The least time of vehicle travelling

【Abstract】This paper uses Dynamic Programming to solve the problem of the least time needed for automobile driving, in order to provide the optimal route for it. First of all, this paper introduces the fundamental principles and ideas of Dynamic Programming and constructs the model of it. Then according to the roads conditions of Beijing, the roads are characterized into the time needed from one node to another. Thus the Adjacency Matrix is produced and then modified due to the Floyd algorithm. Finally the least time between any two nodes is determined. Hence it is convenient for people to choose the least time-consuming route by automobile. Considered the deficiency of this model, the paper offers relevant algorithms to refine it, and concludes the applicability of the model.

Key words: Dynamic Programming; The least-Time; Floyd

1.Introduction

There have been a large amount of researches on the problem of transportation route choosing previously, most of which, however, concentrate on the problem of “bus transferring”. The stochastic equilibrium assignment of bus network system with multi-users are

discussed in literature [1]; in literature [2], a “Aggregate—combustion” algorithm is put forward to quickly finding the optimal route to meet passengers’ different demands. In this paper, we study the choosing of the optimal route based on least time spending for cars(private cars),that is to say, find the optimal route and least time spending on the travel by method of dynamic program, and thus make it convenient for people to travel by private cars. In literature [3], dynamic programming is applied to find the shortest transportation route and the transportation process is divided into several decision-making stages. In literature [4], a dynamic algorithm based on temporal constraint network is put forward to solving the planning and scheduling problem. By combining and applying the basic idea and strategy of dynamic programming in literature [5], we utilize dynamic programming combined with Floyd algorithm to get the least time needed, meanwhile set up relevant model and advance the improvement to avoid deficiency.

1.1 shortest route

The shortest route problem is how to find the shortest route when the starting point, the end point and many probable routes between them are given. The algorithm to solve this problem is called “the shortest route algorithm”, sometimes abbreviated for “route algorithm”. The most common algorithms are Dijkstra algorithm, dynamic programming, A* algorithm, Floyd algorithm, Johnson

algorithm, etc.

This paper adopts the dynamic programming to solve the problem.

1.2 dynamic programming

Dynamic programming is a branch of operations research, and it's a mathematical method to solve the optimization problem of multi-stage decision-making process. This method is proposed by American mathematician R.Bellman in 1950s first. It is proved that dynamic programming is much more valid than linear programming, and usually used to solve problems about optimization.

1.2.1 The basic idea of dynamic programming

In some cases, there may be many possible solutions, with each corresponding to a value, and we hope to find the solution with the optimal value. Dynamic programming is very similar to the "divide-and-conquer "method, the basic idea of both of which is to divide the problem into several sub-problems, first solve the sub-problems, and then get the solution of the original problem from these sub-problems. What is different from the divide-and-conquer method is that, the decomposed sub-problems divided by the dynamic programming are often not mutually independent. When divide and conquer method is used to solve such problems, there will be too many decomposed sub-problems, and some are repeated calculated. If we can save the answer of solved sub-problems, and find out them when

necessary, a lot of repeated counting can be avoided and much time can be saved. We can use a table to record all the answers of solved sub-problems, regardless of whether they will be used later or not, as long as it is calculated. This is the basic idea of dynamic programming.

2. Modeling with dynamic programming

2.1 modeling assumption

1) Assuming that cars travel at a constant speed.

2) Assuming there are no traffic jams, traffic accidents and cars run smoothly.

3) Assuming no traffic lights is encountered, no waiting time.

4) Assuming car turning will not affect speed.

5) Given the heavy traffic jams downtown, this model avoids the choice of city center. In general, the center of the city is always out of congestion. Therefore, this model suggests that people do not choose the driving route downtown. The city center is not marked in the map of our model.

2.2Data Processing

2.2.1 Data sources

This paper uses the urban road network of Beijing and the figure is in proportion to zoom.

