山东农业大学学报(自然科学版),2016,47(6):927-931 VOL.47 NO.6 2016
Journal of Shandong Agricultural University (Natural Science Edition) doi:10.3969/j.issn.1000-2324.2016.06.023

改进的 Dijkstra 算法在多点最优路径组合中的应用

——以大学生出游 App 为例

唐彬文,卓雨嘉,陈小青

集美大学 诚毅学院, 福建 厦门 361021

摘 要: 随着我国经济发展和社会进步，人们满足了日益丰富的物质生活的同时更多追求精神上愉悦，因此，旅游业方兴未艾。面对如火如荼的旅游在线平台的发展，如何为游客提供多点组合的最优观光路径，成为当下迫切的需求。本文摈弃传统的点对点导航模式，利用改进的 Dijkstra 算法设计和开发了一个具有旅游线路推荐功能的大学生出游 App，充分利用现有的移动平台载体，为游客方便快捷地规划合理的游览线路。

关键词: Dijkstra 算法; 旅游; App

中图法分类号: U284.37 **文献标识码**: A **文章编号**: 1000-2324(2016)06-0927-05

The Application of Improved Dijkstra Algorithm in the Selection for Optimal Multi-point Traveling Routes

—Taking the traveling App for college students as a case

TANG Bin-wen, ZHUO Yu-jia, CHEN Xiao-qing

Chengyi College/Jimei University, Xiamen 361021, *China*

Abstract: Along with the economical development and social progress in our country, people not only meet the increasingly rich material life but also pursuit the more spiritual pleasure, therefore, tourism industry is in the ascendant. In the face of the flourishing tourism online platform development, it is pressing how to provide visitors with more combination of the optimal traveling routes. Away from the traditional pattern of point-to-point navigation, this paper used the improved Dijkstra algorithm to design and develop an App with the recommending function for college students to fully use the existing mobile platform carriers to provide reasonable routes for tourists conveniently and quickly.

Keywords: Dijkstra algorithm; traveling; App

Dijkstra 方法是目前公认最好的最短路径计算方法，是由 Dijkstra 于 1959 年提出的。Dijkstra 算法是典型最短路径算法，用于计算一个节点到其他所有节点的最短路径[1]。主要特点是以起始点为中心向外层层扩展，直到扩展到终点为止。Dijkstra 算法的基本思想是源点 V_0 出发，逐步向外扩张，寻找最短路径直至终点 V_n 的过程。经典的 Dijkstra 算法只能分别计算出源点（出发点）到各个点的最短路径。但是现实的旅游线路需求不是点对点的，经常是多点的集合。那么针对多点最优路径组合，经典的 Dijkstra 算法是无法计算的。本项目运用面向对象的程序设计思想，前端设计采用 Html5 风格设计，后台开发采用 php+mysql+apache 组合，最后使用 Cordova 进行 App 的封装。App 核心模块功能旅游线路推荐模块是通过改进 Dijkstra 算法，经过多次循环嵌套、迭代和数组遍历等方法推算出多点的最优路径组合。

1 算法思想

1.1 前提假设

（1）所有节点组成的集合图是赋权无向图。即 V_{n-1} 到 V_n 之间的距离与 V_n 到 V_{n-1} 之间的距离相等；
（2）V_0 为源点（出发点）；
（3）所有路径的组合必须回到源点，形成回路；
（4）为了便于输出结果的解读，算例使用的节点个数为 5，包括源点 V_0。

1.2 算法逻辑

（1）经典 Dijkstra 算法的引入；
（2）改进算法，分别计算出包括源点在内的所有点之间的最短路径；
（3）将最短路径值和最短路径经过的点集分别存入多维数组 Djz 和 Ejz；

收稿日期: 2016-02-10 **修回日期**: 2016-03-05
基金项目: 福建省 2015 年省级大学生创新创业训练计划项目:大学生出游网络服务平台(201513471030)
作者简介: 唐彬文(1982-),男,硕士,讲师,研究方向:管理信息系统和电子商务. E-mail:tom414@jmu.edu.cn

数字优先出版:2016-11-07 http://www.cnki.net

（4）根据游客选择的景点（包括源点）构成一个点集 V_x，$V_x \in V_n$。(n 为景点总数，n 包括源点)；
（5）使用循环迭代判断 $V_{i-1} \rightarrow V_i$ 之间的路径是否包括第三点 $j(i \in x; j \in x)$，如果有则从 V_x 中剔除；
（6）最终得到的点集 V_m 就为最优路径组合($m \in x$)；
（7）m 个节点依次排列形成组合（$V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow \ldots \rightarrow V_m$），按照点对点组合分别从 Djz 和 Ejz 获取值，组成最优路径值和最优路径组合点集。

