固定序 Bellman-Ford 算法的一个改进

韩伟一

(哈尔滨工业大学 经济与管理学院, 150001 哈尔滨)

摘 要: 通过对固定序 Bellman-Ford 算法进行修正, 获得了一种求解边数不大于 k 的最短路问题的新算法.相对于原始算法,修正后的算法通过改变点的标号过程,使得在第 k 次迭代后每一条路径的边数均不超过 k. 新算法被证明是正确的,它的计算复杂性为 $O(km)$. 实验表明,在大规模情形下,相对于修正的先进先出算法,该算法具有显著的竞争优势.

关键词: 算法;Bellman-Ford 算法; 先进先出; 固定序; 最短路问题

中图分类号: TP301.6 **文献标志码**: A **文章编号**: 0367-6234(2014)11-0058-06

An improvement on fixed order Bellman-Ford algorithm

HAN Weiyi

(School of Economic and Management, Harbin Institute of Technology, 150001 Harbin, China)

Abstract: In the paper, Bellman-Ford algorithm with fixed order is modified in order to solve the shortest path problem with not more than k edges. After the k-th iteration, each path must own no more than k edges by modifying the labeling process of the origin algorithm. The modified algorithm proves true and its worst-case complexity is $O(km)$. In contrast to the modified Bellman-Ford algorithm with FIFO order, experiments show that the algorithm has the significant competitive advantage in the large-scale case.

Keywords: algorithm; Bellman-Ford algorithm; FIFO order; fixed order; the shortest path problem

自 1958 年以来,Bellman-Ford 算法一直被公认为是求解负权最短路问题最好的强多项式算法,它的研究历程可概括为 3 个发展阶段[1-3]: 第 1 阶段,动态规划模式阶段, 即原始的 Bellman-Ford 算法;第 2 阶段,基本改进阶段,在第 1 阶段的每次迭代中所有点都要参与计算,而在第 2 阶段中只有在上次迭代中距离标号改变的点才参与下一次迭代, 从而使得计算效率显著提高, 迭代次数明显减少[4-8];第 3 阶段,加速技术阶段,主要体现为上次迭代中距离标号改变的点并不全部参与下一轮迭代计算,因而还可进一步提高计算效率.第 3 阶段的算法一般以第 2 阶段的算法为基础,因而第 2 阶段的算法也称为 Bellman-Ford 算法的基本改进算法.目前, 主要有 5 类基本改进算法, 即先进先出算法[4,8]、后进先出算法[8]、Yen 算法[9]、快速算法[10]和固定序算法[1-2]等.第 3 阶段的主要工作有 1993 年 Goldberg 等[11]提出的拓扑序技术,1981 年 Tarjan[12]提出的装配子树技术,以及 1985 年 Glover 等[13-14]提出的门槛技术等,1996 年Cherkassky 等[4,8]提出的父点技术等.需要指出的是,尽管第 2、3 阶段的算法都具有很高的效率,但由于最短路径中的边数可能会多于 k 条,因而均不能解决边数不大于 k 的最短路问题(或边数 $\leqslant k$ 的最短路问题),为此文献[2]对先进先出算法进行了修正.本文将对固定序算法进行修正,使得修正后的算法能够解决有边数限制的最短路问题,并将该算法与已有的先进先出算法进行比较,用实验说明它在大规模情形下具有显著的竞争优势.

1 固定序算法及其修正

对于一个具有 n 个点、m 条边的有向图 $G(V, E)$,n 个点分别编号为 1, 2, …, n,且规定点 1 为源点,用 $w(i,j)$ 表示有向边 (i,j) 的权.定义 $d(i)$

收稿日期: 2004-03-02.

基金项目: 国家自然科学基金(71101037); 中央高校基本科研业务费专项资金(HIT.HSS.201406).

作者简介: 韩伟一(1974—), 男, 讲师,硕士生导师.

通信作者: 韩伟一, wyhan@hit.edu.cn.

为点 i 的距离标号，h 为迭代次数. 固定序算法如下：

Step 1　初始化. 令 $d(1)=0$，$d(i)=+\infty$，$2\leqslant i\leqslant n$. 把点 1 加入集合 A.对 h 赋值为 1.

Step 2　在第 h 次迭代中，按照编号顺序对 A 中的各点 i 进行计算

$$d(j)=\min_{(i,j)\in E}\{d(j),d(i)+w(i,j)\},$$

如果 $d(j)$ 变小，则当 $j\notin A$ 时，把点 j 加入集合 A. 从集合 A 中去除点 i.