Figure 1

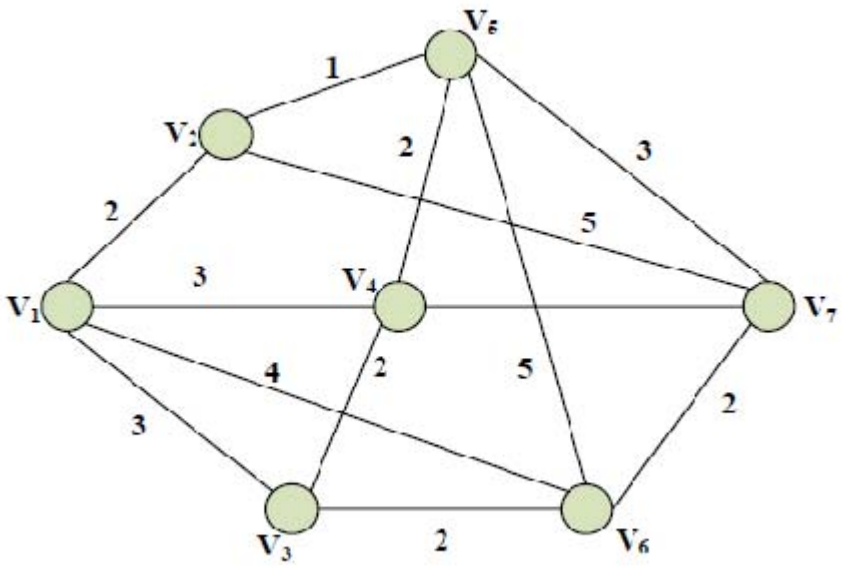
Figure 1 is the map within the Fourth Ring Road in Beijing, we will use the weighted graph to mark out some of the main roads and the cross-road in Beijing, and then abstract the path distance between nodes in order to determine the time needed.

2.2.2 Background Knowledge

(1) Weighted graph

To abstract figure 1 into figure 2, weighted graph method will be used. The following are weighted graph for details.

Each side of a weighted graph is bestowed with a corresponding non-negative real number, which is called the weight of this side. Below is a weighted graph:



Let G= (V, E) be a graph, it can be seen from the above, V= {v1,v2,…,v7},E refers to the distance between nodes, which is called weight value. In the above figure the route (weight) between V1 and V2 is 2.

(2) Adjacency matrix

This paper uses the adjacency matrix to deal with datas. Adjacency matrix is a matrix used to express the relationship between the neighboring vertex. Let G = (V, E) be a graph, V={v1,v2,…,vn}, the adjacency matrix of G is a n-order matrix with the following properties:

[1]. Adjacency matrix of undirected graph must be symmetric, while the adjacency matrix of directed graph is not necessarily symmetric. Thus, n^2 units are needed to store the adjacency matrix which is used to represent the n vertices of a directed graph.

[2].For an undirected graph with n vertices, only right upper or left down elements are needed to deposit with exclusion of the 0 element in the diagonal elements. Therefore, it only needs 1 +2 +...+(n-1) = n (n-1) / 2 units.

