2009 年 4 月 第 4 期
Vol. 30 No. 4 2009

基于节点合并的最短路问题新算法

吕 欣, 李 勇, 邓宏钟, 谭跃进

(国防科技大学 信息系统与管理学院, 湖南 长沙 410073)

E-mail: lvxin_nudt@yahoo.com.cn

摘 要: 提出一个解决非负权网络最短路问题的节点合并算法. 该算法以将距离起始节点最近的邻居节点拉到身边的方法, 与距离最近节点不断合并, 重复这一动作, 最终求得起始节点到其他节点的最短路距离. 与 Dijkstra 算法相比, 节点合并算法不存在节点着色操作, 始终只考虑起始节点的邻居, 实现步骤更加简单, 整个过程可以采用向量化操作, 易于理解和编程实现. 数据试验表明, 节点合并算法求解效率明显高于 Dijkstra 算法.

关 键 词: 最短路; 节点合并; 节点合并算法; Dijkstra 算法

中图分类号: TP311 **文献标识码**: A **文 章 编 号**: 1000-1220(2009)04-0695-05

New Algorithm for Shortest Paths Based on Node Combination

LV Xin, LI Yong, DENG Hong-zhong, TAN Yue-jin

(*School of Information Systems and Management, National University of Defense Technology, Changsha* 410073, *China*)

Abstract: This paper proposes a Node Combination algorithm for the Shortest-Path of real-weighted networks. By the method of dragging the start node's nearest neighbor to itself, it combines the nearest node repeatedly, and finally gains the lengths of the shortest paths between the start node and all other nodes. Compared with the Dijkstra algorithm, it is a new algorithm without node labeling operations, it takes no but the start node's neighbors into account all the time. While the whole process could be manipulated with vectors, it is more comprehensible and convenient for programming. With the experimental evaluation on a variety of networks, it shows that it gains more computing efficiency than Dijkstra.

Key words: shortest path; node combination; node combination algorithm; Dijkstra algorithm

1 引 言

通篇排号最短路(Shortest Path, SP)问题是图论中的经典问题, 也是最基本的网络优化问题之一. 使用节点着色的方法, E. W. Dijkstra[1]在 1959 年给出了求非负权网络中节点 v_i 到网络中所有其他节点最短路距离(Single-Source Shortest Path, SSSP)的一个算法, 其时间复杂度为 $O(N^2)$.

为了进一步提高 Dijkstra 算法的效率, 从提高临时标记节点搜索效率和减少临时标记节点搜索数量两方面出发, 学者们提出了大量改进算法, 这些算法都是通过设计特定的数据结构和采用优化的搜索策略来改进效率的. 其中典型的有使用桶结构的 DIKB、DIKBD、DIKBM、DIKBA 算法, 使用堆结构的 DIKH、DIKR 算法, 使用队列结构的 DIKQ、DIKF 算法等[2]. Cherkassky B. V., Goldberg A. V. 和 Randzik T.[2]对 Dijkstra 算法的 6 种改进算法进行的比较表明, DIKBD 和 DIKBA 效果最好.

当网络中存在负权值时, Dijkstra 算法不能保证一定产生最短路, Bellman-Ford-Moore 算法能解决不含负回路网络的最短路问题[3-5]. 该算法的主要思想是逐次逼近, 每次逼近都是求 D 中从顶点 v_i 到其余各顶点的带限制的最短路. 该算法最多需要 $3N^3/2$ 次运算, 其时间复杂度为 $O(mN)$[6], m 表示边的数量.

对于不含回路的网络, A. V. Goldberg[7]给出的拓扑排序算法能在 $O(mN)$ 的时间复杂度下解决一个节点到所有其他节点的最短路问题.

基于图增长理论, B. Ju. Levit[8]和 U. Pape[9]分别单独提出了解决最短路问题的算法, 一般称为 Pape-Levit 算法, 该算法在最坏情况下复杂度为 $O(N \cdot 2^N)$. S. Pallottino[10]提出了著名的双队列 TQQ 算法, 虽然在最坏的情况下, TQQ 算法的复杂度为 $O(mN)$, 但是 Cherkassky[2]和 F. Benjamin Zhan[11]分别对 17 和 15 种算法求解最短路效率进行了比较, 结果表明, TQQ 算法是速度最快的算法之一.

