哈尔滨工业大学计算机科学与技术学院

《算法设计与分析》

课程报告

学号	
姓名	
报告日期	2020 年 12 月 27 日

1 论文题目

Maximilian Probst Gutenberg, Virginia Vassilevska Williams, and Nicole Wein. 2020. New Algorithms and Hardness for Incremental Single-Source Shortest Paths in Directed Graphs. In Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC '20), June 22–26, 2020, Chicago, IL, USA. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3357713.3384236

2 论文阅读报告

2.1 摘要

本文提出的算法是一个解决动态单源最短路径(SSSP)的问题的近似算法，该算法基于对动态带权有向图各点之间最短距离维护的经典算法 ES-Tree[1]的修改，提出的算法在处理有向增量图的过程中能在$m^{0.626-O(1)}$的时间内对增量图进行更新和查询，且在运行过程中整体的时间复杂度为$O(n^2 \log^4 n/\zeta)$，本算法所维护的距离是一个近似距离,设为d，设真实最短距离为d_0，d 的范围为$d_0 \le d \le \alpha d_0$,其中α为一个常数。同时，根据 k-Cycle 假设可以证明这一时间复杂度已经接近最优的结果。同时可以证明本文的算法在各种密度的图中的更新和查询的时间复杂度都可以逼近最优的情况，是目前已知的对于动态单源最短路径问题最为高效的近似算法。

2.2 问题定义

对于一个有向带权图，在插入或删除边或改变边的权重的过程中，不断更新从某一特定点 s 到其他顶点的最短距离。定义 n 为最大可能顶点数，m 为最大可能边数，W 为加权图中最大权重与最小权重的比值。如果更新操作序列仅由边缘删除和权重增加组成，则说G为减量图；如果更新操作限于边缘插入或权重减少，则说G为增量图。无论哪种情况，我们都说G是部分动态的，如果更新序列混合在一起，我们就说G是完全动态图。

2.3 算法或证明过程

2.3.1 算法基本思想

首先，介绍算法的基本思想。此基本思想的来源是对于无向无权图动态问题的思考，在无向无权图中，如果两个顶点的最短距离超过 3，那么它们之间就不存在一个结点可以从这两个顶点同时到达，假设存在则两点的最短距离应该小于 3。而有向图中这一性质显然不存在，如果出现中间结点 w 存在分别从 u 和 v 两个顶点出发的入边，u 和 v 的距离可以超过 3。

那么在有向图中需要引入一个新的概念：前向邻域。使用 N(u)表示 u 的前向邻域，其中的结点存在一条从 u 出发的入边，且与原点 s 的距离大于 u。引入前向邻域的概念之后，就可以说与原点距离相差较大的两点的前向邻域不存在重叠，这是由于如果 N(u)与 N(v)的重叠结点为 w，那么，由于 $d(w) \ge d(v)$,$d(w) \ge d(u)$,且 d(w)比 d(u)大得有限，那么如果 d(v)与 d(u)差距太大则与事实矛盾，因此本算法的基本思想是在一条最短路径上的顶点的前向邻域并不存在太多的重叠，并假设分析每一个结点的前向邻域假设其都不重叠并不会带来太大的

误差。

2.3.2 算法中用到的假设

(1).k-Cycle 假设：在具有 $O(\log m)$ 位字的 word-RAM 模型中，对于任何常数 $\partial > 0$，都存在一个常数整数 k，不存在可以算法可以在 $O(m^{2-\partial})$ 时间内检测出一个共有 m 条边的图中的 k 环。[2]

(2).OMv3 假设:在带有 $O(\log n)$ 位字的 word-RAM 模型中，使用多项式预处理时间求解 OMv3 的任何算法都需要 $n^{\varpi+1-O(1)}$ 的总时间来解决。[3]

