Team Control Number

20201009824

Problem Chosen

C

2020

ShuWei Cup

Summary Sheet

Summary

When winter comes, the snow on the roads will affect the traffic of the whole city.In order to ensure the normal operation of urban traffic, it is necessary to arrange an efficient and reasonable road snow clearing plan.

In task 1, Floyd algorithm is used to solve the shortest path between any two intersections. Under the condition that the snow removal workload in each region is balanced, the urban area is divided into N sub-regions with the clustering analysis idea, and the road map is transformed into an undirected graph according to the connectivity relationship between intersections in any sub-region. On the basis of the above undirected graph, a new undirected graph is drawn by taking into account the ratio of the width of the road and the width of the snow clearing of the snowmobile at one time and redetermining the number of edges of the two adjacent points. According to the **Hungarian algorithm**, the driving scheme that makes every route in the subregion be cleaned up and the total distance traveled is the shortest is found. The undirected graph is transformed into Euler graph by adding heavy edge method, and the **Euler loop**[4], namely the solution of this problem, is obtained by using The **Fleury algorithm**. In the calculation example of n=10, according to the model, namely the relevant data, it can be calculated that the completion time of snow removal is about 2.9h.

In task 2, the idea of **classification discussion** is adopted to establish the relationship between function and variable in turn, and the two critical values of snow thickness are solved.By constructing a **multi-objective function**, the relationship between snowplow, transporter and sanitation worker and working time and snow thickness is solved by using constraint variables.When the snow thickness is 0.8 meters and the upper limit of working time is 3 hours.We can figure out that we need 14 plows, 39 transporters and 3,250 sanitation workers.

In task 3, the method of analogy is adopted, which is the model in task 1.On this basis, variables are added to obtain a new model, which takes into account the impact

of vehicles on snow removal.

In task 4, the **priority coefficient** is introduced to compare the difference in the number of priority roads among different route schemes, and the value of the priority coefficient depends on the weighted sum of the number of priority roads. In a certain period of time, the route plan with higher priority coefficient is more in line with the standard of snow removal in real life. By generating the **minimum spanning tree algorithm**, the route scheme with the highest priority coefficient can be obtained, that is, the best snow removal scheme considering the priority level of the section.

Key word: Hungarian algorithm Euler loop Fleury algorithm multi-objective function priority coefficient minimum spanning tree algorithm

Content

1. Introduction

1.1 Background

When winter comes, the temperature is dropping, which is followed by massive snowfall. When the snowfall reaches a certain level, it will bring a lot of inconvenience to the more urbanized areas.If snow removal work is not implemented in time, it will seriously affect urban traffic and People's Daily life. At this time, it is particularly important to do snow removal work well.

Every city's sanitation department needs to make a road cleaning plan before large-scale snow removal, which not only considers the number of roads, width, priority and cleaning difficulty, but also considers how much human and material resources, how long it will take, and where the cleared snow will be transported. However, the most important thing is how to give a general plan while considering all the problems comprehensively, so that the cleaning plan can be carried out efficiently and orderly under the condition of limited manpower and material resources.

1.2 Work

The thickness of snow determines the cleaning method, so we need to understand different cleaning methods and consider different methods according to different problems.Whether we give a reasonable snow removal plan or the best snow removal plan, we need to make the plan more efficient under the given conditions. In this paper, we need to solve the following problems：

- When the snow volume is relatively small, the snow can be cleaned to the nearby green belt or leisure area. With the limitattion of the number of cleaning vehicles is fixed, we need to provide a reasonable cleaning plan.
- When the snow volume is relatively large and the municipal sanitation has relatively fixed configurations for snowplows, transport vehicles and sanitation workers, we need to provide the optimal snowplow, transport vehicle and sanitation worker assignment plan so as to complete the snow plow task as soon as possible.
- In the actual road cleaning process, sanitation workers often encounter the situation that there are parked vehicles on both sides of the road, which will undoubtedly increase the difficulty of road cleaning, and the number of parking spaces in the city is very limited. We shoule provide a reasonable snow removal schedule so that we can clear the vehicles parked on both sides of the road in advance, and at the same time, we can resume the use of parking spaces as soon as possible.
- There is a very real problem that there are many overpasses and obvious ramps in cities, and different roads have different cleaning priorities. We should consider the actual situation and road priority level to carry out a comprehensive planning, and finally give an optimal snow removal plan.

2. Problem analysis

2.1 Analysis of question one

The first question is an optimization problem, as soon as possible in order to achieve the purpose of stopping after the roads, the city of the whole area is divided into multiple small area, and the workload is roughly equal to every small area. Multiple small areas need to be cleaned at the same time, and the cleaning vehicle should complete the task of snow removal in about the same time. In the actual process, in order to minimize the working time of snow removal, it is necessary to determine the optimal route planning of the cleaning vehicle in any small area, so as to minimize the empty driving time of the cleaning vehicle, that is, the sum of each empty driving distance is the minimum, so as to complete the snow removal task the fastest. This problem can be solving 0-1 programming model is established first analysis the each clean vehicles need to clean up the route of the area, get the corresponding area of the undirected graph, the use of postal problems of construction of eulerian graph algorithm thought, find out the euler loop, is idle travel the shortest route

Under the assumption that the snow can be cleaned to the nearby green belt or leisure area, task one needs to consider a scheme of snow removal area division and route planning for clean vehicles, thus establishing two models under this module.

2.2 Analysis of question two

The second requirement is that when the volume of snow is relatively large, an optimal allocation scheme for snowplows, transport vehicles and sanitation workers can be provided so that the task of snow clearing can be completed as soon as possible.Because the number of vehicles and sanitation workers provided by the municipal Health Bureau is limited, the minimum requirements for vehicles and sanitation workers should be found within a given working time.Choose to use only plows or both plows and transporters depending on the thickness of the snow.And no matter use a few tool car, must satisfy the time to be the least, the number of tool car is the least.So through the analysis of snow thickness, working time and the number of vehicles using tools to find the best choice to meet the conditions.According to the information inquired, the number of sanitation workers is directly proportional to the number of transport vehicles, so we can get an optimal snowplow, transport vehicles and sanitation workers distribution scheme.

2.3 Analysis of question three

In the third question, consider not only the time it takes to clear the snow, but also the time it takes to clear the vehicles on the road.Only by removing parked cars from the road can the snow clearing work be carried out smoothly.The amount of time it takes to clear the road depends on the depth of snow and the width of the road.Therefore, a reasonable snow removal plan can be obtained by establishing and analyzing the relationship between the three.

2.4 Analysis of question four

For the fourth question, consider not only the difference in the priority of clearing roads, but also the connections between roads with different priorities. The problem is transformed into how to make the routes cleared by the cleaning vehicles within a certain time as many roads with the highest priority as possible, the number of other priority roads is positively correlated with the priority, and the route plan containing the most roads with the highest priority is selected. If the number of roads with the highest priority is the same, then the number of roads with the next highest priority is compared, and so on, an optimal snow removal plan taking into account the priority level of the road segment is obtained.

3. Symbol and Assumptions

3. 1 Symbol Description

Symbols	Definition
n	The number of subregions
S_i	Actual snow removal effort
d_{ik}	Shortest distance from the K intersection point to the I center point
u_1^0	Initial center point
T_c	Snow clearing time
T_d	Empty travel time
d_f	Empty travel distance
f_t	Priority coefficient
m_k	The number of corresponding priority road
w_k	The weight of corresponding priority road
S_{all}	The total area of road snow
V_{all}	The total volume of road snow
h	The thickness of snow on the road
dist	The average thickness of snow
C_j	The average snow removal workload of each intersection
t_h	The upper limit of a given snow removal time

n	The number of snowplows

3.2 Fundamental assumptions

To simplify the given problem and make it more suitable for simulating real life conditions, we propose the following basic assumptions, each of which is reasonable.

- It is assumed that vehicles related to snow removal do not break down and will not be blocked by traffic during the working process.
- Snow cover is the same on all roads.
- In the snow clearing process, the snow clearing speed and driving speed of the snow clearing vehicle are constant.
- It is assumed that the snow will not melt into water during the entire transport.

4. Model

4.0 Preparation:

4.0.1 Calculate the distance based on latitude and longitude

According to the latitude and longitude of 141 intersections given in the title, the difference of latitude and longitude can be used to calculate the actual distance between the two intersections.

Set the latitude and longitude of point A as $(\mathrm{LonA}, \mathrm{LatA})$, set the latitude and longitude of point B as $(\mathrm{LonB}, \mathrm{LatB})$. Taking prime meridian as datum, east longitude is positive, west longitude is negative. Based on the equator, the north latitude is replaced by $(90°-\mathrm{Latitude})$, and the south latitude by $(90°+\mathrm{atitude})$. After treatment, the two points are $(\mathrm{MLonA}, \mathrm{MLatA})$ and $(\mathrm{MLonB}, \mathrm{MLatB})$ respectively. Then according to the trigonometric derivation, the calculation formula of the distance between two points is obtained.

$$\begin{cases} \mathrm{dLon} = \mathrm{LonA} - \mathrm{LonB} \\ \mathrm{dLat} = \mathrm{LatA} - \mathrm{LatB} \\ \mathrm{d} = \sin(\mathrm{dLat}/2)^2 + \cos(\mathrm{LatA}) \times \cos(\mathrm{LatB}) \times \sin(\mathrm{dLon}/2)^2 \\ \mathrm{dist} = 2 \times \mathrm{a} \times \sin\sqrt{\mathrm{a}} \times 6317 \times 1000 \end{cases} \tag{0-1}$$

4.0.2 Convert line workload into point workload

Assume that the width of the road is $w_i\ (i=1,2,\cdots,141)$, the length of the road is $l_i\ (i=1,2,\cdots,141)$, the average thickness of snow cover is h. Since the snow is evenly distributed on the road, the amount of snow removal on each road can be calculated as

$$W_i = w_i \times l_i \times h \quad (i=1,2,\cdots,141) \tag{0-2}$$

As is known to all, roads are not divided as easily as intersections, which requires consideration in assigning work area. So we substitute the average amount of work per intersection for the amount of work per road.The average snow removal workload of each intersection can be seen as the aggregation of all road workloads passing through this intersection, and since each road connects two intersections, the average snow removal workload of each intersection can be expressed as

$$C_j = \frac{1}{2}\sum W_i \ (j \subset i) \qquad (0\text{-}3)$$

Among them, the eligible road i must pass through the intersection j .

4.1 Task 1：Optimal scheduling by region

4.1.1 Area partitioning model for clean vehicles

For any city, its urban transportation network is often very important. In order to achieve the goal of smooth roads after the snow stopped as soon as possibl e, the whole urban area was divided into several small areas, and several small areas were cleaned up at the same time.At the core of this problem, it is necessary to consider the shape rules of the clearing area so that the snow removal tasks in each area are balanced, that is, the snow removal area is approximately equal.

According to the above analysis, we divide the urban area into n small areas (the number of small areas is equal to the number of clean vehicles). n intersection is randomly taken as the initial center point in a large area, and the shortest circuit length between each intersection and different center points is calculated by Floyd algorithm[2]. Then the center point and the corresponding area range are determined by cluster analysis. At the same time, this problem is actually an optimization problem. The 0-1 programming model can be used to solve this multi-objective problem. It is necessary to consider whether the shortest distance between the boundary intersection and the center point in each region is approximately equal at the same time And the amount of snow removal tasks processed in each different area is balanced.

If the intersection of the urban area is divided into an area denoted by X_{ij} , then

$$X_{ij} = \begin{cases} 0, & \text{Intersection J is not divided into area I} \\ 1, & \text{Intersection J is divided into area I} \end{cases} \qquad (1\text{-}1)$$
$$(j = 1, 2, \cdots 141, \quad i = 1.2, \cdots n)$$

n is the number of regions divided.

(1) Establishment of objective functions:

This is a multi-objective problem, so you need to build multiple objective functions.

Objective function a:

Whether the workload is balanced can be reflected by the standard deviation of snow removal area in each region. The smaller the standard deviation, the more balanced the workload.

Actual snow removal workload for the i region

$$S_i = \sum_{j=1}^{141} X_{ij} C_{ij} \quad (i = 1, 2, \cdots, n) \tag{1-2}$$

C_{ij} is the snow removal workload of the j intersection handled by the i area every day.

Average snow surface workload per region

$$\overline{S} = \frac{1}{n} \sum_{i=1}^{n} S_i \quad (i = 1, 2, \cdots, n) \tag{1-3}$$

The standard deviation of regional snow removal workload to be satisfied is as s mall as possible, i.e

$$MIN \sqrt{\frac{1}{n} \sum_{i=1}^{n} S_i - \overline{S}^{\,2}} \quad (i = 1, 2, \cdots, n) \tag{1-4}$$

Objective function b:

In order to facilitate snow removal, the distance between each intersection and th e center should be minimized, i.e

$$MIN \left\{ d_{1j}, d_{2j}, \cdots, d_{nj} \right\} \quad \left(j = 1, 2, \cdots, 141 \right) \tag{1-5}$$

Objective function c:

Whether the shortest distance between the boundary intersection and the center point is approximately equal can be reflected by the standard deviation of the shortest distance from each boundary point to the center point. The smaller the standard deviation, the more balanced the workload.

In the same region, the shortest distance from the KTH intersection point on the boundary to the center point I is d_{ik} , so the average shortest distance from the initial p oint to the boundary point is

$$\overline{d} = \frac{1}{m} \sum_{k=1}^{m} d_{ik} \quad k = 1, 2, \cdots, m \tag{1-6}$$

m is the number of boundary points in the region.

The standard deviation of the shortest distance between the boundary points and t he center point should be as small as possible, i.e

$$MIN \sqrt{\frac{1}{m} \sum_{k=1}^{m} \left(d_{ik} - \overline{d} \right)^2} \quad \left(k = 1, 2, \cdots, m \right) \tag{1-7}$$

The constraint conditions are:

The constraint of regional division. Because there are n areas ,141 intersections and each intersection is divided into a certain area, i.e

$$\sum_{i=1}^{n} X_{ij} = 1 \quad \left(j = 1,2,\cdots,141 \right) \tag{1-8}$$

The 141 intersections are divided into n areas, i.e

$$\sum_{j=1}^{141}\sum_{i=1}^{n} X_{ij} = \mathrm{n} \tag{1-9}$$

The basic constraints are as follows,

$$S.T\begin{cases} \sum_{i=1}^{n} X_{ij} = 1 \quad \left(j = 1,2,\cdots,141 \right) \\ \sum_{j=1}^{141}\sum_{i=1}^{n} X_{ij} = \mathrm{n} \\ X_{ij} = 0 \quad or \quad 1 \end{cases} \tag{1-10}$$

4.1.2 Model Solution Method

In real life, in order to achieve the goal of smooth roads after the snow stopped as soon as possible, the whole urban area should be divided into several small areas, and several small areas should be cleaned up at the same time. N intersections were rando mly selected as the initial center points in the large area, and Floyd algorithm was use d to calculate the shortest path length between each intersection and different center p oints. Then, cluster analysis method was used to determine the center point and the co rresponding area scope to make the objective function fully reach, so as to realize the reasonable division of the area. The algorithm idea is as follows:

Floyd algorithm:

Step 1: Given that each intersection point are $1,2,\cdots,N$, determine the matrix D_0, where element (i,j) is equal to the length of the shortest arc (if any) from vertex i to vertex j. If th ere is no such arc, then $d_{ij}^0 = \infty$. For i, let $d_{ii}^0 = 0$.

Step 2: apply the following recursive formula to the D_m element whose $m = 1,2,\cdots,N$ is determined by the elements of D_{m-1} in turn

$$d_{ij}^m = \min\left\{ d_{im}^{m-1} + d_{mj}^{m-1}, d_{ij}^{m-1} \right\} \tag{1-11}$$

Whenever an element is identified, write down the path it represents.

At the end of the algorithm, the elements (i,j) of the matrix D_n represent the shortest length from the intersection i to the intersection j.