显然, 将 Dijkstra 循环 N 次即可求解所有节点对之间的最短路(All-Pair Shortest Path, APSP), 其时间复杂度为 $O(N^3)$. 求所有节点对之间的最短路的算法还有 Floyd 算法[12], Dantzig 算法[13], 他们的时间复杂度均为 $O(N^3)$. 鉴于本文仅讨论单源最短路问题, 在此不一一介绍.

收稿日期: 2007-12-17 收修改稿日期: 2008-03-03 基金项目: 国家自然科学基金项目(70501032; 70771111)资助. 作者简介: 吕 欣, 男, 1984 年生, 博士研究生, 研究方向为复杂网络理论及应用; 李 勇, 男, 1977 年生, 博士研究生, 研究方向为复杂保障网络仿真与评价; 邓宏钟, 男, 1974 年生, 副教授, 研究方向为复杂系统理论, 分布式人工智能和遗传算法; 谭跃进, 男, 1958 年生, 教授, 博士生导师, 研究方向为系统理论与综合集成.

随着计算机技术的进步和最短路算法研究的深入, 最新的最短路改进算法主要从网络分层[14-16]、网络分割[17]和将各种优化方法结合使用[18]来提高效率. 各种启发式搜索算法也越来越频繁地应用于最短路问题求解中来. 如神经网络[19, 20]、遗传算法[21, 22]、蚁群算法[23-25]、A^* 算法[26]等. 关于应用启发式算法解决最短路问题的综述性文献可以参看[27], 该文对最短路算法研究的相关文献作了详尽的总结.

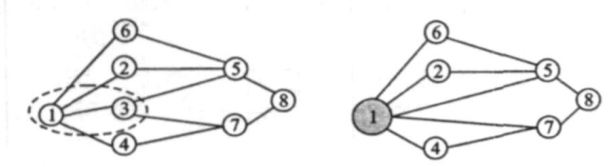
这些最短路算法考虑的是怎样从(from)起始节点走到(to)目标节点的路径选择问题, 尤其是Dijkstra 算法还涉及节点和边的着色操作, 这对算法的理解和程序设计都提高了难度, 将增大算法的时间和内存开销, 影响执行效率.

本文提出了一种基于节点合并的最短路问题新算法, 它从反方向考虑, 即怎样把目标节点拉(drag)到起始节点身边, 将拉过来的节点与起始节点合并(Combine)为新的起始节点, 在这过程中按一个非常简单的规则修改边权值, 始终考察起始节点的邻居节点, 最终所有与起始节点连通的节点将合并成一个节点, 就可以求得起始节点到其他各节点的最短路距离.

节点合并(Node Combination, NC)算法具有形象生动, 易于理解, 便于编程实现和运算效率高等特点.

2 节点合并

定义1 设v_i, v_j 是图$G=(V, E, C)$中的两个节点, 我们用$G(v_i \cdot v_j)$表示将节点v_i 与节点v_j 合并后所得到的图. 所谓将节点v_i 与节点v_j 合并是指将v_i 与v_j 合并成一个新的节点代替v_i 与v_j, 原来分别与v_i 和v_j 相关联的边现在都与新节点关联, 如图2 所示.



(a) 合并前的图 (b) 合并后的图

图1 将节点1 与节点3 进行合并

Fig. 1 Node combination

图1 中节点1 与节点3 合并后形成的新节点保持了与节点5, 7 的连接, 同样地, 节点合并过程也可能产生多重边的情况, 如果图1(a)中节点1 与节点7 有边相连, 则节点1 与节点3 合并后在新节点和节点7 之间会产生二重边. 显然, 无多重边网络两节点合并过程产生的多重边重数最多为二重.

节点合并算法对多重边的处理是选择权值较小的边代替多重边, 这一点在第3 节有详细介绍.

3 节点合并算法

3.1 节点合并算法基本思想

考虑一个无负权值的无向网络, 节点之间的边权值代表相连节点的距离, 要找到某个给定的起始节点到所有其他节点之间的距离, 可以这样考虑: 站在起始节点, 将其他节点一个个拉到起始节点身边. 拉的过程, 就是起始节点和被拉的目标节点的合并过程.

可以作这样一个比方: 假设网络中的节点都是由绳子牵着的, 起始节点放在一个池子里面, 我们站在起始节点, 不断地将最短距离的邻居节点拉过来连同绳子一起丢到池子里面. 这样, 池子外面的节点越来越少, 最后, 所有绳子牵着的节点都被拉到池子里面. 节点合并算法的最大特点就是操作简单, 始终只考虑起始节点的邻居, 重复选择最近的邻居与其合并, 最终得到起始节点与其他节点的最短路距离. 如果网络采用邻接矩阵形式存储, 则每次选择目标节点都只需考察起始节点所在行.

