最短路问题的 Floyd 改进算法

赵礼峰,梁 娟
(南京邮电大学 理学院,江苏 南京 210023)

摘 要:目前在不含负回路的网络中,对于求解任意两节点之间最短路问题的方法有很多,Floyd 算法是最经典的算法之一,但随着节点数量的增加,重复的计算量也随之增大,从而降低了计算效率。为此,文中通过迭代矩阵和下标标注法对 Floyd 算法进行了改进,改进后的算法既能快速地计算出网络中任意两节点之间的最短路长值,又能更直观地找出最短路径。通过具体实例分析表明,Floyd 改进算法减少了重复计算,简化了路径标注方法,提高了计算效率。

关键词:最短路;不含负回路网络;Floyd 改进算法;迭代矩阵
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2014)08-0031-04
doi:10.3969/j.issn.1673-629X.2014.08.008

Improved Floyd Algorithm for Shortest Paths Problem

ZHAO Li-feng, LIANG Juan
(College of Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)

Abstract: At present, there are many algorithms for solving the shortest path between any two points in the network without negative loop. Floyd algorithm is one of the most classical algorithms. But when the number of nodes increases, the amount of repeated calculation also increases which reduces the efficiency of the algorithm. Floyd algorithm is improved in this paper by using iterative matrix and subscript tagging method. The improved algorithm not only can calculate the shortest path weights more quickly but also find shortest paths more directly. The specific instance analysis indicates that the improved algorithm reduces the repeated calculation and simplifies the path tagging method, which lead to the improvement of the computational efficiency.

Key words: the shortest path; network without negative loop; improved Floyd algorithm; iterative matrix

0 引 言

随着科技的迅猛发展,人们做事越来越追求效率,最短路问题也逐渐成为计算机、运筹学、地理信息科学等领域的研究热点之一。由于其在生产实践中的广泛应用,因此对最短路问题的算法以及提高算法的效率的研究也就具有重大的现实意义,国内外诸多专家均对此问题进行了深入的研究[1-9],Floyd 算法[10]就是其中最为常用的一种经典算法。此算法是通过权矩阵和后点标号矩阵来实现的,但随着节点的增加,需要计算的矩阵增多,重复的计算也随之增多,从而大大降低了计算效率。优化矩阵算法[11]虽然对传统的 Floyd 算法进行了优化,但算法只对某些特殊的网络(例如,某个节点无出弧或入弧的网络)较适合,对于一般的网络(每个节点均有出弧和入弧的网络),计算效率仍然不高。Floyd 算法[10]和 Floyd 优化算法[12],对任意两节点之间最短路长的计算,有些是不必要的。例如,当待插入的中间节点不与源点或汇点直接相连的情况。另外,这两种算法都是借用正向追踪法,通过标号矩阵来寻找最短路径的,这样做比较繁琐,不够直观。因此为了改善这些不足之处,文中提出了 Floyd 改进算法。

1 相关知识

定义 1 赋权图:给定有向图 $D=(V,A,W)$,如果 D 中任意一条边 (v_i,v_j) 上均有一个权 w_{ij},则称图 D 是一个赋权图。

定义 2 回路:设 $D=(V,A,W)$ 是一个有向图,D

收稿日期:2013-10-16 **修回日期**:2014-01-18 **网络出版时间**:2014-05-21
基金项目:国家自然科学基金资助项目(61070234,61071167)
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;梁 娟(1988-),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.20140525.1242.003.html

中含有一条起点和终点相同的路,则称为图 D 的一条回路。

定义 3 负回路:设 C 是 D 的一个回路,如果 $w(C)=\sum_{a\in A(C)} w(a)<0$,则称 C 为 D 的负回路[13]。

定义 4 最短路:设 $D=(V,A,W)$ 是一个有向赋权网络,v_i 和 v_j 是 D 中的两个节点,D 的所有 (v_i,v_j) 路中权最小的路称为最短 (v_i,v_j) 路。

