

基于矩阵自定义运算的 Floyd 改进算法

赵礼峰, 黄奕雯

(南京邮电大学理学院, 江苏南京 210046)

摘要: 解决最短路问题的算法层出不穷, 其中最经典的要数 Dijkstra 算法和 Floyd 算法。但 Dijkstra 算法只能得出一对节点间的最短距离, 而 Floyd 算法计算过程十分繁琐。为解决这两种经典算法中的缺陷, 提出一种基于矩阵自定义运算的 Floyd 改进算法。该算法通过自定义矩阵运算得出一个表示两两节点间距离的路权修正矩阵, 再用路权修正矩阵与原距离矩阵进行比较, 选择两矩阵中对应较小元素组成当前最短路权矩阵, 再通过有限次的迭代, 从而得到各顶点间的最短路。通过 MATLAB 仿真, 将该算法推广到随机大规模复杂网络中, 通过运行时间折线图表明, 该算法在节点达到一定数量后运行速度明显优于传统算法, 且在稀疏网络中运行效率非常高, 说明了该算法的有效性。最后, 通过具体应用说明了该算法的实用性。

关键词: 最短路问题; Floyd 算法; 矩阵自定义运算; MATLAB; 稀疏网络

中图分类号: TP301.6

文献标识码: A

文章编号: 1673-629X(2016)10-0041-04

doi: 10.3969/j.issn.1673-629X.2016.10.009

Improved Floyd Algorithm Based on Customized Matrix Operations

ZHAO Li-feng, HUANG Yi-wen

(College of Mathematics and Physics, Nanjing University of Posts and Telecommunications,
Nanjing 210046, China)

Abstract: The algorithm to solve the shortest path problem is endless, and the algorithm of Dijkstra and Floyd is the most typical. However, the Dijkstra algorithm can only get the shortest distance between a pair of nodes, and Floyd algorithm is very cumbersome. For solving their defects in two classical algorithms, an improved Floyd algorithm is proposed based on matrix custom operation. It obtains a modified matrix between two nodes by using a customized matrix calculation, and then compares the modified matrix with the original distance matrix, selecting the smaller matrix elements for composition of the shortest path weight matrix. Through finite iteration, the shortest path is obtained between each vertex. By the MATLAB simulation, the algorithm is extended to stochastic large-scale complex networks. The running time line chart shows that this algorithm runs faster than traditional algorithm after a certain number of nodes, and in sparse network, the efficiency is particularly high, which shows its effectiveness. Finally, by using the specific application, the feasibility of the algorithm is verified.

Key words: shortest path problem; Floyd algorithm; matrix custom operations; MATLAB; sparse network

0 引言

随着社会的发展, 最短路问题在日常生活中的应用越来越广泛, 小到上班上学走哪条路最近, 大到网络路由以及基站的选址, 还有交通旅行、城市规划、电网架设, 最短路问题出现在生活的方方面面。如果掌握了最短路算法, 那么可以给生活带来许多便利。为了解决简单的最短路问题, 提出了许多简便的算法, 如 Dijkstra 算法、Floyd 算法、Kruskal 算法、拓扑排序法、

启发式搜索算法、A* 算法、动态规划算法以及神经网络遗传算法等等^[1-6]。然而 Dijkstra 算法和 Floyd 算法无法解决任意顶点间最短路长的问题, 而且 Floyd 算法十分繁琐。

针对上述问题, 文中提出了一种基于矩阵自定义运算的 Floyd 改进算法。该算法在计算权矩阵时直接在权值旁对路径进行标注, 省去了路径矩阵的求解。同时, 在运算过程中对矩阵元素出现 ∞ 时不进行运

收稿日期: 2015-11-23

修回日期: 2016-03-03

网络出版时间: 2016-08-01

基金项目: 国家自然科学基金资助项目(61304169)

作者简介: 赵礼峰(1959-), 男, 教授, 硕士研究生导师, 研究方向为图论及其在通信中的应用; 黄奕雯(1991-), 女, 硕士研究生, 研究方向为图论及其在通信中的应用。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20160801.0904.024.html>