注意, 算法对节点的合并操作体现在对起始节点到其他节点的边权值修改上, 不必增加额外的计算复杂度. 于是节点合并算法没有繁琐的节点标号操作, 极大的简化了程序设计.

3.2 节点合并算法流程

按照上述思路, 对一个具有N 个节点的非负无向网络$D=(V, E, C)$, 网络邻接矩阵为$C_{N\times N}$, 节点合并算法步骤如下:

Step 0 初始化$C_{N\times N}$ 的所有零元素用∞代替, 即两点之间(包括自己)没有路连接则距离为无穷. 对每个节点v_j 用$d(v_j)$表示起始节点v_i 到v_j 的最短路长度, 开始, 令$d(v_i)=0$, $d(v_j)=\infty$, $j\neq i$.

Step 1 取v_{min}使得$C(v_i, v_{min})=\min\{C(v_i, v_j)\}$, 令$d(v_{min})=C(v_i, v_{min})$. 如果$\min\{C(v_i, v_j)\}=\infty$, 停止.

Step 2 对$\forall v_j\in V$, $C(v_j, v_{min})=\infty$, 修改$C(v_i, v_j)=\min\{C(v_i, v_j), d(v_{min})+C(v_{min}, v_j)\}$, 转Step 1.

可见节点合并算法中仅存在求极小值和简单的赋值操作, 不存在繁琐的节点着色引起的判断操作. 通过重复查找起始节点最近邻居和修改与其他节点距离这两种操作, 最终得到起始节点到其他各节点的最短路距离. 算法中涉及到要操作的变量少, 比较Dijkstra 算法, 节点合并算法极大地简化了程序设计.

下面证明算法的正确性.

定理1 节点合并算法按照最短路距离递增的顺序求解最短路问题.

证明: 由于$d(v_{min})=\min\{C(v_i, v_j)\ |v_j\in V\}$, 只需证明节点合并后对所有$j$, $\min\{C(v_i, v_j)\}\geq d(v_{min})$即可. 即证$\min\{\min\{C(v_i, v_j), d(v_{min})+C(v_{min}, v_j)\}\}\geq d(v_{min})$.

由$d(v_{min})=\min\{C(v_i, v_j)\}$, 有$C(v_i, v_j)\geq d(v_{min})$, 又$d(v_{min})+C(v_{min}, v_j)\geq d(v_{min})$, 显然对任意$j$, 有$\min\{C(v_i, v_j), d(v_{min})+C(v_{min}, v_j)\}\geq d(v_{min})$, 马上得到$\min\{\min\{C(v_i, v_j), d(v_{min})+C(v_{min}, v_j)\}\}\geq d(v_{min})$. 得证.

定理2 通过节点合并算法能够求出无向非负网络起始节点v_i 到其他节点的最短路距离.

证明: 设已经求得最短距离节点的集合为S, 则$S=\{v_i\}$时, 有$C(v_i, v_j)=\infty$, 显然$d(v_k)=\min\{C\ |(v_i, v_j)\ |v_j\in V\}$是$v_i$到$v_k$ 的最短路距离, 而且该距离是所有最短路距离中除了起始节点到自身的距离外($d(v_i)=0$)最小的.

当$|S|>1$ 的某时刻, Step 1 中求得$\min\{C(v_i, v_j)=C(v_i,$

$v_k)\}$, 只要证明$C(v_i, v_k) = d(v_k)$即可.

用反证法, 假设$C(v_i, v_k)$不是节点v_i到v_k的最短路距离, 则存在下述两种情况:

(1) v_i到v_k的最短路径为边(v_i, v_k), 也即假设此时的$C(v_i, v_k)$是通过节点合并操作修改边权值后得到的. 然而, 通过 Step 2 中

$$C(v_i, v_j) = \min\{C(v_i, v_j), d(v_{min}) + C(v_{min}, v_j)\}$$

可知$C(v_i, v_j)$在节点合并过程中直至被合并之前, 是不断递减的, 故存在某节点m, 使得$d(v_m) + C(v_m, v_k) < C(v_i, v_k)$, 与$v_i$到$v_k$的最短路径为边$(v_i, v_k)$矛盾.

