文章编号:1671-4598(2018)07-0143-04 DOI:10.16526/j.cnki.11—4762/tp.2018.07.031 中图分类号:TP393.03 文献标识码:A

适用于无向网络的动态 Dijkstra 算法优化

马慧慧[1]，卢 昱[1]，王增光[2]
(1. 军械工程学院 信息工程系，石家庄 050003；
2. 军械工程学院 装备指挥与管理系，石家庄 050003)

摘要：网络拓扑发生变化时，利用静态 Dijkstra 算法重新计算最短路径树（SPT）会造成冗余计算；动态 Dijkstra 算法解决了这个问题，但目前动态算法一般是基于有向网络模型进行的研究；在已有的动态 Dijkstra 算法基础上，提出适用于无向网络的动态 Dijkstra 算法；算法主要解决了在无向网络中如何确定待更新节点的问题，对网络中的一条边权值增大、减小的处理方法进行了详细描述，并对已有的算法的筛选机制进行了优化；为了验证算法的正确性，用仿真实验实现了该算法并与静态算法进行性能比较；实验结果表明，新算法更能提高节点更新的时间效率。

关键词：路由算法；Dijkstra 算法；无向网络；最短路径树；动态更新

Dynamic Dijkstra Algorithm Optimization for Undirected Networks

Ma Huihui[1], Lu Yu[1], Wang Zengguang[2]
(1. Information Engineering Department, Ordnance Engineering College, Shijiazhuang 050003, China;
2. Equipment Command and Management Department, Ordnance Engineering College, Shijiazhuang 050003, China)

Abstract: Using the static Dijkstra algorithm to recalculate the shortest path tree (SPT) will cause redundant computation when the network topology changes. In order to reduce the computational complexity, a dynamic Dijkstra algorithm for undirected networks is proposed based on the existing dynamic Dijkstra algorithm. The problem of how to determine the nodes to be updated in the undirected network is solved. The algorithm describes the processing method of the increase and decrease of the weight. And the existing algorithms are optimized. In order to verify the correctness of the algorithm, which is implemented by code and compared with its static algorithm. Experimental results show that the new algorithm has more performance.

Keywords: routing algorithm; Dijkstra algorithm; undirected network; shortest path tree; dynamic update

0 引言

路由算法是网络研究中的关键问题[1-2]。为了减小数据在传输过程中的开销，通常寻找节点之间的最短路径。E. Dijkstra提出经典的最短路径算法[3]，以一个节点为根，形成最短路径树（SPT，Shortest Path Tree），适用于寻找单节点到网络中任意节点的最短路径。在实际的网络中，网络的拓扑结构会因为各种原因发生改变，例如链路失效、节点宕机、新节点加入等。在网络模型中，拓扑结构的变化一般总结为以下 4 种情况：边的权值增大或减小、节点增加或删除[4-5]。当网络拓扑小范围发生变化时，通过 Dijkstra 算法从根节点开始重构 SPT，能够解决这个问题。但是这种静态的算法重构 SPT 会造成大量不必要的开销，因为小范围拓扑发生变化，有许多节点在最短路径树中的位置是不发生改变的。

收稿日期:2017-09-07； **修回日期**:2017-11-29。
基金项目:国家社会科学基金军事学资助项目(15GJ003—184)；国家自然科学基金资助项目(61271152)。
作者简介:马慧慧(1992-)，男，山东平度人，硕士研究生，主要从事信息网络安全与控制技术方向的研究。
卢 昱(1960-)，男，河南洛阳人，博士学位，教授，博士生导师，主要从事信息网络安全控制方向的研究。

