第44卷 第7期
2012年7月

哈 尔 滨 工 业 大 学 学 报
JOURNAL OF HARBIN INSTITUTE OF TECHNOLOGY

Vol. 44 No. 7
July 2012

经典 Bellman-Ford 算法的改进及其实验评估

韩伟一

(哈尔滨工业大学 经济与管理学院, 150001 哈尔滨)

摘 要: 针对以高效求解有边数限制的最短路问题,对经典 Bellman-Ford 算法进行了改进. 借鉴划分算法的思想,通过减少距离标号的数目,得到了两个改进算法. 既然已有的改进算法均不能解决有边数限制的最短路问题,因而本算法是经典 Bellman-Ford 算法的全新改进. 相对于经典 Bellman-Ford 算法,改进后的算法不仅可有效地节省存储空间, 而且实验表明能显著地提高计算效率.

关键词: 算法; Bellman-Ford 算法; 划分算法; 最短路问题

中图分类号: TP301.6 **文献标志码**: A **文章编号**: 0367-6234(2012)07-0074-04

Improvement and experimental evaluation on classical Bellman-Ford algorithm

HAN Wei-yi

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

Abstract: Classical Bellman-Ford algorithm is improved to solve the shortest path problem with bounded edge number efficiently. Using the experience of partitioning algorithm for reference, two improved algorithms are obtained, which can decrease the number of distance labels of vertices. Since all existing improved Bellman-Ford algorithms can't solve the shortest path problem with bounded edge number, these two improved algorithms are entirely new. In contrast to the common version of Bellman-Ford algorithm, these two improved algorithms can save storage space efficiently, and can raise computing efficiency remarkably.

Key words: algorithm; Bellman-Ford algorithm; partitioning algorithm; the shortest path problem

Bellman-Ford 算法一般有两种表述,一种为经典的动态规划模式或经典算法[1-3],另外一种为划分模式或划分算法(Partitioning algorithm)[4-7]. 严格意义上, 划分算法并不是普遍公认的经典 Bellman-Ford 算法,而是经过改进的 Bellman-Ford 算法,它把参与第 i 轮和第 $i+1$ 轮迭代的点进行了明确划分,改进了距离标号的计算方式,能够显著提高计算效率. 目前,采用先进先出策略的划分算法是解决负权最短路问题、最短路问题可行性问题的最有竞争力的算法,在算法基础上再加入门槛技术即成为解决负权最短路问题当前公认的最快算法[8-9]. 尽管经典的 Bellman-Ford 算法在解决负权最短路问题、最短路问题可行性问题已经不具有优势,但它仍然是解决有边数限制最短路问题的最好算法[10-11]. 本文将研究经典 Bellman-Ford 算法及其性质,并将对之进行改进,使之能更快地解决有边数限制最短路问题,同时将对改进后的算法进行实验评估.

收稿日期: 2012-04-01.

基金项目: 国家自然科学基金资助项目(70672063); 哈尔滨工业大学青年优秀基金资助项目(2009036).

作者简介: 韩伟一(1974—), 男, 博士, 讲师.

通信作者: 韩伟一, wyhan@mails.gucas.ac.cn.

1 经典 Bellman-Ford 算法的两种形式及其性质

对于一个具有 n 个点、m 条边的有向图 $G(V, E)$, n 个点分别以 1, 2, …, n 为编号,且规定点 1 为始点, $w(i,j)$ 为有向边 (i,j) 的权. 定义 $d_k(i)$ 为点 i 在第 k 次迭代后 i 点的距离标号,则可给出经典 Bellman-Ford 算法的常见形式为:

Step 1 初始化. 令 $d_0(1)=0, d_0(i)=$

$+\infty$,$(2 \leqslant i \leqslant n)$.

Step 2　对每一条边 (i,j) 按照给定顺序进行计算

$$d_k(j) = \min\{d_{k-1}(j), d_{k-1}(i) + w(i,j)\}.$$

Step 3　当所有的点 $i(1 \leqslant i \leqslant n)$ 均满足 $d_{t-1}(i) = d_t(i)$,$(t \leqslant n)$ 时,$d_t(i)$ 就是从点 1 到 i 的最短距离;当存在某个点 j 满足 $d_{n-1}(j) \neq d_n(j)$ 时,可判定存在负循环,算法结束.