(3)Floyd algorithm

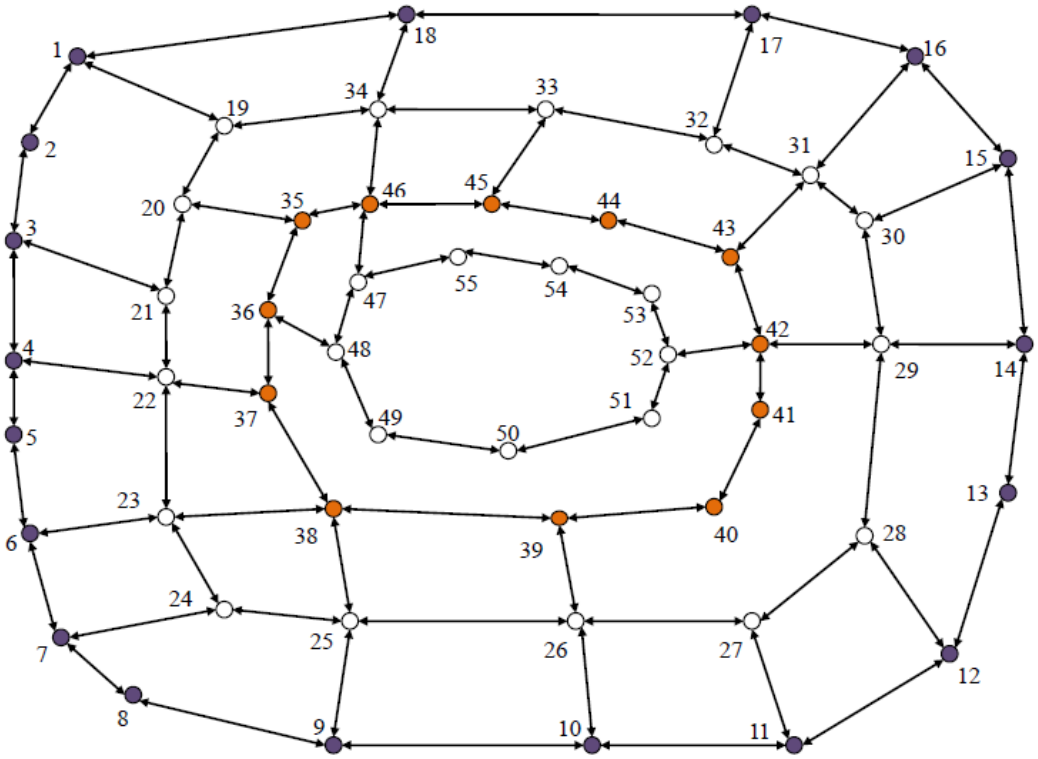
Floyd algorithm is a dynamic programming algorithm (interception method) used to find the shortest route between vertices of a given weighted graph. The advantage is that it is easy to understand, can calculate the shortest distance between any two nodes, and its code is simple. The disadvantage is that it's of relatively high time complexity, not suitable for large data calculations.

The core idea of Floyd algorithm: find the shortest route matrix of any two vertices of a weighted graph through the weight matrix of it. Begin from the weighted graph adjacency matrix A=[a(i,j)]（n×n）, update recursively for n times, that is to say, from the matrix D (0) = A, Construct matrix D (1) by a given formula, similarly, construct D (2) from D(1) by the same formula.……Finally, construct matrix D (n) from D(n-1) with the same formula. The (i , j) element of matrix D (n) is the shortest route length from vertex i to vertex j, and D (n) is called distance matrix of the graph.

2.2.3Data collection

The distance data are obtained from measurement by the network,

the impact on travel time of different routes is considered only when faced with multiple branch ports or no straight lines can be obtained. This paper selected 55 nodes within the Fourth Ring Road in Beijing, and abstract a network map(figure 2) from figure 1. As follows:



Nodes denote note (counterclockwise)

Outer ring: the upper left label 1 to label 18 forms the outer ring; the third ring: from label 19-34; second Ring: From the label 35-46; the most in the loop: from the label 47-55

Considered the size of figure 2, specific weight as well as route distance is not specifically marked up. Further details as below:

The weight value of the line segment between two nodes of a weighted graph denotes the distance between them, specifically; the

weight values of 81 direct routes in Figure 2 are given as follow. (Unit: km)

Outer ring (18 line segments ,18 nodes) from 1 -> 2to18 -> 1 are: 2.747、5.093、3.083、1.492、2.403、2.262、1.567、4.752、4.938、4.075、2.983、3.483、3.849、5.287、2.735、2.926、3.253、4.887

the junction of the outer ring and the third ring: 2.945（1-->19）、2.797（3-->21）、3.066（4-->22）、2.398（6-->23）、3.310（7-->24）、2.459（9-->25）、2.744（10-->26）、3.134（11-->27）、3.142（12-->28）、2.590（14-->29）、2.363（15-->30）、1.695（16-->31）、2.254（17-->32）、2.065（18-->34）

The third ring (16 line segments , 16 nodes) from 19 -> 20to34 -> 19 are：2.202、2.024、3.077、3.471、2.237、2.205、4.804、4.320、1.469、4.185、5.191、1.031、2.443、4.402、2.393、3.951