Region division algorithm based on K-means clustering algorithm:

Step 1: in order to facilitate snow removal, the urban area should be divided into n small areas and cleaned up at the same time. and randomly initializes n central point, which is denoted as

$$u_1^t, u_2^t, \cdots, u_n^t \quad (t=0) \tag{1-12}$$

Among them: t is the number of iterative steps

Step 2: By calculating the distance between each point and the center point, each point is allocated to the nearest center point, forming N regions.

Step 3: Based on these classification points, consider the balance of snow removal task volume in each region at the same time, and change the area where the points that can greatly affect the balance of snow removal task volume in each region are located. The center point is recalculated by changing the mean of all points in the region.

Step 4: Set $t = 1, 2, 3, \ldots$ and repeat the above steps until no points are reallocated to different regions, and output the classification points contained in each region.

According to the model established above and the solution method, the region division diagram is drawn when n=10。

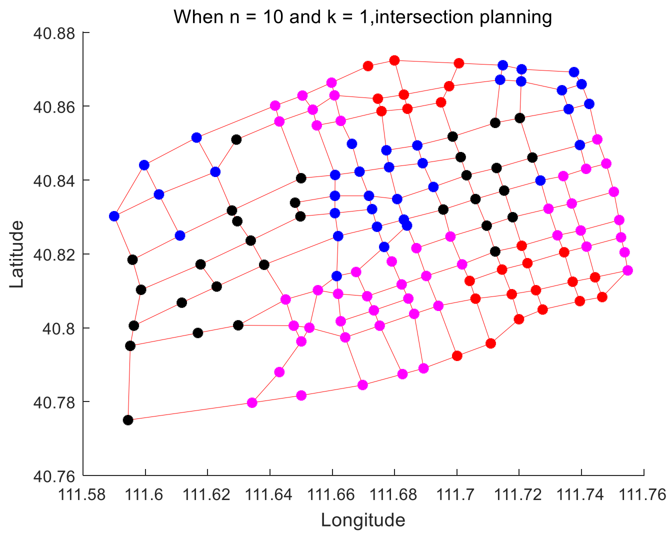


Figure1-1 Road and intersection images

From the figure above, it can be clearly seen that the approximate distribution of each region and most regions are regular graphs.

4.1.3 Route planning model for clean vehicles

According to the regional division model of clean vehicles established before, the regional division is realized reasonably. To minimize snow removal time, determine the best route plan for cleaning vehicles in any small area. The core of this problem is to consider the minimum working time of cleaning vehicles to complete snow removal tasks in this area and find an optimal path for snow removal.

According to the above analysis, as driving over the cleared road will reduce the efficiency of snow clearing, it is hoped that the cleaning car will spend as much time as possible in cleaning the road, and as little time as possible in the empty road.

Establishment of objective function:

The total working time of snow removal completed by the cleaning vehicle is the sum of snow removal time and empty travel time of the cleaning vehicle, i.e

$$T_w = T_c + T_d \tag{1-13}$$

T_c is the snow clearing time and T_d is the empty travel time.

The snow clearing time of the cleaning vehicle is:

$$T_c = \frac{\sum_{j=1}^{141} X_{ij} C_{ij}}{V_c L_w} \quad (i = 1, 2, \cdots, n) \tag{1-14}$$

V_c is the snow clearing speed within unit time, and L_w is the one-time snow clearing width of snowmobile.

The empty travel time of the cleaning vehicle is:

$$T_d = \frac{\sum d_f}{V_d} \tag{1-15}$$

V_d is the driving speed of empty distance in unit time, and d_f is the driving distance of each empty distance.

The total working time of the cleaning vehicle to complete snow removal is as small as possible, and since the total amount of snow removal by the cleaning vehicle is determined, that is, the time of snow removal by the cleaning vehicle is determined, the empty travel time of the cleaning vehicle is as small as possible, i.e

$$MIN \frac{\sum d_f}{V_d} \tag{1-16}$$

The basic constraints are as follows,

$$S.T \begin{cases} \sum_{i=1}^{n} X_{ij} = 1 \quad (j = 1, 2, \cdots, 141) \\ \sum_{j=1}^{141} \sum_{i=1}^{n} X_{ij} = \text{n} \\ X_{ij} = 0 \quad or \quad 1 \end{cases} \tag{1-17}$$

4.1.4 Model Solution Method

In the actual process, in order to minimize the working time of snow removal, the optimal route planning of cleaning vehicles in any small area should be determined to minimize the empty driving time of cleaning vehicles, that is, the sum of each empty driving distance should be minimized. To solve this model, the idea of oil route problem is used. The algorithm idea is as follows:

If the road runs in both directions, we simplify it to an undirected graph

$G=\{V \mid E\}$ in the case that the road route and the adjacent relationship between different intersections are known. Where, the intersection node is the node V of the undirected graph, the road is the edge E of the undirected graph, and the road length D is the weight of the edge of the undirected graph. For undirected graph G, its adjacency matrix is $A=\left(a_{ij}\right)_{v\times v}$

$$a_{ij}=\begin{cases} d_{ij}, & \left(v_i,v_j\right)\in E,, \\ 0, & i=j, \\ \infty, & \left(v_i,v_j\right)\notin E. \end{cases} \tag{1-18}$$

On the basis of the above undirected graph, a new undirected graph is drawn by determining the number of edges between two adjacent intersections by considering the ratio between the width of the road and the width of the snow clearing of the snowmobile at one time. In the new undirected graph, the degree of each vertex can be calculated to determine whether the graph is an Euler graph. If the degree of each vertex is even, the graph is Euler graph, and the Fleury algorithm is used to get Euler loop. Since euler loops pass through all edges, any euler loop is the solution to this problem. If the degree of each vertex in the graph is non-even, it is not an Euler graph, and some parallel edges are added, so that the new graph does not contain odd degree nodes, and the total weight of the added edges is minimum.

If G has only two singularities V_i, when V_j, there is an Euler trace from V_i to V_j, and when returning from V_j to V_i, some edges must be repeated to minimize the total length of the repeated edges, which is converted to find the shortest path from V_i to V_j. Algorithm:

（1）Find the shortest path P between the singularities;

(2)Make $G^{'}=P+G$; $G^{'}$ is eulerian graph, and The Eulerian circuit of $G^{'}$ is the optimal mail route.

In general, if the number of singularities is greater than 2, the path must repeat more edges.

1. Find the shortest path and distance between all singularities of G;

2. Take all singularities of G as nodes (which must be even Numbers), and take the shortest distance between them as the edge weight between nodes to obtain a complete graph G1;

3. The matching edge (V_i ,V_j) in M is written as the set E_{ij} of all edges through the

shortest path between V_i and V_j;

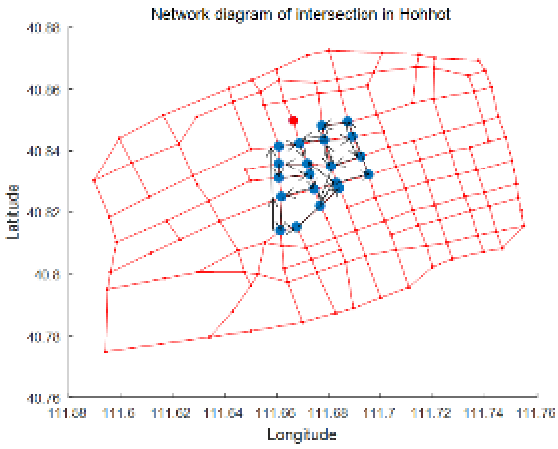
4. Let $G^{'} = G \uplus \{E_{ij} | (V_i, V_j) \in M\}$, $G^{'}$ is Eulerian graph, and find the optimal mail route.

Fleury algorithm[1]:

(1)For any $v_0 \in V(G)$, let $W_0 = v_0$.

(2)Suppose trace $W_i = v_0 e_1 v_1 e_2 \cdots e_i v_i$ is selected, then select edge e_{i+1} from $E - \{e_1, e_2, \cdots, e_i\}$ in the following way; e_{i+1} is associated with v_{i+1}; e_{i+1} cannot be the cut edge of $G_i = G - \{e_1, e_2, \cdots, e_i\}$ unless there is no other side to choose from.

When (2) cannot be executed, the algorithm stops.



According to the above established model and solution method, the route planning diagram in a certain region when n=10 is drawn.

4.2 Task 2: Distribution of human and material resources

4.2.1 Calculate the critical value of snow thickness

Suppose the average thickness of snow cover is h , then the amount of snow cover to be cleaned is

$$V_{all} = S_{all} \times h \qquad (2\text{-}1)$$

In the formula, S_{all} is the total area of road snow， V_{all} is the total volume of road snow cover.

Snow thickness there must be a two threshold, if the snow on the road to thickness is less than the first critical value, then the snow can be swept to the roadside green belts. If the thickness of snow on the road is greater than the first critical value and less than the second threshold, then the snow can't be swept to the roadside green belts, all the overflow of snow will be transported to a large-scale open space. If the thickness of snow on the road is greater than the second critical value,the overflow of snow will be sent to the outside of the city.

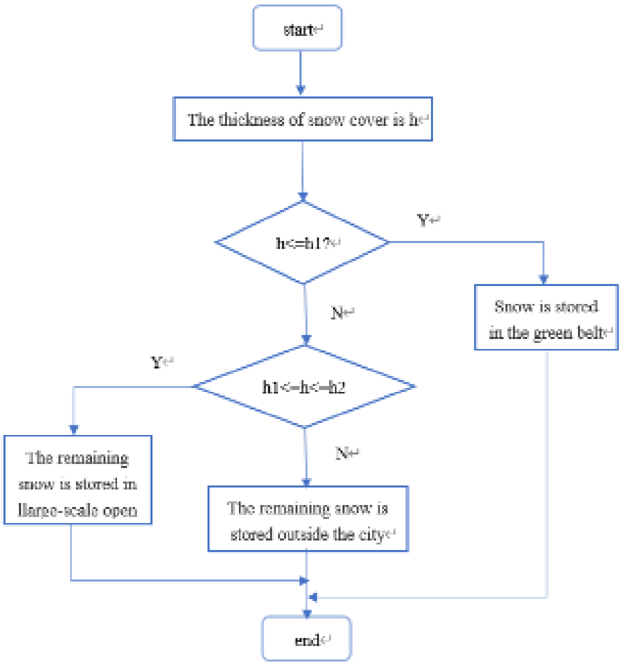


Figure 2-1 Flow chart for determining snow storage location based on critical thickness of snow cover

The amount of snow that can be stored in large-scale open spaces is the key to determining whether the snow on roads should be sent out of the city. Because the snow on the road is very loose, the snow after cleaning is not so loose, and the snow stored in the space will be artificially compressed, so the storage space can store a larger volume of snow. By comparing the density of loose snow with the density of ice pressed to the limit, the relationship between the volume of stored snow and the volume of snow on the road is as follows:

$$v=\frac{V'}{V}=\frac{\frac{\rho'}{m}}{\frac{\rho}{m}}=\frac{\rho'}{\rho}=\frac{0.1}{0.9}=\frac{1}{9} \qquad (2\text{-}2)$$

In fact, the compression of snow volume will not reach the limit, so we take $v=\frac{1}{3}$.Assuming that the snow storage height of large open space is h_{store}, the upper limit of the snow storage of the space is

$$V_{store\max}=S_{area}\times\frac{1}{v}\times h_{store} \qquad (2\text{-}3)$$

In conclusion, the model of critical value of snow thickness can be obtained, as shown below：

$$S.T.\begin{cases} V_{move} = 0, \quad 0 < h < h_1 \\ V_{store} = S_{all} \times (h - h_1), \quad h_1 < h < h_2 \\ V_{transport} = S_{all}(h - h_2), \quad h_2 < h \\ S_{area} = \sum_{j=1}^{30} S_j \\ S_{all} \times (h_2 - h_1) \leq V_{store\,max} \end{cases} \tag{2-4}$$

In the formula , V_{move} is the amount of snow that needs to be transported under the circumstance that the green belt can store snow. V_{store} is the amount of snow that cannot be contained in the green belt and needs to be transported to large open areas. $V_{transport}$ is the amount of snow that need to be transported outside the city, and S_{area} is the sum of area of all intersections where the snow can be stored.

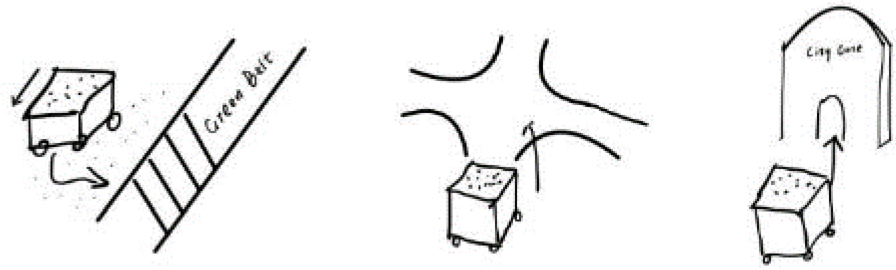


Figure 2-2 In three cases, different snow cover thickness corresponds to different treatment methods

The relationship between the two critical values can be obtained by this model:

$$h_2 = h_1 + \frac{S_{area} \times \frac{1}{v} \times h_{store}}{S_{all}} \tag{2-5}$$

4.2.2 The relationship between snow thickness and working time and manpower and material resources required

According to the data, in general, the snowplow used in the city can clear the snow thickness of the upper limit of 1.5 meters. Moreover, the thicker the snow is, the slower the snow sweeping speed of the snowplow is. In conclusion, the relationship between the snow sweeping speed of the snowplow and the snow thickness can be established,and the relationship is as follows:

$$V_{car} = 10 \times (1.5 - h) \tag{2-6}$$

Because snowplows can only clear one driveway at a time (the width of a driveway is 2 meters according to the information in the attachment), When there are many lanes on a road, many sweeps must be carried out on the same road.We can get the area that each snowplow can clear snow in unit time, so the area of snow clearing per unit time of n vehicles is:

$$s = 2nV_{car} \tag{2-7}$$

In the first question, the total area of road snow can be obtained as S_{all}, so the time it will take for n snowplows to clear all the road snow is

$$t = \frac{S_{all}}{s} \tag{2-8}$$

Given the upper limit t_h of expected snow clearing time, a set of data that exactly satisfies the condition $t \le t_h$ can be found out, and then the minimum vehicle used and the actual time of snow clearing can be obtained.

Therefore, the model of the relationship between the number of snowplows, the snow sweeping thickness and the upper limit time is as follows:

$$S.T.\begin{cases} V = 10 \times (1.5\text{-}h) \\ S = 2nV \\ t = \dfrac{S_{all}}{S} \\ t \le t_h \end{cases} \quad (0 < h < 1.5) \tag{2-9}$$

In the formula, V_{car} is the snow removal speed of a snowplow, h is the snow thickness, s is the area of snow removal per unit time of a snowplow, S_{all} is the total road snow removal area, t_h is the upper limit of a given snow removal time, and t is the actual time of snow removal.

As long as the time limit and the snow thickness are given, the number of plows used can be calculated, and then the work area assigned to each plowing machine can be calculated according to the first question.In the following analysis, we continue to solve the problem by turning the road workload into point workload.

In the formula, n is the number of snowplows actually used, n_1 is the actual number of transporters in use, n_2 is the actual number of sanitation workers dispatched, and t is the actual time it takes to clear the snow.

Find all the large open Spaces in the working area of a snowplow.Transport snow from each equivalent intersection to the nearest large open space in the area.

Transporters move snow from the closer intersections and then from the farther intersections until all the open space in the area is no longer able to hold any snow. So, the snow must be moved out of the city.

If snow can be stored in other nearby areas, it is necessary to compare the distance between the intersection and the outside of the city and the distance between the intersection and the large-scale open space in other areas, and choose a relatively close distance to send the snow to the destination.