对于以上形式的 Bellman-Ford 算法,可作两方面的改进:1) 当 $d_{k-1}(i) = d_{k-2}(i)$ 时,在第 k 次迭代中 Step 2 中边 (i,j) 所进行的计算不会造成点 j 的距离标号的改变,从而对边 (i,j) 所进行的计算可以省略,这样可把 Step 2 的计算量由 $\boldsymbol{\Theta}(m)$ 降低为 $O(m)$;2) 算法结束的判定规则相对复杂,计算量比较大,可作进一步改进.

定义 $\boldsymbol{A}_k$ 为在第 $k-1$ 轮迭代中距离标号减少的点的集合,则经典 Bellman-Ford 算法又可采取下列精简形式为:

Step 1　初始化. 令 $d_0(1) = 0$,$d_0(i) = +\infty$,$(2 \leqslant i \leqslant n)$,把所有点加入集合 $\boldsymbol{A}_1$.

Step 2　对于任意点 $i(1 \leqslant i \leqslant n)$,令 $d_k(i) = d_{k-1}(i)$.

Step 3　按照进入顺序对 $\boldsymbol{A}_k$ 中的各点 i 进行计算为

$$d_k(j) = \min_{(i,j) \in E}\{d_{k-1}(j), d_{k-1}(i) + w(i,j)\}.$$

如果 $d_k(j) < d_{k-1}(j)$,且当点 $j \notin \boldsymbol{B}$ 时,则把点 j 加入集合 $\boldsymbol{A}_{k+1}$.

Step 4　如果集合 $\boldsymbol{A}_{k+1} = \varnothing$,则算法结束,$d(i)$ 就是从点 1 到 i 的最短距离;当集合 $\boldsymbol{A}_{k+1} \neq \varnothing$,且 $k = n-1$ 时,可判定存在负循环,算法结束;当集合 $\boldsymbol{A}_{k+1} \neq \varnothing$,且 $k < n-1$ 时,执行 Step 5.

Step 5　令 $k = k+1$,转到 Step 2.

关于经典 Bellman-Ford 算法的高效实现,文献[8,11]分别介绍了 Bellman 本人的改进思想,但已有的改进算法均不适用于有边数限制最短路问题,且没有专门论述此问题. 上述精简形式可解决有边数限制最短路问题,相对于常见形式计算效率有了显著地改进.

经典 Bellman-Ford 算法无论采用上述哪种形式,都具有以下性质:

性质 1　算法的迭代次数为最短路径树的深度.

性质 2　算法每次迭代的计算量均为 $O(m)$.

性质 3　算法的最坏计算复杂性为 $O(nm)$.

性质 4　算法经过 $n-1$ 次迭代后就可判定负循环的存在.

上述 4 个性质已经由相关文献证实,本文不给出具体证明. 但针对性质 1 将给出以下结论:

定理 1　算法在 k 次迭代后,如果 $d_k(i) < +\infty$,那么可得到从始点 1 到 j 的边数 $\leqslant k$ 的一条最短路径,这条路径的长度恰为 $d_k(i)$.

证明　用数学归纳法证明,显然 $k = 1$ 时,命题成立.

假设 $k = t-1$ 时,命题仍然成立. 假定从始点 1 到 j 的边数 $\leqslant t$ 的最短路径中,点 j 的上一个点为 $i(i \neq j)$. 这样一来,要获得从始点 1 到 j 的边数 $\leqslant t$ 的最短路径,就必须首先获得从始点 1 到 $j(j \neq i)$ 的边数 $\leqslant t-1$ 的最短路径. 那么当 $k = t$ 时

情形 1　如果 $d_t(j) < d_{t-1}(j)$,由常见形式的 Step 2,则有

$$d_t(j) = \min_{(h,j) \in E}\{d_{t-1}(j), d_{t-1}(h) + w(h,j)\}.$$

根据上述假定,那么必有 $d_t(j) = d_{t-1}(i) + w(i,j)$. 再根据归纳法的假设,$d_{t-1}(h)$ 是从始点 1 到 h 的边数 $\leqslant t-1$ 的最短路径,因此命题成立.

情形 2　如果 $d_t(j) = d_{t-1}(j)$,说明在从始点 1 到 $h(h \neq j)$ 的边数 $\leqslant t-1$ 的最短路径的基础上增加边 (h,j) 新形成的路径均不优于从始点 1 到 j 的边数 $\leqslant t-1$ 的最短路径,这样一来从始点 1 到 j 的边数 $\leqslant t-1$ 的最短路径就是从始点 1 到 j 的边数 $\leqslant t$ 的最短路径,命题仍然成立.