The junction of the second and third ring：2.276（20-->35）、3.317（22-->37）、3.271（23-->38）、2.171（25-->38）、1.684（26-->39）、2.164（29-->42）、2.408（31-->43）、2.020（23-->45）、2.845（34-->46）

The second ring (12 line segments ,12 nodes) from 35 - >36 to 46 -> 35 are：3.414、1.740、3.097、4.315、3.739、2.279、1.723、4.042、2.486、2.441、2.214、1.877

The junction of the second ring and the innermost ring: 1.820（36-->48）、1.580（42-->52）、1.973（46-->47）

Innermost ring(9 line segments ,9 nodes) from 47 -> 48to55 -> 47 are：

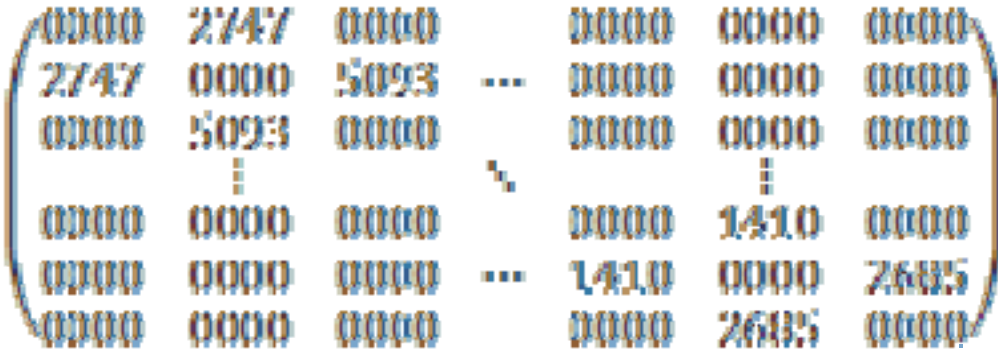
1.635、2.834、1.871、3.076、2.212、2.020、1.410、2.685、1.622

2.2.4Data processing

The connection line of the nodes in figure 2 means that there is direct access between them. Two-way arrows means that cars can travel between .We can regard figure 2 as an undirected graph, because of its adjacency matrix is symmetric.

The model selected a total of 55 nodes, the data of distance between nodes can be saved in the adjacency matrix which has 55 rows and 55 column. Since the adjacency matrix is too large, and the space is limited, this paper just list the adjacency matrix of the first three and the last three nodes, 0 means no directly access.(The unit is km in figure 2,however in the program it's convenient to express with data's fixed-point, the unit of the adjacency matrix of nodes is meters.)

The following matrix is the initial expression of distance between nodes, details can be found in the "in.in" file of the appendix, (program input data file). The initial adjacency matrix denotes that there are 81 direct access driving routes, and the matrix is symmetric.



2.3 Model construction

2.3.1 algorithm process

1) Adjacency matrix initialization. The information of the graph is represented in the adjacency matrix **map**, **dist**, and the **map** to store the initial information. At first, **dist** stored the same information as **map**, if there is a road from Vi to Vj , then map[i,j]=d, d expresses the distance between nodes. Else map[i,j]= 0, then assign **map** to **dist**.

2) Define a matrix **path** to record the information of insertion nodes, path[i,j] refers to the nodes needed to go through from Vi to Vj .initialize path[I,j]= 0.

3)Insert all the nodes into the graph, compare with the original distance, **dist**[i,j] = min(**dist**[i,j], **dist**[i,k]+**dist**[k,j]) If the value of **dist** [i, j] gets smaller, then path[I,j]=k.

4) **Dist** contains the information of the shortest route between two nodes, while **path** contains the information of the shortest route.

