文章编号: 1004 - 6011(2007)02 - 0065 - 03

Dijkstra 矩阵算法

代西武
(北京建筑工程学院 基础科学部,北京 100044)

摘 要: 介绍了 Dijkstra 算法,对 Dijkstra 算法进行改进,提出了计算加权图中任意两点之间最短距离的算法——Dijkstra 矩阵算法,给出了 Dijkstra 矩阵算法在 Matlab 语言中的实现,对一个具体例子,应用 Dijkstra 矩阵算法进行了验算.

关键词: Dijkstra 算法; 最短路问题; 最短距离; 矩阵; Matlab 语言
中图分类号: O151.21　　**文献标识码**: A

Dijkstra's Matrix Algorithm

Dai Xiwu
(Dept. of Basic Sciences, BUCEA Beijing 100044)

Abstract: In this paper, the Dijkstra's algorithm is introduced. By improving Dijkstra's algorithm, the Dijkstra's matrix algorithm, which is to calculate the shortest distance between two arbitrary vertexes in a weighted graph is proposed. The MATLAB source code of Dijkstra's matrix algorithm is supplied, and an example is calculated.

Key words: dijkstra's algorithm; shortest path problem; shortest distance; matrix; matlab

在图论中,Dijkstra 算法[1]是很实用的,可以用来计算加权图里的两个指定顶点 u_0 与 v_0 之间的最短距离,实际上,Dijkstra 算法计算出了从 u_0 到其它所有顶点的最短距离. 在实际问题中,经常需要计算加权图中任意两个顶点之间的最短距离,为此,我们对 Dijkstra 算法进行了改进,提出了 Dijkstra 矩阵算法,该算法比 Dijkstra 算法更容易在计算机上实现,能够计算加权图中任意两个顶点之间的最短距离,方便实用. 进一步,我们用 Matlab 编出了 Dijkstra 矩阵算法的源程序,并对一个例子,进行了计算,得出了满意的结果.

1 Dijkstra 算法

已知图 $G(V, E)$ 及每条边的权 $w(e)$,对于任意指定的二点 u_0、$v_0 \in V(G)$,寻找路 $P(u_0, v_0)$,使得

$$w(P(u_0, v_0)) = \min_{P \in \Omega}\{w(P)\}$$

其中 Ω 是从 u_0 到 v_0 的所有路的集合,$w(P)$ 是路 P 上的各边权之和. 这样的路 $P(u_0, v_0)$ 称为从 u_0 到 v_0 的最短路,$w(P(u_0, v_0))$ 称为从 u_0 到 v_0 的最短路的长度,也可称为从 u_0 到 v_0 的最短距离. Dijkstra 算法可以计算出从 u_0 到其它所有顶点的最短距离.

下面介绍 Dijkstra 算法[1]:

(1):置 $l(u_0) = 0$;对 $v \neq u_0$,置 $l(v) = \infty$; $S_0 = \{u_0\}$; $i = 0$;

(2):对每个 $v \notin S_i$,用 $\min\{l(v), l(u_i) + w(u_iv)\}$ 代替 $l(v)$. 计算 $\min_{v \notin S_i}\{l(v)\}$,并把达到这

收稿日期: 2007 - 01 - 05
作者简介: 代西武(1963 -),男,副教授,研究生,研究方向:图论、计算机科学、数学建模.

个最小值的一个顶点记为 u_{i+1}，置 $S_{i+1} = S_i \cup \{u_{i+1}\}$.

(3)：若 $i = |V| - 1$，则停止. 若 $i < |V| - 1$，则用 $i+1$ 代替 i，并转入第二步.

说明：(1) 当算法结束时，从 u_0 到 v 的距离（最短路的长度）由标号 $l(v)$ 的终值给出. 即算法求出了 u_0 至其它所有顶点的最短路的长度. (2) Dijkstra 算法仅确定了从 u_0 到所有其它顶点的距离，而并未给出实际最短路. 对 Dijkstra 算法稍加修改，即可得到相应的最短路. (3) 对 Dijkstra 算法进行改进，可以算出加权图中任意两个顶点之间的最短距离. 我们下面就详细讨论这一问题.