为了优化算法性能，提出了动态 Dijkstra 算法[6]。动态 Dijkstra 算法的核心思想是尽可能保持现有 SPT 的结构，缩小需要重新计算的节点的范围，降低更新 SPT 过程的复杂程度。文献［7］最早提出动态更新最短路径树算法(DSPT)，但是算法所划定的更新范围仍有冗余。NSPT 算法[8]，Ball—and—String 算法[9-10]是比较高效的动态算法，解决了有边的权重增大减小的状况。文献［11］提出了多条边发生变化时的解决方法。这些算法基于的网络模型大多是有向网络模型，但在实际计算机网络中，一般是数据正反向都能传递的无向网络。在无向网络模型中，两个节点只有连接关系而无先后顺序，因此在算法中无法利用节点先后连接参数。文献［12］提出了较为成熟动态算法的思想，讨论了在网络中一条边权值变化的 SPT 更新算法。但是该算法适用于有向网络模型，同时对于所处理的节点范围没有进行必要的筛选，还是有部分冗余计算。本文在此基础上提出了适用于无向网络的动态更新 SPT 算法，优化了受影响节点的筛选机制，进一步缩小了节点更新范围。

1 Dijkstra 算法基本原理

Dijkstra 算法是求单源最短路径的经典算法。它采用标记法按照路径长度递增的顺序寻找最短路径，首先从源点开始，找出长度最短的一条路径及节点，然后从新节点出发，通过迭代得到从源点到其余各节点的最短路径[13]。为

了跟后来所研究的动态 Dijkstra 算法形成对应，把经典的 Dijkstra 算法称作静态 Dijkstra 算法。

Dijkstra 算法的基本过程[14]如下：在整个网络中设置两个集合，集合 T 是经过计算加入到最短路径树中的节点，集合 S 是还未加入到最短路径树中的节点。假设每个节点 v_j 都有一对参数 (d_j, p_j)，d_j 是从源节点 v_s 到节点 v_j 的最短路径长度，p_j 代表节点 v_j 在最短路径中的父亲节点。

步骤一：初始化，$T = \phi$，S 包含了网络中的所有节点。设置所有节点 $d_j = \infty$，$p_j = [.]$

步骤二：将源节点 v_s 加入集合 T，并从 S 中删除，设置 $d_s = 0$，$p_s \in \phi$。

步骤三：检验集合 T 中的所有节点 v_i 到的 S 中节点 v_j 的距离 l_{ij}，其中 v_i 到 v_j 必须是直接连接的，中间无其他节点。设置 $d_j = \min[d_j, d_i + l_{ij}]$。

步骤四：选取 d_j 最小的节点 v_j，将其从集合 S 中删除，并加入到集合 T 中去，该节点成为最短路径树中的一个新节点。

步骤五：找到新节点 v_j 的父亲节点 p_j，并记录该节点的两个参数 (d_j, p_j)。

步骤六：重复步骤三到步骤五的过程，直到网络中的所有节点全部加入到集合 T 中，此时集合 $S = \phi$，完成整个过程。

2 算法设计

2.1 参量设置

假设存在加权无向网络 $G = (V, E)$，V 表示节点的集合，E 表示边的集合，$|V| = n$，$|E| = m$。G 中的节点用 v 来表示，用下标加以区分。e 代表两个有直接连接的节点之间的边，权值用 $w(e)$ 表示。从根节点的整个网络的最短路径树表示为 SPT（Shortest Path Tree），$d(v)$ 表示从根节点到节点 v 的最短路径长度，$p(v)$ 表示节点 v 的在 SPT 中的父亲节点，$T(v)$ 表示在 SPT 中以 v 为根节点的子最短路径树的节点的集合。对网络中的所有节点设置一个状态参量 $Sta(v)$，当 $Sta(v) = 1$ 表示节点 v 在 SPT′（更新后的 SPT）中的位置已经固定，不再需要处理；$Sta(v) = 0$ 表示节点的位置仍需要更新。设置一个集合 Q，存储等待处理的节点及节点 $d(v)$ 的变化量。该算法讨论的是无向网路，必要时用 $[v_i, v_j]$ 来表示两个节点及所形成的边。v_i、v_j 的先后顺序与表示的结果无关，也就是说 $[v_i, v_j]$ 和 $[v_j, v_i]$ 实际上表示的是同一条边。相比于有向网络，无向网络需要解决的问题更复杂一些。