算,大大简化了运算量。特别是在稀疏矩阵中,由于稀疏矩阵中有大量的 ∞ 元素,那么只需计算其中几个非 ∞ 元素,算法优势显而易见。但是当阶数 n 非常大时,计算依然十分复杂,所以需要借助计算机用计算机语言来实现大型网络的计算。文中借助 MATLAB 实现该算法。

1 相关知识

定义1^[7]赋权图:设 $G=(V,E)$ 为一幅图,给 G 的每一条边 e 赋予一个权值 $w(e)$, $w(e)$ 可以指网络流量、运输费用、物理距离、消耗时间等。若图 G 的所有边都赋予权值,称 G 为赋权图或网络。

定义2^[7]最短路径:在带权图 $G=(V,E)$ 中,若顶点 v_i, v_j 是图 G 的两个顶点,从顶点 v_i 到 v_j 的路径长度定义为路径上各条边的权值之和。从顶点 v_i 到 v_j 可能有多条路径,其中路径长度最小的一条称为顶点 v_i 到 v_j 的最短路径。

定义3^[8]加权有向图的带权邻接矩阵:设 $G=(V,E)$ 是一幅有向图,如果 D 中的每条弧都被赋予一个权值,那么称 D 为加权有向图;设 $G=(V,E)$ 是一幅简单的加权有向图, $v=\{v_1, v_2, \dots, v_n\}$,则 D 的邻接矩阵 $A=(a_{ij})_{n \times n}$ 。其中, $a_{ij}=\begin{cases} w_{ij}, (v_i, v_j) \in E \\ 0, i=j \\ \infty, (v_i, v_j) \notin E \end{cases}$ 。简单有向图是指不含有环也不含有重弧的有向图。

定义4^[3]稀疏网络:用稀疏矩阵存储的网络。

定义5、定义6是文中给出的:

定义5 稀疏矩阵:包含大量0元素的矩阵,在最短路径问题中,可以认为含有大量 ∞ 元素的矩阵。

定义6 矩阵自定义运算 \oplus :现定义一种运算 \oplus ,使得:

$$A = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}, b = (b_1, b_2, \dots, b_n)$$

那么有:

$$A \oplus B = \begin{pmatrix} a_1 + b_1 & a_1 + b_2 & \cdots & a_1 + b_n \\ a_2 + b_1 & a_2 + b_2 & \cdots & a_2 + b_n \\ \vdots & \vdots & \vdots & \vdots \\ a_n + b_1 & a_n + b_2 & \cdots & a_n + b_n \end{pmatrix}$$

这种新定义的运算相当于连接两条经过同一定点的路径。每做一次 \oplus 运算相当于原路径经过一次中间节点。

定理1^[7]:起点 u 到终点 v 的最短路径总是先从起点 u 沿着路径数据达 v_i ,再沿着邻接边 $v_i v$ 到达 v 的,

那么路径 L 必是起点 u 到 v_i 的最短路,且 $d_{uv} = \min\{d_{ui} + b_{vi}\}$ 。其中, d_{uv} 表示起点 u 到终点 v 的最短路径长, b_{vi} 表示 v_i 与 v 的邻接边权值。

定理2^[8]:设网络 $D=(V,A,W)$ 不含负回路, $v=\{v_1, v_2, \dots, v_n\}$,且 D 中存在 (v_1, v_i) 路($i=1, 2, \dots, n$),若 $\overline{u_j}$ 是 D 中最短 (v_1, v_j) 路的权($j=1, 2, \dots, n$),则 $(\overline{u_1}, \overline{u_2}, \dots, \overline{u_n})$ 满足方程

$$\begin{cases} u_1 = 0 \\ u_j = \min_{(v_1, v_j) \in A} \{u_i + \omega_{ij}\}, j=2, 3, \dots, n \end{cases} \quad (1)$$

如果网络 D 中不含非正回路,则方程(1)的解中 u_j 是 D 中最短 (v_1, v_j) 路的权。

2 基于矩阵自定义运算的 Floyd 改进算法

2.1 算法思想

该改进算法旨在对 Floyd 算法进行改进。在不含负回路的网络中,分析传统 Floyd 算法,从每一次对权矩阵的修改来看,首先由第一列每个元素 $d_{i1}(i=1, 2, \dots, n)$ 与第一行每个元素 $d_{1j}(j=1, 2, \dots, n)$ 的和 $d_{i1} + d_{1j}$ 的值组成的 n 阶方阵与 U_0 中元素 d_{ij} 依次比较取较小值;其次由第二列每个元素 $d_{i2}(i=1, 2, \dots, n)$ 与第二行每个元素 $d_{2j}(j=1, 2, \dots, n)$ 的和 $d_{i2} + d_{2j}$ 的值组成的 n 阶方阵与 U_1 中元素 d_{ij} 依次比较取较小值;以此类推,直到计算到第 n 列与第 n 行元素为止,得到最后任意两点间距离^[5,9]。