(2) v_i到v_k的最短路径的倒数第二个节点为v_{k2}, 不妨假设通过节点合并算法得到的最短路径倒数第二个节点为v_{k1}, 则$d(v_{k1}) + C(v_{k1}, v_j) > d(v_{k2}) + C(v_{k2}, v_j)$.

容易验证v_{k2}未被合并, 否则在v_{k2}被合并时必然有

$$C(v_i, v_k) = \min\{C(v_i, v_k), d(v_{k2}) + C(v_{k2}, v_k)\} = d(v_{k2}) + C(v_{k2}, v_k)$$

而此时与

$C(v_i, v_k) = d(v_{k1}) + C(v_{k1}, v_k) > d(v_{k2}) + C(v_{k2}, v_j)$矛盾, 故$v_{k2}$未被合并.

那么有$d(v_{k2}) > C(v_i, v_{k2}) > d(v_{k1})$(否则不会选中$v_{k1}$节点合并), 在$v_i$与$v_{k1}$节点合并时:

$$d(v_{k1}) < d(v_{k2}) < C(v_i, v_{k2}) < d(v_{k1}) + C(v_{k1}, v_k) = C(v_i, v_k)$$

而$\min\{C(v_i, v_j)\} = C(v_i, v_k) < C(v_i, v_{k2})$, 故矛盾. 证毕.

3.3　一个完整的例子

图 2 展示了应用节点合并算法求解节点 1 到其他各节点最短路距离的过程, 其中$d(i)$表示求得的节点 1 到节点i的最短路距离. 下面详解该过程:

合并 1, 2. 起始节点 1 首先找出与自己最近的邻居为节点 2, 马上得到与节点 2 的最短路距离为$d(2) = 5$. 执行合并操作, 修改

$C(1, 5) = \min\{C(1, 5), d(2) + C(2, 5)\} = \min\{\infty, 5 + 10\} = 15$. 合并 1, 3. 新的起始节点 1 找出与自己最近的邻居节点 3, 得到$d(3) = 8$. 执行合并操作, 修改$C(1, 5) = \min\{C(1, 5), d(3) + C(3, 5)\} = \min\{15, 8 + 10\} = 15$. $C(1, 7) = 18$.

合并 1, 6. $d(6) = 8$, 合并 1, 6, 修改$C(1, 5) = 13$.

合并 1, 4. $d(4) = 10$, 合并 1, 4, 修改$C(1, 7) = 17$.

合并 1, 5. $d(5) = 13$, 合并 1, 5, 修改$C(1, 8) = 19$.

合并 1, 7. $d(7) = 17$, 合并 1, 7, 修改$C(1, 8) = 19$.

合并 1, 8. $d(8) = 19$.

所有节点合并为一个节点, 算法终止.

从上述过程可见, 对于一个 8 节点的全连通网络, 节点合并算法通过 7 次节点合并操作, 求出节点 1 到其他各节点(2 到 8)的最短路距离分别为: 5, 8, 10, 13, 8, 17, 19. 容易验证, 该结果与用 Dijkstra 算法得到的从节点 1 到其他各节点的最短路距离是一样的.

4　算法效率比较

4.1　算法复杂度

对于全连通网络, 节点合并算法的计算复杂性由以下部分组成:

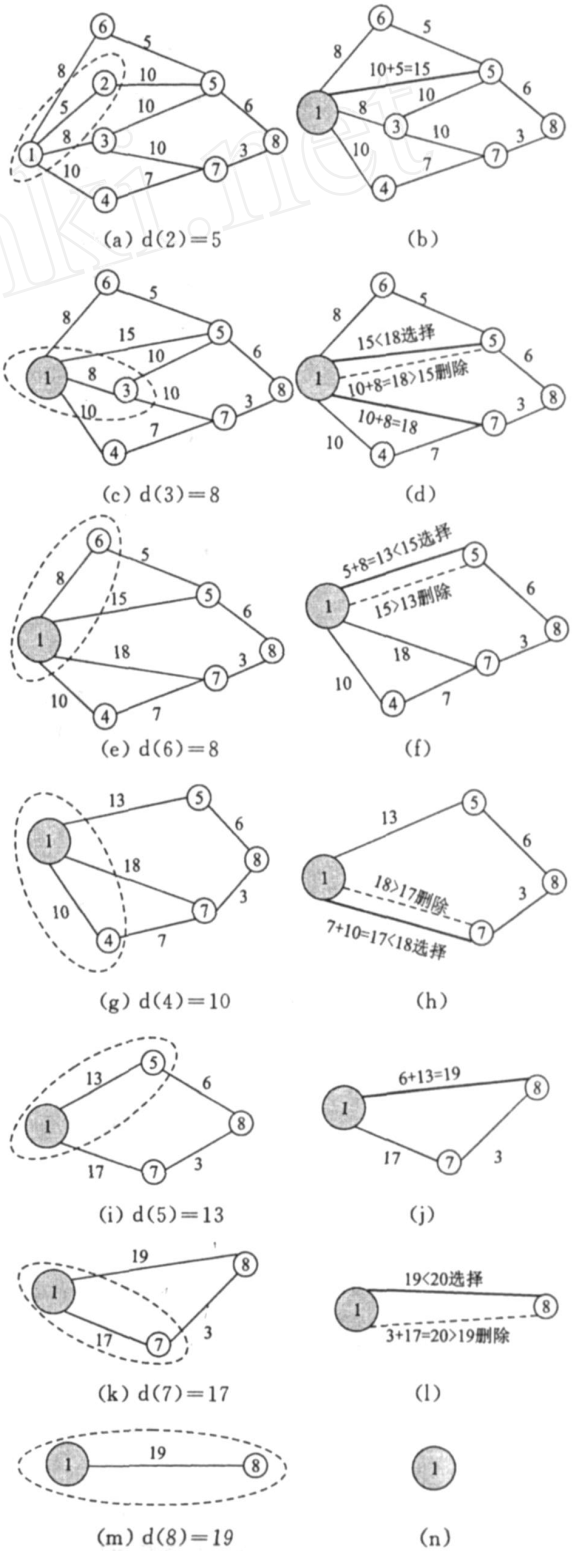


图 2　节点合并算法演示

Fig. 2　Demo of node combination algorithm

(1) 查找最近邻居v_{min}的比较运算: 即 Step 1 中, 对v_i的

所有邻居v_j, 求$C(v_i, v_{min}) = min\{C(v_i, v_j)\}$, 共需比较$n_i - 1$次($n_i$表示节点$v_i$的度);

(2) 修改与起始节点相连边权值的加法运算以及其选择较小边的比较运算: 即 Step 2 中, 对v_{min}的所有邻居v_j, $C(v_i, v_j) = min\{C(v_i, v_j), d(v_{min}) + C(v_{min}, v_j)\}$, 加法和比较运算分别为$n_{min} - 1$次;

(3) 节点合并的实现: 对v_{min}的所有邻居v_j, $C(v_j, v_{min}) = \infty$, 共需$n_{min} - 1$次赋值操作.

算法循环最多$N - 1$次, 由于节点合并过程将导致网络中节点邻居数量的不断减少, 故下面仅考虑算法复杂度的一个上界:

$$(N - 1) \times n_i + 2 \times \sum_{j=1, j \neq i}^{N} (n_j - 1) \doteq (N - 1)(n_i + 2\bar{n} - 2) \doteq 2\bar{n}(N - 1)$$

其中$\bar{n}$表示网络的平均度, 故节点合并算法复杂度为$O(\bar{n}N)$.

注意, 对于大部分的复杂网络尤其是小世界网络和无标度网络, $\bar{n} \ll N$, 故节点合并算法能够极大地提高算法效率.

比较两种算法的过程就可以发现, 节点合并算法通过节点合并代替了传统的节点着色操作, 程序设计也不必开辟额外的内存空间用以记录着色节点. 算法运算过程不必进行着色节点的判断操作, 只存在查找最近邻居的求最小值操作和节点合并过程中起始节点边权值的修改操作, 而这两种操作几乎都基于网络的邻接矩阵和最短路距离向量, 整个算法只需要增加两个记录距离和位置的整型变量(也可以是实型变量), 可以很好的提高运算效率, 节省内存空间.