2.2 算法描述

假设在网络 G 中，关于根节点 v_s 的 SPT 已经构造完成。当检测到有一条边 e_0 的权值发生变化时，需要首先确定 SPT 受影响的范围[15]。算法是分成权值增大和权值减小两种情况进行处理的。

情况一：权值增大时。如果权值增大的边不是 SPT 中的边（例如图 1 中的 $[v_5, v_6]$），由于各节点的最短路径长度（d 值）已经是当前最小的，非 SPT 中的边权值增大不会改变各节点到根节点的 d 值，因此这种情况下不会影响 SPT 结构。如果 SPT 中的一条边的权值增大，假设图 1 中，边 $[v_2, v_6]$ 的权值由 3 变为 10，增加了 7，这将导致以 v_6 为根节点的子树（$T(v_6)$）中的节点到 v_s 的距离都会增加，此时在网络中的位置不一定是路径最短的位置，因此需要对这些节点的位置进行调整。边 $[v_2, v_6]$ 权值增加后影响到的节点范围在图中用虚线圈出。将 $T(v_6)$ 中的节点的 d 值全都增加 7，并把它们的 $Sta(v)$ 参数置为 0，表示节点位置等待更新；原 SPT 中的其他节点 $Sta(v)$ 置为 1，表示节点位置已经固定。

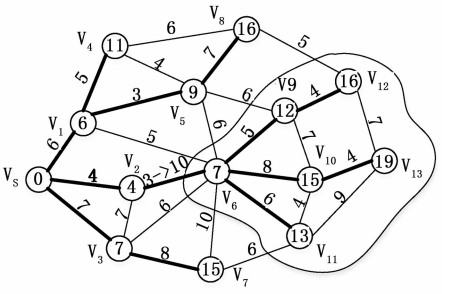

图 1 使用 Dijkstra 算法计算的最短路径树

所有与待更新节点相连的节点都有可能成为新的父亲节点，在文献［12］的算法中，把所有的连接边都纳入到了判断范围之内，但实际仍有部分是无需考虑的。与待更新节点相连的边有三种类型，以 v_9 为例：（a）与 $T(v_6)$ 之外的节点相连的边，例如 $[v_5, v_9]$；（b）与 $T(v_6)$ 内节点相连但不属于 $T(v_6)$ 的边，例如 $[v_{10}, v_9]$；（c）与 $T(v_6)$ 内节点相连但且属于 $T(v_6)$ 的边，例如 $[v_6, v_9][v_9, v_{12}]$。其中 c 类边不纳入考虑范围是显然的；b 类边，两个节点的 d 值加上了相同的增量，不会改变待更新节点的父亲节点；只有 a 类边存在改变待更新节点最短路径的可能。因此在下一步操作之前可以首先把 b、c 类边排除，达到缩减计算范围的目的。该部分判定过程对应于算法 1 中的第 8 行伪代码。在 a 类边中，计算边的对应节点的最短路径值与该边权值之和，如果小于待更新节点的最短路径值，则将这个待更新节点以及对应的边、节点加入到集合 Q 中，并将这个差值 $\delta(v_j) = d(v_i) + w(e) - d(v_j)$ 记录下来。表 1 显示了图 1 中首先加入到 Q 中的项。

表 1 父亲节点可能改变的节点及对应边和增量

序号	节点	边	增量
1	v_6	$[v_1, v_6]$	−3
2	v_6	$[v_3, v_6]$	−1
3	v_9	$[v_5, v_9]$	−4
4	v_{12}	$[v_8, v_{12}]$	−2