与此同时,以标注法直接在代表距离的权值旁的括号中标注路径。当插入某个节点 v_k 后,若计算出的最短路径长度小于原来不经过 v_k 时的长度,那么就将该节点的下标 k 直接标注在权矩阵中对应的元素的右边括号中表明 v_k 为最短路中路过节点,若路过节点不止一个则依次标明。

最后,当加法运算中一个加法项 d_{ij} 出现 ∞ 时,不用计算,直接得 ∞ ;当加法项 d_{ij} 出现0时,同样无需计算,直接写上另一个加法项,从而简化计算。

2.2 算法步骤

同 Floyd 算法一样,规定 $\forall (v_i, v_j) \notin A$,令 $\omega_{ij} = \begin{cases} 0, i=j \\ \infty, i \neq j \end{cases}$ 。

Step0:令 $U_0 = (\omega_{ij})_{n \times n} = (u_{ij}^{(0)})_{n \times n}, k=1$;

Step1:取 $V_k = u_{ik}^{(k-1)} \oplus u_{kj}^{(k-1)} (i=1, 2, \dots, n; j=1, 2, \dots, n), U^{(k)} = \min\{u_{ij}^{(k-1)}, v_{ij}^{(k)}\}, U_k = (u_{ij}^{(k)})_{n \times n}$ 。当 $u_{ij}^{(k)} = v_{ij}^{(k)}$ 时,在 $u_{ij}^{(k)}$ 右边的括号中标上 k (由 k 可得出具体路径);

Step2:如果 $k=n$,结束;否则,令 $k:=k+1$,转

Step1。

2.3 算法复杂度分析

对一个含 n 个节点的网络 G , 求 $d_{ik} \oplus d_{kj}$ 每一项元素需作加法次数 $n \times n$ 次, 由于 $d_{ik} \oplus d_{kj}$ 矩阵有 $n \times n$ 个元素, 每一个元素都需要与 U_{k-1} 中元素进行比较, 因此求一个 U_k 矩阵需比较 $n \times n$ 次。因此, 该算法的时间复杂度约为 $O(n^4)$ 。但实际运算中, 当出现 0 或者 ∞ 时就不用进行加法运算也不用进行比较, 故一般不会达到该算法复杂度, 它的优越性在稀疏网络中尤为明显。

3 算 例

以图 1 为例, 得出各个顶点间的最短路。

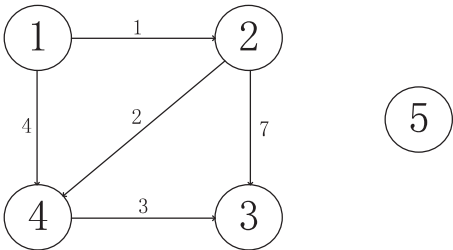


图 1 稀疏网络

解: 由图 1 知, 初始矩阵为 $U_0 = \begin{pmatrix} 0 & 1 & \infty & 4 & \infty \\ \infty & 0 & 7 & 2 & \infty \\ \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & 3 & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$, 由于第一列、最后一列、第三行和最后一行元素皆为 0 或 ∞ , 所以 U_1 、 U_3 和 U_5 不用进行计算, 那么运算结果如下:

$$V_2 = \begin{pmatrix} 1 \\ 0 \\ \infty \\ \infty \\ \infty \end{pmatrix} \oplus (\infty \ 0 \ 7 \ 2 \ \infty) = \begin{pmatrix} \infty & 1 & 8 & 3 & \infty \\ \infty & 0 & 7 & 2 & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

所以, 经过 V_2 和 U_0 的比较, 有

$$U_2 = \begin{pmatrix} 0 & 1 & 8(2) & 3(2) & \infty \\ \infty & 0 & 7 & 2 & \infty \\ \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & 3 & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