2.3.2 control variables illustration used in the model

a) map[i,j] denotes the initial distance from i to j, 0 means no directly access.

b) dist[i,j] refers to the shortest distance from i to j

c) path[I,j] refers to the information of the intermediate junction nodes needed to go through from i to j, 0 indicates that there is no intermediate junction point and they are connected directly.

d) k exhausts all the intermediate junction nodes between i and j

2.3.3 Floyd algorithm for establishing the formula

The state transition equation of the distance between nodes bases on the core idea of Floyd algorithm:

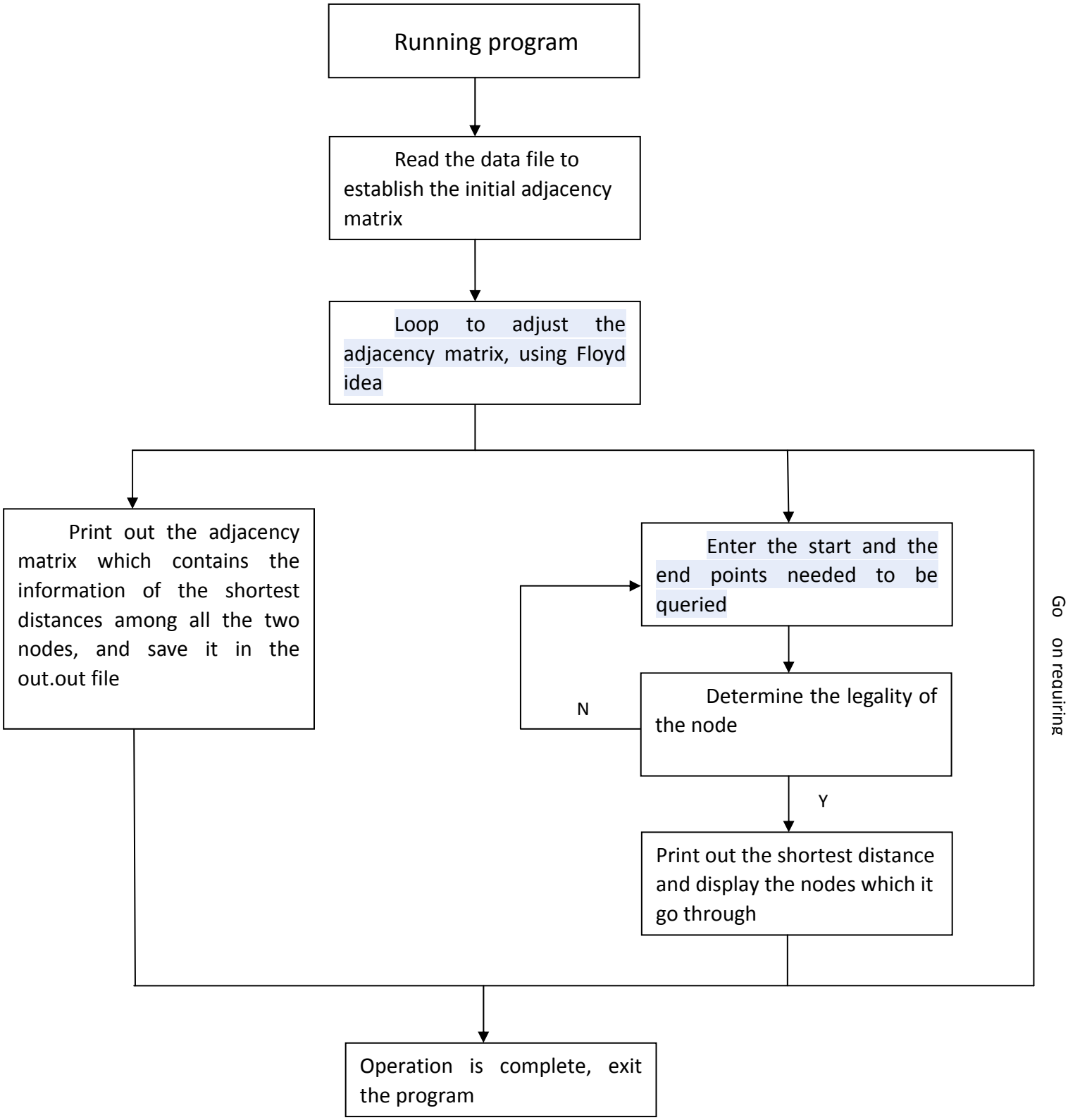
dist[i,j]:=min{dist[i,k]+dist[k,j],dist[i,j]}