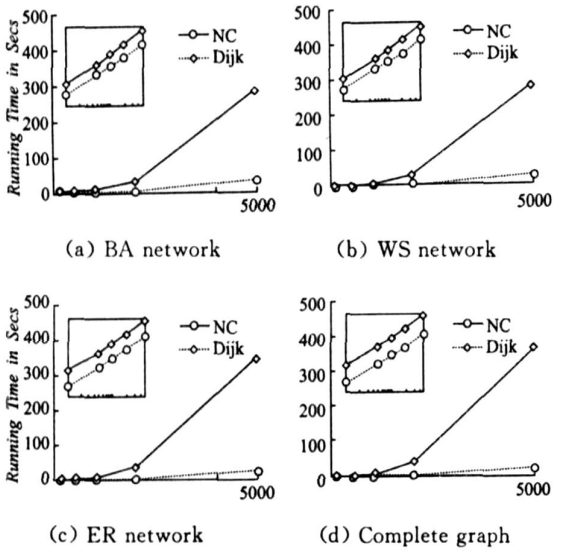


图 3 节点合并算法求解最短路效率比较

Fig. 3 Comparation of running time

简单的说, 节点合并算法省去了Dijkstra 算法的顶点着色操作, 不必开辟额外的内存空间和进行顶点着色的判断操作. 而且整个过程始终从起始节点出发, 即程序完成一轮循环后, 仍然只需重新考察起始节点即可, 这就大大简化了程序.

不难发现节点合并算法的所有步骤几乎都可以用矩阵向量来"批量"操作, 使用Matlab 来编程实现, 可以得到更快的运算速度.

4.2 效率比较

为了比较节点合并算法与Dijkstra 算法的效率, 使用Matlab 7.1 编程, 在Pentium (R) 4 2.4GHz, 512M 内存的计算机上分别计算复杂网络中节点v_1到其他节点的最短路距离, 网络的存储结构采用赋权邻接矩阵的形式.

比较算法所花费的时间(单位, 秒), 采用Barabasi, Albert 和Jeong 提出的BA 无标度网络[28], Watts 和Strogatz 提出的WS 小世界网络[29], Erdos 和 Renyi 引入的 ER 随机网络[30, 31], 以及完全图, 分别设置网络中节点数为 100, 500, 1000, 5000, 比较Dijkstra 算法与节点合并算法的运算效率, 我们发现, 节点合并算法比Dijkstra 算法运算效率高, 可以节省5 倍以上的计算时间, 而且, 随着网络规模的扩大, 算法效率的差异表现得越明显, 当网络为5000 个节点的完全图时, 节点合并算法比Dijkstra 算法快了20 倍以上. 在大规模的复杂网络研究中使用节点合并算法, 将是不错的选择. 图3 展示了两种算法的计算时间比较结果.

5 总 结

节点合并算法基于将目标节点不断拉到身边的新思路, 通过节点合并代替传统的顶点和边着色操作, 合并后形成的新的起始节点代替Dijkstra 算法的已着色顶点集合, 省去了顶点和边着色、着色顶点判断等繁琐的操作, 整个算法仅仅使用两种基本操作: 寻找与起始节点最近的邻居和修改起始节点与其他未求得最短路节点的边权值, 重复对起始节点搜索距离最近的邻居节点, 就可以求得所有到其他节点的最短路距离. 反映到邻接矩阵上, 算法每次都只需要从起始节点所在行出发进行搜索即可.

上述优点导致节点合并算法同时具有较高的计算效率和简洁的程序代码, 其时间复杂度为$O(\bar{n}N)$, 而对于大部分的复杂网络尤其是小世界网络和无标度网络, $\bar{n} \ll N$, 在大规模的复杂网络研究中使用节点合并算法, 将是不错的选择.

该算法已成功应用于十一五装备预研项目"复杂**网络可靠性分析与评价"软件系统, 与*Dijkstra* 算法类似地, 节点合并算法可以很方便地设计于求解最短路径, 我们同样得到了该算法的权矩阵解法, 后者的操作非常简单, 每次只需修改权矩阵的起始节点所在行的元素值, 限于篇幅, 在此不作进一步介绍.

本文仅讨论了非负权无向网络, 容易证明, 节点合并算法对于非负权的有向网络也是适用的.

References:

[1] Dijkstra E W. A note on two problems in connection with graphs[J]. Numeriche Mathematik, 1959, 1(1): 269-271.

[2] Cherkassky B V, Goldberg A V, Radzik T. Shortest paths algorithms: theory and experimental evaluation[C]. In: Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, 1996, 516-525.