2 Dijkstra 矩阵算法 I

Dijkstra 算法只是算出了加权图里的一个指定顶点到其它所有顶点的最短距离，为了计算加权图中任意两顶点之间的最短距离，我们对 Dijkstra 算法进行了修改，提出了 Dijkstra 矩阵算法 I，该算法比 Dijkstra 算法更容易在计算机上实现，能够计算加权图中任意两顶点之间的最短距离.

下面介绍 Dijkstra 矩阵算法 I 的思想. 将加权图 $G(V, E)$ 储存在矩阵 $A = (a_{ij})_{n\times n}$ 里，其中，n 为图 G 的顶点个数，

$$a_{ij} = \begin{cases} 0, & \text{当 } i = j \text{ 时} \\ w_{ij}, & \text{当 } i \neq j\text{，顶点 } v_i \text{ 与 } v_j \text{ 有连边时，} \\ \infty, & \text{当 } i \neq j\text{，顶点 } v_i \text{ 与 } v_j \text{ 无连边时} \end{cases}$$

w_{ij} 为边 v_iv_j 的权重. 显然 A 为对称矩阵. 将 Dijkstra 算法的思想应用于此矩阵的第 k 行，$k = 1, 2, \cdots, n$ 可求出顶点 v_k 到其它各顶点的最短距离，将最短距离还保存在矩阵 A 的第 k 行. 算法结束时，矩阵 A 的元素值就是任意两个顶点之间的最短距离.

注意到，在描述算法 Ⅰ（以及后面的算法 Ⅱ）时，用到了 Matlab 的语句功能，如求数组的元素个数 $length(b)$，求数组中某个元素的下标 $find(b1 == a\_id)$，等.

Dijkstra 矩阵算法 I

1. [输入加权图，保存在矩阵 $A = (a_{ij})_{n\times n}$ 里]

2. [对矩阵 A 进行操作，求任意两个顶点间的最短距离]

循环 k 以 1 为步长，从 1 到 $n-1$，执行

2.1 $b \leftarrow \{1, 2, \cdots, k-1, k+1, k+2, \cdots, n\}$; $kk \leftarrow length(b)$; $a\_id \leftarrow k$

2.2 循环 反复执行下列语句，直到 $kk = 0$

2.2.1 循环 j 以 1 为步长，从 1 到 kk，执行

$te \leftarrow a(k, a\_id) + a(a\_id, b(j))$;

若 $te < a(k, b(j))$, $a(k, b(j)) \leftarrow te$

2.2.2 $miid \leftarrow 1$

2.2.3 循环 j 以 1 为步长，从 2 到 kk，执行若

$a(k, b(j)) < a(k, b(miid))$, $miid \leftarrow j$

2.2.4 $a\_id \leftarrow b(miid)$; $b \leftarrow [b(1:miid-1), b(miid+1:kk)]$; %在数组 b[ ]中去掉一个元素 $kk \leftarrow length(b)$ %数组 b[ ]的长度减少了 1

3. [对矩阵 A 的最后一行赋值，即求 v_n 与其它顶点间的最短距离]

循环 i 以 1 为步长，从 1 到 $n-1$，执行 $a(n, i) \leftarrow a(i, n)$

4. [算法结束]

3 Dijkstra 矩阵算法 Ⅱ

Dijkstra 矩阵算法 Ⅰ只是简单地将 Dijkstra 算法的思想应用到矩阵的每一行，这样有很多的重复计算，效率不高，为了提高效率，我们提出下面的 Dijkstra 矩阵算法 Ⅱ.