Step 3　如果集合 A 为空集，则算法结束，$d(i)$ 就是从点 1 到点 i 的最短距离；当集合 A 非空且 $h=n-1$ 时，可判定存在负循环，算法结束；当集合 A 非空且 $h<n-1$ 时，执行 Step 4.

Step 4　令 $h=h+1$，转到 Step 2.

尽管固定序算法在求解负权最短路问题上具有较高的计算效率，但由于求得的是最短路径，而最短路径中可能会多于 k 条边，因而它也无法直接求解边数 $\leqslant k$ 的最短路问题，必须对它加以修正.

在对固定序算法修正以前，需要引入一些新的定义和符号：1) 定义 $d_0(i)$ 为上次迭代完成后点 i 的距离标号；2) 在修正的算法中，定义集合 A 为上轮迭代中距离标号变小的点的集合，定义集合 B 为本轮迭代中距离变小的点的集合. 修正后的固定序算法如下：

Step 1　初始化. 令 $d(1)=0$，$d(i)=+\infty$，$2\leqslant i\leqslant n$，把点 1 加入集合 A.迭代次数 h 赋初值为 1，则第 0 次迭代的距离标号分别为

$$d^0(1)=0,d^0(i)=+\infty,\ 2\leqslant i\leqslant n.$$

Step 2　在第 h 次迭代中，按照编号顺序对 A 中的各点 i 进行计算

$$d(j)=\min_{(i,j)\in E}\{d(j),d^0(i)+w(i,j)\},$$

如果 $d(j)<d^0(j)$，且当点 $j\notin B$ 时，把点 j 加入集合 B.从集合 A 中去除点 i.

Step 3　如果集合 B 为空集或 $h=k$ 时，则算法结束，$d(i)$ 就是从点 1 到点 i 的最短距离；当集合 B 非空且 $h<k$ 时，执行 Step 4.

Step 4　清空集合 A，并把集合 B 中的元素转到集合 A，再清空集合 B，同时对任意的点 $i(1\leqslant i\leqslant n)$，令

$$d^0(i)=d(i),$$

再令 $h=h+1$，转到 Step 2.

2　修正的先进先出算法

文献[2]对基于先进先出序的 Bellman-Ford 算法进行了修正，使之能够解决边数不大于 k 的最短路问题，特称为修正的先进先出算法.作为 Bellman-Ford 算法的一种基本改进算法，先进先出算法被公认为是解决负权最短路问题最快、最有影响力的算法，第 3 阶段的加速算法都是以它为基础进行改进的.在文献[2]中，修正的先进先出算法不仅可以求解边数 $\leqslant k$ 的最短路问题，而且显示出了卓越的计算效率，相对于原始的 Bellman-Ford 算法效率可提高近 30 倍.需要指出的是，原始的 Bellman-Ford 算法可以求解边数 $\leqslant k$ 的最短路问题，但第 2、3 阶段的改进算法均不可以.

修正的先进先出算法仍然采用相应固定序算法的问题描述和符号，算法如下：

Step 1　初始化. 令 $d(1)=0$，$d(i)=+\infty$，$2\leqslant i\leqslant n$，把点 1 加入集合 A.迭代次数 h 赋初值为 1，则第 0 次迭代的距离标号分别为

$$d^0(1)=0,d^0(i)=+\infty,\ 2\leqslant i\leqslant n.$$

Step 2　在第 h 次迭代中，按照进入顺序对 A 中的各点 i 进行计算

$$d(j)=\min_{(i,j)\in E}\{d(j),d^0(i)+w(i,j)\},$$

如果 $d(j)<d^0(j)$，且当点 $j\notin B$ 时，把点 j 加入集合 B.从集合 A 中去除点 i.

Step 3　如果集合 B 为空集或 $h=k$ 时，则算法结束，$d(i)$ 就是从点 1 到点 i 的最短距离；当集合 B 非空且 $h<k$ 时，执行 Step 4.

Step 4　清空集合 A，并把集合 B 中的元素转到集合 A，再清空集合 B，同时对任意的点 $i(1\leqslant i\leqslant n)$，令

$$d^0(i)=d(i),$$

再令 $h=h+1$，转到 Step 2.

需要指出的是，本文对文献[2]中的修正的先进先出算法的表述错误进行了纠正，即把 $d(j)=\min_{(i,j)\in E}\{d^0(i),d^0(i)+w(i,j)\}$ 修正为 $d(j)=\min_{(i,j)\in E}\{d(j),d^0(i)+w(i,j)\}$.

