第 36卷第 1期　　　数学的实践与认识　　　Vol. 36 No. 1
2006年 1月　MATHEMATICS IN PRACTICE AND THEORY　Jan., 2006

Kth最短路径的 Bellman改进算法

李　杰,　刘思峰,　任盈盈,　贾迎宾,　商红岩

(南京航空航天大学经济与管理学院, 南京　210016)

摘要:　基于对 Bellman算法的改进,得到了求解 kth最短路的新算法.改进算法的优势在于从 Bellman算法只能解决最短路问题拓展到求解 kth最短路问题,而且可以考虑权重为负数的情况.与传统算法相比,新算法更易于理解.

关键词:　$k$th最短路;最短路径;次短路径;路径追踪

1　引　　言

在最短路问题中, Bellman算法由于考虑到权重为负值的情况,而显得独树一帜,而且在表格中直接进行求解,使得求解过程比较简单、直观,容易掌握,而给人留下比较深的印象.可是它只能解决最短路问题,还没有应用到 kth最短路径的求解上面来,本文主要研究如何用 Bellman改进算法来解决 kth最短路径问题,以使得其应用领域更加广泛.

2　问题的定义和算法介绍

2.1　问题的定义

定义 1　算法 ($*$) Suppose $A = [a_{1j}]_{1\times n}$, $B = [b_{ji}]_{n\times n}$, then $C = A * B$ is defined as $C = [c_{1i}]_{1\times n}$, where $c_{1i} = \min\{a_{11} + b_{1i}, a_{12} + b_{2i}, \cdots, a_{1n} + b_{ni}\}$, for $j = 1, 2, \cdots, n$; $i = 1, 2, \cdots, n$.

定义 2　算法 ($*/$) Suppose $A = [a_{1j}]_{1\times n}$, $B = [b_{ji}]_{n\times n}$, $C = [c_{1i}]_{1\times n}$, and C is defined as in definition 1, then $D = A * B / C$ is defined as $D = [d_{1i}]_{1\times n}$, where $d_{1i} = \min\{[a_{11} + b_{1i}, a_{12} + b_{2i}, \cdots, a_{1n} + b_{ni}] / c_{1i}\}$, where / means take out c_{1i} from the set $A * B$, in the other words, if find out any $a_{1s} + b_{si} = c_{1i}$ $(1 \leq s \leq n)$, then we take out it from the set, and make the minimum value of the rest as d_{1i}.

注　在"/"运算时,得注意,如果被"/"的是个值,则去除集合中和这个值相同的项;如果是表达式,则只去掉和表达式一样的项.例如,假设 $c_{1i} = a_{1n} + b_{ni}$,则 $d_{1i} = \min\{[a_{11} + b_{1i}, a_{12} + b_{2i}, \cdots, a_{1n} + b_{ni}] / (a_{1n} + b_{ni})\} = \min\{a_{11} + b_{1i}, a_{12} + b_{2i}, \cdots, a_{1(n-1)} + b_{(n-1)i}\}$.

按照定义 2的方式,我们可以定义 ($*$ / /), $E = A * B / C / D$,只是从原来的 $A * B$集合中每次都同时去掉和 c_{ij}, d_{ij}相同的项,所谓的"/"运算,类似于集合运算中的"-",只不过这不是直接去除一个集合,而是在一个集合列中每次去掉另一个集合中对应的项.为简便起见,我们把 E改写成: $E = A * B / CD$.如果存在更多层次的"/"运算,按相似的理解来简化.

2.2　算法介绍

2.2.1　最短路的 Bellman算法介绍

收稿日期: 2005-09-28

按照上面的定义,下面我们将重新叙述一下最短路的 Bellman算法.

给定网络 $G=(V,E)$,其中 $V=\{vi|1\leq i\leq n\}$为网络节点的集合,n为节点的个数;$E=\{(vi,vj)|1\leq i,j\leq n,$ 且 $i\neq j\}$是链路的集合,其中链路是二元组(vi,vj),简记为 Eij,同时假定对于每一条链路 Eij来说均存在一个权值 Wij与其对应.

Bellman算法的优势就在于它不限定权值 Wij必须为正值,对于 $Wij\leq 0$的情况,也可以用 Bellman算法求解,而且它可以同时求出从某点到其它点的最短路.

我们把权值储存在 $b[n][n]$中,并把 b矩阵中对应于所要求的最短路的起点到其它点的权值储存到 $a[n]$中,并定义 $c[s][n]$为初始点(对应于 a的初始点)经过 s步到所有点的最短路权值集合(a矩阵是初始点到其它点的权值集合,b矩阵包含了所有点之间权值,a相当于 b矩阵中对应于初始点的那一行).