The corresponding time formula under the constant speed:

time[i,j]:=dist[i,j]/30000*60(min)

(speed is 30km/h)

2.3.4 Program flow chart of the algorithm



2.4 Model improvement

As the actual road conditions in different sections can not be the same, and it is impossible to not encounter jams, especially in traffic peak time, etc. The following series of figures shows the road conditions on a certain date within the Fourth Ring, Beijing at16:00,19:00,22:00

Road condition at 16:00

Road condition at 19:00

Road condition at 22:00

red denote jam, orange denote slow and green denote smooth

It can be easily seen from the above figure that the following are congestion-prone roads:

Xizhimen Bridge, Deshengmen Bridge, Andingmen，Xibianmen Bridge，Dongbianmen Bridge，Zuoanmen Bridge，East Chang'an avenue， Huayuan Bridge， Hangtian Bridge， Suzhou Bridge， Liuli Bridge， Yuquanying Bridge， Wanfang Bridge， Muxidi Bridge， Guomao， Jingguang Bridge， Taiyanggong Bridge， Madian Bridge， Jianxiang Bridge， Wanquanhe Bridge， Yuegezhuang Bridge， Beishatan Bridge， Qinghe Bridge，

Xiaojiahe Bridge.

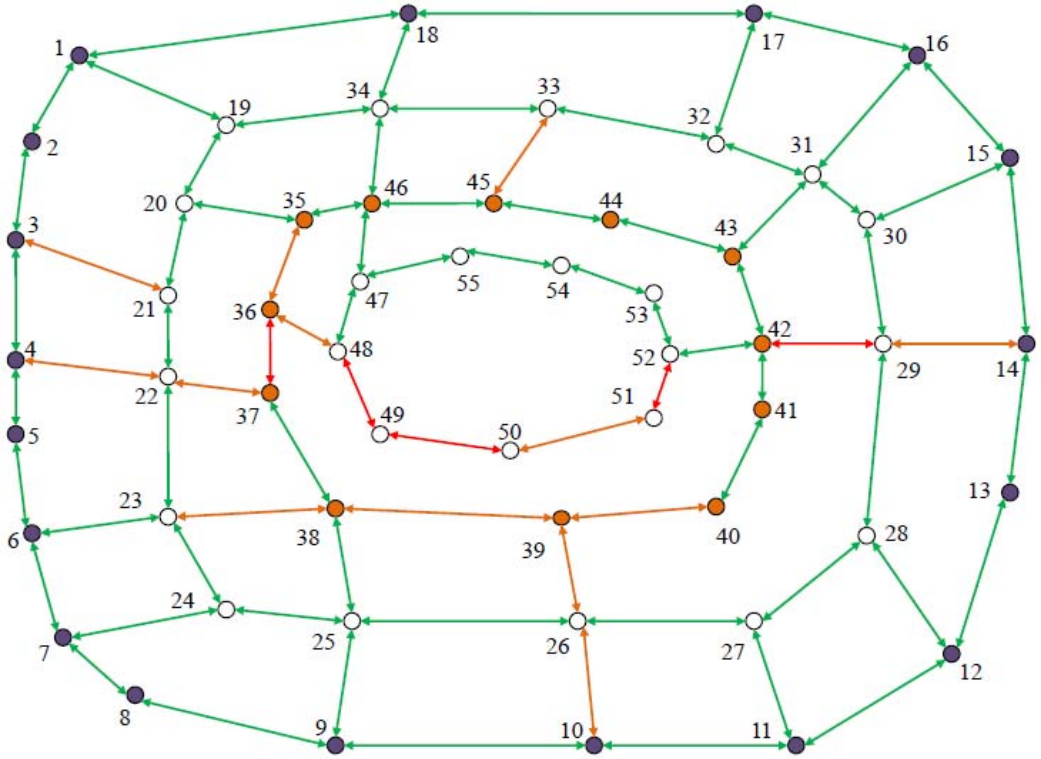
This paper decided to add the weight factor to different routes of different rings, which is divided into three levels, smooth, slow, jam. this model assumes that cars are traveling at constant speed, so we add the weight factor to identify different road conditions as distance of the path varies due to traffic. Following are the assumptions:

a) smooth corresponds 1.2

b) slow corresponds 1.4

c) jam corresponds 1.6

According to these three conditions, different colors denote different road conditions, as figure 3 shows.