According to the thickness of snow, the amount of snow cover that the whole city needs to transport can be calculated as

$$V_{store} = S_{all} \times (h - h_1), \quad h > h_1 \tag{2-10}$$

If there are n_1 transporters, then the volume to be transported for each transporter is

$$V = \frac{V_{store}}{n_1} \tag{2-11}$$

Each intersection has the nearest large open space, so the total distance needed to transport the excess snow from each intersection to the nearest large open space is

$$L_{all} = \sum l_{i,j} \tag{2-12}$$

The average distance traveled by each transport vehicle is

$$L = \frac{\sum l_{i,j}}{n_1} \tag{2-13}$$

Under normal circumstances, the driving speed of transport vehicles is $30\,km/h$, the capacity of a transport vehicle is $16\,m^3$, after compaction can be loaded with snow is $48\,m^3$.So we can figure out the volume of snow per truck per hour transport is

$$v = \frac{48}{\frac{L}{30}} = \frac{1440}{L} \tag{2-14}$$

So the time it takes n_1 cars to transport all the snow away is

$$t = \frac{V_{store}}{nv} \tag{2-15}$$

The upper limit of the time to transport snow is t_h', because it takes time for the snow sweeper to load the snow into the transport vehicle after sweeping the snow, so given this time difference t', make $t_h' = t_h + t'$, when the time spent is just enough to meet $t \le t_h'$, the minimum number of vehicles to be transported can be figured out.

Therefore, the relationship between the actual transport time, the number of transport vehicles and the snow cover thickness is modeled as follows:

$$S.T.\begin{cases} V_{store} = S_{all} \times (h - h_1), \quad h > h_1 \\ V = \dfrac{V_{store}}{n_1} \\ L = \dfrac{\sum l_{i,j}}{n_1} \\ v = \dfrac{48}{\frac{L}{30}} = \dfrac{1440}{L} \\ t = \dfrac{V_{store}}{nv} \\ t \le t_h^{'} \end{cases} \tag{2-16}$$

In the formula, V_{store} is the amount of snow that needs to be transported in the whole city, v is the volume of snow transported by each transport vehicle per hour, and $t_h^{'}$ is the upper limit of time for snow transported.

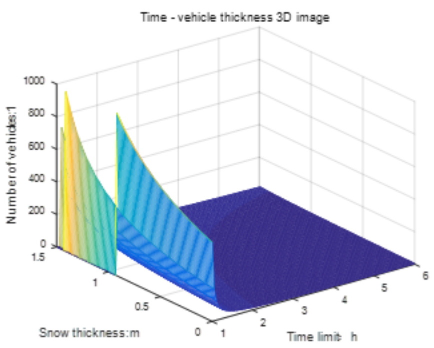
According to the information in the newspaper, the number of transport vehicles is in direct proportion to the number of sanitation workers, and the ratio of the two is about 3:250, so the number of sanitation workers can be obtained by this ratio.

$$n_2 = \frac{250}{3} \times n_1 \tag{2-17}$$

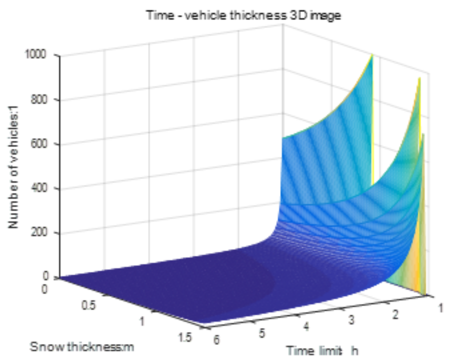
4.2.3 Solution of the model

When the snow cover on a given road exceeds 0.5m, a snowplow is used to clear the snow cover, that is, the first critical value of snow cover thickness is given.According to the first model, the critical value of the second snow thickness can be obtained. (h_1=0.5m h_2=0.713488m)

According to the second model, the relationship between the number of snow plows and the snow cover thickness and the upper limit time can be obtained, and the relationship between the actual working time and the snow cover thickness and the upper limit time can also be obtained.These relationships are represented by three-dimensional images, as shown in Figure 2-2 and 2-3.

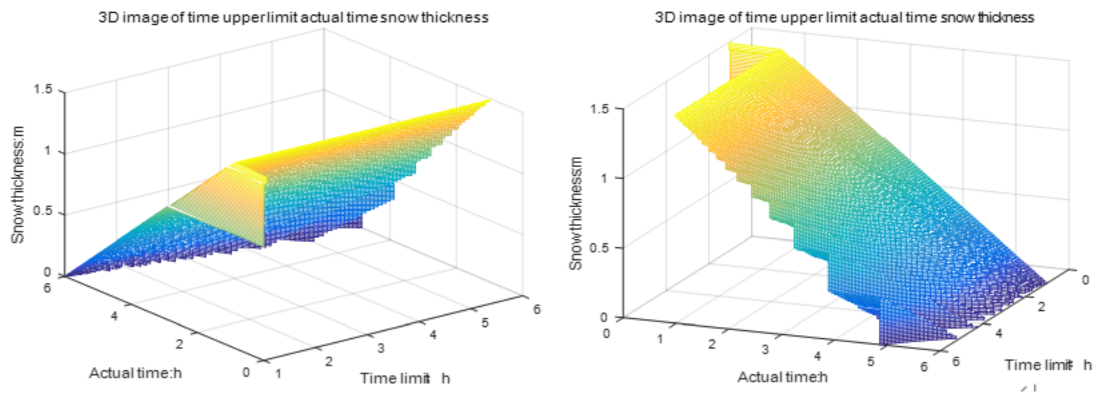


(a)



(b)

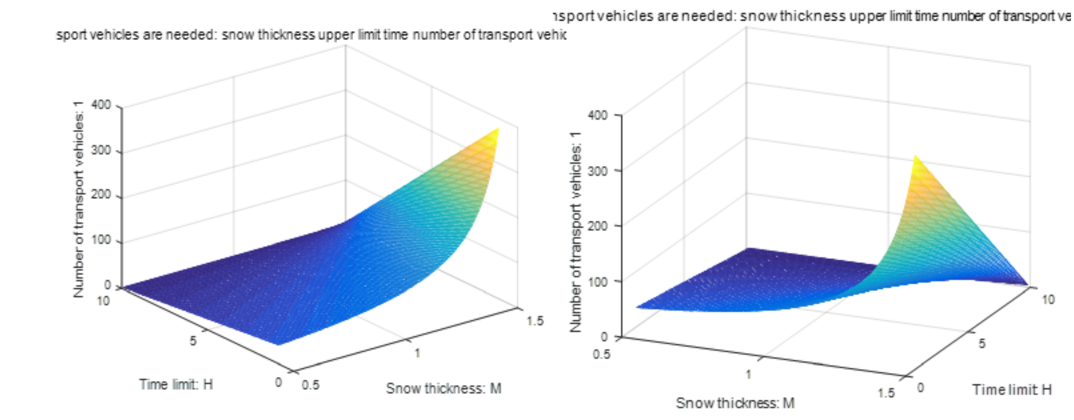
Figure2-3 Relationship between snowplow number and snow thickness and upper limit time



(a) (b)

Figure2-4 The relationship between actual time and snow cover thickness and upper limit time

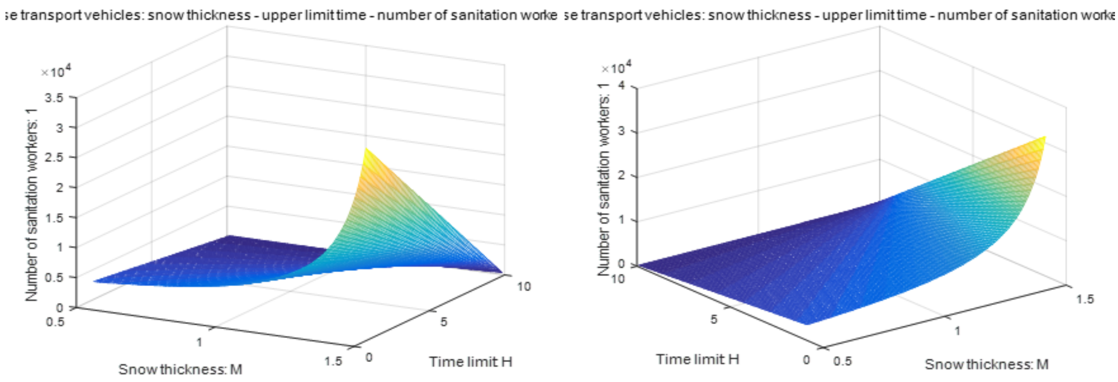
Under the condition of the upper limit time given by the second model, the relationship between the number of transport vehicles, the snow cover thickness and the upper limit time can be obtained according to the third model, which can be represented by three-dimensional images, as shown in Figure 2-4.



(a) (b)

Figure2-5 Relationship between the number of transport vehicles and snow cover thickness and upper limit time

According to the proportion, the relationship between the number of sanitation workers, snow thickness and time limit can be obtained, as shown in Figure 2-5.



(a) (b)

Figure2-6 The relationship between the number of sanitation workers and the snow thickness and the expected time limit

Example: When the snow cover on a given road exceeds 0.5m, a snowplow is used to clear snow, namely, the first critical value of snow cover thickness is given.According to the first model, the critical value of the second snow thickness can be obtained.Assuming that it takes one hour for a snowplow to load the snow into a transport vehicle after sweeping the snow, the data are shown in the table below:

Table 2-1 The relationship between all human resources and snow thickness and upper limit time (example)

The maximum time given (h)	Road snow thickness (m)	The actual time (h)	Minimum number of plows	Minimum number of carriers	Minimum number of sanitation workers
3	0.8	2.97	14	39	3250

4.3 Task 3: Clear vehicles and resume parking

4.3.1 The model of clearing road vehicles

According to some news, it is forbidden to stop on the road in winter when the road is cleared of snow.When parking interferes with snow clearing and drivers are not present, the traffic police department will forcibly remove the vehicle from the scene and store it in the designated place.So cars on the road will prolong the time it takes to clear the entire road.

According to the regional division model of clean vehicles established before, the regional division is realized reasonably. To minimize snow removal time, determine the best route plan for cleaning vehicles in any small area. The core of this problem is to consider the minimum working time of cleaning vehicles to complete snow removal tasks in this area and find an optimal path for snow removal.

According to the above analysis, as driving over the cleared road will reduce the efficiency of snow clearing, it is hoped that the cleaning car will spend as much time as possible in cleaning the road, and as little time as possible in the empty road.

Establishment of objective function:

The total working time of the cleaning vehicle for snow removal is the sum of the cleaning vehicle's snow removal time, empty travel time and the time of clearing vehicles on the road, i.e:

$$T_w = T_c + T_d + T_t \qquad (3\text{-}1)$$

In the formula, T_c is the snow clearing time and T_d is the empty travel time, T_t is the time to clear the vehicles on the road.

The snow clearing time of the cleaning vehicle is:

$$T_c = \frac{\sum_{j=1}^{141} X_{ij} C_{ij}}{V_c L_w} \quad (\ i = 1,\ 2, \cdots, n) \qquad (3\text{-}2)$$

In the formula, V_c is the snow clearing speed within unit time, and L_w i is the one-time clear snow width of the snowmobile.

The empty travel time of the cleaning vehicle is:

$$T_d = \frac{\sum d_f}{V_d} \qquad (3\text{-}3)$$

In the formula, V_d is the driving speed of empty distance in unit time, and d_f is the driving distance of each empty distance.

The study found that the time it takes to clear cars parked on the road is related to the snow thickness of the road, and the relationship is as follows:

$$T_d = kh \qquad (3\text{-}4)$$

In the formula, k is the proportion coefficient and h is the road snow cover thickness.

To keep the total working time of the cleaning vehicle to a minimum.And since the total amount of snow clearing by the cleaning vehicle is determined, that is, the time of snow clearing by the cleaning vehicle is determined, the time of empty journey of the cleaning vehicle and the time of clearing vehicles on the road is minimized, i.e

$$MIN(\frac{\sum d_f}{V_d} + kh) \qquad (3\text{-}5)$$

The basic constraints are as follows,

$$S.T.\begin{cases} \sum_{i=1}^{n} X_{ij} = 1 \quad (j = 1, 2, \cdots, 141) \\ \sum_{j=1}^{141} \sum_{i=1}^{n} X_{ij} = \text{n} \\ X_{ij} = 0 \quad or \quad 1 \end{cases} \qquad (3\text{-}6)$$

To sum up, under the condition of considering the clearing of vehicles on the road, the model of completing the clearing of vehicles on the road to restore the parking space as soon as possible is as follows:

$$S.T.\begin{cases}\sum_{i=1}^{n} X_{ij}=1 \quad (j=1,2,\cdots,141) \\ \sum_{j=1}^{141}\sum_{i=1}^{n} X_{ij}=\text{n} \\ X_{ij}=0 \quad or \quad 1 \\ MIN(\frac{\sum d_f}{V_d}+kh) \end{cases} \tag{3-7}$$

4.4 Task 4: Consider road priorities

4.4.1 Planning model based on different priority of road

In real life, as there are many overpasses in cities and the road slopes are different, each road will be affected by different snowfall weather. In order to minimize the negative impact of snowfall, priority is given to restoring traffic on heavily trafficked roads as well as on more important roads, so road clearance has different priorities. The core of the problem is to consider how to clean the routes cleared by vehicles within a certain period of time to maximize the number of roads with the highest priority, while the number of other priority roads is positively correlated with the priority.

According to the above analysis and combined with the regional division model established in the first question, the urban area was first divided into N sub-regions. When the cleaning time is fixed, the number of cleaning route schemes in any subregion is large, and the route scheme with the highest priority and the largest number of roads is selected. If the number of roads with the highest priority is the same, then compare the number of roads at the next priority level, and so on.

(1)Establishment of objective function:

Objective function a:

Whether the workload is balanced can be reflected by the standard deviation of snow removal area in each region. The smaller the standard deviation, the more balanced the workload.

Actual snow removal workload for the i region

$$S_i=\sum_{j=1}^{141} X_{ij}C_{ij} \quad (i=1,2,\cdots,n) \tag{4-1}$$

C_{ij} is the snow removal workload of the j intersection handled by the i area every day.

Average snow surface workload per region

$$\overline{S}=\frac{1}{n}\sum_{i=1}^{n} S_i \quad (i=1,2,\cdots,n) \tag{4-2}$$

The standard deviation of regional snow removal workload to be satisfied is as s mall as possible, i.e

$$MIN\sqrt{\frac{1}{n}\sum_{i=1}^{n} S_i-\overline{S}^{\ 2}} \quad (i=1, 2,\cdots,n) \tag{4-3}$$

Objective function b:

In order to compare the snow removal benefits of different schemes, the priority coefficient f_t is introduced, i.e

$$\sum_{k=1}^{6} m_k \quad \left(k=1,2,3,4,5,6\right) \tag{4-4}$$

m_k is the number of the corresponding priority roads, w_k is the weight of the corresponding priority roads and $w_{k+1} >> w_k$.

Within a certain cleaning time, the higher the priority coefficient, the more likely the scheme is to meet the target requirements, i.e

$$MAX\sum w_k m_k \tag{4-5}$$

The constraint condition is: the number of roads in any subregion is equal to the number of roads at each priority, i.e

$$\sum_{k=1}^{6} m_k = N \quad \left(k=1,2,3,4,5,6\right) \tag{4-6}$$

N is the number of roads in the subregion.

$$S.T\begin{cases}\sum_{i=1}^{n} X_{ij}=1 \quad \left(j=1,2,\cdots,141\right) \\ \sum_{k=1}^{6} m_k = N \quad \left(k=1,2,3,4,5,6\right) \\ \sum_{j=1}^{141}\sum_{i=1}^{n} X_{ij}=\text{n} \\ X_{ij}=0 \quad or \quad 1\end{cases} \tag{4-7}$$

4.4.2 Model Solution Method

For the above model, when cleaning time must be different solutions roadmap is different, thus can get a big probability through the intersection, and according to the actual road map to map the undirected graph, and using minimum spanning tree algorithm, a weight on the side of the picture with the road priority weights for the inverse relationship, generated a after each vertex and edge weight sum of the minimum tree, is the optimal solution of the model.

Minimum spanning tree[3]: in a given undirected graph G = (V, E), (u, v) on behalf of connected vertex u and v ($\left(u,v\right)\in E$), and w (u, v) represent the weight of the edge, if there are T as a subset of the E ($T \subseteq E$) and (V, T) as the tree, and

$w(T) = \sum_{(u,v)\in T} w(u,v)$ is the minimum, then this T for minimum spanning tree of G.