则最短路的 Bellman算法可以表述如下:

1) 按照算法“* ”,求得 $c[1][n]=a*b$;

2) $c[s][n]=c[s-1][n]*b$;

3) 归纳为 $c[s][n]=a*b^s$(注意的是这个运算只能从前往后运算,不能先算 b^s,且 b^s也是代表 $b*b$连乘 s次);

4) 检查 $c[s][n]$是否等于 $c[s-1][n]$,等于则结束,且最短路的权值就为 $c[s-1][n]$;否则,$s=s+1$,转入第(3)步.(算法中所有的 $c[s][n]$,$c[s-1][n]$指的都是一个集合,而不是这些集合的最后一项).

但是上面的算法没有考虑成圈的情况,下面我们介绍避圈的方法,在介绍避圈时,首先引入路径追踪问题.

我们定义 $p[i][s][m](m=1,2,\cdots,s)$为储存经过 s步到 i点的最短路每一中间点,但是不储存起点和终点,如果需要插入的点个数少于 s,则不足的在后面补充 0,例如第四步从 4点到 1点的路径为 4-3-2-1,则 $p[1][4][1]=3$,$p[1][4][2]=2$,$p[1][4][3]=p[1][4][4]=0$.

另外如果插入新的非零点,例如从 $p[i][s][m]$ $(m=1,2\cdots,s)$ 变成 $p[i][s+1][t](t=1,2\cdots,s,s+1)$插入了非零点 j,则从 $p[i][s][m](m=1,2\cdots,s)$的最后往前搜索,一直搜索到非零为止,并把 j插入到非零点的后面,而原来的零点往后挪一位,这个可以通过指针的插入实现! 而如果增加的点是零点,即最短路经不变,则只需在 $p[i][s][s]$后面补充一个 0即可,即变成 $p[i][s+1][s+1]$了.

2.2.2 包含避圈处理的 **Bellman**最短路算法

此外我们还定义一个临时数组 temp $[i][s][m](m=1,2\cdots,s)$;假设我们是求 h点到其它点的最短路,则可以直接令 $c[s][h]=0$,$p[h][s][m]=0(m=1,2\cdots,s)$,即到自身不存在最短路或 kth-shortest path问题.(下面讨论时,$i=h$,可以不讨论.)

1) 按照算法“* ”,求出 $c[1][i]=\min\{a_{11}+b_{1i},a_{12}+b_{2i},\cdots,a_{1n}+b_{ni}\}$,假定取得 $c[1][i]=a_{1x}+b_{xi}$,则 $p[i][1][1]=x$;第一步不存在避圈问题.

2) 当 $2\leq s$时,$c[s][i]=\min\{c_{(s-1)1}+b_{1i},c_{(s-1)2}+b_{2i},\cdots,c_{(s-1)n}+b_{ni}\}$,假定取得 $c[s][i]=c_{(s-1)j}+b_{ji}$,如果 $c[s][i]=c[s-1][i]$,则 $p[i][s][s]=0$,$p[i][s][t]=p[j][s-1][t](t=1,2,\cdots,s-1)$,其中 $1\leq m\leq s-1$;否则按照插入新点的规则,把点 j插入到

$p[i][s-1][t](t=1,2,\cdots,s-1)$中,作为 $temp[i][s][m]$ $(m=1,2,\cdots,s)$,并转入第 3)步;

3) 检验 $temp[i][s][m](m=1,2,\cdots,s)$中是否存在非零项重复,如果非零项出现次数大于 1次,则说明此路径存在圈,当舍去.重新计算 $c[s][i]=\min\{[c_{(s-1)1}+b_{1i},c_{(s-1)2}+b_{2i},\cdots,c_{(s-1)n}+b_{ni}]/[c_{(s-1)j}+b_{ji}]\}$,并转入第 2)步;如果非零项出现次数都小于等于 1次,则 $p[i][s][m]=temp[i][s][m]$ $(m=1,2,\cdots,s)$,并转入第 4)步;

4) 当所有的 $c[s][i](1\leq i\leq n)$都计算出来后,则检查 $c[s][i](1\leq i\leq n)$,是否都全都等于 $c[s-1][i]$,如果全相等,则说明第 $s-1$步已经达到最短路,我们就把 $s-1$步的相关数组 $c[s-1][i]$和 $p[i][s-1][s-1]$取出来,作为最终的最短路的参数;否则, $s=s+1$,转入第 2)步.

2.2.3 包含避圈处理的 Bellman次短路算法

先按 2.2.2中的算法求出第 s步最短路,并在此基础上来求第 s步次短路.下面就来介绍次短路的求法.