从集合 Q 中选取 $\delta(v_j)$ 最小的一组数据，对应的边的另一个节点 v_i 就是该节点 v_j 的新父亲节点。将以 v_j 为根节

点子树中的所有节点的最短路径值加上该增量值（负值），状态参量修改为 1，删除集合 Q 中所有跟这些节点有关的项。

定理 1：跟随 v_j 加入到新 SPT 中的 $T(v_i)$ 子树中的节点，处于当前的最短路径。

证明 1：对 $T(v_j)$ 中的任意节点 v_t，假设存在 v_x 使得 $d(v_x)+w([v_x,v_t])<d'(v_t)$，又 $d'(v_t)=d(v_t)+\delta(v_j)$，那么 $d(v_x)+w([v_x,v_t])-d(v_t)<\delta(v_j)$，即 $\delta(v_j)$ 不是 Q 中最小的，与题设矛盾，从而反证。

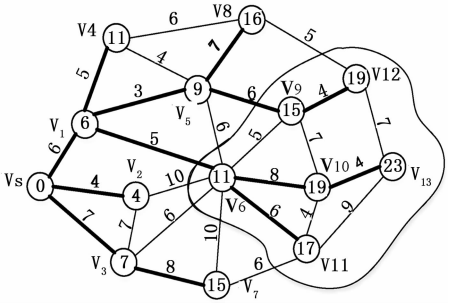

图 2　使用动态 Dijkstra 算法更新后的树

由于更新节点最短路径值 d 的减小，会使与之相连的节点的父亲节点发生改变，因此需要将 $d(v_i)+w(e)-d(v_j)<0$ 的节点对应的项加入到集合 Q 中去，重复更新过程。直到 Q 变成空集为止，退出循环，整个更新过程结束。该部分算法的伪码描述见算法 1，更新之后的 SPT 如图 2 所示。

算法 1：权值增大的 SPT 更新算法。
Input：网络 G，根节点 v_s，边 e 的权值从 $w(e)$ 变为 $w'(e)$；
Output：以 v_s 为根节点的新 SPT'
1) Initialize SPT from G
2) wait until one edge $e_0=[v_i,v_j]$: $w(e_0)\rightarrow w'(e_0)$
3) if $w'(e_0)-w(e_0)>0$, then
4) if $e_0\notin E(SPT)$　不处理　//不是 SPT 上的边无影响，不处理
5) else if $e_0\in E(SPT)$
6) 初始化．　$\delta=w'(e_0)-w(e_0)$；$\forall v\in V(SPT)$ 设置为 $Sta(v)=1$；若 $v_i=p(v_j)$，$\forall v\in T(v_j)$，$d(v)=d(v)+\delta$，$Sta(v)=0$
7) end
8) for all $e=[v_i,v_j]\wedge Sta(v_i)=1\wedge Sta(v_j)=0$ do
9) if $\delta(v_j)=d(v_i)+w(e)-d(v_j)<0$ then
10) add $\{v_i,v_j,\delta(v_j)\}$ to Q
11) end if
12) end for
13) end if
14) while $Q\neq\phi$
15) $\{v_i,v_j,\delta(v_j)\}\leftarrow \min Q$，修改 $p(v_j)=v_i$
16) for all $v\in T(v_j)$ do
17) 修改 $d(v)=d(v)+\delta(v_j)$，$Sta(v)=1$，delete $\forall v\in T(v_j)$ from Q
18) end for
19) for all $e=[v_i,v_j]\wedge Sta(v_i)=1\wedge Sta(v_j)=0$ do
20) if $\delta(v_j)=d(v_i)+w(e)-d(v_j)<0$ and $\{v_i,v_j,\delta(v_j)\}\notin Q$ do
21) add $\{v_i,v_j,\delta(v_j)\}$ to Q
22) end if
23) end for
24) end while

情况二：权值减小时。与情况一类似，权值减小的边出现的情况有两种，或不属于 SPT 中的边，或属于 SPT 中的边。这两种情况都有可能造成 SPT 结构的变化，因此边的权值减小受到影响节点的情况相对更复杂。

