Application of Dijkstra Algorithm in Optimal Route Selection Under the Background of TPACK Education Model

Fengling Wang(✉), Yan Li, Qingling Chen, and Guoxian Wang

Heihe University, Heihe, Heilongjiang, China
78623392@qq.com

Abstract. With the advent of the era of globalization and information technology, all aspects of people's lives have been deeply influenced by Internet technologies such as mobile Internet and big data technology. TPACK is presented in this paper under the background of education mode based on network data automation and other related theory, using the Dijkstra algorithm in 31 cities between tourism route design, and through the MATLAB software programming software realization route network shortest route choice, makes the tourists in the process of route choice by according to their actual demand the best travel path. So as to achieve the goal of energy conservation and environmental protection, green travel.

Keywords: Greedy algorithm · Dijkstra algorithm · Route selection

1 Introduction

With the development of economic globalization and tourism market boundaries without borders, tourism market expanded rapidly, consumer demand for the development of transportation and communications and more diverse and more changes, so people for their travel route selection is becoming more and more diversified.It is important of the following several aspects to travel route choice in design based on the Dijkstra algorithm.

First, According to the geographical location (latitude and longitude) design the shortest circuit travel scheme.

Educational Science Research Project of Heihe University: TPACK for mathematics normal university students under normal professional certification standards Research on the cultivation of teaching ability (JYZ202101).

X. Jiang (Ed.): MLICOM 2021, LNICST 438, pp. 48–66, 2022.
https://doi.org/10.1007/978-3-031-04409-0_6

Second, If the traveler starts from Harbin and stays for 3 days in each city, they can choose the air or railway (express sleeper or bullet train) to design the most economical travel booking plan on the Internet.

Third, To take a comprehensive view of money, time and convenience, set up your evaluation criteria, build a mathematical model, revise your program.

2 Application of Dijkstra Algorithm in Optimal Route Selection

In the first problem, the shortest path is related to the longitude and latitude of each city, so the longitude and latitude of each city are listed in the form of matrix, and the Dijkstra algorithm is used to find the shortest path through 31 cities, and the shortest road map is made through MATLAB software [1].

Problem two of the main economic issues related to drive cost, so the use of "timetable" software, through the actual search, getting any a city to the rest of the minimum cost data of 30 cities results when using greedy method. Greedy method is selecting a measure standard, and then pressing measure this sort of input city, a city and a sequential input. If the sum of this input and the partial optimal solution currently constituted in the sense of this measure does not yield a viable solution, the city is not added to the decomposition. This paper is to use this method to solve, and through these datas to do programming, then get about the most economic route. To find the most time-saving route and the most economical approach is the same, To find any city to the other 31 cities in the least time data, and get the most time-saving and the most economic route travel by the greedy method and programming,

2.1 Considering the Shortest Travel Route

It will pass through all the provincial capitals, municipalities directly under the central government, so we choose the longitude and latitude to solve the problem. Latitude and longitude is one of the effective methods to calculate the shortest distance between two points on earth, this article through to the city between latitude and longitude, and latitude and longitude data list matrix, using the Dijkstra algorithm to make the program, it is concluded that the shortest travel route, based on the online survey of 31 provinces data as an example, through the MATLAB software to make the shortest route map. Finally, the shortest travel routes and corresponding road maps for provincial capitals.The shortest path problem can be described as given a directed graph, also known as a network, for each arc, corresponding weights, and given two vertices in, let is a path from, when the weight of the path is defined, the sum of the weights of all the arcs in this path, denoted as. The shortest path is to find the path with the least weight among all the paths from PI to PI, that is, to find a path from PI to PI that minimizes all the paths from PI to PI in D, which is called the shortest path from PI to PI.This method is called Dijkstra's algorithm [2]. The basic principle of Dijkstra's algorithm is that if the shortest path from to, then it must also be the shortest path to. follow the formatting instructions for headings given in Table 1.