综合上述两种情形,命题对于 $k = t$ 时也成立,从而定理 1 正确.

应用定理 1,只需把经典 Bellman-Ford 算法的迭代数限定为 k,就可解决限制边数 $\leqslant k$ 的最短路问题.

2　经典 Bellman-Ford 算法的改进

尽管精简形式的经典 Bellman-Ford 算法已经具有较高的计算效率,但距离标号的数目最多可达到 n^2 个. 相对于常见形式的 Bellman-Ford 算法,又增加了集合 $\boldsymbol{A}_k$,这将进一步增加算法的存储空间. 针对上述两点,本文将得到新的改进算法.

定义 $d_0(i)$ 为上次迭代后点 i 的距离标号,同时定义集合 $\boldsymbol{A}$ 为上轮迭代中距离标号变小的点的集合. 本文给出第 1 个改进算法为:

Step 1　初始化. 另 $d(1) = 0$,$d(i) = +\infty$,$(2 \leqslant i \leqslant n)$,把所有点加入集合 $\boldsymbol{A}$.

Step 2　在第 k 次迭代中,按照进入顺序对集合 $\boldsymbol{A}$ 中的各点 i 进行计算为

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

如果 $d(j) < d^0(j)$,且当点 $j \notin \boldsymbol{B}$ 时,把点 j 加

入集合 $\boldsymbol{B}$.

Step 3 如果集合 $\boldsymbol{B}=\varnothing$,则算法结束,$d(i)$ 就是从点1 ~ i 的最短距离;当集合 $\boldsymbol{B}\neq\varnothing$,且 $k=n-1$ 时,可判定存在负循环,算法结束;当集合 $\boldsymbol{B}\neq\varnothing$,且 $k<n-1$ 时,执行 Step 4.

Step 4 清空集合 $\boldsymbol{A}$,并把集合 $\boldsymbol{B}$ 中的元素按先进先出顺序转到集合 $\boldsymbol{A}$,再清空集合 $\boldsymbol{B}$,同时对任意的点 $i(1\leqslant i\leqslant n)$,令

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

再令 $k=k+1$,转到 Step 2.

对照精简形式的算法,第1改进算法用集合 $\boldsymbol{A}$ 代替了精简形式中的 $\boldsymbol{A}_k(1\leqslant k\leqslant n-1)$. 另外,第1改进算法中,仅仅使用了距离标号 $d_0(i)$ 和 $d(i)$,$(1\leqslant i\leqslant n)$,而精简形式一般要使用至少 n^2 个距离标号. 因而,第1改进算法显著提高了存储效率.

上述改进算法事实上借鉴了划分算法的思想,也就是参与第 k 次迭代计算的点放在集合 $\boldsymbol{A}$ 中,而把参加第 $k+1$ 次迭代计算的点放在集合 $\boldsymbol{B}$ 中. 但二者也有根本的区别,主要区别在于划分算法根本没有用到上一轮的距离标号 $d_0(i)$,仅仅使用了距离标号 $d(i)$. 为了显示这个区别,特给出以下划分算法作出比较.

Step 1 初始化. 令 $d(1)=0$,$d(i)=+\infty$,$(2\leqslant i\leqslant n)$,把所有点加入集合 $\boldsymbol{A}$.

Step 2 在第 k 次迭代中,按照进入顺序对集合 $\boldsymbol{A}$ 中的各点 i 进行计算为

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

如果 $d(j)<d^0(j)$,则当 $j\in\boldsymbol{A}$ 且 $j\notin\boldsymbol{B}$ 时,把点 j 加入集合 $\boldsymbol{B}$;而当 $j\notin A$ 时,把点 j 加入集合 $\boldsymbol{A}$.

Step 3 如果集合 $\boldsymbol{B}=\varnothing$,则算法结束,$d(i)$ 就是从点1到 i 的最短距离;当集合 $\boldsymbol{B}\neq\varnothing$,且 $k=n-1$ 时,可判定存在负循环,算法结束;当集合 $\boldsymbol{B}\neq\varnothing$,且 $k<n-1$ 时,执行 Step 4.

Step 4 清空集合 $\boldsymbol{A}$,并把集合 $\boldsymbol{B}$ 中的元素按进入顺序转到集合 $\boldsymbol{A}$,再清空集合 $\boldsymbol{B}$,同时对任意的点 $i(1\leqslant i\leqslant n)$,令

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

再令 $k=k+1$,转到 Step 2.