万方数据

$$V_4 = \begin{pmatrix} 3(2) \\ 2 \\ \infty \\ 0 \\ \infty \end{pmatrix} \oplus (\infty \ \infty \ 3 \ 0 \ \infty) = \begin{pmatrix} \infty & \infty & 6 & 3 & \infty \\ \infty & \infty & 5 & 2 & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 3 & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

所以, 经过 V_4 和 U_2 的比较, 有

$$U_4 = \begin{pmatrix} 0 & 1 & 6(24) & 3(2) & \infty \\ \infty & 0 & 5(4) & 2 & \infty \\ \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & 3 & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

由于最后一行不用计算, 所以 $U_5 = U_4$ 即为最终得出的结果。

4 算法的仿真与可行性分析

利用 MATLAB^[10] 对文中提出的算法进行仿真。首先运行普通网络, 接着运行稀疏网络, 同时与传统算法的运行作对比, 通过运行时间的长短说明该算法的优越性。由仿真结果可知, 该算法可以推广到大规模随机复杂网络中, 进一步说明了该算法的实用性。

由于是随机生成的网络, 每次的运行结果有所差别, 所以对它的运行时间取一个平均值。表 1 为对 10 阶, 20 阶, 直到 100 阶矩阵运行 20 次取得的平均结果, 将它的运行时间与传统 Floyd 算法进行比较。图 2 为该改进算法与传统算法运行时间相比的折线图。

表 1 算法运行时间

网络规模 (节点数)	传统 Floyd 算法 运行时间/s	改进 Floyd 算法 运行时间/s	稀疏矩阵中 改进 Floyd 算法 运行时间/s
10	0.017 4	0.007 8	0.006 1
20	0.028 4	0.016 5	0.006 5
30	0.045 1	0.033 5	0.008 5
40	0.085 6	0.082 2	0.013 1
50	0.128 9	0.138 0	0.020 7
60	0.447 6	0.235 0	0.037 1
70	0.871 4	0.302 1	0.040 3
80	1.300 1	0.473 5	0.062 7
90	1.701 5	0.635 5	0.071 4
100	2.322 2	0.898 2	0.102 9

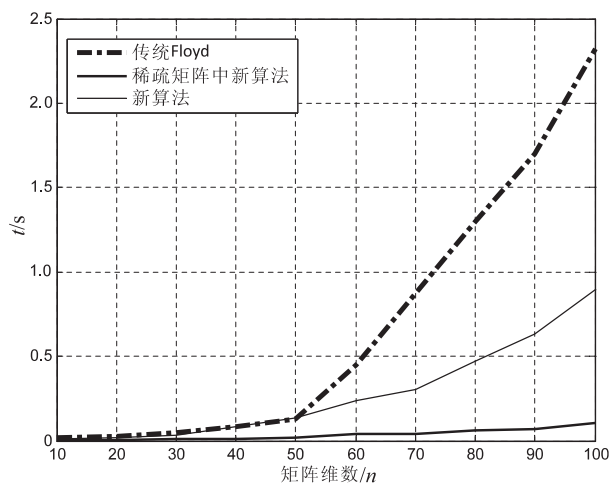


图 2 算法运行时间对比折线图

5 实际应用举例

最短路问题的应用非常广泛,举一个实际生活中存在的旅游班车选乘问题^[11-14]的例子来说明最短路问题在生活中的应用。

例:南京旅游局开通了一条旅行专线,途中经过 5 个景点(奥体中心(A)、夫子庙(B)、中华门(C)、玄武湖(D)、中山陵(E)),每班车的发车时间固定(见表 2)。如果在某一时刻出发从一个景点到另一个景点,讨论不同时刻出发去某一景点的最短时间。(将途中经过的时间看作是路的权值,那么,求最短时间的问题就可以看作是求最短路问题,即可以用文中求最短路的方法来解决。)

表 2 班车时刻表

起始站	到达站	发班时间	行驶时长/min
奥体中心	中山陵	6:50、8:30、10:30、12:30、13:30	100
奥体中心	夫子庙	6:30-18:30 每 30 min 一班	45
奥体中心	中华门	6:10-17:50 每 30 min 一班	70
奥体中心	玄武湖	6:00、13:00	80
夫子庙	中山陵	7:00-16:30 每 30 min 一班	40
夫子庙	中华门	6:00-18:00 每 10 min 一班	15
夫子庙	玄武湖	7:00-18:00 每 30 min 一班	20
中华门	玄武湖	7:00-18:00 每 30 min 一班	15
中华门	中山陵	8:00、11:00、14:00、17:00	30
玄武湖	中山陵	7:00、9:00、11:30、13:00、15:00、17:00	15

问:某人在 11:30 要从奥体中心出发去中山陵游玩,怎样乘车才能最快到达?