The figure above use three colors to denote three different conditions, that is green for smooth, orange for slow, red for jam. Among these 81 roads, 5 are jam, 13 are slow ,the other 63 are smooth. Details are as follows:

Jam(5): 29-->42； 36-->37； 48-->49； 49-->50； 51-->52

Slow(13): 3-->21；4-->22；10-->26；14-->29；22-->37；23-->38；26-->39；33-->45； 35-->36； 36-->48； 38-->39； 39-->40； 50-->51

Others are smooth.

After adding the weight factor, the distances change correspondingly, in our algorithm realization it needs to change the corresponding control variables, i.e. the two two-dimensional arrays **map** and **dist** (control

variable description is in 2.3.2). Thus our best route choice (saved in the array **path**) will also change accordingly. Details are in the “out_weight. Out" file in the appendix. (output data file after adding weights factor and programming improvement).

The state transition equation of the distance between nodes base on the core idea of Floyd algorithm:

dist[i,j]:=min{a*dist[i,k]+b*dist[k,j],dist[i,j]}

Note: a and b is the weights of dist [i, k],dist[k,j] respectively, which is determined by what road condition they are.

Add other external factors to travel time effects, such as a traffic jam, traffic accidents, encountering traffic lights in the process of driving etc. Unify these external factors into a constant factor c,

The corresponding time formula under the constant speed:

time[i,j]:=dist[i,j]/30000*60(min)+c

3. Model results

(1)The least time and the shortest route of all nodes

In 2.2.4 section, we can get the initial adjacency matrix. By the Floyd algorithm we get the final adjacency matrix as following (this paper just list adjacency matrix of the first and the last three nodes, 0 means no directly access. The details are in the “out. out" file (program output data files) and the "out-weight. out" file for the improved model (output data

file after adding weights factor and programming improvement).There are a total of 2970 results about the shortest routes among all the two nodes of all 55 nodes. Details are in the appendix.

$$\begin{pmatrix} 0000 & 2747 & 7840 & & 16990 & 15580 & 12895 \\ 2747 & 0000 & 5093 & \cdots & 19737 & 18327 & 15642 \\ 7840 & 5093 & 0000 & & 16664 & 15624 & 12569 \\ & \vdots & & \ddots & & \vdots & \\ 16990 & 19737 & 16664 & & 0000 & 1410 & 4095 \\ 15580 & 18327 & 15254 & \cdots & 1410 & 0000 & 2685 \\ 12895 & 15642 & 12569 & & 4095 & 2685 & 0000 \end{pmatrix}$$

Adjacency Matrix without weights

0	3296.4	9408	20388	18696	15474
3296.4	0	6111.6	23684.4	21992.4	18770.4
9408	6111.6	0	20556.2	18864.2	15642.2
20388	2368.4	20556.2	0	1692	4914
18696	21992.4	18864.2	1692	0	3222
15474	18770.4	15642.2	4914	3222	0

Adjacency Matrix with weights

(2) The least time and the shortest route of specific nodes

Among the three kinds of road conditions in figure 3, we select 3 groups to analyze the results, respectively corresponding to the smooth road , the slow and the jam.

1. smooth road (1-<35)

========== Best results without weights =========

Beginning: 1

End: 35

Least time: 14.846min

Shortest distance: 7423m

Route: 1-->19-->20-->35

passed nodes: 2

========== Best results with weights =========

Beginning: 1

End: 35

Least time: 17.8125min

Shortest distance: 8907.6m

Route: 1-->19-->20-->35

passed nodes: 2

========= ========= ========= ========= ========= ========= ========== ========== =====

Conclusion: from the two results we can see that if the road condition is smooth, there will be no effect on the route-choosing strategy regardless of whether add weight factors or not.