2　算法实现

2.1　算法案例

算法通过实测 30 多个算例，能够满足 20 个节点以内的最优组合计算。本项目为周边游项目（短期旅游项目），为了良好的用户体验和论文阐述清晰，便以厦门地区四个主要景点的实际距离赋权为算例。图 1 是以四个景点构建的赋权无向图。并将节点之间的距离存入初始矩阵。

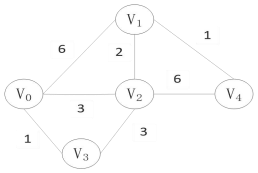

图 1 赋权无向图
Fig.1 Undirected weighted graph

0	6	3	1	0
6	0	2	0	1
3	2	0	3	6
1	0	3	0	0
0	1	6	0	0

图 2 初始矩阵
Fig.2 The initial matrix

2.2　构建类计算最短路径和最短路径经过的点集

2.2.1 改进的 Dijkstra 算法

```
Class Dijkstra
{…
Public function calculate(){
for($i=0;$i<$this->node;$i++)
{  if($this->G[$this->source][$i]>0)
{  $d[$i] = $this->G[$this->source][$i];
}  else
{  $d[$i] = 1000000;//不相邻的节点初始值为无穷大，用 1000000 表示}
if($i==$this->source)     $d[$i]=0; //以自身为源点，其距离值为 0
}  ……
```

2.2.2 初始算法解释

源点变量 Source（包括出发点在内的节点集合）是变化的范围为[0，4]；
当 Source=0 时对数组$d 进行赋值，$d=(0,6,3,1, 1000000)；
当 Source=1 时对数组$d 进行赋值，$d=(6,0,2, 1000000,1)
以此类推

```
……
$V = Array();
for($i=0;$i<=$this->node-1;$i++)
$V[$i]=$i;
for($l=0;$l<$this->node-1;$l++)
{//  查找剩余节点中距离源点最近的节点
$min = 100000; $min_v = 0;
for each($V as $key=>$value)
{ if($d[$value] < $min)
{ $min = $d[$value];
$min_v = $value;
}  }
Array_splice($V,array_search($min_v,$V),1);
for each($V as $key=>$value)
```

```
{ if($this->G[$min_v][$value]!=0&&$d[$value]>$d[$min_v]+$this->G[$min_v][$value])
{ $d[$value] = $d[$min_v]+$this->G[$min_v][$value];
$E[$value]=$E[$min_v];
Array_push($E[$value],$min_v);
} } }    }……
```

2.2.3 最优点集算法解释

①用数组$V 存放包括源点在内的所有节点；

②数组遍历距离源点最近的节点 *X*，并从数组$V 中，去除该节点;

③比较其他点 *Y* 与源点的距离$d[*Y*]是否大于$d[*X*]（*X* 与源点距离）+$G[*X*][*Y*](节点 *X* 与 *Y* 的距离)，是则用该距离存入最短路径数组$d,同时更新最短路径组合覆盖原有的二维数组$E[$value]，并将该节点的编号添加在$E[$value]的尾部。

④循环以上操作则可以得出：

$d=Array ([0] => 0 [1] => 5 [2] => 3 [3] => 1 [4] => 6)

$E=Array ([0] => Array ([0] => 0) [1] => Array ([0] => 0 [1] => 2) [2] => Array ([0] => 0) [3] => Array ([0] => 0) [4] => Array ([0] => 0 [1] => 2 [2] => 1))

例如：V_0 到 V_4 的最短距离为：$d [4]=6；经过的路径为$E[4]=$V_0 \rightarrow V_2 \rightarrow V_1$ 。

2.2.4 节点间最短路径值和最短路径经过节点集的计算

```
$jz=Array(array(0,6,3,1,0), Array(6,0,2,0,1),
Array(3,2,0,3,6), array(1,0,3,0,0), Array(0,1,6,0,0));
$node=5;$Djz=Array(array());$Ejz=Array(array(array()));
for($source=0;$source<=4;$source++){
$D = new Dijkstra($jz,$node,$source);
$D->calculate();
Array_push($Djz,$D->distance);
Array_push($Ejz,$D->E);
}……
```

多源点算法解释：分别以 0~4，五个节点为源点求出与其他点的最短距离存入二维数组$Djz，和最短路径节点集存入三维数组$Ejz。

2.2.5 结果输出及解释

$Djz 为：

Array ([0] => Array ([0] => 0 [1] => 5 [2] => 3 [3] => 1 [4] => 6) [1] => Array ([0] => 5 [1] => 0 [2] => 2 [3] => 5 [4] => 1) [2] => Array ([0] => 3 [1] => 2 [2] => 0 [3] => 3 [4] => 3) [3] => Array ([0] => 1 [1] => 5 [2] => 3 [3] => 0 [4] => 6) [4] => Array ([0] => 6 [1] => 1 [2] => 3 [3] => 6 [4] => 0))