Table 1. Latitude and longitude

Number	Province	Capital	Latitude	Longitude
1	Heilongjiang province	Ha erbin	45.44	126.36
2	Jilin	Changchun	43.54	125.19
3	Xinjiang	Wulumuqi	43.45	87.36
4	Liaoning	Shenyang	41.48	123.25
5	Neimenggu	Huhehaote	40.48	111.41
6	Beijing	Beijing	39.55	116.24
7	Tianjin	Tianjin	39.02	117.12
8	Ningxia	Yinchuan	38.27	106.16
9	Hebei	Shijiazhuang	38.02	114.3
10	Shanxi	Taiyuan	37.54	112.33
11	Shandong	Jinnan	36.4	117
12	Qinghai	xinning	36.38	101.48
13	Ganshu	Lanzhou	36.04	103.51
14	Henan	Zhengzhou	34.46	113.4
15	Shanxi	Xian	34.17	108.57
16	Jiangshu	Nanjing	32.03	118.46
17	Anhui	Hefei	31.52	117.17
18	Shanghai	Shanghai	31.14	121.29
19	Sichuan	Chengdu	30.4	104.04
20	Hubei	Wuhan	30.35	114.17
21	Zhejiang	Hangzhou	30.16	120.1
22	Xizhang	Lasha	29.39	91.08
23	Chongqing	Chongqing	29.35	106.33
24	Jiangxi	Nanchang	28.4	115.55
25	Hunan	Changsha	28.12	112.59
26	Guizhou	guiyang	26.35	106.42
27	Gujian	fuzhou	26.05	119.18
28	Yunnan	Hunming	25.04	102.42
29	Guangdong	guangzhou	23.08	113.14
30	Guangxi	Nanning	22.48	108.19

In this paper, in the network, the weight of the arc represents the direct distance from to, through the use of MATLAB programming.

```
program Project1;
findroute.m
points =[
    126.36      45.44
    125.19      43.54
    87.36       43.45
    123.25      41.48
    111.41      40.48
    116.24      39.55
    117.12      39.02
    106.16      38.27
    114.3       38.02
    112.33      37.54
    117         36.4
    101.48      36.38
    103.51      36.04
    113.4       34.46
    108.57      34.17
    118.46      32.03
```

```
    117.17      31.52
    121.29      31.14
    104.04      30.4
    114.17      30.35
    120.1       30.16
    91.08       29.39
    106.33      29.35
    115.55      28.4
    112.59      28.12
    106.42      26.35
    119.18      26.05
    102.42      25.04
    121.3       25.03
    113.14      23.08
    108.19      22.48
    114.15  22.28
    ];
rand('seed',0);
R=6371.11;
for i=1:length(points(:,1))
for j=1:length(points(:,1))
    tabledis(i,j)=distance(points(i,2),points(i,1),points(j,2),points(j,1))*2*R*pi/360;
end;
end;
p=[1:length(points(:,1))]';
optlength=routelength(p,tabledis)

for jjj=1:6
    for kkk=1:60
```

```
        if mod(kkk,20)==0
            disp(['kkk=' num2str(kkk)]);
        end;
        p=randperm(length(points(:,1)))';
        p=optimizeroute(p,tabledis);
        if optlength>routelength(p,tabledis)+1e-10
            figure(1);
            hold off;
            plot([points(p,1); points(p(1),1)],[points(p,2);points(p(1),2)],'k','linewidth',2);
            hold on;
            plot(points(p,1),points(p,2),'r.','markersize',30);
            axis([85 128 15 47]);
            text(95,45,['No. ' num2str(jjj) ' : '
num2str(routelength(p,tabledis))],'fontsize',20);
            drawnow
            optlength=routelength(p,tabledis);
            print('-dpng',['r' num2str(jjj) '.png']);
            break;
        end
    end
end;
```