Floyd 算法的基本定理:设网络 D 不含负回路,网络 $D(i,j,k)$ 中最短 (v_i,v_j) 路记为 $P_{ij}^{(k)}(i,j=1,2,\cdots,n;k=0,1,\cdots,n)$,则 $u_{ij}^{(k)}=w(P_{ij}^{(k)})(i,j=1,2,\cdots,n;k=0,1,\cdots,n)$ 当且仅当 $u_{ij}^{(k)}(i,j=1,2,\cdots,n;k=0,1,\cdots,n)$ 满足下面的方程

$$\begin{cases}u_{ij}^{(0)}=w_{ij}\\ u_{ij}^{(k)}=\min\{u_{ij}^{(k-1)},u_{il}^{(k-1)}+u_{lj}^{(k-1)}\},i,j,l=1,2,\cdots,n\end{cases}$$

2 Floyd 改进算法

2.1 算法思想

文中的算法思想旨在通过对 Floyd 算法进行改进,从而更加简便有效地计算不含负回路网络中任意两节点之间的最短路。初始权矩阵 $\boldsymbol{D}^{(0)}$ 可由网络图直接得到,由 $\boldsymbol{D}^{(0)}$ 计算 $\boldsymbol{D}^{(1)}$ 时,$\boldsymbol{D}^{(1)}$ 中的元素是经过多次经转后得到的最短路长,即经过该次迭代计算出的最短路径可以经过多个中间节点。在此基础上计算 $\boldsymbol{D}^{(2)},\boldsymbol{D}^{(3)},\cdots,\boldsymbol{D}^{(k)}$,当算出第 $k+1$ 个矩阵时,如果 $\boldsymbol{D}^{(k+1)}=\boldsymbol{D}^{(k)}$,那么 $\boldsymbol{D}^{(k)}$ 中的元素就是对应的节点对之间的最短路长值。

在计算权矩阵时,有时插入的中间节点与源点、汇点均不直接相连,因此计算出来的最短路长值不会改变,故这些点可以不参与计算。另外,在计算最短路长前,先将待插入节点进行路长比较,若 $d_{il}^{(k)}\geqslant d_{ij}^{(k)}$ 或 $d_{lj}^{(k)}\geqslant d_{ij}^{(k)}$,则表明插入该节点后,不会使原来的路长变短,无需再计算 $d_{ij}^{(k+1)}$(因为 $d_{ij}^{(k+1)}=d_{ij}^{(k)}$),接着对下一个节点进行搜索。文中采用下标直接标注的方法来标注最短路径,当插入某个节点 v_l 后,如果计算出来的最短路长值不会比原来的短,那么就将该节点的下标 l 直接标注在权矩阵中对应的元素的右下角,表明最短路径经过该节点。

2.2 算法步骤

在不含负回路的网络中,有节点 $v_1,v_2,\cdots,v_n$,用 w_{ij} 表示节点 v_i 和节点 v_j 之间的连线长,用 d_{ij} 表示从节点 v_i 出发到节点 v_j 时的路长[14]。

改进算法步骤如下:

Step1:作初始权矩阵 $\boldsymbol{D}^{(0)}=(d_{ij}^{(0)})_{n\times n}(i,j=1,2,\cdots,n)$,$k=0$。

其中,$d_{ij}^{(0)}=\begin{cases}0 & i=j\text{ 时}\\ \infty & i,j\text{ 不相连或无路时}\\ w_{ij} & i,j\text{ 相连时}\end{cases}$

Step2:在 $\boldsymbol{D}^{(k)}$ 中,若 $i=j$,则 $d_{ij}^{(k+1)}=0$;否则转 Step3。

Step3:若 $d_{il}^{(k)}=\infty$ 或 $d_{lj}^{(k)}=\infty(l=1,2,\cdots,n)$,且 $d_{ij}^{(k)}\neq\infty$,则 $d_{ij}^{(k+1)}=d_{ij}^{(k)}$,否则转 Step4。

Step4:若 $d_{il}^{(k)}\geqslant d_{ij}^{(k)}$ 或 $d_{lj}^{(k)}\geqslant d_{ij}^{(k)}(l=1,2,\cdots,n$ 且 $l\neq i,j)$,则 $d_{ij}^{(k+1)}=d_{ij}^{(k)}$,否则转 Step5。

Step5:若 $i<j$,则

$$d_{ij}^{(k+1)}=\min\{d_{ij}^{(k)},\min_{l<i}\{d_{il}^{(k+1)}+d_{lj}^{(k+1)}\},\min_{i<l<j}\{d_{il}^{(k+1)}+d_{lj}^{(k)}\},\min_{j<l}\{d_{il}^{(k)}+d_{lj}^{(k)}\}\};$$