解:把换乘旅行班车问题运用到图论中转化为最短路问题。首先,令奥体中心为 A 点(在本例中作为发点),中山陵作为 E 点(在本例中作为收点),做出旅行班车的路线图,因为旅行班车停靠点时间固定,所以

不同的出发时间,会有不同的时间图,即图中边上的权值不同。图 3 为 11:30 时出发的路线图,为赋权无回路有向图。

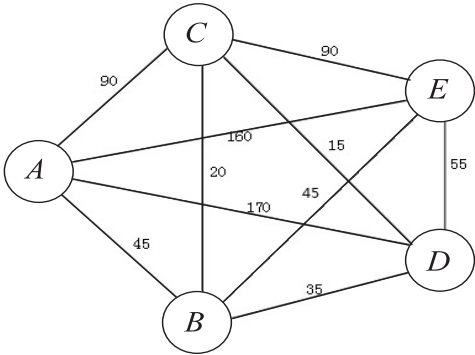


图 3 景点分布图

初始矩阵为:

$$U_0 = \begin{pmatrix} 0 & 45 & 90 & 170 & 160 \\ \infty & 0 & 20 & 35 & 45 \\ \infty & \infty & 0 & 15 & 90 \\ \infty & \infty & \infty & 0 & 55 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

用以上算法求解最终得到:

$$U_4 = \begin{pmatrix} 0 & 45 & 65(B) & 80(B) & 90(B) \\ \infty & 0 & 20 & 35 & 45 \\ \infty & \infty & 0 & 15 & 70(D) \\ \infty & \infty & \infty & 0 & 55 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

由于最后一行不用计算,所以 $U_5 = U_4$ 即为最终得出的结果。

即,从奥体中心(A)到中山陵(E)最快需要 90 min,行驶路线为:奥体中心(A)→夫子庙(B)→中山陵(E)。

6 结束语

通过应用实例可以看出该算法在实际运用中的作用;经过程序测试,算法能够得到任意节点间最短路。由仿真结果可以看出,当网络节点较小时,该算法在一般网络中与传统算法相比,运行时间并没有得到明显提高;但是,当节点增多到 50 个以上时,该算法明显快于传统算法。同时,在稀疏网络中,无论从时间复杂度,还是空间复杂度来说,文中算法优越性明显。

参考文献:

[1] 张德全,吴果林,刘登峰. 最短路问题的 Floyd 加速算法与优化[J]. 计算机工程与应用,2009,45(17):41-43.
[2] 张德全,吴果林. 最短路问题的 Floyd 算法优化[J]. 许昌学院学报,2009,28(2):10-13.

方案达到了云主机资源监测和云硬盘动态分配的目的,具有可行性和有效性。

参考文献:

[1] Sahasrabudhe S S, Sonawani S S. Comparing OpenStack and VMware[C]//Proceedings of international conference on advances in electronics, computers and communications. [s. l.]:[s. n.],2014:1-4.

[2] Wang Endong, Wu Nan, Li Xu. QoS-oriented monitoring model of cloud computing resources availability[C]//Proceedings of fifth international conference on computational and information sciences. [s. l.]:[s. n.],2013:1537-1540.

[3] Zhang Zehua, Zhang Xuejie. Realization of open cloud computing federation based on mobile agent [C]//Proceedings of IEEE international conference on intelligent computing and intelligent systems. [s. l.]:IEEE,2009:642-646.

[4] Bist M, Wariya M, Agarwal A. Comparing delta, open stack and Xen cloud platforms;a survey on open source IaaS[C]//Proceedings of IEEE 3rd international conference on advance computing conference. [s. l.]:IEEE,2013:96-100.