firstordercc.m

```
function r=firstordercc(p,d,tabledis)
lold=routelength(p,tabledis);
lll=length(p(:,1));
lengtharray=zeros(lll,1);
for i=1:lll
    p0=[p;p];
    p0(i:i+d,:)=flipud(p0(i:i+d,:));
    p0=p0(i:i+lll-1,:);
```

```
    lengtharray(i)=routelength(p0,tabledis);
end;
[mmm,i]=min(lengtharray);
    p0=[p;p];
    p0(i:i+d,:)=flipud(p0(i:i+d,:));
    p0=p0(i:i+lll-1,:);
if lold>mmm
    r=p0;
else
    r=p;
end;
```

optimizeroute.m

```
function p=optimizeroute(p,tabledis)
for j=1:100
    alength=routelength(p,tabledis);
for i=length(p):-1:1
 p=firstordercc(p,i,tabledis);
end
if alength-routelength(p,tabledis)<1e-10
    break;
end
end;
```

routelength.m

```
function r=routelength(p,tabledis)
r=tabledis(p(1),p(end));
for i=1:length(p)-1
    r=r+tabledis(p(i),p(i+1));
end;
```

Draw the following Fig. 1 and get the shortest path of 15591.0705 km.

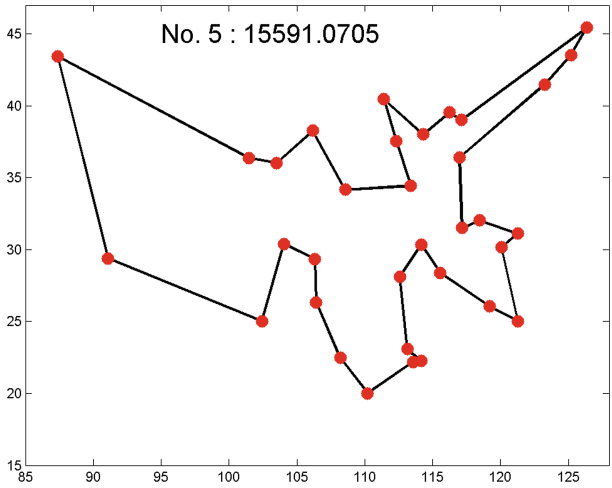

Fig. 1. The shortest path

Namely shortest route is: from Harbin to changchun to shenyang to jinan, tohefei, nanjing to Shanghai to hangzhou to Taipei to fuzhou to nanchang to wuhan, changsha, guangzhou to haikou to nanning to guiyang to chongqing and chengdu to kunming to Lhasa, urumqi, xining to lanzhou, yinchuan to xi'an to zhengzhou, taiyuan to Hohhot to shijiazhuang to Beijing to tianjin to Harbin.

Advantages of the model: since the data used in this problem is only the longitude and latitude of each city, which is not changed by other things, so there are no human and geographical factors, and the result is very accurate.

Disadvantages of the model: the spherical distance is more accurate to calculate the distance between two places, while the actual data used is the straight-line distance between two places, so there will be errors in the results, but it will not affect the travel route.

2.2 Choice of the Cheapest Travel Route by Greedy Algorithm

This paper requires to design a travel booking plan on the Internet for Mr. Zhou to start from Harbin on May 1st, pass through 34 provincial capitals, municipalities directly under the central government, and finally return to Harbin, and stay in each city for 3 days. According to the actual situation, the most economical travel route between any one city and the other 30 cities is first found, and the matrix is listed. Using this matrix, the program is made by greedy method, and then the result is made by C++. The greedy method is an improved grading method. It begins with a travel salesman problem description and selects a metric. The N input cities are then sorted by this metric, one at a time. If the sum of this input and the partial optimal solution currently constituted in the sense of this measure does not yield a viable solution, the city is not added to the

decomposition. The hierarchical processing method which can get the optimal solution in a certain sense of measurement is called greedy method, and this paper is to use the greedy method to solve. First of all, use the "train schedule" to find out the most economical travel cost data between any city and other cities, the most time-saving travel data, the most convenient travel data, and list the data table.