否则

$$d_{ij}^{(k+1)}=\min\{d_{ij}^{(k)},\min_{l<j}\{d_{il}^{(k+1)}+d_{lj}^{(k+1)}\},\min_{j<l<i}\{d_{il}^{(k)}+d_{lj}^{(k+1)}\},\min_{i<l}\{d_{il}^{(k)}+d_{lj}^{(k)}\}\}$$

若 $d_{ij}^{(k+1)}=d_{il}^{(k)}+d_{lj}^{(k)}$,且 $d_{ij}^{(k+1)}<d_{ij}^{(k)}$,则将 l 标注在权矩阵 $\boldsymbol{D}^{(k+1)}$ 中的元素 $d_{ij}^{(k+1)}$ 的右下角,表明经过该次迭代后,节点 v_i 与节点 v_j 之间的最短路径经过节点 v_l,并且路长值为 $d_{ij}^{(k+1)}$。

Step6:若 $\boldsymbol{D}^{(k+1)}=\boldsymbol{D}^{(k)}$,结束;否则 $k=k+1$,转 Step2。

2.3 算法的复杂度分析

在计算节点 v_i 到节点 v_j 的最短路长 $d_{ij}^{(k+1)}$ 时,Floyd 算法将 $\boldsymbol{D}^{(k)}$ 中第 i 行的元素和第 j 列对应的元素相加,一共做了 n 次加法,但在 Floyd 改进算法中 $l\neq i,j$,所以,最多需要计算 $n-2$ 次加法;另外,在计算 $d_{ij}^{(k+1)}$ 之前,先令 $d_{ii}^{(k+1)}=0$,接着比较路长,即如果 $d_{il}^{(k)}\geqslant d_{ij}^{(k)}$ 或 $d_{lj}^{(k)}\geqslant d_{ij}^{(k)}$,则不用计算 $d_{ij}^{(k+1)}$,特别地,当第 i 行或第 j 列的元素都大于等于 $d_{ij}^{(k)}$ 时,此时直接得 $d_{ij}^{(k+1)}=d_{ij}^{(k)}$,加法的计算量为 0 次。因此,加法计算量的取值范围为 $0\sim n-2$ 次,是一个随机变量,记为 X。为了计算方便,假设 X 是连续随机变量[15],由 $\int_0^{n-2}\frac{x}{n-2}\mathrm{d}x=\frac{n-2}{2}$ 知 X 的数学期望为 $\frac{n-2}{2}$,即用 Floyd 改进算法计算任意两节点之间的最短路长 $d_{ij}^{(k+1)}$ 时,加法计算量为 $\frac{n-2}{2}$,又因为权矩阵中有 n^2 个元素,所以在实际应用中,Floyd 改进算法的复杂度约为 $O(\frac{1}{2}n^3)$。

2.4 算法的可行性分析

Floyd 算法的可行性和正确性是由 Floyd 算法的基本定理保证的。文中的改进算法是在 Floyd 算法的基础上,先令 $d_{ii}^{(k+1)}=0$,显然对于 Step3,插入节点 v_l 后,由于其和节点对 v_i、v_j 均不直接相连,计算得到的

路长值不变,故可以不参与计算。在不含负回路的网络中,如果 $d_{il}^{(k)} \geqslant d_{ij}^{(k)}$ 或 $d_{lj}^{(k)} \geqslant d_{ij}^{(k)}$,则表明插入节点后的路长值大于或等于原来的路长值,所以也可以不参与计算。Step5 中计算 $d_{ij}^{(k+1)}$ 可归纳为 $d_{ij}^{(k+1)} = \min\{d_{ij}^{(k)}, d_{il} + d_{lj}\}$,其中,$d_{il} + d_{lj}$ 为最新计算出的值[16]。而 Floyd 算法中 $d_{ij}^{(k+1)} = \min\{d_{ij}^{(k)}, d_{il}^{(k)} + d_{lj}^{(k)}\}$,是经过一次经转后的最短路长。改进算法计算出的 v_i 和 v_j 的最短路径可以经过多个中间节点,经过多次经转。改进算法是对 Floyd 算法的改进,就如同化学反应中的催化剂(催化剂的作用仅是提高化学反应速率),提高了计算速度和效率,与 Floyd 算法本质没有太大区别。Step6 对算法作结束判断,若 $\boldsymbol{D}^{(k+1)} = \boldsymbol{D}^{(k)}$,则结束[17],否则继续迭代,由于 $k+1 \leqslant n$,因此算法必然会在有限步后终止,不会出现死循环。所以,算法是可行的,也是正确的。