比较两种修正算法，可以看出两种算法仅仅在 Step 2 有微小的差别，也就是修正的先进先出算法参与计算的点按照先进先出的顺序，而修正的固定序算法则按照事先给定的编号顺序.两个算法满足如下定理.

定理 1　对于同一个问题，修正的先进先出算法和修正的固定序算法不仅具有相同的迭代次数，而且每一次迭代使用的集合 A 和迭代后形成的集合 B 也是相同的.

证明　两个算法仅仅在 Step 2 存在一定的差

别,其他步骤都是一样的.在 Step 2 中,修正的先进先出算法按照先进先出的顺序对 A 中的点进行计算,而修正的固定序算法按照给定的顺序对 A 中的点进行计算.如果每一次迭代使用的集合 A 是相同的,而且得到的计算结果也是相同的,那么两个算法无疑具有同样的迭代次数.

事实上,如果集合 A 是相同的,由 $d(j)=\min\limits_{(i,j)\in E}\{d(j),d^0(i)+w(i,j)\}$ 可知,两个算法的计算结果必定是一致的,也就是集合 B 中的点也是相同的.既然两个算法给定的初始集合 A 是相同的,且下一次参与计算的集合 A 是由本次迭代计算得到的集合 B 转换而来,这样就保证了每一次迭代使用的集合 A 和迭代后形成的集合 B 总相同.因而,两算法每一次迭代的计算结果必然相同.命题得证.

定理 1 间接说明,本文的算法确实可以求解边数 $\leqslant k$ 的最短路问题.也进一步表明,修正的先进先出算法和修正的固定序算法应该具有同样的计算复杂度.由文献[2],有如下引理.

引理 1 算法在 k 次迭代后,如果 $d(i)<+\infty$,那么可得到从点 1 到点 i 的边数不大于 k 的一条最短路径,这条路径的长度恰为 $d(i)$.

证明 参见文献[2]的定理 1.

定理 2 修正的先进先出算法和修正的固定序算法最坏情形下的计算复杂度为 $O(km)$.

证明 在每次迭代中,由于每一条边参与 Step 2 的计算过程最多一次,因而每一次迭代的计算复杂度为 $O(m)$.根据引理 1,对于求解边数 $\leqslant k$ 的最短路问题,迭代次数不会超过 k.因而,两个算法最坏情形下的计算复杂度为 $O(km)$.

需要指出的是,文中提到的 3 个阶段的 Bellman-Ford 算法最坏情形下的计算复杂性均为 $O(nm)$.

3 算法的实验评估

由于修正的先进先出算法、修正的固定序算法两种算法具有相同的、最坏情形的计算复杂度,因而只能通过实验来比较两个算法的计算效率.本文将进行两个实验:1)设定最短路径中的边数上限 k 为 $n/4$,对两种算法在多种稀疏程度、多个规模水平下进行评估,以测试稀疏程度和计算规模两个因素对两个算法的影响,这里 n 为有向图的点数;2)给定计算规模,针对不同的 k 对两种算法进行评估,以测试不同 k 对算法的影响.实验所用有向图的权重服从均匀分布,可取 1~100 000 之间的任一整数.每种图实验 10 000 个例子.实验用的计算机为联想 ThinkPad 笔记本,CPU 为 Intel i5-2410M 2.30 GHz,内存为2.0 G.算法的比较都使用相同的例子.算法程序均采用结构化语言 Delphi7.0.

3.1 稀疏程度和计算规模的影响实验

本文用边数与点数之比来表示图的稀疏程度.在每种图中,设计了 10、50、200 和 1 000 等 4 种稀疏情形进行评估,主要原因是它们代表了 4 种典型的稀疏程度,即:1)当稀疏程度≤10 时,一般认为是超稀疏的;2)当稀疏程度≤ $\log n$ 时,一般认为是稀疏的,按照本文的实验规模,50 接近于这个水平;3)当稀疏程度接近于 $\sqrt{n}$ 时,一般认为是超稠密的,按照本文的实验规模,1 000 接近于超稠密水平,而 200 可认为是稠密的水平.之所以不选择更大的稀疏程度进行实验,根本原因是:如果稠密程度 >2 000 时,无法在 > 100 000 个点的规模水平下开展实验,将发生内存溢出,不能形成完整的实验结果以进行对比分析.