To solve this problem, the approximate road priority can be divided into three categories, you can get the corresponding road map.

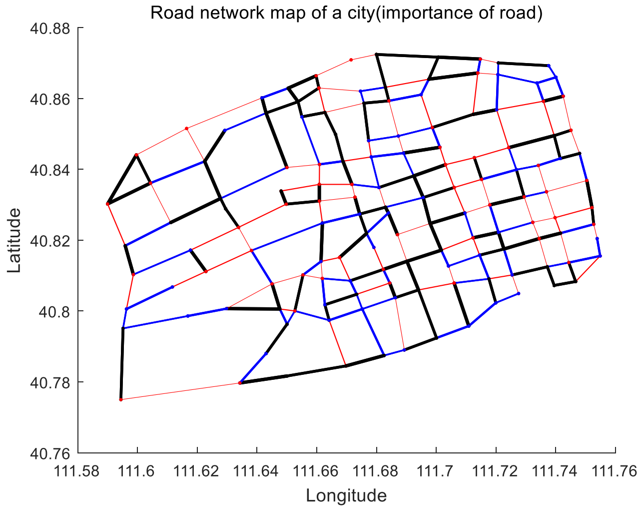


Figure 4-1 Distribution of different priority routes

Use the minimum spanning tree method to draw the following figure.

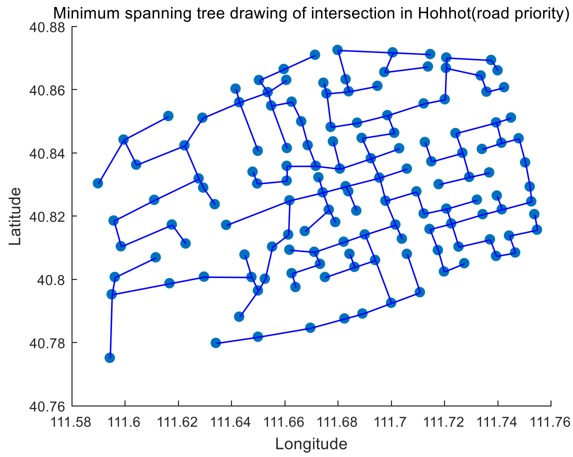


Figure 4-2 Route planning under minimum spanning tree method

The route plan is the cleaning route plan with a large priority coefficient.

5.Test the Models

For the model constructed in Question 4, the optimal snow removal plan that takes into account the priority level of the section can be obtained by bringing relevant road data into the model and drawing it into the road map. At the same time, the software is used to draw a road map for the road with higher priority, while the road with lower priority is not shown on the map. The rationality of the model is verified by comparing the degree of similarity between the two.

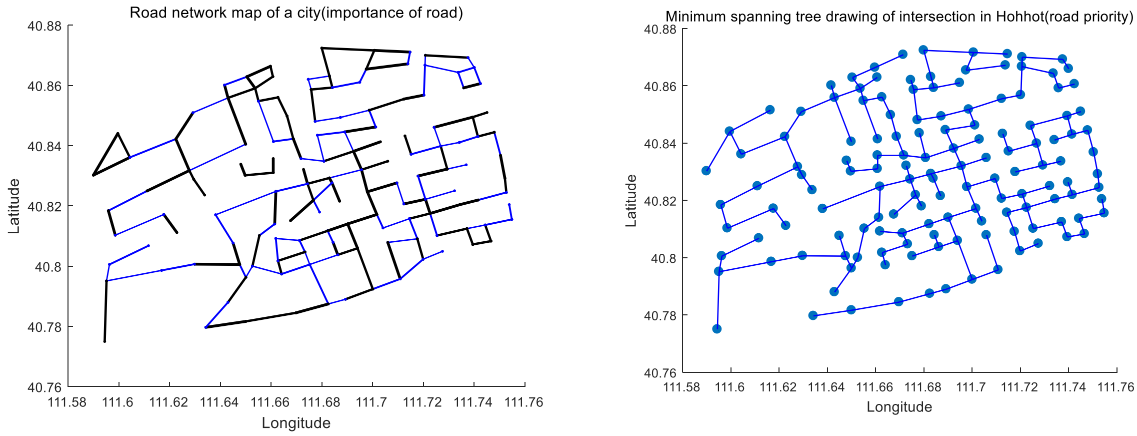


Figure 5-1 Comparison of similarity between the two figures

Through the observation of the two figures, it is found that the similarity between the two figures is relatively high, which proves the feasibility and rationality of the model.

6. Sensitivity Analysis

The global and local sensitive factors are found in the four models.

For the model we established in question 1, the area is divided into n small areas, and the division principle has volatility: 1. When the actual workload of each road is not completely equal to the workload of intersection, that is, there is error, we need to calculate the sensitivity coefficient and critical point to judge. The following attempts to solve the problem with dynamic balance algorithm: the threshold has fluctuation. The threshold values of five examples are given below, as well as the drawing of the first type of partition.

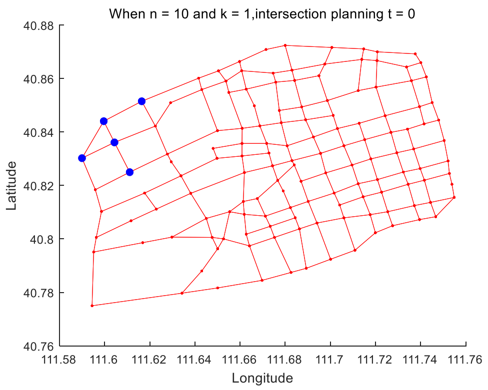


Figure 6-1 The threshold of this graph is t = 0

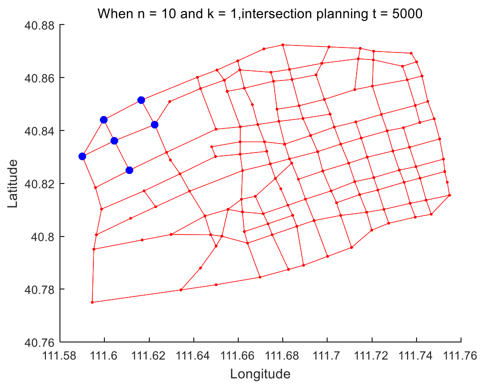


Figure 6-2 The threshold of this graph is t = 5000

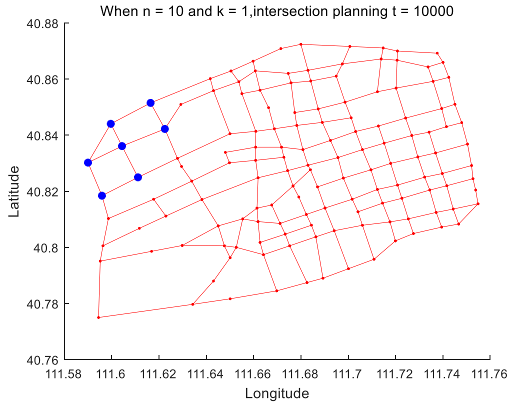


Figure 6-3 The threshold of this graph is t = 10000

In the first type of zoning, it can be seen that the threshold value with a difference of 10000 only affects 1-2 assigned intersection points, so it can not be judged that the influence factor of threshold value is low. Therefore, we draw the region division of the whole map as follows:

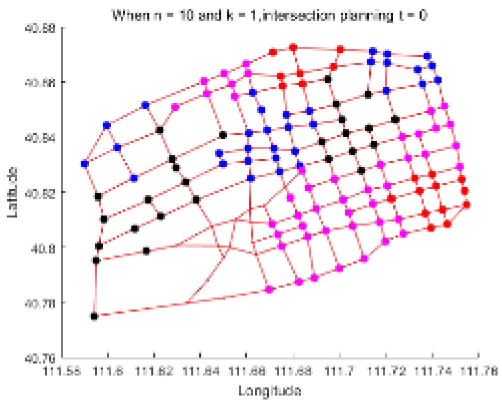


Figure 6-4

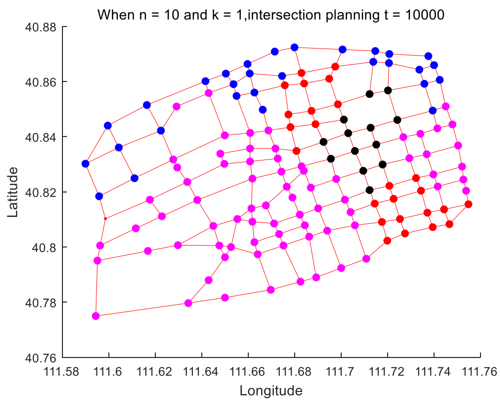


Figure 6-5

It is obvious that there is no threshold, which leads to the overflow of points, which leads to the problem of incomplete allocation. However, when the threshold is 10000. In the process of point allocation, there are some left over problems in the region. Because the threshold is too large, it leads to the uneven distribution of intersection points, so that the number of areas is not n. in the process of dynamic analysis, we constantly find a more appropriate threshold to make the workload of each area more average.

For the model we established in question 2, considering that the thickness of snow washed into the nearby green belt or leisure area is h, h is 0.5m in the calculation example. When the area of each snow storage point is known, the snow storage height is 2 meters and the compression amount is 3 times, the number of transport vehicles required is as follows:

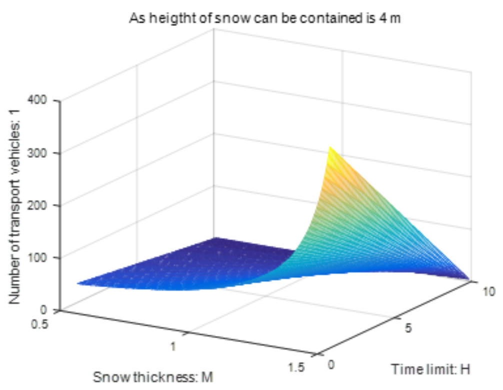


Figure 6-6

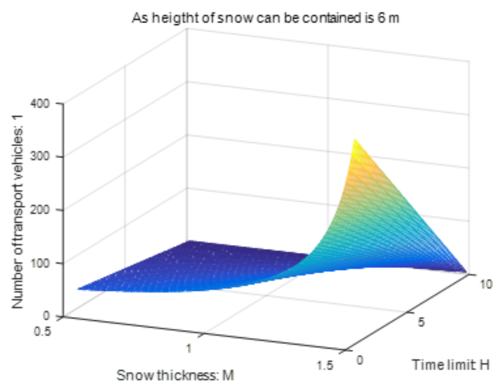


Figure 6-7

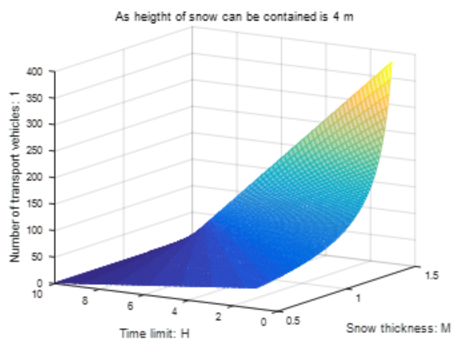


Figure 6-8

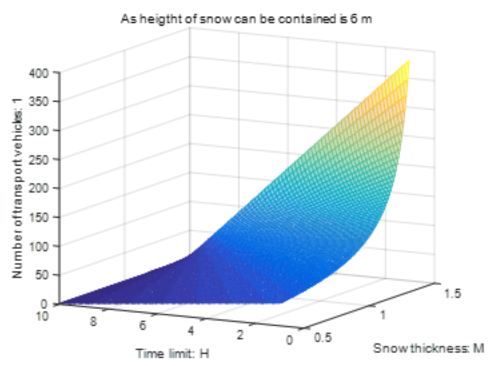


Figure 6-9

It is very obvious that the change of the height of snow storage leads to the change of the number of transport vehicles, and also directly affects the number of sanitation workers.

In the process of transportation, because the snow thickness directly affects the times and modes of transportation, when the snow thickness is 0.8 meters, we draw the workload image of each snow point and intersection point.

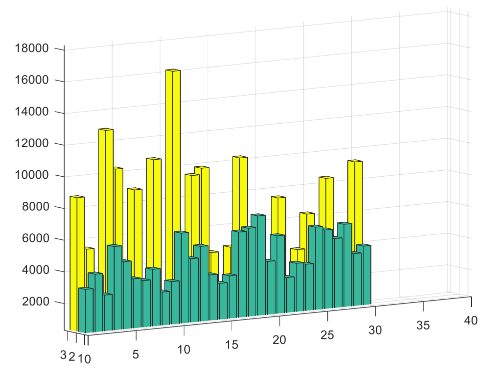
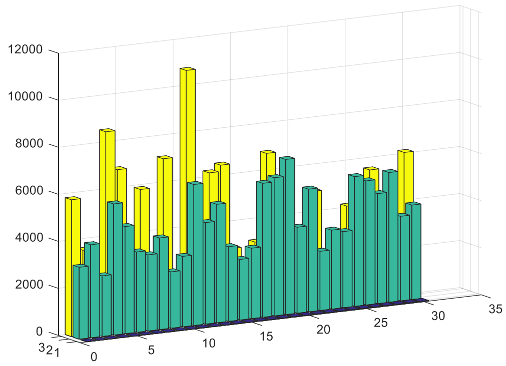


Figure 6-10

It can be seen from the image that basically all the snow points exceed the amount that can be stored, so it needs to be transported by flow, and even most of the snow needs to be transported to the outside of the city. Adjust the snow thickness, and when the height is the upper limit of the height that can be stored inside the city, the image is drawn as follows:



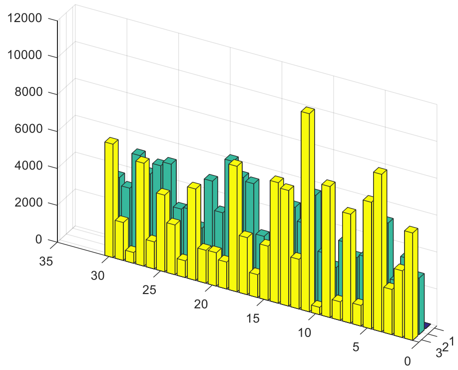


Figure 6-11 Figure 6-12

It is obvious from the figure that the circulation and transportation between snow spots is needed. Therefore, in the sensitivity analysis, we found the above sensitive factors and found the threshold which is more in line with the actual through dynamic analysis.

7.Strengths and Weakness

7.1 Strengths

- This paper considers the workload balance of each snowplow, which is closer to the real life.
- In this paper, the human and material resources required by different thickness of snow cover are considered and the relationship between them is found.

7.2 Weaknesses

- The problem of snow melting into water during transportation is not considered in this paper.
- In this paper, the calculated snow thickness is large, which is suitable for the case of heavy snow day, but not suitable for the case of especially little snow.

8.Conclusion

In this paper, through the above problem, we found the road snow removal is influenced by multiple factors. Under the premise that the snow removal time cannot be too long, not only the unnecessary driving distance of the snow removal vehicle should be minimized, but also the priority of the road and the road surface condition should be considered to make the best snow removal route plan. In the process of building the model, some simplified processing of the model will lead to slight deviation of the result. However, due to the irresistible factor of the actual situation, the probability of the uncertain result will increase. More consideration is given to the influence of uncertain factors in making the plan, which makes the plan more scientific and reasonable and ensures the smooth passage of citizens.