3　应用实例

在图 1 的网络图中,求任意两节点之间的最短路。

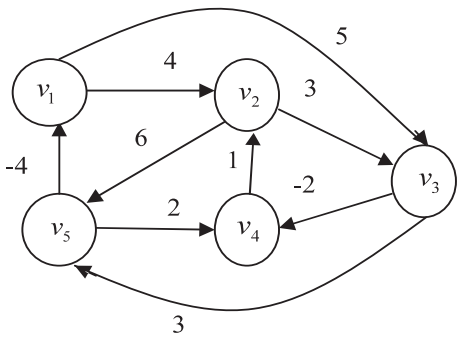

图 1　有向网络图

容易看出,该有向网络图中不含负回路,记初始矩阵 $\boldsymbol{D}^{(0)} = (d_{ij}^{(0)})_{5\times5}$,则:

$$\boldsymbol{D}^{(0)} = \begin{bmatrix} 0 & 4 & 5 & \infty & \infty \\ \infty & 0 & 3 & \infty & 6 \\ \infty & \infty & 0 & -2 & 3 \\ \infty & 1 & \infty & 0 & \infty \\ -4 & \infty & \infty & 2 & 0 \end{bmatrix}$$

首先根据 $\boldsymbol{D}^{(0)}$ 计算 $\boldsymbol{D}^{(1)}$,由 Step2 得 $d_{ii}^{(1)} = 0(i = 1, 2, \cdots, 5)$,由 Step3 知,在 $\boldsymbol{D}^{(0)}$ 中,元素 $d_{ij}^{(0)}$ 所在的行或列中有元素 ∞,则 $d_{ij}^{(1)} = d_{ij}^{(0)}$,即 $d_{12}^{(1)} = 4, d_{13}^{(1)} = 5, d_{23}^{(1)} = 3, d_{25}^{(1)} = 6, d_{34}^{(1)} = -2, d_{35}^{(1)} = 3, d_{42}^{(1)} = 1, d_{51}^{(1)} = -4, d_{54}^{(1)} = 2$。

此时 $\boldsymbol{D}^{(0)}$ 中的其余值为 ∞。因此,由 Step5 计算:

$d_{14}^{(1)} = \min\{d_{14}^{(0)}, d_{12}^{(1)} + d_{24}^{(0)}, d_{13}^{(1)} + d_{34}^{(0)}, d_{15}^{(0)} + d_{54}^{(0)}\} = \min\{\infty, 4+\infty, 5+(-2), \infty+2\} = d_{13}^{(1)} + d_{34}^{(0)} = 3$, $l = 3$

$d_{21}^{(1)} = \min\{d_{21}^{(0)}, d_{23}^{(0)} + d_{31}^{(0)}, d_{24}^{(0)} + d_{41}^{(0)}, d_{25}^{(0)} + d_{51}^{(0)}\} = \min\{\infty, 3+\infty, \infty+\infty, 6+(-4)\} = d_{25}^{(0)} + d_{51}^{(0)} = 2, l = 5$

同理可计算出

$d_{15}^{(1)} = 8, l = 3; d_{24}^{(1)} = 1, l = 3; d_{31}^{(1)} = -1, l = 5; d_{32}^{(1)} = -1, l = 4; d_{41}^{(1)} = 3, l = 2; d_{43}^{(1)} = 4, l = 2; d_{45}^{(1)} = 7, l = 2, 3; d_{52}^{(1)} = 0, l = 1; d_{53}^{(1)} = 1, l = 1$

经计算得出 $\boldsymbol{D}^{(1)}$ 如下:

$$\boldsymbol{D}^{(1)} = \begin{bmatrix} 0 & 4 & 5 & 3_3 & 8_3 \\ 2_5 & 0 & 3 & 1_3 & 6 \\ -1_5 & -1_4 & 0 & -2 & 3 \\ 3_2 & 1 & 4_2 & 0 & 7_{23} \\ -4 & 0_1 & 1_1 & 2 & 0 \end{bmatrix}$$