2.3.3 算法初步设计

根据前向邻域的思想，可以对 ES-tree 进行合理的修改，在保证误差不太大的情况下大幅降低时间复杂度。首先，该算法需要定义一个数组为 A_u，其中 $A_u[i]$ 表示的顶点 v，d(s,v)=i，且 $v \subseteq N(u)$，当 d(u)需要减少的时候，对于 A_u 中的元素需要修改最短路径的顶点的范围为 $A_u[d(u)^{new}+2,n]$(表示的是从 $A_u[d(u)^{new}+2]$ 到 t 的顶点)，本算法的近似的出发点是更新过程中对于一部分 u 不需要每次更新 d(u)的时候都更新 $A_u[d(u)^{new}+2,n]$ 这一部分顶点。由于需要控制误差在一定范围内，因此需要对于何时更新进行一定的限制。首先，如果 $A_u[d(u)^{new}+2,n]$ 的大小小于 $O(n^{2/3})$ 那么定义 u 为一个轻结点，反之称 u 为一个重结点，对于轻结点，由于更新该节点之后需要更新的结点并不多，因此可以每次更新轻结点之后直接更新其后续结点；对于重结点 u，由于后续需要更新的结点数量较多，因此只有当 d(u)至少减少了 $n^{1/3}$ 之后才更新其后续结点，根据前面的假设可以证明，这种算法的时间复杂度为 $O(n^2 \log^2 n/\zeta)$，相比于 ES-tree 来说有了较大的性能提升，且此方法维护的最小路径 d(u)满足 $d_0(u) \le d(u) \le (1+\xi)d_0$

2.3.4 算法正确性证明

算法的正确性只需要证明估计出的距离 d(u)满足条件 $d_0(u) \le d(u) \le (1+\xi)d_0$，首先分析误差的产生是由于在更新重结点的时候并不是每次更新都更新其前向邻域中的结点，这是此算法中产生误差的唯一原因。首先需要指出：

（1）轻顶点不造成任何误差

（2）我们可以限制由前向邻域重叠的重顶点对造成的误差

（3）任何成对不相交前向邻域的最短路径上的重顶点的数量很小。

首先提出此算法中存在的一个不变式，

不变式 2.1. 在每次边缘更新后，如果 $v \in N(u)$，则 $|d(v)-d(u)| \le$

$$d(t) \le \sum_{i=0}^{k} d(s_{i+1}) - d(t_i^{'}) + \sum_{i=0}^{k-1} d(t_{i+1}^{'}) - d(s_{i+1})$$

$$< \sum_{0}^{k} d(t_i^{'}, s_{i+1}) + 3kn^{1/3} \le d(s,t) + n^{2/3}\zeta$$

证明：①不妨假设 d(u) ≤d(v),那么需要证明的就是在 d(v)不减少的时候，d(u)并不会减少太多。这一点是由更新结点的性质决定的，由于更新的方式是 d(u)减少 $n^{1/3}$ 的时候对 u 的前向邻域中的结点进行更新，而每次更新之后前向邻域中的任意结点 v 与原点 s 的距离 d(v)最大为 d(u)+1，因此当 d(u) ≤d(v)时该不变式成立。②当 d(u) ≥d(v)时，需要证明的是当 d(v)减少太多时 v 将不会留在 N(u)中，这一点同样是由更新结点的方法决定的，每当 d(v)减少 $n^{1/3}$ 时将更新 v 在 A_u 中的位置，当 d(v) ≤d(u)+2 时，v 将被移出 N(u)。故此不变式成立，证毕。

以下假设所有涉及的证明中如果不特别提及，结点都是重结点（轻结点不需要考虑误差问题）：

在增量图中，考虑一条最短路径 $\pi_{s,t}$ 表示从 s 到 t 的最短路径，其中 s 与 t 均为增量图 G 中的顶点，假设 t_0 =s，对于所有的常数 i，假设 s_{i+1} 是 $\pi_{s,t}$ 中在 t_i 后的第一个重结点，t_{i+1} 是 $\pi_{s,t}$ 中的最后一个前向邻域与 s_{i+1} 的前向邻域相交的结点（t_{i+1} 可能等于 s_{i+1}），这样我们就得到了顶点对$(s_1,t_1),(s_2,t_2),\ldots\ldots(s_k,t_k)$,此外，令 s_{k+1}=t，由于 s_{i+1} 是重结点，因此可以证明顶点对的数量 k≤ $2n/\gamma$ 。

对于每一个 i，令 v_i 是 N(t_i)∩N(s_i)的结点集合，由不变式 2.1 可知|d(s_i)-d(v_i)|≤ $n^{1/3}$ 且 |d(t_i)-d(v_i)|≤ $n^{1/3}$，因此，d(t_i)-d(s_i)≤$2n^{1/3}$ 。令 $t_i^{'}$ 是 t_i 在 $\pi_{s,t}$ 中的后继结点，那么经过与上述同理的证明可以得出 d($t_i^{'}$)-d(s_i)≤$3n^{1/3}$ 。