本文提出的第1个改进算法尽管减少了算法的存储空间,但算法仍可以进一步改进,也就是可减少 Step 4 的计算量,进一步改进算法(第2改进算法)为:

Step 1 初始化. 另 $d(1)=0$,$d(i)=+\infty$,$(2\leqslant i\leqslant n)$,把所有点加入集合 $\boldsymbol{A}$.

Step 2 具有两种情形:

情形1 在第 $t=2k-1(k\geqslant 1)$ 次迭代中,按照进入顺序对集合 $\boldsymbol{A}$ 中的各点 i 进行计算为

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

如果 $d(j)<d^0(j)$,当点 $j\notin\boldsymbol{B}$ 时,把点 j 加入集合 $\boldsymbol{B}$.

把点 i 从集合 $\boldsymbol{A}$ 中移除.

情形2 在第 $t=2k(k\geqslant 1)$ 次迭代中,按照进入顺序对集合 $\boldsymbol{B}$ 中的各点 i 进行计算为

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

如果 $d(j)<d^0(j)$,且当点 $j\notin\boldsymbol{A}$ 时,把点 j 加入集合 $\boldsymbol{A}$.

把点 i 从集合 $\boldsymbol{B}$ 中移除.

Step 3 也分两种情形:

情形1 在奇数次迭代中,如果集合 $\boldsymbol{B}=\varnothing$,则算法结束,$d(i)$ 就是从点1到 i 的最短距离;当集合 $\boldsymbol{B}\neq\varnothing$,且 $t=n-1$ 时,可判定存在负循环,算法结束;当集合 $\boldsymbol{B}\neq\varnothing$,且 $t<n-1$ 时,执行 Step 4.

情形2 在偶数次迭代中,如果集合 $\boldsymbol{A}=\varnothing$,则算法结束,$d(i)$ 就是从点1到 i 的最短距离;当集合 $\boldsymbol{A}\neq\varnothing$,且 $t=n-1$ 时,可判定存在负循环,算法结束;当集合 $\boldsymbol{A}\neq\varnothing$,且 $k<n-1$ 时,执行 Step 4 .

Step 4 对任意的点 $i(1\leqslant i\leqslant n)$,令

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

再令 $t=t+1$,转到 Step 2.

比较本文提出的两个改进算法,可看出第2个改进算法减少了把集合 $\boldsymbol{B}$ 的元素转移到集合 $\boldsymbol{A}$、再清空集合 $\boldsymbol{B}$ 这个环节所发生的计算量.

关于简化算法和两个改进算法均满足下列结论:

定理2 对于同一个最短路问题,如果3个算法(经典 Bellman-Ford 算法的精简形式、第1改进算法和第2改进算法)的初始距离标号是相同的,那么3个算法不仅具有相同的迭代次数,而且每一次迭代后距离标号改变的点形成的集合也是相同的.

证 明 在经典 Bellman-Ford 算法的常见形式中,在每一次迭代中每一条边都要参与计算. 如果在第 k 次迭代后,点 i 的距离标号没有改变,那么点 i 的关联边 (i,j) 不会影响点 j 在 $k+1$ 次迭代后的距离标号,因而只考虑在第 k 次迭代后距离标号变化的点 u 的关联边 (u,v),这样就形成了经典 Bellman-Ford 算法的精简形式. 而两个改进算法与精简形式的区别仅仅表现为实现方法上的

不同. 因而，这 3 个算法必然具有相同的迭代次数，且每一次迭代后距离标号改变的点形成的集合也是相同的.

尽管由经典 Bellman-Ford 算法的精简形式推广到第 1 改进算法应该是非常自然的，但与第 1 改进算法相类似的算法形式尚未在相关文献发现. 究其原因，可能是划分算法的提出造成了经典 Bellman-Ford 算法在计算效率方面的劣势，因而对它的研究鲜有关注.