以上值转化为多点最优矩阵为：

0	5	3	1	6
5	0	2	5	1
3	2	0	3	3
1	5	3	0	6
6	1	3	6	0

图 3 最优矩阵

Fig.3 Optimal matrix

三维数组$Ejz 为：

Array ([0] => Array ([0] => Array ([0] => 0) [1] => Array ([0] => 0 [1] => 2) [2] => Array ([0] => 0) [3] => Array ([0] => 0) [4] => Array ([0] => 0 [1] => 2 [2] => 1)) [1] => Array ([0] => Array ([0] => 1 [1] => 2) [1] => Array ([0] => 1) [2] => Array ([0] => 1) [3] => Array ([0] => 1 [1] => 2) [4] => Array ([0] => 1)) [2] => Array ([0] => Array ([0] => 2) [1] => Array ([0] => 2) [2] => Array ([0] => 2) [3] => Array ([0] => 2) [4] => Array ([0] => 2 [1] => 1)) [3] => Array ([0] => Array ([0] => 3) [1] => Array ([0] => 3 [1] => 2 [2] => 1) [2] => Array ([0] => 3) [3] => Array ([0] => 3) [4] => Array ([0] => 3 [1] => 2 [2] => 1)) [4] => Array ([0] => Array ([0] => 4 [1] => 1 [2] => 2) [1] => Array ([0] => 4) [2] => Array ([0] => 4 [1] => 1) [3] => Array ([0] => 4 [1] => 1 [2] => 2) [4] => Array ([0] => 4)))

输出结果解释：例如$Ejz[0][1]=(0,2),表示到达 V_1 的最短路径点集为(V_0, V_2)。

2.3 多点最优组合计算

2.3.1 最短距离算法

```
Class best plan{
……
Function sourced($x,$y,$Ejz,$jz){ //计算节点之间最短距离的值
$sum=0;
for($i=0;$i<count($Ejz[$x][$y])-1;$i++)
{$sum=$sum+$jz[$Ejz[$x][$y][$i]][$Ejz[$x][$y][$i+1]];  }
$sum=$sum+$jz[$Ejz[$x][$y][$i]][$y]; return ($sum);
}……
```

算法解释：三维数组$Ejz[*x*][*y*][*i*]，*x* 与 *y* 表示两个节点，第三维 *i* 表示它们之间最短路径经过的节点集。用循环计算出两点之间的最短路径$sum。

2.3.2 交叉节点剔除算法

```
Function source all(){
$Us=$Vs;//存储剩余的节点
$Os=$Vs;//存储初始值
$n=count($Vs);
for ($h=0;$h<$n;$h++){
for($i=1;$i<$n;$i++)
{Unset($Us[0]);
if($h<>0)
Unset($Us[$h]);
Unset($Us[$i]);
$new Us=array_values($Us);
for($j=0;$j<count($newUs);$j++)
{
if(in_array($newUs[$j],$Ejz[$Os[$h]][$Os[$i]]))
{
for each($Vs as $key=>$value)
{if($value==$newUs[$j])
Unset($Vs[$key]); }    }    }
$Us=$Os;}}
$this->new Vs=array_values($Vs);}……
```

2.3.3 算法解释

(1) 根据游客提交的景点组合作为节点存入数组$Vs，$Vs=(0,a,b,c,d),0 为源点，a,b,c,d 为景点对应的节点（0 为源点为必选项，其余节点可任意组合）；

(2) 将$Vs 付给$Os，因为$Vs 将进行删除重复节点操作，需要一个完整的初始值做重复的遍历；故定义$Us 数组存放经过删除操作的$Vs 数组；

(3) 通过三层嵌套的数组迭代，剔除路径交叉时的重复节点。依次剔除$Vs 中两个节点的 *x*, *y*，并获取对应三维数组$Ejz[*x*][*y*]的值，遍历剩余节点数组$Us，判断是否有节点与$Ejz[*x*][*y*]中的节点重复，有将其从$Vs 中删除。

(4) 以上代码段若$Vs=Array(0,1,2,3,4)，输出结果为 Array ([0] => 0 [1] => 3 [2] => 4)，即$newVs= Array(0,3,4)。