References

[1] Jiarong Shi,Matlab Program design and mathematical experiment and modeling,Xidian University Press,11-2019,page 162-165

[2] Guo Jiani Hu Jiuxiang Lu Zhengding,A Domain Splitting Algorithm for Parallel Grid Generation,J.Huazhong Univ. of Sci.&Tech.,Vol.27 No.7,Jul.1999

[3] ZHENG Jin-Hua and CAI Zi-Xing,RESTRICTED GENETIC ALGORITHM OF AREA SEARCHING BASED ON AUTOMATIC AREA PARTING,JOURNAL OF COMPUTER RESEARCH&DEVELOPMENT,Vol.37 No.4,Apr.2000

[4] Shuhe Wang,Fundamentals of mathematical modeling,page 11-13

Appendix

<< maltab version: 7.1(R2016b) >>

List of attachments:
Annex 1:Matlab main function code (file: main.m)
Annex 2:Matlab Floyd subfunction (file: Floyd.m)
Annex 3:Matlab Findline subfunction (file: findline.m)
Annex 4:Matlab Lontodistance subfunction code (file: lontodistance.m)
Annex 5:Matlab Prim subfunction code (file: prim.m)
Annex 6:C++ Fleury algorithm code (file: test.cpp)

Matlab main function code (file: main.m):

```
%% Drawing a network map of a city
map_node_data = load('excel1_node.txt');
map_link_data = load('excel1_link.txt');
map_location_data = load('excel1_location.txt');
figure(1) %Open figure 1 to draw the horizontal and vertical coordinates of intersection points in a city
for i=1:length(map_node_data(:,1))
    plot(map_node_data(i,2),map_node_data(i,3),'b*','Linewidth',1)
    hold on
end
title('Scatter diagram of intersection in Hohhot')
xlabel('Longitude')
ylabel('Latitude')
clear i
%% figure(2)Draw the connection of the intersection（A city is connected by roads）
figure(2)
for i=1:length(map_link_data(:,1))
    number_1 = map_link_data(i,1);
    number_2 = map_link_data(i,2);
    width = map_link_data(i,3);
    for j=1:length(map_node_data(:,1))
        if number_1 == map_node_data(j,1)
            number_1_x = map_node_data(j,2);
            number_1_y = map_node_data(j,3);
        end
    end
    for k=1:length(map_node_data(:,1))
        if number_2 == map_node_data(k,1)
            number_2_x = map_node_data(k,2);
            number_2_y = map_node_data(k,3);
        end
    end
    width_drop = width/3;
    plot([number_1_x number_2_x],[number_1_y number_2_y],'r.','Linewidth',10,'LineStyle','-
','Linewidth',width_drop)
    hold on
end
xlabel('Longitude')
ylabel('Latitude')
```

```
title('Huhhot intersection network map')
clear i j k number_1_x number_1_y number_2_x number_2_y width number_1 number_2 width_drop
%% Drawing the network map of streets with different colors
figure(3)
%The width of purple road is 2
%The width of red road is 4
%The width of blue road is 6
%The width of black road is 8
hold on
for i=1:length(map_link_data(:,1))
    number_1 = map_link_data(i,1);
    number_2 = map_link_data(i,2);
    width = map_link_data(i,3);
    for j=1:length(map_node_data(:,1))
        if number_1 == map_node_data(j,1)
            number_1_x = map_node_data(j,2);
            number_1_y = map_node_data(j,3);
        end
    end
    for k=1:length(map_node_data(:,1))
        if number_2 == map_node_data(k,1)
            number_2_x = map_node_data(k,2);
            number_2_y = map_node_data(k,3);
        end
    end
    if width == 2
        plot([number_1_x number_2_x],[number_1_y number_2_y],'m.','Linewidth',10,'LineStyle','-
','Linewidth',0.1)
    elseif width == 4
        plot([number_1_x number_2_x],[number_1_y number_2_y],'r.','Linewidth',10,'LineStyle','-
','Linewidth',0.1)
    elseif width == 6
        plot([number_1_x number_2_x],[number_1_y number_2_y],'b.','Linewidth',10,'LineStyle','-
','Linewidth',0.1)
    elseif width == 8
        plot([number_1_x number_2_x],[number_1_y number_2_y],'k.','Linewidth',10,'LineStyle','-
','Linewidth',0.1)
    else
        continue
    end
end
xlabel('Longitude')
ylabel('Latitude')
title('Huhhot intersection network map(different road width)')
clear i j k number_1_x number_1_y number_2_x number_2_y width number_1 number_2
%% Potential snow locations are found in Figure 4
figure(4)
hold on
for i=1:length(map_link_data(:,1))
    number_1 = map_link_data(i,1);
    number_2 = map_link_data(i,2);
    width = map_link_data(i,3);
    for j=1:length(map_node_data(:,1))
        if number_1 == map_node_data(j,1)
            number_1_x = map_node_data(j,2);
            number_1_y = map_node_data(j,3);
        end
    end
    for k=1:length(map_node_data(:,1))
        if number_2 == map_node_data(k,1)
```

```
                number_2_x = map_node_data(k,2);
                number_2_y = map_node_data(k,3);
            end
        end
        if width == 2
            plot([number_1_x number_2_x],[number_1_y number_2_y],'m.','Linewidth',10,'LineStyle','-
','Linewidth',0.1)
        elseif width == 4
            plot([number_1_x number_2_x],[number_1_y number_2_y],'r.','Linewidth',10,'LineStyle','-
','Linewidth',0.1)
        elseif width == 6
            plot([number_1_x number_2_x],[number_1_y number_2_y],'b.','Linewidth',10,'LineStyle','-
','Linewidth',0.1)
        elseif width == 8
            plot([number_1_x number_2_x],[number_1_y number_2_y],'k.','Linewidth',10,'LineStyle','-
','Linewidth',0.1)
        else
            continue
        end
end
xlabel('Longitude')
ylabel('Latitude')
title('Network map of a city intersection(different road width),potential snow location')
clear i j k number_1_x number_1_y number_2_x number_2_y width number_1 number_2
for i = 1:length(map_location_data(:,1))
        plot(map_location_data(i,1),map_location_data(i,2),'b+','Linewidth',2)
end
clear i
k=0;
matrix_of_Potential_snow = [];%Establish a matrix to store the label, longitude, latitude and snow area
of potential snow cover location
for i = 1:length(map_location_data(:,1))
        location_x = map_location_data(i,1);
        location_y = map_location_data(i,2);
        area_space = map_location_data(i,3);
        for j = 1:length(map_node_data(:,1))
            if (location_x == map_node_data(j,2)) && (location_y == map_node_data(j,3))
                number = map_node_data(j,1);
                fprintf('The location of the intersection with potential snow cover is
obtained:%d\n',number)
                fprintf('\t\tThe location of the intersection is:%f\n',location_x)
                fprintf('\t\tThe location of the intersection is:%f\n',location_y)
                k = k + 1;
                matrix_of_Potential_snow(i,1) = number;
                matrix_of_Potential_snow(i,2) = location_x;
                matrix_of_Potential_snow(i,3) = location_y;
                matrix_of_Potential_snow(i,4) = area_space;
            end
        end
end
fprintf('There are %d potential snow locations \n',k)
clear i j k location_x location_y area_space number
save('OUTPUT_Potential_snow_information.txt','matrix_of_Potential_snow','-ascii');
%% The distance between two adjacent points is calculated by longitude and latitude, and stored in the
141 * 141 matrix(only the distance directly connected)is obtained
matrix_of_linkdistance = inf(141,141);%A 141 * 141 matrix is established to store the distance
between any two points
for i =1:length(map_link_data(:,1))
        position_number_1 = map_link_data(i,1);%The first column of each row in the link is the starting
point
```

```
	position_number_2 = map_link_data(i,2);%The second column of each row in the link is the destination
	position_number_1_Lon = map_node_data(position_number_1,2);%Longitude of starting point
	position_number_1_Lat = map_node_data(position_number_1,3);%Latitude of starting point
	position_number_2_Lon = map_node_data(position_number_2,2);%Longitude of the end point
	position_number_2_Lat = map_node_data(position_number_2,3);%Latitude of the end point
	dis = 
Lontodistance(position_number_1_Lon,position_number_1_Lat,position_number_2_Lon,position_number_2_Lat);%The actual distance between the two points is calculated
	matrix_of_linkdistance(position_number_1,position_number_2) = dis;%Matrix stored in the distance between any two points
	matrix_of_linkdistance(position_number_2,position_number_1) = dis;
end
for j = 1:length(matrix_of_linkdistance(:,1))
	matrix_of_linkdistance(j,j) = 0;
end
fprintf('\nhe actual distance between two adjacent points is calculated successfully!!!\n\n')
clear i j position_number_1 position_number_2 position_number_1_Lon position_number_1_Lat position_number_2_Lon position_number_2_Lat dis
%% The distance between any two points by using Floyd algorithm through direct matrix(distance unit is meter(m))
[D,path] = Floyd(matrix_of_linkdistance);
matrix_of_point_linkdistance = D;%Get the actual distance between any two points
matrix_of_path_linkdistance = path;%route
fprintf('Floyd algorithm is successfully used to obtain the actual distance between any two points!!!\n\n')
clear D path
%% Then a matrix of workload is established;the workload between any two points is stored, and the length of the road is multiplied by the width of the road
matrix_of_linkworktime = inf(141,141);%Build a matrix of workload
matrix_of_linkworktime = matrix_of_linkdistance;
for i = 1:length(matrix_of_linkworktime(:,1))
	position_number_1 = map_link_data(i,1);%The first column of each row in the link is the starting point
	position_number_2 = map_link_data(i,2);%The second column of each row in the link is the destination
	worktime = map_link_data(i,3);%Workload of two adjacent points
	matrix_of_linkworktime(position_number_1,position_number_2) = worktime * 
matrix_of_linkworktime(position_number_1,position_number_2);%Road length times road width
	matrix_of_linkworktime(position_number_2,position_number_1) = 
matrix_of_linkworktime(position_number_1,position_number_2);
end
fprintf('The workload between two adjacent points is calculated successfully!!!\n\n\tMethods adopted:Road length times road width\tGet the road area\tRoad area is the workload of snow clearing!!!\n\n')
clear i position_number_1 position_number_2 worktime
%% The same method is used to get the workload between any two points
[D,path] = Floyd(matrix_of_linkworktime);
matrix_of_point_linkworktime = D;%Get the workload between any two points
matrix_of_path_linkworktime = path;% route
fprintf('Floyd algorithm is successfully used to calculate the workload between any two points!!!\n\n')
clear D path
%% Prim algorithm, calculate the minimum spanning tree, and draw the minimum number of spanning in Figure 5
[Edge,weight] = Prim(matrix_of_linkworktime);%Calling the prim function
figure(5)
xlabel('Longitude')
ylabel('Latitude')
title('Minimum spanning tree drawing of intersection in Hohhot')
hold on
```

```
scatter(map_node_data(:,2),map_node_data(:,3),50,'filled')
hold on
for i = 1:size(Edge,2)
	plot(map_node_data(Edge(:,i),2),map_node_data(Edge(:,i),3),'r-','Linewidth',1)
end
fprintf('Successful minimum spanning tree drawing!!!\n')
%% Draw intersection scatter
figure(6)
hold on
for i = 1:length(map_node_data(:,1))
	plot(map_node_data(i,2),map_node_data(i,3),'r.','Linewidth',1)
end
clear i
%% Calculate total workload
worktime_all = 0;
for i = 1:length(matrix_of_linkworktime(:,1))
	for j = i:length(matrix_of_linkworktime(:,1))
		if matrix_of_linkworktime(i,j) ~= inf
			worktime_all = worktime_all + matrix_of_linkworktime(i,j);
		end
	end
end
fprintf('\nThe total workload is:%f square meter\n\n',worktime_all)
clear i j
%% Calculate the workload of 1 minute for each equipment
width_car = 2;%It is assumed that the transverse distance that each machine can sweep is 2m
speed_car_1 = 30;%The speed of each machine is assumed to be 30 km / h without snow sweeping
height_of_snow = 0.5;%Suppose the thickness of the snow
fprintf('\nWhen the thickness of snow is:%f\n',height_of_snow)
speed_car_2 = 10*(1.5 - height_of_snow);%The speed of each snow sweeper varies with the thickness
of snow
fprintf('At this time, the speed of the snow sweeper is%f\n',speed_car_2)
disp('The relationship between the speed of snow sweeping equipment and the change of snow
thickness:Vs = 10*(1.5 - h)')
fprintf('\t\tThe unit of snow sweeping speed is:km/h\tThe thickness of snow is:m\n\n')
car_snow_area = (speed_car_2/60) * width_car * 1000;%The area that the snow sweeper can clear per
minute
number_of_car = 10;
time_to_clear_snow = (worktime_all / (car_snow_area * number_of_car)) / 60;
fprintf('It is assumed that the snow cleaning workload of each equipment is approximately the
same\n\n')
fprintf('When the number of snow sweepers is： %d \t The total time for clearing snow is as follows:%f
hour\n\n',number_of_car,time_to_clear_snow)
fprintf('The workload of each snow sweeper is obtained as follows:%f\n',worktime_all/number_of_car)
%% alculate the weight of each point: the sum of the weights of all directly connected connecting lines
divided by 2
map_node_data_worktime = map_node_data;
map_node_data_worktime(:,4) = zeros(length(map_node_data(:,1)),1);%The fourth column is used to
store the weight of each point
for i = 1:length(matrix_of_linkworktime(:,1))
	for j = 1:length(matrix_of_linkworktime(1,:))
		if matrix_of_linkworktime(i,j) ~= inf
			map_node_data_worktime(i,4) = map_node_data_worktime(i,4) +
matrix_of_linkworktime(i,j);
		end
	end
	map_node_data_worktime(i,4) = map_node_data_worktime(i,4) / 2;%Weight of each intersection
point
end
clear i j
```

```
sum(map_node_data_worktime(:,4))
if sum(map_node_data_worktime(:,4)) == worktime_all
    fprintf('\nThe weight of each point is calculated as correct!!!\n\t Calculation method: divide the sum of weights of directly connected straight lines by 2\n\n')
else
    fprintf('\nalculation error!!!\n')
end
%% Zoning The first snow sweeper
worktime_all_per_car = worktime_all / number_of_car;
%Set threshold time
time = 5000;%Set the threshold to 5000
worktime_all_per_car_time = worktime_all_per_car + time;
fprintf('The upper limit of the workload of each snow sweeper is given as follows:%f\n\n\n',worktime_all_per_car_time)
figure(6)
xlabel('Longitude')
ylabel('Latitude')
title('Zoning (snow sweeper Regional Planning)')
figure(7)%Draw the intersection of Hohhot City on Figure 7
xlabel('Longitude')
ylabel('Latitude')
title('When n = 10 and k = 1,intersection planning')
hold on
for i=1:length(map_link_data(:,1))
    number_1 = map_link_data(i,1);
    number_2 = map_link_data(i,2);
    width = map_link_data(i,3);
    for j=1:length(map_node_data(:,1))
        if number_1 == map_node_data(j,1)
            number_1_x = map_node_data(j,2);
            number_1_y = map_node_data(j,3);
        end
    end
    for k=1:length(map_node_data(:,1))
        if number_2 == map_node_data(k,1)
            number_2_x = map_node_data(k,2);
            number_2_y = map_node_data(k,3);
        end
    end
    plot([number_1_x number_2_x],[number_1_y number_2_y],'r.','Linewidth',10,'LineStyle','-','Linewidth',0.1)
end
plot(map_node_data(1,2),map_node_data(1,3),'b+')%Determine this point as the boundary point
%% Divide the first area
number1_worktime_all = 0;%The amount of work needed to store the first snow sweeper
number1_worktime_point = [];%The label used to store the working intersection of the first snow sweeper
matrix_of_point_linkdistance_new = matrix_of_point_linkdistance;%A matrix containing the distance of a labeled point
matrix_of_point_linkdistance_number1 = matrix_of_point_linkdistance_new(1,:);%The first row is assigned a new matrix to hold a used intersection
map_node_data_worktime_new = map_node_data_worktime;%Set up a workload matrix for storing labeled points
fprintf('\n\n-----------------Scope planning of the first snow sweeper:-----------------\n\n')
while number1_worktime_all <= worktime_all_per_car_time
    [number,position] = min(matrix_of_point_linkdistance_number1);
    if number1_worktime_all + map_node_data_worktime_new(position,4)<= worktime_all_per_car_time
        fprintf('The shortest distance found is:%f \t And the label of the intersection is:%d\n',number,position)
```

```
            matrix_of_point_linkdistance_number1(position) = inf;%After using the intersection, it is
assigned to infinity
            number1_worktime_point = [number1_worktime_point,position];
            %Calculate the total workload of the first snow sweeper
            number1_worktime_all = number1_worktime_all +
map_node_data_worktime_new(position,4);
            fprintf('\t\tThe workload of the intersection found is as
follows:%f\n',map_node_data_worktime_new(position,4))
            fprintf('\t\tAnd the total workload of the first vehicle at this time
is:%f\n\n',number1_worktime_all)
            map_node_data_worktime_new(position,4) = inf;%The workload of the intersection is
assigned to infinity to prevent it from being used again in future planning
            matrix_of_point_linkdistance_new(position,:) =
inf(1,length(matrix_of_point_linkdistance_new(1,:)));
            matrix_of_point_linkdistance_new(:,position) =
inf(length(matrix_of_point_linkdistance_new(1,:)),1);
      else
            break
      end
end
clear number position
figure(7)
hold on
for i = 1:length(number1_worktime_point)

plot(map_node_data(number1_worktime_point(i),2),map_node_data(number1_worktime_point(i),3),'b
*','Linewidth',2)
end
clear i
%% Second snow sweeper
%Divide the second area
number2_worktime_all = 0;
number2_worktime_point = [];
number2 = 4;
matrix_of_point_linkdistance_number2 = matrix_of_point_linkdistance_new(number2,:);
fprintf('\n\n-----------------Scope planning of the second snow sweeper:-----------------\n\n')
while number2_worktime_all <= worktime_all_per_car_time
      [number,position] = min(matrix_of_point_linkdistance_number2);
      if number2_worktime_all + map_node_data_worktime_new(position,4)<=
worktime_all_per_car_time
            fprintf('The shortest distance found is:%f \t And the label of the intersection
is:%d\n',number,position)
            matrix_of_point_linkdistance_number2(position) = inf;
            number2_worktime_point = [number2_worktime_point,position];
            number2_worktime_all = number2_worktime_all +
map_node_data_worktime_new(position,4);
            fprintf('\t\tThe workload of the intersection found is as
follows:%f\n',map_node_data_worktime_new(position,4))
            fprintf('\t\tThe total workload of the second vehicle at this time is as
follows:%f\n\n',number2_worktime_all)
            map_node_data_worktime_new(position,4) = inf;
            matrix_of_point_linkdistance_new(position,:) =
inf(1,length(matrix_of_point_linkdistance_new(1,:)));
            matrix_of_point_linkdistance_new(:,position) =
inf(length(matrix_of_point_linkdistance_new(1,:)),1);
      else
            break
      end
end
clear number position
```

```
figure(7)
hold on
for i = 1:length(number2_worktime_point)

plot(map_node_data(number2_worktime_point(i),2),map_node_data(number2_worktime_point(i),3),'k
*','Linewidth',2)
end
clear i
%% The third snow sweeper
number3_worktime_all = 0;
number3_worktime_point = [];
for i = 1:length(matrix_of_point_linkdistance_new(:,1))
    if matrix_of_point_linkdistance_new(i,i) == 0
        number3 = i;
        break
    end
end
clear i
%%
matrix_of_point_linkdistance_number3 = matrix_of_point_linkdistance_new(number3,:);
fprintf('\n\n-----------------Scope planning of the third snow sweeper:-----------------\n\n')
while number3_worktime_all <= worktime_all_per_car_time
    [number,position] = min(matrix_of_point_linkdistance_number3);
    if number3_worktime_all + map_node_data_worktime_new(position,4)<=
worktime_all_per_car_time
        fprintf('The shortest distance found is:%f \t And the label of the intersection
is:%d\n',number,position)
        matrix_of_point_linkdistance_number3(position) = inf;
        number3_worktime_point = [number3_worktime_point,position];
        number3_worktime_all = number3_worktime_all +
map_node_data_worktime_new(position,4);
        fprintf('\t\tThe workload of the intersection found is as
follows:%f\n',map_node_data_worktime_new(position,4))