经比较不难发现 $\boldsymbol{D}^{(1)} \neq \boldsymbol{D}^{(0)}$,因此需要计算 $\boldsymbol{D}^{(2)}$。由 Step2 得 $d_{ii}^{(2)} = 0(i = 1, 2, \cdots, 5)$,因为 $\boldsymbol{D}^{(1)}$ 中的元素没有 ∞,所以跳过 Step3。由 Step4 知,如果元素 $d_{ij}^{(1)}$ 是所在的行或列中元素的最小值,则 $d_{ij}^{(2)} = d_{ij}^{(1)}$。即 $d_{14}^{(2)} = 3, d_{24}^{(2)} = 1, d_{32}^{(2)} = -1, d_{34}^{(2)} = -2, d_{35}^{(2)} = 3, d_{42}^{(2)} = 1, d_{51}^{(2)} = -4, d_{53}^{(2)} = 1$。

对于 $d_{31}^{(2)}$:由于 $\boldsymbol{D}^{(1)}$ 中的第三行的元素中值比 $d_{31}^{(1)}$ 小的元素($d_{33}^{(1)}$ 除外)只有 $d_{34}^{(1)}$,而第一列的元素中值比 $d_{31}^{(1)}$ 小的元素($d_{11}^{(1)}$ 除外)只有 $d_{51}^{(1)}$,因此有

$$d_{31}^{(2)} = \min\{d_{31}^{(1)}, d_{34}^{(1)} + d_{41}^{(1)}, d_{35}^{(1)} + d_{51}^{(1)}\} = \min\{-1, -2+3, 3+(-4)\} = d_{31}^{(1)} = -1$$

同理可计算出 $\boldsymbol{D}^{(2)}$ 中的其他元素。

经计算得出 $\boldsymbol{D}^{(2)}$ 如下:

$$\boldsymbol{D}^{(2)} = \begin{bmatrix} 0 & 4 & 5 & 3_3 & 8_3 \\ 2_5 & 0 & 3 & 1_3 & 6 \\ -1_5 & -1_4 & 0 & -2 & 3 \\ 3_{25} & 1 & 4_2 & 0 & 7_{23} \\ -4 & 0_1 & 1_1 & -1_{13} & 0 \end{bmatrix}$$

经比较不难发现 $\boldsymbol{D}^{(2)} \neq \boldsymbol{D}^{(1)}$,接着根据 $\boldsymbol{D}^{(2)}$ 计算 $\boldsymbol{D}^{(3)}$,经计算可得 $\boldsymbol{D}^{(3)} = \boldsymbol{D}^{(2)}$,算法结束。根据 $\boldsymbol{D}^{(2)}$ 直接得到任意两节点 v_i 和 v_j 之间的最短路长值为 $d_{ij}^{(2)}$,元素 $d_{ij}^{(2)}$ 的下标为节点 v_i 到节点 v_j 的最短路径经过的节点,见表 1。

4　改进算法与 Floyd 算法、优化矩阵算法的比较

对于文献[11]中的例题,用文献[11]中的优化矩阵算法计算需要 75 次加法,若用文中的改进算法,只需计算 33 次加法即可。若用文献[10]中的 Floyd 算法计算文中的应用实例,需要计算 625 次加法,用优化矩阵算法计算需要 255 次加法,而用改进算法只需计算 105 次加法。由此可见,改进算法比 Floyd 算法和优化矩阵算法的复杂度要低,其运算过程更为简捷。

表 1 所有两节点间的最短路径与最短路长

节点	最短路径	路长
1,2	①→②	4
1,3	①→③	5
1,4	①→③→④	3
1,5	①→③→⑤	8
2,1	②→⑤→①	2
2,3	②→③	3
2,4	②→③→④	1
2,5	②→⑤	6
3,1	③→⑤→①	-1
3,2	③→④→②	-1
3,4	③→④	-2
3,5	③→⑤	3
4,1	④→②→⑤→①	3
4,2	④→②	1
4,3	④→②→③	4
4,5	④→②→③→⑤	7
5,1	⑤→①	-4
5,2	⑤→①→②	0
5,3	⑤→①→③	1
5,4	⑤→①→③→④	-1