对于不属于 SPT 中的边权值减小，首先比较这条边对应的两个节点最短路径值的大小，d 值比较小的节点可能成为比较大的节点的新父亲节点。具体能否构成此影响，还要看“准父亲节点”的 d 值与边的新权值之和跟另一个节点 d 值的大小关系。如果“准父亲节点”的 d 值与边的新权值之和比较小，则修改另一个节点的父亲节点，以此节点为根的子树中的节点是此次受到影响的范围；如果“准父亲节点”的 d 值与边的新权值之和比较大，那么此次权值的变更对 SPT 没有影响。

对于属于 SPT 中的边权值减小，需要首先判断这条边的两个节点在树中的父子关系，找到子节点，以子节点为根的子树中的节点到根节点的最短距离 d 会因此变得更小，因此这仍是 SPT 的一部分。网络中的其他节点的最短路径会有一种向这部分节点“聚集”的趋势，因此剩下的节点即是此次权值变更受到影响的节点范围。算法的其他处理过程与情况一类似，在这里不再过多描述。该部分算法的初始化部分的伪码描述见算法 2。

算法 2：权值减小的 SPT 更新算法。
1) /* $w'(e_0)-w(e_0)<0$ */
2) if $e_0\notin E(SPT)$
3) 若 $d(v_i)<d(v_j)$　//找出 d 比较大的节点，因为它的父亲节点可能改变
4) if $d(v_i)+w'(e_0)>d(v_j)$　不处理　//未改变 SPT
5) else
6) 修改 $p(v_j)=v_i$，计算 $\delta(v_j)=d(v_i)+w'(e)-d(v_j)$
7) 初始化．　$\forall v\in V(SPT)$ 设置为 $Sta(v)=0$；　$\forall v\in T(v_j)$，$d(v)=d(v)+\delta(v_j)$，$Sta(v)=1$
8) end
9) else if $e_0\in E(SPT)$
10) 若 $p(v_j)=v_i$，计算 $\delta=w'(e_0)-w(e_0)$
11) 初始化．　$\forall v\in V(SPT)$ 设置为 $Sta(v)=0$；　$\forall v\in T(v_j)$，$d(v)=d(v)+\delta$，$Sta(v)=1$
12) end

3 实验验证与分析

为了验证算法的正确性，本节进行实验验证。实验运行主机配置为：CPU 主频 3.20 GHz，内存大小为 4 GB，操做系统为 Windows 7 旗舰版。设置网络节点数为 N 的无向网络，N 取 [100，1000]，步长值为 100。赋予网络连接边以及边的权值，设置节点的平均度为 5，权值范围为 [20，80]。首先根据生成的网络计算出 SPT，随机选择一条边的改变权值，考查静态 Dijkstra 算法和本文提出的动态 Dijkstra 算法在更新 SPT 所需要的时间。鉴于当权值增大和权值减小时算法的处理过程有所差异，所以在实验中也分两种情况分别进行比较。实验结果如图 3 图 4 所示。

图 3 显示了当一条边的权值增大时，在不同网络规模下，静态 Dijkstra 算法和动态 Dijkstra 算法更新 SPT 所需要的时间；图 4 显示了当一条边的权值减小时，在不同网络规模下，静态 Dijkstra 算法和动态 Dijkstra 算法更新 SPT 所需要的时间。两种情况下动态算法都比静态算法的时间开销要小，能在相对短的时间内更新 SPT，体现出了更好的性能。结合图 3 图 4 来看，在节点数目相同时，权值增大和减小静态 Dijkstra 算法更新 SPT 所用的时间几乎一样，这是因为当检测到有边的权值发生变化时，静态 Dijkstra 算法都是从根节点开始更新 SPT；两种情况下动态 Dijkstra 算法的变化趋势几乎一样，因为两部分算法的时间复杂度比较接近。