        fprintf('\t\tThe total workload of the third vehicle at this time is as
follows:%f\n\n',number3_worktime_all)
        map_node_data_worktime_new(position,4) = inf;
        matrix_of_point_linkdistance_new(position,:) =
inf(1,length(matrix_of_point_linkdistance_new(1,:)));
        matrix_of_point_linkdistance_new(:,position) =
inf(length(matrix_of_point_linkdistance_new(1,:)),1);
    else
        break
    end
end
clear number position
figure(7)
hold on
for i = 1:length(number3_worktime_point)

plot(map_node_data(number3_worktime_point(i),2),map_node_data(number3_worktime_point(i),3),'
m*','Linewidth',2)
end
clear i
%% The fourth snow sweeper
number4_worktime_all = 0;
number4_worktime_point = [];
for i = 1:length(matrix_of_point_linkdistance_new(:,1))
    if matrix_of_point_linkdistance_new(i,i) == 0
        number4 = i;
        break
```

```
        end
end
clear i
%%
matrix_of_point_linkdistance_number4 = matrix_of_point_linkdistance_new(number4,:);
fprintf('\n\n----------------Scope planning of the fourth snow sweeper:----------------\n\n')
while number4_worktime_all <= worktime_all_per_car_time
        [number,position] = min(matrix_of_point_linkdistance_number4);
        if number4_worktime_all + map_node_data_worktime_new(position,4)<=
worktime_all_per_car_time
                fprintf('The shortest distance found is:%f \t And the label of the intersection
is:%d\n',number,position)
                matrix_of_point_linkdistance_number4(position) = inf;
                number4_worktime_point = [number4_worktime_point,position];
                number4_worktime_all = number4_worktime_all +
map_node_data_worktime_new(position,4);
                fprintf('\t\tThe workload of the intersection found is as
follows:%f\n',map_node_data_worktime_new(position,4))
                fprintf('\t\tAnd the total workload of the fourth vehicle at this time
is:%f\n\n',number4_worktime_all)
                map_node_data_worktime_new(position,4) = inf;
                matrix_of_point_linkdistance_new(position,:) =
inf(1,length(matrix_of_point_linkdistance_new(1,:)));
                matrix_of_point_linkdistance_new(:,position) =
inf(length(matrix_of_point_linkdistance_new(1,:)),1);
        else
                break
        end
end
clear number position
figure(7)
hold on
for i = 1:length(number4_worktime_point)

plot(map_node_data(number4_worktime_point(i),2),map_node_data(number4_worktime_point(i),3),'b
*','Linewidth',2)
end
clear i
%% The fifth snow sweeper
number5_worktime_all = 0;
number5_worktime_point = [];
for i = 1:length(matrix_of_point_linkdistance_new(:,1))
        if matrix_of_point_linkdistance_new(i,i) == 0
                number5 = i;
                break
        end
end
clear i
%%
matrix_of_point_linkdistance_number5 = matrix_of_point_linkdistance_new(number5,:);
fprintf('\n\n----------------Scope planning of the fifth snow sweeper:----------------\n\n')
while number5_worktime_all <= worktime_all_per_car_time
        [number,position] = min(matrix_of_point_linkdistance_number5);
        if number5_worktime_all + map_node_data_worktime_new(position,4)<=
worktime_all_per_car_time
                fprintf('The shortest distance found is:%f \t And the label of the intersection
is:%d\n',number,position)
                matrix_of_point_linkdistance_number5(position) = inf;
                number5_worktime_point = [number5_worktime_point,position];
                number5_worktime_all = number5_worktime_all +
```

```
map_node_data_worktime_new(position,4);
            fprintf('\t\tThe workload of the intersection found is as
follows:%f\n',map_node_data_worktime_new(position,4))
            fprintf('\t\tThe total workload of the fifth vehicle at this time is as
follows:%f\n\n',number5_worktime_all)
            map_node_data_worktime_new(position,4) = inf;
            matrix_of_point_linkdistance_new(position,:) =
inf(1,length(matrix_of_point_linkdistance_new(1,:)));
            matrix_of_point_linkdistance_new(:,position) =
inf(length(matrix_of_point_linkdistance_new(1,:)),1);
      else
            break
      end
end
clear number position
figure(7)
hold on
for i = 1:length(number5_worktime_point)

plot(map_node_data(number5_worktime_point(i),2),map_node_data(number5_worktime_point(i),3),'r
*','Linewidth',2)
end
clear i
%% The sixth snow sweeper
number6_worktime_all = 0;
number6_worktime_point = [];
for i = 1:length(matrix_of_point_linkdistance_new(:,1))
      if matrix_of_point_linkdistance_new(i,i) == 0
            number6 = i;
            break
      end
end
clear i
%%
matrix_of_point_linkdistance_number6 = matrix_of_point_linkdistance_new(number6,:);
fprintf('\n\n-----------------Scope planning of the sixth snow sweeper:-----------------\n\n')
while number6_worktime_all <= worktime_all_per_car_time
      [number,position] = min(matrix_of_point_linkdistance_number6);
      if number6_worktime_all + map_node_data_worktime_new(position,4)<=
worktime_all_per_car_time
            fprintf('The shortest distance found is:%f \t And the label of the intersection
is:%d\n',number,position)
            matrix_of_point_linkdistance_number6(position) = inf;
            number6_worktime_point = [number6_worktime_point,position];
            number6_worktime_all = number6_worktime_all +
map_node_data_worktime_new(position,4);
            fprintf('\t\tThe workload of the intersection found is as
follows:%f\n',map_node_data_worktime_new(position,4))
            fprintf('\t\tThe total workload of the sixth vehicle at this time is as
follows:%f\n\n',number6_worktime_all)
            map_node_data_worktime_new(position,4) = inf;
            matrix_of_point_linkdistance_new(position,:) =
inf(1,length(matrix_of_point_linkdistance_new(1,:)));
            matrix_of_point_linkdistance_new(:,position) =
inf(length(matrix_of_point_linkdistance_new(1,:)),1);
      else
            break
      end
end
clear number position
```

```
figure(7)
hold on
for i = 1:length(number6_worktime_point)

plot(map_node_data(number6_worktime_point(i),2),map_node_data(number6_worktime_point(i),3),'k
*','Linewidth',2)
end
clear i
%% The seventh snow sweeper
number7_worktime_all = 0;
number7_worktime_point = [];
for i = 1:length(matrix_of_point_linkdistance_new(:,1))
    if matrix_of_point_linkdistance_new(i,i) == 0
        number7 = i;
        break
    end
end
clear i
%%
matrix_of_point_linkdistance_number7 = matrix_of_point_linkdistance_new(number7,:);
fprintf('\n\n-----------------Scope planning for the seventh snow sweeper:-----------------\n\n')
while number7_worktime_all <= worktime_all_per_car_time
    [number,position] = min(matrix_of_point_linkdistance_number7);
    if number7_worktime_all + map_node_data_worktime_new(position,4)<=
worktime_all_per_car_time
        fprintf('The shortest distance found is:%f \t And the label of the intersection
is:%d\n',number,position)
        matrix_of_point_linkdistance_number7(position) = inf;
        number7_worktime_point = [number7_worktime_point,position];
        number7_worktime_all = number7_worktime_all +
map_node_data_worktime_new(position,4);
        fprintf('\t\tThe workload of the intersection found is as
follows:%f\n',map_node_data_worktime_new(position,4))
        fprintf('\t\tAnd the total workload of the seventh vehicle at this time is as
follows:%f\n\n',number7_worktime_all)
        map_node_data_worktime_new(position,4) = inf;
        matrix_of_point_linkdistance_new(position,:) =
inf(1,length(matrix_of_point_linkdistance_new(1,:)));
        matrix_of_point_linkdistance_new(:,position) =
inf(length(matrix_of_point_linkdistance_new(1,:)),1);
    else
        break
    end
end
clear number position
figure(7)
hold on
for i = 1:length(number7_worktime_point)

plot(map_node_data(number7_worktime_point(i),2),map_node_data(number7_worktime_point(i),3),'b
*','Linewidth',2)
end
clear i
%% Snow sweeper No.8
number8_worktime_all = 0;
number8_worktime_point = [];
for i = 1:length(matrix_of_point_linkdistance_new(:,1))
    if matrix_of_point_linkdistance_new(i,i) == 0
        number8 = i;
        break
```

```
        end
end
clear i
%%
matrix_of_point_linkdistance_number8 = matrix_of_point_linkdistance_new(number8,:);
fprintf('\n\n-----------------Scope planning of the eighth snow sweeper:-----------------\n\n')
while number8_worktime_all <= worktime_all_per_car_time
        [number,position] = min(matrix_of_point_linkdistance_number8);
        if number8_worktime_all + map_node_data_worktime_new(position,4)<=
worktime_all_per_car_time
                fprintf('The shortest distance found is:%f \t And the label of the intersection
is:%d\n',number,position)
                matrix_of_point_linkdistance_number8(position) = inf;
                number8_worktime_point = [number8_worktime_point,position];
                number8_worktime_all = number8_worktime_all +
map_node_data_worktime_new(position,4);
                fprintf('\t\tThe workload of the intersection found is as
follows:%f\n',map_node_data_worktime_new(position,4))
                fprintf('\t\tAt this time, the total workload of the eighth vehicle is as
follows:%f\n\n',number8_worktime_all)
                map_node_data_worktime_new(position,4) = inf;
                matrix_of_point_linkdistance_new(position,:) =
inf(1,length(matrix_of_point_linkdistance_new(1,:)));
                matrix_of_point_linkdistance_new(:,position) =
inf(length(matrix_of_point_linkdistance_new(1,:)),1);
        else
                break
        end
end
clear number position
figure(7)
hold on
for i = 1:length(number8_worktime_point)

plot(map_node_data(number8_worktime_point(i),2),map_node_data(number8_worktime_point(i),3),'
m*','Linewidth',2)
end
clear i
%% The ninth snow sweeper
number9_worktime_all = 0;
number9_worktime_point = [];
for i = 1:length(matrix_of_point_linkdistance_new(:,1))
        if matrix_of_point_linkdistance_new(i,i) == 0
                number9 = i;
                break
        end
end
clear i
%%
matrix_of_point_linkdistance_number9 = matrix_of_point_linkdistance_new(number9,:);
fprintf('\n\n-----------------Scope planning of the ninth snow sweeper:-----------------\n\n')
while number9_worktime_all <= worktime_all_per_car_time
        [number,position] = min(matrix_of_point_linkdistance_number9);
        if number9_worktime_all + map_node_data_worktime_new(position,4)<=
worktime_all_per_car_time
                fprintf('The shortest distance found is:%f \t And the label of the intersection
is:%d\n',number,position)
                matrix_of_point_linkdistance_number9(position) = inf;
                number9_worktime_point = [number9_worktime_point,position];
                number9_worktime_all = number9_worktime_all +
```

```
map_node_data_worktime_new(position,4);
            fprintf('\t\tThe workload of the intersection found is as
follows:%f\n',map_node_data_worktime_new(position,4))
            fprintf('\t\tAnd the total workload of the ninth vehicle at this time is as
follows:%f\n\n',number9_worktime_all)
            map_node_data_worktime_new(position,4) = inf;
            matrix_of_point_linkdistance_new(position,:) =
inf(1,length(matrix_of_point_linkdistance_new(1,:)));
            matrix_of_point_linkdistance_new(:,position) =
inf(length(matrix_of_point_linkdistance_new(1,:)),1);
      else
            break
      end
end
clear number position
figure(7)
hold on
for i = 1:length(number9_worktime_point)

plot(map_node_data(number9_worktime_point(i),2),map_node_data(number9_worktime_point(i),3),'r
*','Linewidth',2)
end
clear i
%% he tenth snow sweeper
number10_worktime_all = 0;
number10_worktime_point = [];
for i = 1:length(matrix_of_point_linkdistance_new(:,1))
      if matrix_of_point_linkdistance_new(i,i) == 0
            number10 = i;
            break
      end
end
clear i
%%
matrix_of_point_linkdistance_number10 = matrix_of_point_linkdistance_new(number10,:);
fprintf('\n\n-----------------Scope planning of the 10th snow sweeper:-----------------\n\n')
while number10_worktime_all <= worktime_all_per_car_time
      [number,position] = min(matrix_of_point_linkdistance_number10);
      if number10_worktime_all + map_node_data_worktime_new(position,4)<=
worktime_all_per_car_time
            fprintf('The shortest distance found is:%f \t And the label of the intersection
is:%d\n',number,position)
            matrix_of_point_linkdistance_number10(position) = inf;
            number10_worktime_point = [number10_worktime_point,position];
            number10_worktime_all = number10_worktime_all +
map_node_data_worktime_new(position,4);
            fprintf('\t\tThe workload of the intersection found is as
follows:%f\n',map_node_data_worktime_new(position,4))
            fprintf('\t\tAnd the total workload of the tenth vehicle at this time
is:%f\n\n',number10_worktime_all)
            map_node_data_worktime_new(position,4) = inf;
            matrix_of_point_linkdistance_new(position,:) =
inf(1,length(matrix_of_point_linkdistance_new(1,:)));
            matrix_of_point_linkdistance_new(:,position) =
inf(length(matrix_of_point_linkdistance_new(1,:)),1);
      else
            break
      end
end
clear number position
```

```
figure(7)
hold on
for i = 1:length(number10_worktime_point)

plot(map_node_data(number10_worktime_point(i),2),map_node_data(number10_worktime_point(i),3)
,'m*','Linewidth',2)
end
clear i
%% Path planning(for example,route planning within the scope of the fourth snow sweeper plan)
figure(8)%Draw the intersection of Hohhot City on figure 8
hold on
for i=1:length(map_link_data(:,1))
    number_1 = map_link_data(i,1);
    number_2 = map_link_data(i,2);
    width = map_link_data(i,3);
    for j=1:length(map_node_data(:,1))
        if number_1 == map_node_data(j,1)
            number_1_x = map_node_data(j,2);
            number_1_y = map_node_data(j,3);
        end
    end
    for k=1:length(map_node_data(:,1))
        if number_2 == map_node_data(k,1)
            number_2_x = map_node_data(k,2);
            number_2_y = map_node_data(k,3);
        end
    end
    plot([number_1_x number_2_x],[number_1_y number_2_y],'r.','Linewidth',10,'LineStyle','-
','Linewidth',0.1)
end
xlabel('Longitude')
ylabel('Latitude')
title('Network diagram of intersection in Hohhot')
for i = 1:length(number4_worktime_point)

plot(map_node_data(number4_worktime_point(i),2),map_node_data(number4_worktime_point(i),3),'b
*','Linewidth',2)
    c=num2str(number4_worktime_point(i));%Number to character

text(map_node_data(number4_worktime_point(i),2),map_node_data(number4_worktime_point(i),3),c)
;%Mark text on the drawing
end
clear i j k number_1_x number_1_y number_2_x number_2_y width number_1 number_2 ccl
%%
number4_worktime_point
for i = 1:length(map_node_data(:,1))
    if map_node_data(i,2) >= 111.6956 && map_node_data(i,2)<=111.6957 &&
map_node_data(i,3)>=40.832 && map_node_data(i,3)<=40.83205
        number4_add_point = i;
    end
end
fprintf('\nTo form a more regular range, add points:%d\n',number4_add_point)
plot(map_node_data(number4_add_point,2),map_node_data(number4_add_point,3),'k*','Linewidth',3)
clear i c
for i = 1:length(map_node_data(:,1))
    if map_node_data(i,2) >= 111.6662 && map_node_data(i,2)<=111.6664 &&
map_node_data(i,3)>=40.8497 && map_node_data(i,3)<=40.8499
        number4_delate_point = i;
    end
end
```

```
fprintf('\nTo form a more regular range, delete the point:%d\n',number4_delate_point)
plot(map_node_data(number4_delate_point,2),map_node_data(number4_delate_point,3),'r*','Linewidt
h',3)
clear i c
number4_worktime_point_new = number4_worktime_point;
for j = 1:length(number4_worktime_point_new)
    if number4_worktime_point_new(j) == number4_delate_point
        number4_worktime_point_new(j) = number4_add_point;
    end
end
%Add another point to make the scope rule
for i = 1:length(map_node_data(:,1))
    if map_node_data(i,2) >= 111.6676479 && map_node_data(i,2)<=111.6676481 &&
map_node_data(i,3)>=40.8151019 && map_node_data(i,3)<=40.8151021
        number4_add_point = i;
    end
end
fprintf('\nTo form a more regular range, add points:%d\n',number4_add_point)
number4_worktime_point_new = [number4_worktime_point_new,number4_add_point];
plot(map_node_data(number4_add_point,2),map_node_data(number4_add_point,3),'k*','Linewidth',3)
clear i j c
fprintf('Planning range point of the fourth snow sweeper:\n')
number4_worktime_point_new
%% On the new distance matrix of the fourth snow sweeping mechanism frame
matrix_of_number4_linkdistance=[];%A new matrix is constructed to store the location information of
the intersection within the planning range of the fourth snow sweeper
%We need to get the connection between the points and get the situation matrix
matrix_of_number4_linkline = [];%A matrix is constructed to store the connection of the planned area
intersection of the fourth snow sweeper
for i = 1:length(number4_worktime_point_new)%Cycle through the points traversed by the fourth
snow sweeper
    %       for j = 1:length(number4_worktime_point_new)
    point_view_a = number4_worktime_point_new(i);%Take each point in turn
    for j = 1:length(map_link_data(:,1))%In the connection information matrix, traverse in turn
        if point_view_a == map_link_data(j,1)%Find the starting point from the point above
            for k = 1:length(number4_worktime_point_new)%Traverse the other points in the fourth
snow sweeper again
                if map_link_data(j,2) == number4_worktime_point_new(k)%f the traversal point is
found to be consistent with the label of the second point in the connection information table
                    point_view_b = map_link_data(j,2);%Assign the label of subsequent
connected points to point_ view_b
                    matrix_of_number4_linkline =
[matrix_of_number4_linkline;point_view_a,point_view_b];
                end
            end
        end
    end
    %       end
end
clear i j k point_view_a point_view_b
fprintf('Get the connection of the points within the planning range of the fourth snow sweeper!!!as
follows:\n')
matrix_of_number4_linkline
fprintf('\t\tAnd the number of vertices of the Euler graph is as
follows:%d\n',length(number4_worktime_point_new))
fprintf('\t\tAnd the number of edges of the Euler graph is:%d\n\n',length(matrix_of_number4_linkline))
matrix_of_number4_linkline_c = matrix_of_number4_linkline;%Fleury algorithm for C + +
for i = 1:length(number4_worktime_point_new)
    number = number4_worktime_point_new(i);
    for j = 1:length(matrix_of_number4_linkline_c(:,1))
```

```
            if number == matrix_of_number4_linkline_c(j,1)
                matrix_of_number4_linkline_c(j,1) = i;
            end
            if number == matrix_of_number4_linkline_c(j,2)
                matrix_of_number4_linkline_c(j,2) = i;
            end
        end
end
clear i j number
fprintf('\nGet the connection of the fourth snow sweeper planning area intersection for C++!!!\n\n')
matrix_of_number4_linkline_c
matrix_of_number4_linkdistance =
inf(length(matrix_of_number4_linkline),length(matrix_of_number4_linkline));
for i = 1:length(matrix_of_number4_linkdistance(:,1))
        matrix_of_number4_linkdistance(i,i) = 0;
end
clear i
for i = 1:length(matrix_of_number4_linkline(:,1))
        point_a = matrix_of_number4_linkline(i,1);
        point_b = matrix_of_number4_linkline(i,2);
        for j = 1:length(number4_worktime_point_new)
            if point_a == number4_worktime_point_new(j)
                point_position_a = j;
            end
            for k = 1:length(number4_worktime_point_new)
                if point_b == number4_worktime_point_new(k)
                    point_position_b = k;
                    matrix_of_number4_linkdistance(point_position_a,point_position_b) =
matrix_of_linkdistance(point_a,point_b);
                end
            end
        end
end
clear i j point_a point_b point_position_a point_position_b k