List the matrix through the data table, the matrix using greedy algorithm, through C++ editing

```
program Project1;
#ifndef AdjTWGraph_H
#define AdjTWGraph_H
#include <vector>
#include <iostream>
using namespace std;
const int MaxV=100;
struct Edge
 {
   int dest;
   int weight;
   Edge * next;
    Edge(){}
    Edge(int d,int w):dest(d),weight(w),next(NULL){}
};
struct item
 {   int data;
   Edge * adj;
};
```

```
class AdjTWGraph
 {
private:
   item vertices[MaxV];
   int numV,numE;
public :
   AdjTWGraph();
   ~AdjTWGraph();
    int NumV(){return numV;}
    int NumE(){return numE;}
   int GetValue(const int i);
   int GetWeight(const int v1,const int v2);
   void InsertV(const int & vertex);
   void InsertE(const int v1,const int v2,int weight);
   friend ostream& operator<<(ostream& os,  AdjTWGraph & m)
    {    for (int i = 0; i < m.numV ; i++)     {
          for (int j = 0; j < m.numV; j++)
             os << right << m.GetWeight(i,j) << " ";
          os << endl;
       }
       return os;
   }
   friend istream& operator>>(istream& is, AdjTWGraph & m)
    {   int t;
       for (int i = 0; i < m.NumV(); i++)
          for (int j = 0; j < m.NumV(); j++)
```

```
        {
          is >> t;     m.InsertE(i,j,t);
        }
      return is;
   }
};
AdjTWGraph::AdjTWGraph()
 {
   for(int i=0;i<MaxV;i++)     vertices[i].adj=NULL;
   numV=0;numE=0;
}
AdjTWGraph::~AdjTWGraph()
 {
   for(int i=0;i<numV;i++)
    {
      Edge * p=vertices[i].adj,*q;
      while(p!=NULL)
       {
         q=p->next;delete p;p=q;
      }
   }
}
 int AdjTWGraph::GetValue(const int i){    return vertices[i].data;  }
int AdjTWGraph::GetWeight(const int v1,const int v2)
 {
   Edge *p=vertices[v1].adj;
```

```
    while(p!=NULL && p->dest<v2) p=p->next;
     if(v2!=p->dest)    {    return 0;    }
    return p->weight;
}
 void AdjTWGraph::InsertV(const int & v) { vertices[numV].data=v; numV++;  }
void AdjTWGraph::InsertE(const int v1,const int v2,int weight)
 {
    Edge * q=new Edge(v2,weight);
    if(vertices[v1].adj==NULL) vertices[v1].adj=q;
    else
     {
        Edge *curr=vertices[v1].adj,*pre=NULL;
         while(curr!=NULL && curr->dest<v2)    {    pre=curr;curr=curr->next;    }
         if(pre==NULL){    q->next=vertices[v1].adj;vertices[v1].adj=q;        }
         else    {    q->next=pre->next;pre->next=q;    }
    }
    numE++;
}
 #endif
//---------------------------------- tsp.cpp文件-----------------------------------------------
#include "AdjtwGraph.h"
#include <fstream>
#include <vector>
#include <algorithm>
#include <ctime>
#include <queue>