[5] Yang Chao-Tung, Liu Yu-Tso, Liu Jung-Chun, et al. Implementation of a cloud IaaS with dynamic resource allocation method using OpenStack [C]//Proceedings of international conference on parallel and distributed computing, applications and technologies. [s. l.]:[s. n.],2013:71-78.

[6] Bermudez I, Traverso S, Munafo M, et al. A distributed architecture for the monitoring of clouds and CDNs; applications to Amazon AWS[J]. IEEE Transactions on Network and Service Management,2014,11(4):516-529.

[7] Kuo Yen-Hung, Jeng Yu-Lin, Chen Juei-Nan. A hybrid cloud storage architecture for service operational high availability[C]//Proceedings of IEEE 37th annual computer software and applications conference workshops. [s. l.]:IEEE,2013:487-492.

(上接第 44 页)

[3] 吴果林,金 珍,邓小方. 稀疏网络的 Floyd 动态优化算法 [J]. 江西师范大学学报:自然科学版,2013,37(1):28-32.

[4] 邓方安,雍龙泉,周 涛,等. 基于“矩阵乘法”的网络最短路径算法[J]. 电子学报,2009,37(7):1594-1598.

[5] 赵礼峰,梁 娟. 最短路问题的 Floyd 改进算法[J]. 计算机技术与发展,2014,24(8):31-34.

[6] 邹桂芳,张培爱. 网络优化中最短路问题的改进 Floyd 算法[J]. 科学技术与工程,2011,11(28):6875-6878.

[7] 谢 政. 网络算法与复杂性理论[M]. 长沙:国防科技大学出版社,2003.

[8] 刘焕淋,陈 勇. 通信网图论及应用[M]. 北京:人民邮电出版社,2010.

[9] Houghall S. The Floyd-warshall algorithm on graphs with

[8] Lu X, Zhou W, Song J. Key issues of future network management [C]//Proceedings of IEEE international conference on computer application and system modeling. [s. l.]:IEEE,2010:649-653.

[9] 戢 友. OpenStack 开源云-王者归来[M]. 北京:清华大学出版社,2014:145-150.

[10] Zhu Zhenyu, Chen Hui, Wu Lianguo. Cloud computing model based on multiservice access dynamic detection [C]//Proceedings of IEEE 3rd international conference on cloud computing and intelligence systems. [s. l.]:IEEE,2014:461-464.

[11] Bing H. Research and implementation of future network computer based on cloud computing [C]//Proceedings of 2010 third international symposium on knowledge acquisition and modeling. [s. l.]:[s. n.],2010:406-408.

[12] Alabbadi M M. Cloud computing for education and learning: education and learning as a service (ELaaS) [C]//Proceedings of IEEE international conference on interactive collaborative learning. [s. l.]:IEEE,2011:589-594.

[13] Saibharath S, Geethakumari G. Design and implementation of a forensic framework for cloud in OpenStack cloud platform [C]//Proceedings of international conference on advances in computing, communications and informatics. [s. l.]:[s. n.],2014:645-650.

[14] Rossigneux F, Lefevre L, Gelas J P, et al. A generic and extensible framework for monitoring energy consumption of OpenStack clouds [C]//Proceedings of IEEE fourth international conference on big data and cloud computing. [s. l.]:IEEE,2014:696-702.

[15] Ristov S, Gusev M, Donevski A. Security vulnerability assessment of OpenStack cloud [C]//Proceedings of IEEE international conference on computational intelligence, communication systems and networks. [s. l.]:IEEE,2014:95-100.

negative cycles[J]. Information Processing Letters,2010,110:279-281.

[10] 刘卫国. MATLAB 程序设计与应用[M]. 北京:高等教育出版社,2006.

[11] 李 科,袁 明. 小件快运中的最短运输时间问题[J]. 山东交通科技,2011(5):23-25.

[12] Feillet D, Dejax P, Gendreau M. Traveling salesman problems with profits[J]. Transportation Science,2005,39(2):188-205.

[13] Braess D, Nagurney A, Wakolbinger T. On a paradox of traffic planning[J]. Transportation Science,2005,39(4):446-450.

[14] Zhan F B. Three fastest shortest path algorithms on real road networks[J]. Journal of Geographic Information and Decision Analysis,1997,1(1):69-82.