%symmetric
for i = 1:length(matrix_of_number4_linkdistance(:,1))
        for j = i:length(matrix_of_number4_linkdistance(:,1))
            if matrix_of_number4_linkdistance(i,j) ~= inf
                matrix_of_number4_linkdistance(j,i) = matrix_of_number4_linkdistance(i,j);
            end
        end
end
fprintf('\n\nSymmetry operation completed!!!\n\n')
clear i j
%% The fourth snow sweeper uses minimum spanning tree calculation
%The location information of intersection points is put into the new matrix
map_node_data_number4 = [];%Storage location information
for i = 1:length(number4_worktime_point_new)
        map_node_data_number4(i,1) = number4_worktime_point_new(i);
        map_node_data_number4(i,2) = map_node_data(number4_worktime_point_new(i),2);
        map_node_data_number4(i,3) = map_node_data(number4_worktime_point_new(i),3);
end
clear i
%Calling the prim function
[Edge_number4,weight_number4] = Prim(matrix_of_number4_linkdistance)
figure(8)
hold on
scatter(map_node_data_number4(:,2),map_node_data_number4(:,3),50,'filled')
hold on
fprintf('Drawing successfully with minimum spanning tree!!!\n')
```

```
clear i
%%
Q2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
% sum(map_location_data(:,3))
fprintf('-------------------------------Q2---------------------------------\n')
fprintf('The plane area of snow storage is obtained:%f square meter\n',sum(map_location_data(:,3)))
%% %%%%%%%%%%%%%%%%%%%%%% Suppose: the task needs to be completed within the
specified time, and the number of snow sweepers needed to be solved is n
up_time = input('\n\nPlease enter the upper limit (unit: H) of completion time required:\n');
%The upper limit of completion time is up_ Time hours
height_of_snow_unknown = input('\nPlease enter the accumulated snow thickness in M note: the
thickness should not exceed 1.5m\n');
%The accumulated snow thickness is height_ Of_ snow_ Unknown rice
Vs_car = 10*(1.5 - height_of_snow_unknown);%Snow sweeping speed of snow sweeper km / h
fprintf('\nIn this case, the speed of the snow sweeper is:%f km/h',Vs_car)
Vs_car_min = (Vs_car / 60)*1000;%Speed of snow sweeper m / min
fprintf('\t\tThe speed of the snow sweeper is:%f m/min',Vs_car_min)
area_car_per_min = Vs_car_min * 2;%Snow sweeper minute area of snow sweeper
fprintf('\nIt is known that the area that each snow sweeper can clean per minute is as follows:%f square
meter\n',area_car_per_min)
fprintf('\nThe areas known to need to be cleaned up are:%f square meter\n',worktime_all)
up_time_min = 60 * up_time;%Time to minutes
for n = 1:1000%Look for the minimum number of vehicles that can be provided in a time frame
    car_number_predict = n;%Suppose you need n snow sweepers
    area_car_min_all = area_car_per_min * n;%Total area cleared by N snow sweepers per minute
    time_predict = worktime_all / area_car_min_all;%The cleaning time of N snow sweepers is
calculated
    if time_predict <= up_time_min
        fprintf('\n\nFound it in %f h During the period, the minimum number of snow sweepers to be
provided is:%d\n',up_time,car_number_predict)
        fprintf('\tAnd in the case of% d vehicles, the cleaning time
is:%f\n',car_number_predict,time_predict/60)
        break
    end
end
clear n
%% Meshgrid draws images
fprintf('Draw a three-dimensional image of online time snow thickness number of vehicles\n')
up_time_100 = linspace(6,1,100);
height_of_snow_unknown_100 = linspace(0,1.49,100);
number_car_matrix = [];%Build a matrix to store the number of vehicles
time_predict_100_100 = [];%Establish a matrix to store the actual time required
[up_time_matrix_100,height_of_snow_unknown_matrix_100]=meshgrid(up_time_100,height_of_sno
w_unknown_100);
for i = 1:100
    for j = 1:100
        Vs_car_100 = 10*(1.5 - height_of_snow_unknown_100(j));%Snow sweeping speed of snow
sweeper km / h
        Vs_car_min_100 = (Vs_car_100 / 60)*1000;%Speed of snow sweeper M / min
        area_car_per_min_100 = Vs_car_min_100 * 2;%Snow sweeper minute area of snow sweeper
        up_time_min_100 = 60 * up_time_100(i);%Time to minutes
        for n = 1:1000%Look for the minimum number of vehicles that can be provided in a time
frame
            car_number_predict_100 = n;%Suppose you need n snow sweepers
            area_car_min_all_100 = area_car_per_min_100 * n;%Total area cleared by N snow
sweepers per minute
            time_predict_100 = worktime_all / area_car_min_all_100;%The cleaning time of N
snow sweepers is calculated
            if time_predict_100 <= up_time_min_100
```

```
                    number_car_matrix(i,j) = car_number_predict_100;
                    time_predict_100_100(i,j) = time_predict_100/60;
                    break
                end
            end
        end
end
clear Vs_car_100 Vs_car_min_100 area_car_per_min_100 up_time_min_100 n
car_number_predict_100 area_car_min_all_100 time_predict_100 i j
% surf(up_time_matrix_100,height_of_snow_unknown_matrix_100,number_car_matrix)
figure(9)
mesh(up_time_matrix_100,height_of_snow_unknown_matrix_100,number_car_matrix)
xlabel('Time limit：h')
ylabel('Snow thickness:m')
zlabel('Number of vehicles:1')
title('Time - vehicle thickness 3D image')
fprintf('\n3D image rendering completed!!!\n\n')
figure(10)
mesh(up_time_matrix_100,time_predict_100_100,height_of_snow_unknown_matrix_100)
xlabel('Time limit：h')
zlabel('Snow thickness:m')
ylabel('Actual time:h')
title('3D image of time upper limit actual time snow thickness')
fprintf('\n3D image rendering completed!!!\n\n')
%%
figure(11)
xlabel('Longitude')
ylabel('Latitude')
title('Hohhot')
hold on
for i=1:length(map_link_data(:,1))
    number_1 = map_link_data(i,1);
    number_2 = map_link_data(i,2);
    width = map_link_data(i,3);
    for j=1:length(map_node_data(:,1))
        if number_1 == map_node_data(j,1)
            number_1_x = map_node_data(j,2);
            number_1_y = map_node_data(j,3);
        end
    end
    for k=1:length(map_node_data(:,1))
        if number_2 == map_node_data(k,1)
            number_2_x = map_node_data(k,2);
            number_2_y = map_node_data(k,3);
        end
    end
    plot([number_1_x number_2_x],[number_1_y number_2_y],'r.','Linewidth',10,'LineStyle','-
','Linewidth',0.1)
end
clear number_1 number_2 number_1_x number_1_y number_2_x number_2_y i j k
%% %%%%%%%%%%%%%%%%%%%%Set the height that can store snow to 2M and compress 3 times
the volume
height_of_container = 6;
%The upper limit of snow height is calculated so that it can be stored in the city without being
transported to the outside of the city
contain_snow_V = sum(map_location_data(:,3))*height_of_container;
fprintf('\nThe volume of snow that can be stored in large open space in the city is as follows:%fcubic
metre\n\n',contain_snow_V)
height_of_snow_contain_incity = (contain_snow_V) / worktime_all + 0.5;%This 0.5 is the minimum
upper limit value of the transport vehicle, calculated in M
```

```
fprintf('When all of them exist in the open large space in the city, the maximum upper limit of Road area snow is as follows:%f metre\n\n',height_of_snow_contain_incity)
fprintf('\nIn the case of given storage snow height, find the lowest road can not be transported to the snow height outside the city!!!\n\n')
%% Each point goes to the nearest snow storage location
%Create a matrix to store the distance and label of each intersection to the nearest snow point
matrix_of_point_to_Potential_snow = [];%The first column: the intersection label; the second column: the mark of the intersection where snow is stored; the third column: the corresponding distance m; the fourth column: the area of snow to be stored
for i = 1:length(map_node_data(:,1))
    matrix_of_point_to_Potential_snow(i,1) = i;
    distance = inf;%Set the distance between two points as inf
    for j = 1:length(matrix_of_Potential_snow)
        if distance >= matrix_of_point_linkdistance(i,matrix_of_Potential_snow(j,1))
            distance = matrix_of_point_linkdistance(i,matrix_of_Potential_snow(j,1));
            matrix_of_point_to_Potential_snow(i,2) = matrix_of_Potential_snow(j);
            matrix_of_point_to_Potential_snow(i,3) = distance;
            matrix_of_point_to_Potential_snow(i,4) = map_node_data_worktime(i,4);
        end
    end
end
fprintf('The labeling matrix information of intersection and corresponding snow storage intersection is established!!!\n\n')
fprintf('The matrix is:matrix_of_point_to_Potential_snow\n')
save('matrix_of_everynode_to_potential_snow_node.txt','matrix_of_point_to_Potential_snow','-ascii');
fprintf('\nThe information table is output in the txt file:matrix_of_everynode_to_potential_snow_node!!!\n')
clear i j distance
%% Calculate the volume information of snow storage point
matrix_of_Potential_snow_new = matrix_of_Potential_snow;%A matrix is established to represent the snow storage volume and other information of snow storage points
for i = 1:length(matrix_of_Potential_snow(:,1))
    matrix_of_Potential_snow_new(i,5) = matrix_of_Potential_snow(i,4) * height_of_container;%Ideal snow storage volume at each point
end
clear i
for i = 1:length(matrix_of_Potential_snow(:,1))
    matrix_output=[];%Output the label of each snow storage point
    matrix_of_Potential_snow_new(i,6) = 0;
    for j = 1:length(matrix_of_point_to_Potential_snow(:,1))
        if matrix_of_point_to_Potential_snow(j,2) == matrix_of_Potential_snow(i,1)
            matrix_of_Potential_snow_new(i,6) = matrix_of_Potential_snow_new(i,6) + matrix_of_point_to_Potential_snow(j,4);%The area of snow to be stored in each snow storage point is calculated by superposition
            matrix_output = [matrix_output,matrix_of_point_to_Potential_snow(j,1)];
        end
    end
    fprintf('\n\nThe label is:%d The ideal number of intersections to be managed is:\n',matrix_of_Potential_snow(i,1))
    matrix_output

end
clear i j matrix_output
fprintf('\nThe volume information of snow storage point is calculated!!! The fifth column of the matrix is in cubic meters and the sixth column is in square meters\n\n')
%%
%The height of the snow input through the is:height_of_snow_unknown
fprintf('\n The height of snow input through the is:\n',height_of_snow_unknown)
if height_of_snow_unknown > 0.5
    for i = 1:length(matrix_of_Potential_snow_new(:,1))
```

```
            matrix_of_Potential_snow_new(i,7) = matrix_of_Potential_snow_new(i,6) *
(height_of_snow_unknown-0.5);%The seventh column is the volume of snow to be stored
        end
end
fprintf('\nChange the matrix, the seventh column is the volume of snow to be stored\n')
clear i
%% 3D image:
car_trans_number_matrix = [];%Create matrix
people_number_matrix = [];%Create matrix
height_of_snow_2 = linspace(1.49,0.51,100);
for i = 1:length(height_of_snow_2)
    if height_of_snow_2(i) <=0.5
        continue
    else
        height_of_snow_trans = height_of_snow_2(i) - 0.5;%Height of snow to be transported (m)
        V_of_snow_trans = height_of_snow_trans * worktime_all;%Volume of snow to be
transported (m3)
        %The volume of transport vehicle is 4 * 2 * 2 m3
        trans_car_V = 16;%Volume of transport vehicle
        trans_car_V_snow = 16*3;%Three times the volume of compression to calculate, a single
transport of 48 cubic meters of snow
        time_to_trans = V_of_snow_trans / trans_car_V_snow;
        time_to_trans = ceil(time_to_trans);
        up_time_to_trans = linspace(1,10,100);%input('Please enter the time limit (unit: H) to
complete the task:');
        %Number of times to be transported per hour
        for j = 1:length(up_time_to_trans)
            trans_time_per_hour = time_to_trans / up_time_to_trans(j);%Number of trips per hour
            V_car_trans_snow = 30;%The speed of the transport vehicle is assumed to be 30 km / h
            distance_to_trans_snow = sum(matrix_of_point_to_Potential_snow(:,3));
            distance_to_trans_snow_everytime =
distance_to_trans_snow/length(matrix_of_point_to_Potential_snow(:,1));
            distance_to_trans_snow_all = distance_to_trans_snow_everytime * time_to_trans;
            V_car_trans_snow_meter_per_min = V_car_trans_snow * 1000 / 60;%Speed of
transport vehicle in m / min
            up_time_to_trans_min = up_time_to_trans(j) * 60;%Upper limit time converted to
minutes
            for n = 1:100000
                if distance_to_trans_snow_all / ( n * V_car_trans_snow_meter_per_min) <=
up_time_to_trans_min
                    car_trans_number_matrix(i,j) = n;
                    people_number_matrix(i,j) = car_trans_number_matrix(i,j) * 2500 /
30;%Ratio of sanitation workers to transport vehicles
                    people_number_matrix(i,j) = ceil(people_number_matrix(i,j));
                    break
                end
            end
        end
    end
end
fprintf('\n XYZ matrix calculation finished!!!\n')
clear i n j
[height_of_snow_2_100,up_time_to_trans_100] = meshgrid(height_of_snow_2,up_time_to_trans);
figure(12)
mesh(height_of_snow_2_100,up_time_to_trans_100,car_trans_number_matrix)
title('When transport vehicles are needed: snow thickness upper limit time number of transport vehicles
3D image')
xlabel('Snow thickness: M')
ylabel('Time limit: H')
zlabel('Number of transport vehicles: 1')
```

```
fprintf('\nSnow thickness - upper limit time - number of transport vehicles 3D image rendering
completed!!!\n\n')
figure(13)
mesh(height_of_snow_2_100,up_time_to_trans_100,people_number_matrix)
title('eed to use transport vehicles: snow thickness - upper limit time - number of sanitation workers 3D
image')
xlabel('Snow thickness: M')
ylabel('Time limit: H')
zlabel('Number of sanitation workers: 1')
fprintf('\nSnow thickness - upper limit time - number of sanitation workers, 3D image rendering
completed!!!\n\n')
%% Transportation time
%Snow cleaning time has been determined, and then calculate the transportation time of snow
fprintf('!!!Tip: the snow height you entered above is:%f m!!!(The number of vehicles used to sweep
snow)\n',height_of_snow_unknown)
height_of_snow_2 = input('\nPlease enter the actual snow height (unit: m and no more than 1.5m):');
if height_of_snow_2 <=0.5
    fprintf('\n\tDo not need to use transport vehicles, snow can be directly stored in the green
belt!!!\n')
else
    height_of_snow_trans = height_of_snow_2 - 0.5;%Height of snow to be transported (M)
    V_of_snow_trans = height_of_snow_trans * worktime_all;%Volume of snow to be transported
(M3)
    fprintf('The volume of snow to be transported is:%fcubic metre\n',V_of_snow_trans)
    trans_car_V = 16;
    trans_car_V_snow = 16*3;
    time_to_trans = V_of_snow_trans / trans_car_V_snow;
    time_to_trans = ceil(time_to_trans);
    fprintf('\nRound up!!!\t')
    fprintf('The total number of times to be transported is:%d times\n',time_to_trans)
    fprintf('!!!Tip: the upper limit of time you have entered to complete the task is:%f h!!!(this time
limit was used to calculate the number of snow sweepers)\n',up_time)
    up_time_to_trans = input('Please enter the time limit (unit: H) to complete the task:');
    trans_time_per_hour = time_to_trans / up_time_to_trans;
    V_car_trans_snow = 30;
    fprintf('\nThe speed of transport vehicles is:%f km/h',V_car_trans_snow)
    fprintf('The speed of transport vehicles is:： %f m/min\n\n',V_car_trans_snow* 1000/60)
    distance_to_trans_snow = sum(matrix_of_point_to_Potential_snow(:,3));
    fprintf('\nThe results show that the total distance of transportation is as follows:%f
m\n',distance_to_trans_snow)
    distance_to_trans_snow_everytime =
distance_to_trans_snow/length(matrix_of_point_to_Potential_snow(:,1));
    fprintf('\nThe average distance of each transportation point is obtained as follows:%f
m\n',distance_to_trans_snow_everytime)
    distance_to_trans_snow_all = distance_to_trans_snow_everytime * time_to_trans;
    fprintf('\nThe total distance to be transported is obtained as follows:%f
m\n',distance_to_trans_snow_all)
    V_car_trans_snow_meter_per_min = V_car_trans_snow * 1000 / 60;
    up_time_to_trans_min = up_time_to_trans * 60;
    for n = 1:10000
        if distance_to_trans_snow_all / ( n * V_car_trans_snow_meter_per_min) <=
up_time_to_trans_min
            car_trans_number = n;
            fprintf('\n\n!!!The minimum number of transport vehicles found
is:%d !!!\n',car_trans_number)
            people_number = car_trans_number * 2500 / 30;%Ratio of sanitation workers to
transport vehicles
            people_number = ceil(people_number);
            fprintf('!!!The minimum number of sanitation workers was
found： %d !!!\n\n',people_number)
```

```
                break
            end
        end
    end
    clear i n
    %%
    Q34%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    fprintf('\n\n----------------------Q4-----------------------\n\n')
    importance_road_data = load('excel2.txt');
    figure(14)
    hold on
    for i=1:length(importance_road_data(:,1))
        number_1 = importance_road_data(i,1);
        number_2 = importance_road_data(i,2);
        importance = importance_road_data(i,3);
        for j=1:length(map_node_data(:,1))
            if number_1 == map_node_data(j,1)
                number_1_x = map_node_data(j,2);
                number_1_y = map_node_data(j,3);
            end
        end
        for k=1:length(map_node_data(:,1))
            if number_2 == map_node_data(k,1)
                number_2_x = map_node_data(k,2);
                number_2_y = map_node_data(k,3);
            end
        end
        if importance == 6 || importance == 5
            plot([number_1_x number_2_x],[number_1_y number_2_y],'k.','Linewidth',10,'LineStyle','-
    ','Linewidth',importance/3)
        elseif importance == 4 || importance == 3
            plot([number_1_x number_2_x],[number_1_y number_2_y],'b.','Linewidth',10,'LineStyle','-
    ','Linewidth',importance/3)
        else% importance == 2 || importance == 1
    %           plot([number_1_x number_2_x],[number_1_y
    number_2_y],'r.','Linewidth',10,'LineStyle','-','Linewidth',importance/3)
        end
    end
    xlabel('Longitude')
    ylabel('Latitude')
    title('Road network map of a city(importance of road)')
    clear i j k number_1_x number_1_y number_2_x number_2_y width number_1 number_2 importance
    %% minimum spanning tree
    matrix_of_link_importance = inf(length(map_node_data(:,1)),length(map_node_data(:,1)));%Build a
    matrix to store the weight of importance
    for i = 1:length(importance_road_data(:,1))
        number_1 = importance_road_data(i,1);
        number_2 = importance_road_data(i,2);
        importance_road = importance_road_data(i,3);
        if importance_road == 6 || importance_road == 5
            matrix_of_link_importance(number_1,number_2) = 10;
            matrix_of_link_importance(number_2,number_1) = 10;
        elseif importance_road == 4 || importance_road == 3
            matrix_of_link_importance(number_1,number_2) = 11;
            matrix_of_link_importance(number_2,number_1) = 11;
        else
            matrix_of_link_importance(number_1,number_2) = 12;
            matrix_of_link_importance(number_2,number_1) = 12;
        end
    end
```

```
fprintf('\n!!!Get the adjacency matrix of the importance between adjacent points!!!\n')
clear i number_1 number_2 importance_road
%% Prim algorithm, calculate the minimum spanning tree, and draw the minimum spanning tree in
Figure 15
[Edge,weight] = Prim(matrix_of_link_importance);
figure(15)
xlabel('Longitude')
ylabel('Latitude')
title('Minimum spanning tree drawing of intersection in Hohhot(road priority)')
hold on
scatter(map_node_data(:,2),map_node_data(:,3),50,'filled')
hold on
for i = 1:size(Edge,2)
    plot(map_node_data(Edge(:,i),2),map_node_data(Edge(:,i),3),'b-','Linewidth',1)
end
fprintf('Successful minimum spanning tree drawing!!!\n')
clear i
%%
fprintf('\n\n\n!!!****************End****************!!!\n')
```