2. Slow road (23-<26)

========== Best results without weights =========

Beginning: 23

End: 26

Least time: 18.492min

Shortest distance: 9246m

Route: 23-->24-->25-->26

Passed nodes: 2

========== Best results with weights==========

Beginning: 23

End: 26

Least time: 22.1904min

Shortest distance: 11095.2m

Route: 23-->24-->25-->26

Passed nodes: 2

========== ========== ========== ========== ========== ========== ========== ==

Conclusion: from the two results we can see that if the road condition is slow, there will be little effect on the route-choosing strategy regardless of whether add weight factors or not.

3. jam road(47-<51)

========== Best results without weights ==========

Beginning: 47

End: 51

Least time: 18.832min

Shortest distance: 9416m

Route: 47-->48->49-->50-->51

Passed nodes: 3

========== Best results with weights ==========

Beginning: 47

End: 51

Least time: 25.6472min

Shortest distance: 12823.6m

Route: 47-->55-->54-->53-->52-->51

Passed nodes: 4

========== ========== ========== ========== ========== ========== ========== ======

Conclusion: from the two results we can see that if the road condition is jam, there will be obvious effect on the route-choosing strategy regard whether add weight factors or not. Although the passed node after adding weight may be increased, a distant route which takes less time may be a better alternative to avoid traffic congestion which will bring more waiting time.

(3) Results checking

This paper uses network to test the model's correctness and feasibility.

1. smooth road(1->35)

According to the little-time principle, network shows the result: The distance is 7.3km, 13min

If we didn't add the weight factors to the model, the result shows the shortest road is 7423m, and it takes 14min to reach. That is to say, there exists certain error in our model for data measurement, but its feasibility is good, and the provided route is exact.

2. Slow road(23->26)

According to the short-time principle, network shows the result: The distance is 9.2km, 20min

If we didn't add the weight to the model, the result shows the shortest road is 9246m, and it takes 18.5min to reach, after added the weight, it will take 22min to reach. So, adding weight can improve the result.

3. Jam road(47->51)

According to the short-time principle, network shows the result: The distance is 10.3km, 23min

If we didn't add the weight to the model, the result shows the shortest road is 9416m, and it takes 18.8min to reach, after added the weight, it will take 25.6min to reach. So, adding weight can change the selection of the route.

4. Conclusion:

This paper adopts Floyd algorithm to solve the least time problem of car driving dynamically. It selects 55 nodes and it solved out the optimal solution between any two nodes. The solving process is efficient, the results are reliable and practical, and is worthy to be popularized. Original model doesn't consider

the road condition, after improvement, the new model does not only applicable to solving the least time problem, it can also be applied to vehicles transporting problems, economic management of resource allocation, task equilibrating problem to get the optimal solution. The application prospect is wide.

5. References

[1]. Zhou Jing. Stochastic equilibrium assignment model of transportation network with multi-user. The journal of Southeast University, 2001.2 (in Chinese)

[2]. Yu Li na. “Aggregate—combustion” arithmetic and its application in the query system of transit network. The journal of Liaoning Technical University 2008.7 (in Chinese)

[3]. Sun Xiao yan. Solve the transport problems of shortest route by dynamic programming. The journal of Kunming University 2010.2 (2) :223-224 (in Chinese)

[4]. XU Rui, Xu Xiao fei, Cui Ping-Yuan Dynamic Planning and Scheduling Algorithm Based on Temporal Constraint Network. The journal of Harbin Institute of Technology 2004.2 (10) :188-194

[5]. Pingxiao Hu. Application of Dynamic Programming in algorithm Design. The journal of Anhui Institute of Electronics and Information Technology 2004 (2) (in Chinese)

[6]. Qian Song Di. Operations research [M] Beijing: Tsinghua

University Press, 2002 (in Chinese)

[7]. Jiang qi Yuan. Mathematical models (third edition) Higher Education Press, 2003 (in Chinese)