本文同时选择了 12 类规模水平进行试验评估,其中小规模水平(点数 < 10 000 的图)4 种、中规模水平(点数介于 10 000 ~ 100 000 的图)4 种、大规模水平(点数 > 100 000 的图)4 种.k 取值为 $n/4$.之所以设定最短路径中的边数上限 k 为 $n/4$,主要是保证尽可能多的迭代次数.相关评估结果如表 1 ~ 12 所示.

表 1 2 000 个点下的平均计算时间 ms

稀疏程度	修正的先进先出算法	修正的固定序算法
10	0.28	0.45
50	0.91	1.42
200	3.19	5.37
1 000	14.61	27.03

表 2 4 000 个点下的平均计算时间 ms

稀疏程度	修正的先进先出算法	修正的固定序算法
10	0.64	0.93
50	2.04	3.07
200	8.19	11.94
1 000	31.80	56.16

表 3 6 000 个点下的平均计算时间 ms

稀疏程度	修正的先进先出算法	修正的固定序算法
10	1.04	1.50
50	3.27	4.69
200	13.20	18.11
1 000	52.78	89.78

表 4　8 000 个点下的平均计算时间　ms

稀疏程度	修正的先进先出算法	修正的固定序算法
10	1. 45	1. 95
50	5. 23	6. 87
200	19. 07	25. 73
1 000	74. 05	122. 11

表 5　20 000 个点下的平均计算时间　ms

稀疏程度	修正的先进先出算法	修正的固定序算法
10	4. 49	5. 56
50	18. 70	18. 56
200	57. 72	67. 59
1 000	230. 13	346. 72

表 6　40 000 个点下的平均计算时间　ms

稀疏程度	修正的先进先出算法	修正的固定序算法
10	11. 88	12. 37
50	45. 95	41. 05
200	145. 76	149. 13
1 000	552. 08	728. 76

表 7　60 000 个点下的平均计算时间　ms

稀疏程度	修正的先进先出算法	修正的固定序算法
10	21. 94	20. 08
50	84. 02	68. 42
200	259. 12	239. 17
1 000	1 031. 78	1 204. 72

表 8　80 000 个点下的平均计算时间　ms

稀疏程度	修正的先进先出算法	修正的固定序算法
10	33. 28	25. 15
50	104. 85	64. 89
200	247. 94	190. 53
1 000	906. 89	899. 48

表 9　120 000 个点下的平均计算时间　ms

稀疏程度	修正的先进先出算法	修正的固定序算法
10	63. 45	39. 83
50	175. 83	103. 97
200	416. 71	312. 66
1 000	1 472. 65	1 410. 24

表 10　140 000 个点下的平均计算时间　ms

稀疏程度	修正的先进先出算法	修正的固定序算法
10	78. 87	47. 81
50	218. 79	123. 18
200	563. 15	415. 91
1 000	1 795. 45	16 94. 27

表 11　160 000 个点下的平均计算时间　ms

稀疏程度	修正的先进先出算法	修正的固定序算法
10	94. 54	55. 86
50	262. 96	140. 45
200	596. 61	433. 86
1 000	2 105. 35	1 976. 65

表 12　180 000 个点下的平均计算时间　ms

稀疏程度	修正的先进先出算法	修正的固定序算法
10	113. 25	64. 47
50	306. 53	166. 96
200	730. 92	509. 35
1 000	内存不足	内存不足

根据表 1~12 的实验数据，可得如下结论：

1）相对于修正的先进先出算法，在既定的规模水平下，随着稀疏程度的增加，修正的固定序算法的竞争优势将越来越不显著甚至丧失. 如在 6 000 个点的规模水平下，修正的固定序算法在稀疏程度为 10 时具有更快的计算效率，但当稀疏程度增加为 1 000 时却丧失了竞争优势.

2）相对于修正的先进先出算法，在既定的稀疏程度下，随着规模水平的增大，修正的固定序算法的竞争优势将越来越显著. 如在稀疏程度为 10 的情形下，当规模水平为 2 000 个点时，修正的固定序算法比修正的先进先出算法计算效率平均慢 60. 7%，而当规模水平为 180 000 个点时，修正的固定序算法反而比修正的先进先出算法计算效率平均快 75. 6%.

3）相对于修正的先进先出算法，当规模水平足够大时，在所有的稀疏程度下，修正的固定序算法将具备完全的竞争优势. 如当规模水平 < 10 000 个点时，修正的固定序算法在 4 种稀疏程度情形下均处于劣势，而当规模水平 > 80 000 个点时，这个算法在所有情形下均赢得了优势.