Matlab Floyd subfunction (file: Floyd.m):

```
function [D,path]=Floyd(A)
n=length(A);
D=A;
path=zeros(n);
for i=1:n
    for j=1:n
        if D(i,j)~=inf
            path(i,j)=j;%The following point of I is J
        end
    end
end
for k=1:n
    for i=1:n
        for j=1:n
            if D(i,k)+D(k,j)<D(i,j)
                D(i,j)=D(i,k)+D(k,j);
                path(i,j)=path(i,k);
            end
        end
    end
end
```

Matlab Findline subfunction (file: findline.m):

```
function L=findline(path,i,j)
%Subroutine,convenient to find the best route
L(1)=i;
L(2)=path(i,j);
k=2;
while L(k)~=j
```

```
        k=k+1;
        L(k)=path(L(k-1),j);
end
```

Matlab Lontodistance subfunction code (file: lontodistance.m):

```
%% function Lontodistance,It is used to calculate the actual distance between two points
function [distance] = Lontodistance(Lon1,Lat1,Lon2,Lat2)
dlon = Lon1- Lon2;
dlat = Lat1 - Lat2;
a = sin(dlat*pi/360)^2 + cos(Lat1*pi/180)*cos(Lat2*pi/180)*(sin(dlon*pi/360)^2);
distance = 2*asin(sqrt(a))*6371*1000;
```

Matlab Prim subfunction code (file: prim.m):

```
%% function Solving minimum spanning tree with prim function
function [Edge,weight] = Prim(W)
n = length(W)%Number of vertices
U = 1;
Ubar = 2:n;
[v1,pos1] = min(W(U,Ubar));
weight = v1;
Edge = [1;Ubar(pos1)];
U = [U,Ubar(pos1)];
Ubar(pos1) = [];
for i = 1:n-2
    [v1,pos1] = min(W(U,Ubar));
    [v2,pos2] = min(v1);
    Edge = [Edge,[U(pos1(pos2));Ubar(pos2)]];
    weight = weight + v2;%Update weight
    U = [U,Ubar(pos2)];%Update u and Ubar
    Ubar(pos2)= [];
end
```

C++ Fleury algorithm code (file: test.cpp):

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<string.h>
#include<algorithm>
#include<vector>
using namespace std;
//fleury
const int N = 1005;
```

```
int n, m, flag, top, sum, du, ans[5005], map[N][N];

void dfs(int x)
{
	top++;
	ans[top] = x;
	for(int i = 1; i <= n; i++)
	{
		if(map[x][i] >0)
		{
			map[x][i]=0;
			map[i][x]=0;
			dfs(i);
			break;
		}
	}
}

void fleury(int x)
{
	top = 1;
	ans[top] = x;
	while(top > 0)
	{
		int k = 0;
		for(int i = 1; i <= n; i++)//Judge whether it is extensible
		{
			if(map[ans[top]][i] >0)//If there is an edge starting from ans [top],then it is extensible
			{
				k = 1;
				break;
			}
		}
		if(k == 0)//This point X has no other edge to go first(i.e. not extensible),so it is output
		{
			printf("%d ", ans[top]);
			top--;
		}
		else if(k == 1)//If so,which route can DFS extend
		{
			top--;//This needs to be noted,in order to backtrack
			dfs(ans[top+1]);
		}
	}
}
int main()
{
	int i;
	printf("input n&m\n");
	while(scanf("%d%d", &n, &m) != EOF)
	{
		//memset(du,0,sizeof(du));
		memset(map, 0, sizeof(map));

		for(i = 1; i <= m; i++)
		{
			int x, y;
```

```
            printf("input x&y\n");
            scanf("%d%d", &x, &y);
            map[x][y]=1; //Record the edge, because it is an undirected graph,so add two edges. 
There may be more than one edge between two points
            map[y][x]=1;
        }
        flag = 1; //Flag marks the start point.If the degrees of all points are even, search from 1
        sum = 0;
        for(i = 1; i <= n; i++)
        {
            du=0;
            for(int j=1; j<=n; j++)
                du+=map[i][j];
            if(du % 2 == 1)
            {
                sum++;
                flag = i;//If there are odd edges,search from the odd sides
            }
        }
        if(sum == 0 || sum == 2)
            fleury(flag);
    }
    return 0;
}
```