如果 u 是一个轻结点且(u，v)是一条边，那么可以得出 d(v) ≤d(u)+1,在更新中如果发生以下事件:a)插入边(u，v)，b)d(u)减小,如果发生了第一种情况，那么 d(v)最大值为 d(u)+1,如果发生了第二种情况，那么如果 d(v)≥d(u)+1 将被减少到 d(u)+1，因此可以得出结论：d(s_{i+1})-d($t_i^{'}$)=d($t_i^{'}$,s_{i+1}),(d(u,v)表示从 u 到 v 的最短距离)。因此我们可以得出以下不等式：

$$d(t) \le \sum_{i=0}^{k} d(s_{i+1}) - d(t_i^{'}) + \sum_{i=0}^{k-1} d(t_{i+1}^{'}) - d(s_{i+1})$$

$$< \sum_{0}^{k} d(t_i^{'}, s_{i+1}) + 3kn^{1/3} \le d(s,t) + n^{2/3}\zeta$$

根据最后这一不等式可以得出，如果 d(s,t)> $n^{2/3}$，那么 d(t)≤(1+ζ)d(s,t)。如果 d(s,t)≤

$n^{2/3}$，那么可以根据 ES-tree 得出 d(t)的准确值，那么 d(t)≤(1+ζ)d(s,t)也是成立的，因此此算法的正确性得到了证明。这一算法已经在时间复杂度上对于现有算法有了一定的突破，但是效果并不是十分明显。以下以此算法为基础进行算法优化。

2.3.5 算法优化

通过对上述初步算法设计的思考，如果需要降低时间复杂度那么很显然需要在不明显增大误差的程度上使得 ES-tree 更加“懒惰”，也就是更新结点的操作更不频繁。由此，优化算法的基本点是：上述初步算法设计中根据结点的轻重来决定更新的方式，优化算法中将轻重程度细化，特别地，对于从 0 到 logn-1 中的每一个 i。我们都有一个惰性 ES-tree 来处理与原点距离在 2^i 到 2^{i+1} 之间的结点的 ES-tree。同时，我们可以发现，处理较大距离的 ES-tree 可以承受更多的加性误差，因此在设置过程中我们可以将其设置得更加懒惰。因此，算法优化的出发角度就是将图中结点的层次分得更加细致，不同范围的结点使用不用的懒惰程度的 ES-tree。因此，算法需要解决的问题是怎么合理地更改顶点的轻重程度。和初步算法中的更改方式不同，根据上述初步算法的分析，每次在一个结点的轻重程度的改变的时候需要更新该结点的前向邻域中的所有结点的有关值。但是由于在优化算法中将轻重程度分为了更加细化的层级，那么如果和初步设计中的更新方式一样的话，由于|N(u)|的大小可能是 Ω(n),那么整体的运行时间将会达到Ω(n^2 D),其中 D 表示的是整体 ES-tree 树的深度。这样的时间复杂度是优化算法不能接受的，因此，我们需要提出新的更新结点的方式。

不妨假设结点 u 的沉重度为 h(u),此算法的更新方式可以总结为以下三点：

(1).如果 d(u)的值是 $2^{h(u)}$ 的倍数，那么将扫描 N(u)中的所有顶点，如果需要的话减小前向邻域中结点与原点的距离。

(2).如果 d(u)的值是 $2^{h(u)}$ 的倍数，那么根据需要增加 u 的沉重程度

(3).不管 d(u)的值如何，检查 u 是否离开其他结点 w 的前向邻域，如果是的话，在必要的时候减小 w 的沉重程度。

可以看出，如果采用这种更新方式更新的次数一定是比初步设计的算法少的，那么时间复杂度一定会有所下降，因此现在的关键是证明这种算法的正确性以及证明其时间复杂度是否有了明显的下降。

2.3.6 优化算法正确性证明

在证明之前，先定义几个符号：定义τ 为每一棵给定的 ES-tree 的深度，$d_\tau(v)$ 表示对于其中任意一个顶点 v 与原点 s 的距离估计，由于更新并不是每一次都发生，会因此为了更加符合定义，将初步设计中的 A_u 改称为$Cache_u$。需要证明的命题是：$d(s,t) \le d(t) \le (1+\xi)d(s,t)$，其中 d(s,t)∈[$\tau$,2 τ]。