3. 2　k 的影响实验

首先选择 20 000 个点和 160 000 个点两个规模水平，k 分别取值为 5、10、15 和 $n/4$ 进行实验，如表 13 ~ 15 所示. 鉴于 k 为 5、稀疏程度为 10 时，计算结果出现异常，又增加了两个规模水平（即 60 000 个点和 100 000 个点）进行实验，以给出稳定的规律，如表 16 所示.

根据表 13~16，可得如下结论：

1）由表 15 可以看出，修正的固定序算法在大规模情形下具有压倒性的竞争优势，16 种情形下仅当 k 为 5、稀疏程度为 10 时弱于修正的先进先出算法.

2) 由表 15 还可以看出,在给定的规模水平和稀疏程度情形下,当 k 取适当值时对修正的固定序算法非常有利,但当 k 进一步变大或变小时这种有利现象将减弱或消失. 如在规模水平为 160 000 个点、稀疏程度为 10 的情形下,k 为 10 时最为有利.

3) 由表 16 可以看出, 当 k 和稀疏程度同时较小时, 修正的先进先出算法具有绝对的竞争优势,并随着规模的增大优势愈加突出.需要指出的是,在表 16 中出现了一个反常值,即计算时间值 0.78,但从两个算法的计算时间比来看,规律是十分明显的.

表 13 20 000 个点、不同 k 下的平均计算时间 ms

稀疏程度	修正的先进先出算法				修正的固定序算法			
	$k=5$	$k=10$	$k=15$	$k=n/4$	$k=5$	$k=10$	$k=15$	$k=n/4$
10	0.39	8.20	14.13	15.593	0.73	9.75	16.56	20.35
50	14.44	47.22	54.49	56.070	15.47	51.43	61.93	66.14
200	65.49	150.63	170.27	172.810	86.96	204.93	235.54	242.40
1 000	309.76	624.04	692.80	707.760	518.63	1 052.87	1 184.35	1 212.59

表 14 160 000 个点、不同 k 下的平均计算时间 ms

稀疏程度	修正的先进先出算法				修正的固定序算法			
	$k=5$	$k=10$	$k=15$	$k=n/4$	$k=5$	$k=10$	$k=15$	$k=n/4$
10	0.860	57.14	90.69	94.54	4.430	25.51	42.75	55.86
50	49.762	199.10	251.34	262.90	25.654	92.96	126.48	140.45
200	182.930	477.05	578.42	596.60	120.750	319.47	400.48	433.86
1 000	762.550	1 683.87	2 019.84	2 105.35	691.340	1 544.21	1 860.48	1 976.65

表 15 修正的先进先出算法与修正的固定序算法计算时间之比

稀疏程度	20 000 个点				160 000 个点			
	$k=5$	$k=10$	$k=15$	$k=n/4$	$k=5$	$k=10$	$k=15$	$k=n/4$
10	0.534	0.841	0.853	0.766	0.194	2.240	2.121	1.692
50	0.933	0.918	0.880	0.848	1.940	2.142	1.987	1.872
200	0.753	0.735	0.723	0.713	1.515	1.493	1.444	1.375
1 000	0.597	0.593	0.585	0.584	1.103	1.090	1.086	1.065

注: 两个算法的计算时间由表 13 和表 14 给出.

表 16 k 为 5、稀疏程度为 10、不同规模水平下的平均计算时间 ms

规模	修正的先进先出算法	修正的固定序算法	两个算法的时间比
20 000	0.390	0.730	0.534
60 000	0.780	1.920	0.408
100 000	0.580	2.780	0.210
160 000	0.860	4.430	0.194

由上述两个实验可以看出,尽管两个算法具有同样的计算复杂性,但是在不同参数、稀疏程度和规模水平下却体现了不同的计算效率.因为两个算法在Step 2中分别采取了不同的数据结构,其中修正的先进先出算法采用了优先队列来存储参与迭代的点,而修正的固定序算法则是逐一判断各点是否可参与迭代.后者相对简单,易于实现,又能在大规模情形下体现出明显的竞争优势,因而具有一定的理论和实践价值.

4 结 论

1) 本文通过对固定序 Bellman-Ford 算法进行修正,获得了一种求解边数≤ k 的最短路问题的新算法——修正的固定序算法.

2) 在大规模情形下,相对于文献[2]提出的修正的先进先出算法,修正的固定序算法相对简单、易于实现,实验表明它具有显著的竞争优势,仅仅在少数情形下处于落后状态. (下转第 69 页)