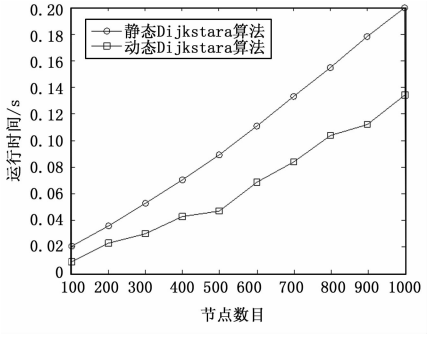

图 3 权值增大时更新 SPT 的时间与节点数的关系

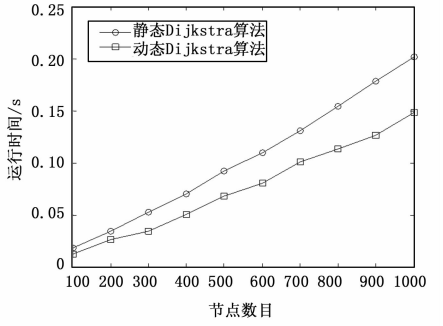

图 4 权值减小时更新 SPT 的时间与节点数的关系

4 结束语

当网络节点或边发生变动时，动态路由算法尽可能地对已有的最短路径树做最小改动来保持 SPT 的稳定，节省了计算时间，比静态算法更具有优势。本文提出了适用于无向网络的 Dijkstra 动态算法，并对算法中节点的更新范围作了进一步优化，证明了改进的合理性。最后通过实验，比较了当网络中的一条边的权值变化时，动态算法和静态算法更新 SPT 所需的时间，验证了所提出算法的可行性和优越性。文章只提出了边的权值发生变化的有关算法，在下一步研究中，将对节点增删以及多边、多节点变化的情况进行讨论，并进一步通过实验来验证所得出的结论。

参考文献：

[1] 张 钟．大规模网络图上的最短路径问题探究 [D]．中国科技大学，2014.

[2] 谢希仁．计算机网络（第四版）[M]．北京：电子工业出版社，2003.

[3] Dijkstra E. A Note Two Problems in Connection with Graphs [J]. Numerical Mathemat. 1959，1：269－271.

[4] 陈庆军．基于隐私保护的跨域路由策略优化：一种密码学的方法 [D]．南京：南京大学，2016.

[5] 肖乾才，李明奇，郭文强．多链路权值增大的动态最短路径算法 [J]．计算机科学，2012，39 (4)：114－118.

[6] 张水舰，刘学军，杨 洋．动态随机最短路径算法研究 [J]．物理学报，2012，61 (16)：1－10.

[7] Spira P，Pan A. On Finding and Updating Spanning Trees and Shortest Paths [J]. SIAM J. Computer. 1975，4 (3)：375－380.

[8] Narvaez P，Siu K Y，Tzeng H Y. New Dynamic Algorithns for Shortest Path Tree Coputation [J]. IEEE/ACM Trans. Networking. 2000，8 (6)：734－746.

[9] Narvaez P，Siu K Y，Tzeng H Y. New Dynamic SPT Algorithm Based on a Ball－and－String Model [J]. IEEE/ACM Trans. Networking. 2001，9：706－718.

[10] Chan E P F，Yang Y Y. Shortest Path Tree Computation in Dynamic Graphs [J]. IEEE Transaction on Computers. 2009，58 (4)：541－557.

[11] 孙知信，高艳娟，王文鼐．更新最短路径树的完全动态算法 [J]．吉林大学学报（工学版），2007，37 (4)：860－864.

[12] Xiao B，Cao J N. Dynamic Update of Shortest Path Tree in OSPF [A]. Proceeding of the 7th International Symposium on Parallel Architectures [C]. 2004：18－23.

[13] 章永龙．Dijkstra 最短路径算法优化 [J]．南昌工程学院学报，2006 (3)：30－33.

[14] 王战红，孙明明，姚 瑶．Dijkstra 算法的分析与改进 [J]．湖北第二师范学院学报，2008，25 (8)：12－14.

[15] 刘代波，侯孟书，武泽旭，等．一种高效的最短路径树动态更新算法 [J]．计算机科学，2011，38 (7)：96－99.