3　算法实验及评估

由于只有经典 Bellman-Ford 算法的常见形式、第 1 改进算法和第 2 改进算法可解决有边数限制最短路问题，而其他已有的改进算法（如划分算法）均不适用，因而只对上述 3 个算法进行实验和评估. 3 个算法测试用的有向图均为完全图，均使用同样的例子进行测试，实验规模分别取点数 n 为 100、200、500、1 000、2 000、3 000、5 000、6 000 和 8 000 等 9 个规模水平. 有向图的权重服从均匀分布，可取 1 ~ 100 000 之间的任一整数. 图结构采用矩阵描述. 由于 3 个算法具有相同的迭代步数，因而算法试验以求得最短路径的迭代步数为参考标准. 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. 每一个规模水平分别测试 1 000 个随机生成的例子. 实验结果如表 1 所示.

表 1　经典 Bellman-Ford 算法 3 种实现形式的计算时间

ms

规模 n	常见形式	第 1 改进算法	第 2 改进算法
100	0.217 97	0.091 06	0.090 61
200	1.220 05	0.411 10	0.406 30
500	16.936 00	2.990 00	2.977 50
1 000	217.932 00	13.645 00	13.522 00
2 000	1 204.540 00	58.054 00	57.361 00
3 000	2 933.000 00	136.175 00	135.496 00
5 000	11 310.213 00	394.257 00	391.924 00
6 000	17 298.807 00	574.22 800	574.035 00
8 000	31 538.446 00	1 043.419 00	1 045.242 00

根据上述实验数据，可以得到：

1）第 2 改进算法在规模 < 6 000 以内具有绝对的优势，但仅仅略优于第 1 改进算法. 相对于第 1 改进算法，第 2 改进算法主要是减少了把集合 $\boldsymbol{B}$ 的元素转移到集合 $\boldsymbol{A}$、再清空集合 $\boldsymbol{B}$ 这个环节，因而可提高计算效率，这一点通过实验数据得以证实. 而规模 > 6 000 后，反而第 1 改进算法占有优势.

2）经典 Bellman-Ford 算法的常见形式确实需要更多的计算量，两个改进算法在规模≥6000 个点时，计算效率提高了近 30 倍.

4　结　论

1）对经典 Bellman-Ford 算法进行了改进，得到两个改进算法. 这两个改进算法可高效地求解有边数限制的最短路问题. 算法实验表明，两个改进算法计算效率显著，在规模≥6 000 个点时比经典 Bellman-Ford 算法快大约 30 倍.

2）尽管文献[1]宣称可改进算法的实现形式以提高计算效率，但已有改进算法均不适用于有边数限制最短路问题，这决定了两个改进算法是经典 Bellman-Ford 算法的全新改进，将使得 Bellman-Ford 算法在解决有边数限制的最短路问题上更具竞争力.

参考文献：

[1] BELLMAN R E. On a routing problem[J]. Quarterly of Applied Mathematics, 1958,16(1): 87 - 90.

[2] LEWANDDOWSKI S. Shortest paths and negative cycle detection in graphs with negative weights[R]. Technical Report, Stuttgart University, 2010.

[3] 韩伟一，王铮. 负权最短路问题的新算法[J]. 运筹学学报, 2007,11(1): 111 - 120.

[4] CHERKASSKY B V, GEORGIADIS L, GOLDBERG A V, *et al.* Shortest-path feasibility algorithm: an experimental evaluation[J]. ACM Journal of Experimental Algorithmics, 2009,14(7): 1 - 37.

[5] GLOVER F, KLINGMAN D, PHILLIPS N V. A new polynomially bounded shortest path algorithm[J]. Operations Research, 1985, 33(1): 65 - 72.

[6] GLOVER F, KLINGMAN D, PHILLIPS N V, *et al.* New polynomial shortest path algorithms and its computational attributes[J]. Management Science, 1986, 31(9): 1106 - 1128.

[7] HUNG M S, DIVOKY J J. A computational study of efficient shortest path algorithms[J]. Computer & Operations Research, 1988,15(6): 567 - 576.

[8] AHUJA R K, MAGNANTI K, ORLIN J B. Network Flows-theory, algorithm, and applications[M]. New Jersey: Prentice Hall, Englewood Cliffs, 1993.

[9] 孙强，杨宗源. 求受顶点数限制的最短路径问题的一个算法[J]. 计算机工程, 2002,28(9): 73 - 74.

[10] SAIGAL R. A constrained shortest path route problem [J]. Operations Research, 1968,16: 205 - 209.

[11] CHERKASSKY B V, GOLDBERG A V. Negative-cycle detection algorithm[C]//Proceedings of the Fourth Annual European Symposium on Algorithms. London, UK: Springer-Verlag, 1996: 349 - 363.

（编辑　张　红）