```

```
using namespace std;
ofstream fout("out.txt");
int N;
AdjTWGraph g;
struct Node
 {   int currentIndex;
    int level;
    Node * previous;
     Node(int L = 0, int V = 0, Node *p = NULL):level(L),currentIndex(V), previous(p) {}
};
class TspBase
 {
protected:
    vector<int> currentPath;
    vector<int> bestPath;
    int cv;
    int bestV;
    Node * root;

    int SumV();
    void EnumImplicit(int k);
    void BackTrackImplicit(int k);

    void EnumExplicit(Node * r);
    void BackTrackExplicit(Node * r);
    void FIFOBB();
```

```
    bool Valid(Node *p,int v)  //
        {    bool flag = true;
          for(Node *r = p; r->level > 0 && v; r = r->previous)  flag = r->currentIndex !=v;
          return flag;
        }
    void StoreX(Node * p) //
        {for(Node *r = p; r->level >0 ; r = r->previous )
{    currentPath[r->level-1] = r->currentIndex;    }
        }
    void Print();
public:
     TspBase(){currentPath.resize(N);    bestPath.resize(N);    }
~TspBase(){currentPath.resize(0);bestPath.resize(0);}

    void TspEnumImplicit();
    void TspBackTrackImplicit();

    void TspEnumExplicit();
    void TspBackTrackExplicit();

    void TspBB();

    void TspGreedy();

    void DataClear(bool flag)
```

```
    {   currentPath.resize(N);       bestPath.resize(N);
        if(flag)       { Node * p=root,*q;
                  while(p!=NULL) {q=p->previous; delete p; p=q;}
        }
    }
};
void TspBase::TspGreedy()
 {    fout<<"TspGreedy ........"<<endl;
bestV = 0;
    vector<int> NEAR(N); //
    NEAR[0] = -1;
    for (int i = 1; i < N; i++)
      NEAR[i] = 0;
    bestPath[0] = 1;
    int t;
    for (int s = 1; s < N; s++)
     {
      int j = 1;
      while (j < N && NEAR[j] < 0)
         j++;
      int K = j;
      for (int k = j + 1; k < N; k++)
        if (NEAR[k] >= 0 &&  g.GetWeight(k,NEAR[k]) < g.GetWeight(j,NEAR[j]))
             j = k;
      bestPath[s] = j + 1;
      bestV +=g.GetWeight(j,NEAR[j]);
```

```
    NEAR[j] = -1;
    for (k = K; k < N; k++) //调整NEAR值
      if (NEAR[k] >= 0)
        NEAR[k] = j;
    t = j;
  }
  bestV += g.GetWeight(t,0);
  fout<<"the shortest path is  ";
  for(unsigned w = 0; w < N; w++)
    fout<<bestPath[w] <<"--";
  fout<<"1"<<endl;
  fout<<"minimum money is  "<<bestV<<endl;      /* the shortist time  the most
convience */
}

int main(int argc, char* argv[])
 {  int m,n;
  ifstream fin("data.txt");
  if(fin.bad()) return 1;
  fin >> m >> n;
  N = n;
  for(int i=0;i<N;i++)  g.InsertV(i);
  fin >> g;
  TspBase it;
  it.TspGreedy();    it.DataClear(false);
  return 0;
}
```