Dijkstra 矩阵算法 Ⅱ

1. [输入加权图，保存在矩阵 $A = (a_{ij})_{n\times n}$ 里]

2. [对矩阵 A 进行操作，求任意两个顶点间的最短距离]

循环 k 以 1 为步长，从 1 到 $n-1$，执行

2.1 $b \leftarrow \{1, 2, \cdots, k-1, k+1, k+2, \cdots, n\}$; $kk \leftarrow length(b)$; $a\_id \leftarrow k$; $b1 \leftarrow \{k+1, k+2, \cdots, n\}$; $kk1 \leftarrow length(b1)$

2.2 循环 反复执行下列语句，直到 $kk = 0$

2.2.1 循环 j 以 1 为步长，从 1 到 $kk1$，执行

$te \leftarrow a(k, a\_id) + a(a\_id, b1(j))$;

若 $te < a(k, b1(j))$, $a(k, b1(j)) \leftarrow te$

2.2.2 $miid \leftarrow 1$

2.2.3 循环 j 以 1 为步长，从 2 到 kk，执行

若 $a(k, b(j)) < a(k, b(miid))$, $miid \leftarrow j$

2.2.4 $a\_id \leftarrow b(miid)$;

$b \leftarrow [b(1:miid-1), b(miid+1:kk)]$; %在数组 b[ ]中去掉一个元素 $kk \leftarrow length(b)$ %数组 b[ ]的长度减少了 1

若 $a\_id > k$，执行

$miid1 \leftarrow find(b1 == a\_id)$;

$b_1 \leftarrow [b1(1:miid1-1), b1(miid1+1:kk1)]$;

$kk1 \leftarrow length(b1)$

2.3　循环 j 以 1 为步长,从 $k+1$ 到 n,执行

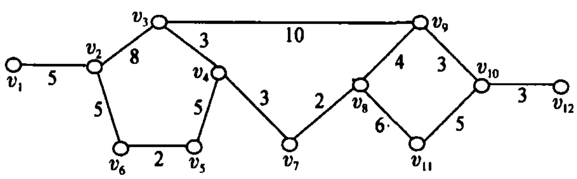
$a(j,k) \leftarrow a(k,j)$

3. [算法结束]

算法 II与算法 I比较,在 2.2.1 处循环的次数随着 k 的增加而减少,循环体的执行总次数会减少一半.

4　Dijkstra 矩阵算法 II在 Matlab 中的实现及算例

为了验证 Dijkstra 算法 II的计算效果,对下面的加权图:



我们用 Matlab 编出了 Dijkstra 矩阵算法 II的源程序如下:

```
function dij_m()
 %This is Dijikstra 's matrix algorithm
 % input the graph G to matrix a
n = 12;
a = ones(n) + inf;
for i = 1:n
    a(i,i) = 0;
end;
a(1,2) = 5;
a(2,3) = 8; a(2,6) = 5;
a(3,4) = 3; a(3,9) = 10;
a(4,5) = 5; a(4,7) = 3;
a(5,6) = 2;
a(7,8) = 2;
a(8,9) = 4; a(8,11) = 6;
a(9,10) = 3;
a(10,11) = 5; a(10,12) = 3;
for i = 2:n
    for j = 1:(i - 1)
        a(i,j) = a(j,i);
    end;
end;
 % The main program:
for k = 1:(n - 1)
    b = [1:(k - 1),(k + 1):n];
    kk = length(b);
    a_id = k;
    b1 = [(k + 1):n];
    kk1 = length(b1);
    while kk > 0
        for j = 1:kk1
            te = a(k,a_id) + a(a_id,b1(j));
            if te < a(k,b1(j))
                a(k,b1(j)) = te;
            end;
        end;
        miid = 1;
        for j = 2:kk
        if a(k,b(j)) < a(k,b(miid)) miid = j;
            end;
    end;
    a_id = b(miid);
    b = [b(1:miid - 1),b(miid + 1:kk)];
    kk = length(b);
    if a_id > k
        miid1 = find(b1 == a_id);
        b1 = [b1(1:miid1 - 1),b1(miid1 + 1:kk1)];
            kk1 = length(b1);
        end;
    end;
    for j = (k + 1):n
        a(j,k) = a(k,j);
    end;
end;
 % Output the result:
a
```

运行可得到计算结果是:

(下转第 71 页)

定理 3　假设在 E 上存在一个满足下列条件的非负 Lyapunov 函数 $V(X,t)\in C_0$:

