                            lbin,
                            ubin)

        # 设置用多线程还是多进程
        self.PoolType = PoolType
        if self.PoolType == 'Thread':
```

```
            self.pool = ThreadPool(mp.cpu_count() * 10)  # 设置池的大小
        elif self.PoolType == 'Process':
            self.pool = ProcessPool(mp.cpu_count())  # 设置池的大小

    def evalVars(self, Vars):  # 目标函数，采用多线程加速计算
        needTimeStart = time.time()

        N = Vars.shape[0]
        args = list(zip(Vars, list(range(N))))
        resultList = list(self.pool.map(subVars, args))
        fList = [i[0].tolist()[0] for i in resultList]
        CVList = [i[1].tolist() for i in resultList]
        f, CV = [np.array(fList), np.array(CVList)]

        Config.callTimes += 1
        # print(f'callTimes: {Config.callTimes}, needTime: {time.time() - needTimeStart}')
        return f, CV

        # 测试某个兵种类型的数量
        np.sum(Vars[:, Config.pointSize:][0][Vars[:, :Config.pointSize][0] == 1])


def subVars(args):
    # needTimeStart = time.time()

    Vars, indexN = args
    f1 = []
    f2 = []
    # 当前预测的策略个数
    polySize = 1
```

```
var = Vars
'''
遍历每种方案
'''

# 各点兵种 a0,a1,a2,a3,a4 ∈ {0,1,2,3,4,5}
everyPointWeaponRowList = var[:Config.pointSize]
# 各点兵力数量 b0,b1,b2,b3,b4 ∈ {0,1,2,...,100}
everyPointWeaponSizeRowList = var[Config.pointSize:]

'''建立当前策略方案模型'''
ploy = Ploy(everyPointWeaponRowList, everyPointWeaponSizeRowList, Config.targetCamp)
pointIdList = tuple(range(ploy.adjacencyMatrix.shape[0]))

'''
max f1 当前方案的威胁系数之和
'''
totalDangerScore = 0
for iPointId in pointIdList:
    '''
    遍历每个点，计算当前点 i 的威胁系数
    描述:
        威胁系数和自身装备实力以及距离其他点的距离有关
    '''
    # i 点的威胁系数
    currentDangerScore = 0

    # 当前点火力参数
    fireParam = ploy.getFireParam(iPointId)
    # 当前点兵力数量
    weaponSizeParam = ploy.getWeaponSizeParam(iPointId)
```

```
            # 当前点被攻击难度
            beAttackedDifficultyParam = ploy.getBeAttackedDifficultyParam(iPointId)
            KDanger = 100

            '''
            威胁系数
            '''
            # i 点周边点数
            connectedCountParam = ploy.getAroundConnectedCount(iPointId)

            # j 点对 i 点的威胁系数贡献
            value = ((fireParam * weaponSizeParam * 0.1) * connectedCountParam * beAttackedDifficultyParam *
10) / KDanger

            currentDangerScore += value

            # 计算当前方案的总威胁系数
            totalDangerScore += currentDangerScore
        f1.append(totalDangerScore)

        f = np.array(np.matrix(f1).T)

        # 利用可行性法则处理约束条件
        # 构建违反约束程度矩阵
        """
        s.t.
        兵种数量限制

        各点兵种 a0,a1,a2,a3,a4 ∈ {0,1,2,3,4,5}
        各点兵力数量 b0,b1,b2,b3,b4 ∈ {0,1,2,...,8225}
        各方布置 10 个防空点
```

```
    """
    # 各点兵种 a0,a1,a2,a3,a4 ∈ {0,1,2,3,4,5}
    everyPointWeaponRowList = Vars[[i for i in range(0, Config.pointSize)]]
    # 各点兵力数量 b0,b1,b2,b3,b4 ∈ {0,1,2,...,100}
    everyPointWeaponSizeRowList = Vars[[i for i in range(Config.pointSize, Config.pointSize * 2)]]
    '''
    各个兵种使用数量
    '''
    everyWeaponCounter = [
        0 for i in range(len(tuple(Arms)))
    ]
    for weaponType in Arms:
        currentWeaponSizeList = everyPointWeaponSizeRowList[everyPointWeaponRowList == weaponType.value]
        everyWeaponCounter[weaponType.value] = np.sum(currentWeaponSizeList)

    CV_ = np.array(everyWeaponCounter)
    """
    违反约束程度矩阵
    """
    CV = []
    for weaponType in Arms:
        currentCV = CV_[[weaponType.value]]
        CV.append(currentCV - Config.limit[Config.targetCamp][weaponType])

    CV = np.hstack(CV)

    # self.callTimes += 1
    # print(f' threadId: {indexN}, callTimes: {self.callTimes}, needTime: {time.time() - needTimeStart}')
    return f, CV
```

```
import random
import geatpy as ea  # import geatpy

if __name__ == '__main__':
    # 实例化问题对象
    problem = MyProblem2()
    # 构建算法
    algorithm = ea.soea_SGA_templet(
        problem,
        ea.Population(Encoding='BG',  # 种群的染色体是'BG'编码
                      NIND=100),  # 种群数量
        MAXGEN=9999,  # 最大进化代数
        logTras=1,
        trappedValue=1e-6,  # 单目标优化陷入停滞的判断阈值。
        maxTrappedCount=1000  # 进化停滞计数器最大上限值。
    )  # 表示每隔多少代记录一次日志信息，0 表示不记录。
    # algorithm.mutOper.Pm = 0.2  # 修改变异算子的变异概率
    # algorithm.recOper.XOVR = 0.7  # 修改交叉算子的交叉概率

    # 求解
    res = ea.optimize(algorithm,
                      seed=1,
                      # prophet=prophet,
                      verbose=True,
                      drawing=1,
                      outputMsg=True,
                      drawLog=True,
                      saveFlag=True,
                      dirName='result_blue',
                      )
```