引理 2.1. 在任何情况下，对任意τ 和任何 t∈V，都有 d(s,t) ≤ $d_\tau(t)$

证明：可以证明只有当 v 具有一条入边且入边的尾结点估计值比 d(v)小 1 以上使，才能让 d(v)减小，根据算法中的更新过程，仿照初步设计算法中的证明可以简单证明这一命题。

引理 2.2. 对于所有顶点$u \in V$，包含v的$Cache_u$ 的索引只能随时间减小。

证明：这一点性质是由增量图的性质决定的，即无论是从 $Cache_u$ 中移除 v 或是向 $Cache_u$ 中加入 v，v 的位置都是在 $Cache_u$ [d(v)],因为 d(v)是单调递减的，因此这一引理成立。

根据不变式 2.1，可以很容易地得出下列两个不变式。

不变式 2. 2. 对于所有的 u，v∈V，在每次处理更新之后，如果 v∈N(u),那么|d(v)-d(u)|≤ $2^{h(u)}$

引理 2. 3. 对于所有的 u ∈V，|N(u)|≥ $(2^{h(u)}-1)^{6n\log n/\zeta\tau}$

引理 2. 4. 在每次边缘更新之后，对每一个 t∈V，以及对于每一个 τ，d(s,t) ≤ $k \le \frac{n}{(2^h-1)^{\frac{6n\log n}{\zeta\tau}}} \le \frac{\zeta\tau}{6(2^h-1)\log n}$，如果有 d(s,t) ∈[τ ,2 τ]，那么就有 $d_\tau(t) \le (1+\zeta)$d(s,t)。

证明：和初步设计算法中的证明一样，假设 t_0 =s，对于所有的常数 i，假设 s_{i+1} 是 $\pi_{s,t}$ 中在 t_i 后的第一个重结点，t_{i+1} 是 $\pi_{s,t}$ 中的最后一个前向邻域与 s_{i+1} 的前向邻域相交的结点（t_{i+1} 可能等于 s_{i+1}），这样我们就得到了顶点对 $(s_1,t_1),(s_2,t_2),\ldots\ldots(s_k,t_k)$,此外，令 s_{k+1}=t。

根据定义以及引理 2.3，可以证得 $k \le \frac{n}{(2^h-1)^{\frac{6n\log n}{\zeta\tau}}} \le \frac{\zeta\tau}{6(2^h-1)\log n}$.

对于所有的 i，假设 v_i 是 N(t_i)∩N(s_i)的结点集合，根据不变式 2.2 可以得出 $d_\tau(t_i)-d_\tau(s_i) \le 2^{h+1}$，令 $t_i^{'}$ 是 t_i 在 $\pi_{s,t}$ 中的后继结点，那么经过与上述同理的证明可以得出 $d_\tau(t_i^{'})-d_\tau(s_i) \le 3*2^h$。

现在，令 $h_{\max}$ =logn 是最大的沉重度等级，我们将 h 初始化为 $h_{\max}$，根据顶点对 (s_i,t_i) 队 h 进行分割，分为不同的沉重度等级，在每一个沉重度等级中仿照初始算法中的证明，将分段的证明累加之后可以得到 $d_\tau(t) \le (1+\zeta)$d(s,t)，具体证明过程与前文证明一致，不再赘述证毕。

至此，可以证得优化算法的正确性是可以保证的，即可以保证 $d(s,t) \le d(t) \le (1+\xi)d(s,t)$。

2.4 实验结论

2.4.1 优化算法时间复杂度理论证明

首先给出结论，优化算法的时间复杂度为 $O(n^2/\zeta)$，证明如下：

不变式 2. 3. 在任何情况下，对所有 u∈V 以及所有正整数 i，有 h(u)<i≤logn，| $Cache_u$

[CacheInd(u, 2^i), $\tau_{\max}$]$\leq (2^i-1)^{\frac{12n\log n}{\zeta\tau}}$.