[3] Bellman R E. On a routing problem [J]. Quarterly of Applied

Mathematics, 1958, 16(1): 87-90.
[4] Ford L R. Network flow theory[R]. Santa Monica, California: The Rand Corp., 1956, 923.
[5] Moore E F. The shortest path through a maze[C]. In: Proceedings of the International Symposium on the Theory of Switching, 1959, 285-292.
[6] Evans J R, Minieka E. Optimization algorithms for networks and graphs[M]. 2nd ed., Marcel Dekker, New York, 1992.
[7] Goldberg A V, Radzik T. A heuristic improvement of the bellman-ford algorithm[J]. Applied Mathematics Letters, 1993, 6(3): 3-6.
[8] Levit B J, Livshits B N. Neleneinye setevye transportnye zadachi[Z]. Transport, Moscow, Russian, 1972.
[9] Pape U. Implementation and efficiency of moore algorithms for the shortest rout problem[J]. Math. Programming, 1974, 7(1): 212-222.
[10] Pallottino S. Shortest-path methods: complexity, interrelations and new propositions[J]. Networks, 1984, 14(14): 257-267.
[11] Zhan F B. Three fastest shortest path algorithms on real road networks: data structures and procedures[J]. Journal of Geographic Information and Decision Analysis, 2001, 1(1): 69-82.
[12] Floyd R W. Algorithm 97: shortest path[J]. Communications ACM, 1962, 6(1): 345-350.
[13] Dantzig G B. All shortest routes in a graph[C]. Paper read at International Symposium, Rome, Italy, 1967, 91-92.
[14] Peter S, Schultes D. Highway hierarchies hasten exact shortest path queries[C]. In: Proceedings 17th European Symposium on Algorithms (ESA), Springer, 2005.
[15] Pettie S, Ramachangran V. A shortest path algorithm for real-weighted undirected graphs[J]. Society for Industrial and Applied Mathematics, 2005, 34(6): 1398-1431.
[16] Thorup M. Undirected single-source shortest paths with positive integer weights in linear time[J]. Journal of the ACM, 1999, 46(3): 362-394.
[17] Rolf H M, Schilling H, Schütz B. Partitioning graphs to speed up Dijkstra's algorithm[C]. In: Proceedings of the 4th Workshop on Experimental and Efficient Algorithms (WEA), 2005, 189-202.
[18] Martin H, Frank S, Thomas W. Combining speed-up techniques for shortest-path computations[J]. ACM Journal of Experimental Algorithmics, 2005, 10: 269-284.
[19] Xia Y, Wang J. A discrete-time recurrent neural network for shortest-path routing[J]. IEEE Transactions on Automatic Control, Nov. 2000, 45(11): 2129-2134.
[20] Nasir S H, Mohamed K H, Seng T G. Implementation of recurrent neural network algorithm for shortest path calculation in network routing[C]. In: International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN), 2002.
[21] Sachith A, Baladasan G, Saluka K. A genetic algorithm approach to solve the shortest path problem for road maps[C]. In: Proceedings of the International Conference on Information and Automation, 2005.
[22] Davies C, Lingras P. Genetic algorithms for rerouting shortest paths in dynamic and stochastic networks[J]. European Journal of Operational Research, 2003, 144(1): 27-38.
[23] Min L Y, Yang J Y. A shortest-path routing based on ant algorithm[J]. Journal of Communication and Computer, 2005, 2(9): 67-70.
[24] Pejas J, Piegat A, Pluciski M. Application of the ant colony algorithm for the path planning[M]. Springer-Verlag, London, 2005, 345-352.
[25] Hui F, Zhen H, Li J J, et al. Solving a shortest path problem by ant algorithm[C]. In: Proceedings of 2004 International Conference on Machine Learning and Cybernetics, 2004, 5(5): 3174-3177.
[26] Goldberg A V, Harrelson C. Computing the shortest path: A* search meets graph theory[C]. In: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2005, 156-165.
[27] Fu L, Sun D, Rilett L R. Heuristic shortest path algorithms for transportation applications: state of the art[J]. Computers and Operations Research, 2006, 33(11): 3324-3343.
[28] Barabási A L, Albert R, Jeong H. Mean-field theory for scale-free random networks[J]. Physica A: Statistical Mechanics and its Applications, 1999, 272(1-2): 173-187.
[29] Watts D J, Strogatz S H. Collective dynamics of 'small-world' networks[J]. Nature, 1998, 393(6684): 440-442.
[30] Erdos P, Rényi A. On random graphs[J]. Publicationes Mathematicae-Debrecen, 1959, 6: 290-297.
[31] Erdos P, Rényi A. On the evolution of random graphs[J]. Publ. Math. Inst. Hung. Acad. Sci, 1960, 5(1): 17-61.