我们定义 $q[i][s][m](m=1,2,\cdots,s)$为储存经过 s步到 i点的次短路每一中间点,但是不储存起点和终点,如果需要插入的点个数少于 s,则不足的在后面补充 0. 插入新的非零点的方法,和最短路的 $p[i][s][m](m=1,2,\cdots,s)$一样. 并定义 $d[s][n]$为经过 s步到所有点的次短路权值集合.

此外我们还定义一个临时数组 $judge[i][s][s]$;假设我们是求 h点到其它点的次短路,则可以直接令 $d[s][h]=0,p[h][s][m]=0(m=1,2,\cdots,s)$,即到自身不存在最短路或 sth-shortest path问题.(下面讨论时,$i=h$,可以不讨论.)

1) 按照算法"* 1",求出 $d[1][i]=\min\{[a_{11}+b_{1i},a_{12}+b_{2i},\cdots,a_{1n}+b_{ni}]/c_{1i}\}$,假定取得 $d[1][i]=a_{1j}+b_{ji}$,则 $q[i][1][1]=j$; 第一步不存在避圈问题.

2) 当 $2\leq s$时,$d[s][i]=\min\{[c_{(s-1)1}+b_{1i},c_{(s-1)2}+b_{2i},\cdots,c_{(s-1)n}+b_{ni}]U[d_{(s-1)1}+b_{1i},d_{(s-1)2}+b_{2i},\cdots,d_{(s-1)n}+b_{ni}]\}/c[s][i]\}$.

情况①:假定取得的 $d[s][i]=c_{(s-1)j}+b_{ji}$,如果 $d[s][i]=d[s-1][i]$,则 $q[i][s][s]=0$, $q[i][s][t]=p[j][s-1][t]$,其中 $1\leq t\leq s-1$;否则按照插入新点的规则,把点 j插入到 $p[i][s-1][t](t=1,2,\cdots,s-1)$中,作为 $judge[i][s][m](m=1,2,\cdots,s)$,并转入第 3)步;

情况②:假定取得的 $d[s][i]=d_{(s-1)j}+b_{ji}$,如果 $d[s][i]=d[s-1][i]$,则 $q[i][s][s]=0,q[i][s][t]=q[i][s-1][t]$,其中 $1\leq t\leq s-1$;否则按照插入新点的规则,把点 j插入到 $q[i][s-1][t]$ $(t=1,2,\cdots,s-1)$中,作为 $judge[i][s][m](m=1,2,\cdots,s)$,并转入第 3)步;

3) 检验 $judge[i][s][m](m=1,2,\cdots,s)$中是否存在非零项重复,如果非零项出现次数大于 1次,则说明此路径存在圈,当舍去.如果是情况①,则 $d[s][i]=\min\{[c_{(s-1)1}+b_{1i},c_{(s-1)2}+b_{2i},\cdots,c_{(s-1)n}+b_{ni}]U[d_{(s-1)1}+b_{1i},d_{(s-1)2}+b_{2i},\cdots,d_{(s-1)n}+b_{ni}]\{/c[s][i]/[c_{(s-1)j}+b_{ji}]\}$,并转入第 2)步;如果是情况②,则 $d[s][i]=\min\{[c_{(s-1)1}+b_{1i},c_{(s-1)2}+b_{2i},\cdots,c_{(s-1)n}+b_{ni}]U[d_{(s-1)1}+b_{1i},d_{(s-1)2}+b_{2i},\cdots,d_{(s-1)n}+b_{ni}]\}/c[s][i]/[d_{(s-1)j}+b_{ji}]\}$,并转入第 2)步;

如果非零项出现次数都小于等于 1次,则 $q[i][s][m]=$ judge$[i][s][m]$ $(m=1,2,\cdots,s)$,并转入第 4)步;

4) 当所有的 $d[s][i]$, $1\leq i\leq n$都计算出来后,则检查 $d[s][i]$, $1\leq i\leq n$,是否都全都等于 $d[s-1][i]$,如果全相等,则说明第 $s-1$步已经达到最短路,我们就把 $s-1$步的相关数组 $d[s-1][i]$ $(i=1,2,\cdots,n)$和 $p[i][s-1][t]$ $(i=1,2,\cdots,n;\ t=1,2,\cdots,s-1)$取出来,作为最终的最短路的参数;否则,$s=s+1$,转入第 2)步.

注 对于 Kth-shortest path 问题,可以比照次短路和最短路的差别,定义运算("* // … /").