(i) $V(0,t)\equiv 0$, $V_\delta=\inf\limits_{t>0,|X|>\delta}V(X,t)>0$,当 $\delta>0$ 时;

(ii) 对于每 $\delta>0$,存在一个有界定正函数 $f_\delta(t)$,使得

$$\frac{d^0 V}{dt}\leq -f_\delta(t)$$

在域 $\{|X|>\delta\}\times\{t>t_0\}$ 中成立;

(iii) 对于每 $\varepsilon>0$,存在一个 $\gamma>0$,使得对于 $t_0\leq s\leq t$,有

$$E\exp\left\{\gamma\int_s^t|\zeta(u,\omega)|\,du\right\}\leq\exp\{\varepsilon(t-s)\}$$

则方程(1)的解 $X\equiv 0$ 对于 $t\geq t_0$ 在(4)型小随机扰动下是稳定的.

类似定理 2 可证明定理 3.

参考文献:

[1] Malkin, I. G. Theory of stability of motion[J]. 2nd Rev. edn. "Nauka", Moscow. 1966

[2] 胡宣达. 随机微分方程稳定性理论[M]. 南京:南京大学出版社,1990

[3] 黄薇. 随机扰动下系统的稳定性问题[J]. 重庆大学学报,1995,18(4):81 - 85

[4] 席富宝. 一维扩散过程的小随机扰动[J]. 数学学报,1998,41(1):199 - 204

[5] Sun Jitao, Zhang Yinping, Wu Qidi. Less conservative conditions for asymptotic stability of impulsive control systems[J]. IEEE Trans Automatic Control, 2003, 48(5): 829 - 831

[6] Soliman A A. Stability criteria of perturbed impulsive differential systems[J]. Applied Mathematics and Computation, 2003, 134: 445 - 457

[7] Sun Jitao, Zhang Yinping. Impulsive control of a nuclear sp in generator[J]. J of Computational and App lied Mathematics, 2003, 157(1): 235 - 242

[8] Soliman A A. On stability of impulsive differential systems[J]. Applied Mathematics and Computation, 2002, 133: 105 - 117

[9] Zhang Yu, Sun Jitao. Bound of the solutions of impulsive differential systems with time-varying delay[J]. Applied Mathematics and Computation, 2004, 154: 279 - 288

[10] 彭国强,黄立宏. 马尔可夫调配的随机微分方程的指数稳定性[J]. 广西师范大学学报,2005,23(2):44 - 47

[责任编辑:佟启巾]

(上接第 67 页)

0	5	13	16	12	10	19	21	23	26	27	29
5	0	8	11	7	5	14	16	18	21	22	24
13	8	0	3	8	10	6	8	10	13	14	16
16	11	3	0	5	7	3	5	9	12	11	15
12	7	8	5	0	2	8	10	14	17	16	20
10	5	10	7	2	0	10	12	16	19	18	22
19	14	6	3	8	10	0	2	6	9	8	12
21	16	8	5	10	12	2	0	4	7	6	10
23	18	10	9	14	16	6	4	0	3	8	6
26	21	13	12	17	19	9	7	3	0	5	3
27	22	14	11	16	18	8	6	8	5	0	8
29	24	16	15	20	22	12	10	6	3	8	0

若计算其它的例子,只需要在源程序中输入矩阵时作相应的改变即可.

参考文献:

[1] J.A. 邦迪, U.S.R. 默蒂. 图论及其应用[M]. 北京:科学出版社,1984

[2] Edward B. Magrab. MATLAB 原理与工程应用[M]. 高会生,李新叶,胡智奇,等译. 北京:电子工业出版社, 2002

[3] 余冬梅,张秋余,马少林,等. Dijkstra 算法的优化[J]. 计算机工程, 2004, 30(22):145 - 146

[4] 李擎,宋顶立,张双江,等. 两种改进的最优路径规划算法[J]. 北京科技大学学报, 2005, 27(3):367 - 370

[5] 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

[责任编辑:佟启巾]