5 结束语

文中提出的 Floyd 改进算法,在计算最短路长时,由于将与源点、汇点均不直接相连的节点插入后,不会使路长变短,故不必参与计算;而对于那些待插入节点的路长不比原来路长短的节点,也无需参与计算,因此在计算权矩阵时,就不需要对每个节点都进行计算,从而减少了计算量。同时将那些使最短路长变短的节点的下标记下,并将其标在权矩阵中对应的元素的右下角,这样可以更方便地找到最短路径。因此采用 Floyd 改进算法来计算不含负回路网络中的最短路,降低了计算量,提高了计算效率,同时使得最短路径的寻找更加直观有效。

[1] Loachim I. A dual programming algorithm for the shortest path problem[J]. Networks,1998,31(2):193-204.

[2] Jin W,Chen S,Jiang H. Finding the K shortest paths in a time-schedule network with constraints on arcs[J]. Computers & Operations Research,2013,40(12):2975-2982.

[3] 赵建宏,杨建宇,雷维礼. 一种新的最短路径算法[J]. 电子科技大学学报,2005,34(6):778-781.

[4] Kuipers F,van Mieghem P,Korkmaz T,et al. An overview of constraint-based path selection algorithms for QoS routing [J]. IEEE Communications Magazine,2002,40(12):50-55.

[5] Cormen T H,Leiserson C E,Rivest R L. Introduction to algorithms[M]. 2nd ed. Cambrige,USA:MIT Press,2001.

[6] 陈益富,卢 潇,丁豪杰. 对 Dijkstra 算法的优化策略研究[J]. 计算机技术与发展,2006,16(9):73-75.

[7] Festa P,Guerriero F,Laganà D,et al. Solving the shortest path tour problem[J]. European Journal of Operational Research,2013,230(3):464-474.

[8] 朱绍伟,徐夫田,腾兆明. 一种改进蚁群算法求解最短路径的应用[J]. 计算机技术与发展,2011,21(7):202-205.

[9] 王海英,黄 强,李传涛,等. 网络算法及其 MATLAB 实现[M]. 北京:北京航空航天大学出版社,2010.

[10] 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.

[11] 林华珍,周根贵. 求解最短路问题的一种优化矩阵算法[J]. 长江大学学报自然科学版:理工(上旬),2007,4(4):14-16.

[12] 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.

[13] 孙小军,刘三阳,焦建民. 基于最小树权矩阵法的改进算法[J]. 计算机工程与设计,2005,26(12):3274-3275.

[14] 龚 劬. 图论与网络最优化算法[M]. 重庆:重庆大学出版社,2009.

[15] 茆诗松,贺思辉. 概率论与统计学[M]. 武汉:武汉大学出版社,2010.

[16] 任 刚,王 炜. 交通网络最短路权矩阵的迭代算法[J]. 交通与计算机,2005,23(5):8-12.

[17] 赵礼峰,蒋腾飞. 无回路网络最短路径的一种新算法[J]. 计算机技术与发展,2013,23(2):105-107.

参考文献:

(上接第 30 页)

engine room automation[J]. Expert System with Applications,2005,29(2):256-263.

[12] Wu M C,Lo Y F,Hsu S H. A fuzzy CBR technique for generating product ideas[J]. Expert Systems with Applications,2008,34(1):530-540.

[13] Gupta K M,Montazemi A R. Empirical evaluation of retrieval in case-based reasoning systems using modified cosine matching function[J]. IEEE Transactions on Systems,Man and Cybernetics-Part A:Systems and Humans,1997,27(5):601-612.

[14] Pal K,Campbell J A. A hybrid system for decision making about assets in English divorce case[J]. Lecture Notes in Computer Science,1995,1020:152-165.

[15] 张本生,于永利. CBR 系统案例搜索中的混合相似性度量方法[J]. 系统工程理论与实践,2002,22(3):131-136.

[16] Liao T W,Zhang Zhiming. A review of similarity measures for fuzzy systems[C]//Proceedings of the fifth IEEE international conference on fuzzy system. New Orleans,LA:IEEE,1996:930-935.

最短路问题的Floyd改进算法

作者： 赵礼峰， 梁娟， ZHAO Li-feng， LIANG Juan
作者单位： 南京邮电大学 理学院,江苏 南京,210023
刊名： 计算机技术与发展 ISTIC
英文刊名： Computer Technology and Development
年，卷(期)： 2014(8)