The shortest path is 1--2--3--34--4--29--30--31--32--33--6--5--8--7--25--26--27--28--24--21--22--23--17--18--19--20--14--15--16--13--10--11--12--9--1

2.3 Choice of the Most Time-Saving Path by Greedy Algorithm

After query to get about any city to the other 33 cities in the bus the least time, with the data listed matrix, the matrix using greedy algorithm, through C++ editing

TspGreedy .....

The shortest path is 1--3--11--5--8--10--31--30--27--12--6--9--15--16--25--22--19--20--26--28--21--14--17--18--13--7--4--33--32--34--2--23--29--24--1

The shortist time is 3621.

With the original 31 provincial capital city, municipality directly under the central government, Harbin to shenyang to nanchang, to hefei, to hangzhou to taiyuan to yinchuan to wuhan to nanjing to xining to kunming, guiyang, chongqing to lanzhou to xi'an, chengdu, guangzhou to haikou, nanning to changsha to Shanghai to jinanto Beijing to shijiazhuang to day Beijing to changchun to Lhasa to zhengzhou to urumqi to Harbin to plus three days stay in each city, a total of 3621 min.

Advantages of the model: the data used in this question are obtained by consulting the train schedule, the data is more accurate, and the most economical, time-saving and convenient results of tourism are practical. If you want to travel all over China and save time, you can start from any starting point according to the travel route in this article and return to Harbin to ensure time saving, economy and convenience.

Disadvantages of the model: in this problem, a few communication methods are replaced by aircraft. Since the price of air tickets varies greatly at different times, the airfare used in this problem is not necessarily accurate and the flight time is not fixed, which leads to errors in the optimal economic value obtained.

3 A Comparative Study

In the first step, the Dijkstra algorithm is used to program the five programs together, and the five shortest route diagrams are obtained. The fifth diagram is the most accurate, with a total length of 15591 km, which means the shortest route of the first step and the corresponding road map are made. This article has solved the most economical, the most time-saving and the most convenient travel route, that is, the most economical cost is 11,789 yuan, the most time-saving time is 2431 min. Due to this problem are basic factors to consider in the process of tourism, so this article petitions walking routes using range is very wide, such as the first question in the shortest route problem, if you have wanted to travel around the provinces, municipalities directly under the central government, or on one section of the route, to walk for the shortest route, can be implemented according to the result of this paper. Of course, if you want to save time, economy and convenience, you can choose the route according to the relevant results made in this paper.

In this paper, we use the latitude and longitude data are obtained by the query, the journey time, cost and the number of transfer is done with train schedules and aircraft moment query website of the query, due to the different transit times of the day and month of a year is different, can lead to drive the time error, but the total time of error will not affect tourism, because Mr. Wang to stay for three days in each city, so the whole travel for a long time. The first three questions in this paper use the data investigated, through C++ programming, to make the results, the results are more accurate. It will be cheaper, less time-consuming and more convenient to follow these routes. Based on the survey between the most direct the most economic data, because a lot of data in "timetable", passes through many data are compared to choose the smallest data, which will result in this paper by use of the data has a tiny error, but due to a wide range of tourism, the cost will be very big, so the error of the results won't have influence, not as a problem to consider. Due to the different factors considered in this paper, the tourist routes are also different. These routes are suitable for the problem of time-saving, economic and convenient and the shortest route in the process of tourism. It can be compared with the various walking routes in this paper to take the corresponding tourist routes.

4 Conclusion

This paper takes the cheapest matrix as the equation to establish the model that Mr. Zhou starts from Harbin, travels to provincial capitals, municipalities directly under the central government, and finally returns to Harbin. Namely the most economical, the most time-saving, the most convenient route. This paper uses a lot of computer programming, programming process used in the data error is very small, so after programming to make the route is more accurate.

According to the longitude and latitude of each city, the model lists the relevant longitude and latitude data matrix, and uses MATLAB programming to make the shortest path problem in the first question. According to the investigation of route, ticket price and transfer times, the model lists the cost, time and transfer number between two places into a matrix, and it is convenient to consider the least number of transfer times on the way. Route due to the weather changes, the temporary adjustment, finally there is a certain price volatility, and according to the problem of constraint conditions, to seek the most economical route, between basic train, does not train to replace with aircraft routes, fares will be greatly change, here we only consider the optimal scheme. Findings of the data into a data table, and use Matlab and C++ programming to make the trip route, and the corresponding road map. After the analysis of the problem, this paper designed the corresponding Dijkstra algorithm, MATLAB algorithm, greedy method, etc., and found the corresponding data, in the form of data matrix used in the algorithm, because the accuracy of the data is higher, that is, the reliability of the results is higher. And because the problem of the model is more comprehensive, combined with the actual situation to solve the problem, so the established model can be closely connected with the actual, so that the model has a good universality and generalization.

References

1. Nikolić, M., Teodorović, D.: Transit network design by bee colony optimization. Expert Syst. Appl. **40**(15), 5945–5955 (2013)
2. Nassi, C.D., da Costa, F.C.D.C.: Use of the analytic hierarchy process to evaluate transit fare system. Res. Transp. Econ. **36**(1), 50–62 (2012)