2.3.4 多点最优路径组合算法

```
Function best(){
//存储路径的节点集合
$programme=Array(array());
//定义总距离 Sum all 存储总距离的集合
```

```
$sumall=0;
$n=count($newVs);
for($i=0;$i<$n-1;$i++){
for($j=0;$j<count($Ejz[$newVs[$i]][$newVs[$i+1]]);$j++)
Array_push($programme,$Ejz[$newVs[$i]][$newVs[$i+1]][$j]);
$sum all+=$this->sourced($new Vs[$i],$new Vs[$i+1], $Ejz, $jz);
}
for($j=0;$j<count($Ejz[$new Vs[$n-1]][0]);$j++)
Array_push($programme,$Ejz[$newVs[$n-1]][0][$j]);
$sumall+=$this->sourced($newVs[$n-1],0,$Ejz,$jz);
Print_r($sum all);
Array_shift($programme);
$programme[]=0;//补充源点 0，形成回路
Print_r($programme); } }
……
```

2.3.5 算法解释

(1) 方法 best()用于计算最优的距离和路径，假设从源点 V_0 出发并回到源点 V_0；

(2) 调用三维数组$Ejz，用 for 循环分别将$new Vs 中的 $V_0 \rightarrow V_1$，$V_1 \rightarrow V_2$，…$V_{n-2} \rightarrow V_{n-1}$ 的节点距离和节点路径累加存入总距离$sum all 和节点集合二维数组$programme；

(3) 调用三维数组$Ejz，将 $V_{n-1} \rightarrow V_0$ 两节点的最短距离和最短路径节点集合分别加入总距离$sum all 和节点集合二维数组$programme。

2.4 多点最优路径组合计算结果

表 1 不同组合的最短距离

Table 1 The shortest distraction in different arrays

$Vs（选中节点） Links in selection for traveling routes	最短距离 The shortest distance
Array(0,1)	10
Array(0,1,3)	11
Array(0,1,2,3)	11
Array(0,1,2,3,4)	13

表 2 不同组合的最优路径

Table 2 The optimal routes in different arrays

最优路径节点集合　The optimal links set
Array ([0] => 0 [1] => 2 [2] => 1 [3] => 2 [4] => 0)
Array ([0] => 0 [1] => 2 [2] => 1 [3] => 2 [4] => 3 [5] => 0)
Array ([0] => 0 [1] => 2 [2] => 1 [3] => 2 [4] => 3 [5] => 0)
Array ([0] => 0 [1] => 3 [2] => 2 [3] => 1 [4] => 4 [5] => 1 [6] => 2 [7] => 0)

(1) Array(0,1,2,3,4)为选中所有的景点，得出最短路径为 13；

(2) 最优路径节点集合为：$V_0 \rightarrow V_3 \rightarrow V_2 \rightarrow V_1 \rightarrow V_4 \rightarrow V_1 \rightarrow V_2 \rightarrow V_0$

(3) 因为节点 V_2 属于重复节点，故(0,1,3)和(0,1,2,3)两种组合得到结果是一样的。

3 结束语

通过利用 PHP+MYSQL+APACHE 技术设计和开发出了一个以大学生及其家人团体为目标客户群的 App，希望以此 App 构建的平台能给那些计划旅行，对旅游线路组合信息检索具有需求的学生或家长用户提供帮助。程序内核是通过改进 Dijkstra 算法构建多点最优路径组合模型，并具体实现其应用功能。经过实测，能够满足 20 个节点以内的多点赋权无向图的优化组合运算。不仅在技术上取得突破，而且有效弥补了 App 应用在该领域的空缺。

参考文献

[1] Hofner P, Moller B. Dijkstra, Floyd and Warshall meet Kleene[J]. Formal Aspects of Computing, 2012,24(4-6):459-476

[2] Chang JR, Jheng YH, Chang CH, *et al.* An Efficient Algorithm for Vehicle Guidance Combining Dijkstra and A* Algorithm with Fuzzy Inference Theory[J]. Journal of Internet Technology, 2015,16(2):189-200

[3] Payette S. Hopper and Dijkstra: Crisis, Revolution, and the Future of Programming[J]. IEEE Annals of the History of Computing, 2014,36(4):64-73

[4] Daylight EG, Wirth N, Hoare T, *et al*. The Dawn of Software Engineering: From Turing to Dijkstra[J]. IEEE Annals of the History of Computing, 2014,36(1):71-73

[5] 王树西,吴政学.改进的Dijkstra最短路径算法及其应用研究[J].计算机科学,2012,39(5):223-228

[6] 潘若愚,褚 伟,杨善林.基于Dijkstra-PD-ACO算法的大城市公交线路优化与评价方法研究[J].中国管理科学,2015,23(9):106-115