2.3 算法时间复杂度分析

从上面的算法中我们可以看出,次短路算法的复杂度和最短路的复杂度是类同的,假定最短路 Bellman算法的复杂度是 $O(1)$的话,则次短路 Bellman算法的复杂度是 $a^*O(1)$,其中 a是一个常数.同理,kth 最短路的复杂度也是同量级的,只是存在常数倍数关系.

2.4 应用实例

下面我们将用一种表格形式来展现次短路求法的过程.

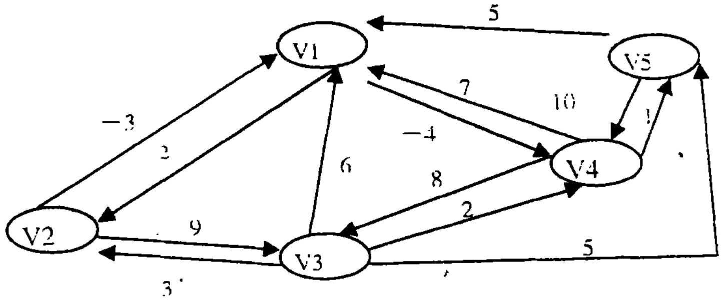

图 1

根据图 1,我们可以给出权重矩阵 B,并在此基础上,求点 V3到各点的次短路和最短路.

$$B=\begin{bmatrix}0 & 2 & \infty & 7 & \infty\\ -3 & 0 & 3 & \infty & \infty\\ 6 & 9 & 0 & 2 & 5\\ -4 & \infty & 8 & 0 & 1\\ 5 & \infty & \infty & 10 & 0\end{bmatrix}$$

相应的可以取得点 V3到各点的有向弧权重集合 $A=[6\ 9\ 0\ 2\ 5]$. 我们假定 $l(s)$为经过 s步得到的最短路权重集合, $g(s)$为经过 s步得到的次短路权重集合.

表格最下方对应的是第 s步到相应点的最(次)短路路径. 因为 $l(3)=l(2)$, $g(3)=g(2)$,所以 $l(2)$, $g(2)$对应的就是最终的最短路和次短路的权重集合.

3 结 论

通过实例检验,用 Bellman改进算法能够较快地找到所需的 $k$th最短路径,而且用表格的形式表现,较为直观.

表 1

	$V1$	$V2$	$V3$	$V4$	$V5$	A	$l(1)$	$g(1)$	$l(2)$	$g(2)$	$l(3)$	$g(3)$
$v1$	0	2	∞	7	∞	6	−2	6	−2	6	−2	6
$v2$	−3	0	3	∞	∞	9	8	9	0	8	0	8
$v3$	6	9	0	2	5	0	0	0	0	0	0	0
$v4$	−4	∞	8	0	5	2	2	13	2	5	2	5
$v5$	5	∞	∞	10	0	5	5	7	5	7	5	7
$v3 \to v1$							3→4→1	3→1	3→4→1	3→1	3→4→1	3→1
$v3 \to v2$							3→1→2	3→2	3→4→1→2	3→1→2	3→4→1→2	3→1→2
$v3 \to v3$							3→3	3→3	3→3	3→3	3→3	3→3
$v3 \to v4$							3→4	3→1→4	3→4	3→1→4	3→4	3→1→4
$v3 \to v5$							3→5	3→4→5	3→5	3→4→5	3→5	3→4→5

参考文献:

[1] 严蔚敏,吴伟民主编. 数据结构[M]. 北京: 清华大学出版社.

[2] 王明中,谢剑英,陈应麟. 一种新的 $k$th最短路径搜索算法[A]. 计算机工程与应用, 2004. 30.

[3] 吴敏,苏厚勤,王明中. $k(\leq 3)$条渐次最短路径搜索算法的研究及其实现技术[A]. 计算机应用与软件, 2004, 21(8).

[4] 李引珍,郭耀煌. 运输网络最短路径关键点问题研究[A]. 铁道学报, 2004, 26(6).

[5] 《运筹学》教材编写组. 运筹学[M]. 北京: 清华大学出版社.

[6] 王树禾. 图论及其算法[M]. 合肥: 中国科技大学出版社, 1990.

An Improved Bellman Algorithm for the Kth-shortest Path Problem

LI Jie, LIU Si-feng, REN Ying-ying, JIA Ying-bin, SHANG Hong-yan

(College of Economics & Management, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract Based on the Bellman algorithm, we make an improvement to get a new algorithm of solving the kth shortest path problem. The advantage of this method is expanding the coverage of Bellman algorithm from solving the shortest path to the kth-shortest path, and taking the minus weight value into account. Furthermore, the new algorithm itself is not far to seek, comparing with the traditional one.

Keywords kth-shortest path; Shortest path 2nd-shortest path; Path tracing