证明：由于初始时，$Cache_u$ 是空的，因此初始时 $Cache_u$ 满足不变量，以下是一些可能造成不变式不成立的情况，只要证明在下述情况中不变式均成立即可。

(1)h(u)减小：根据引理 2.4 可以得出，当 h(u)减小的时候此不变式是成立的。

(2) $Cache_u$ 中加入一个顶点 v：仅由于插入了边（u，v）才会发生这种情况。 但是，在将 v添加到 $Cache_u$ 之后，我们直接调用过程 IncrHeaviness(u)，根据更新结点的性质可以得出此不变式成立（具体证明与引理 2.4 证明基本一致）

由此可见，不变式 2.3 成立

引理 2.5. 对于所有$u\in V$和所有整数 i，0≤i≤logn, 算法在整个更新序列的过程中最多扫描 $Cache_u$ $O(\tau\log^2 n/2^i)$ 次。

引理 2.6. 在整个过程中，整体的时间复杂度为$O(n^2\log^4 n/\zeta)$。

证明：根据引理 2.5 可以得出扫描的最多次数，而每次更新需要摊销的时间复杂度为 $O(n^2\log^2 n/\zeta)$，因此引理 2.6 可以得到证明，因此可以得出，此算法的时间复杂度证毕。

2.4.2 算法实际运行性能

在小规模的增量图上运行此优化算法与经典 ES-tree 效率并无较大时间差别，在个人的实际运行过程中并无明显的差别，原因可能是测试的图规模太小，而论文中并没有给出实际运行的结果，因此暂时无法证明实际运行过程中此算法有明显优势。

3 领域综述

对于动态单源最短路径问题（SSSP），最为经典的确定性算法是 1981 年提出的 ES-tree 算法，该算法将图中的顶点分层，分层的依据是改顶点到特定的某一点的距离，并提取特定点到所有顶点的最短路径，将图简化成树，在插入时沿对应的最短路径搜索是否需要修改其他顶点的层数，以此来维护图中的最短路径。该算法的时间复杂度为$O(mnW)$, 目前暂无确定性算法可以突破这一时间复杂度的下限，且根据 k-Cycle 假设可以证明在组合问题的算法中 ES-tree 已经接近于最优算法，因此后续算法大多从近似算法入手进行修改。

后续 Henzinger 等人基于 ES-tree 提出了使用蒙特卡洛方法对顶点进行概率性随机选择的方法进行更新[4]，而此种算法需要所给的图需要是根据某种特定的顺序建立的，因此这种方法在使用过程中受限较大，该算法主要应用于无权图和减量图，但是在一定的情况下有扩展到增量图的可能且此算法的更新时间复杂度为 $mn^{9/10+O(1)}+\log W$ 。

2020 年，Probst 和 Wulff-Nilsen 同样提出了一种随机更新的算法[5]，该算法借鉴了 Henzinger 的算法中对于蒙特卡洛方法的使用，运用拓扑排序与随机选择相结合的思想，由于在拓扑排序的过程中使用的均是减量图的性质，因此此算法只适用于减量图，扩展到增量图的难度极大，本算法的时间复杂度为 $O(\min\{mn^{3/4}+\log W, m^{3/4}n^{5/4}\log W\})$ $O(\min\{mn^{3/4}+\log W, m^{3/4}n^{5/4}\log W\})$，时间复杂度上有相当程度的提升，但是使用的情

况受限程度也比较大。且前述两种近似算法均无法输出最短的路径的具体路径，智能输出估计出的最短路径。

总的来说，目前在解决 SSSP 问题的主流算法中，ES-tree 算法是确定性算法中时间复杂度最低的算法，也是最为成熟、应用场景最为广泛的算法，后续算法也都是基于 ES-tree 算法进行的改进。后续提出的算法大多是近似算法，大多是采用随机更新的方法，在降低时间复杂度的同时保证最短路径的值与真实最短路径的差距为线性关系。因此，在 SSSP 问题的解决中，减量图和无向图都有了专有的解决方式，但是对于增量图的解决方法暂时还处于欠缺的状态。基于 Henzinger 提出的优化方法可能在于对图中顶点连通性的思考，本文涉及的论文中提出的算法弥补了这一空白，提出的算法使用的限制较小，且在大多数情况下都可以逼近该情况下的最优解决时间。

4 方法不足与改进

4.1 方法不足

本论文主要的优化算法只讨论了增量图的情况，并没有讨论减量图的情况，而在单源动态最短路径问题中增量图和减量图的地位是同等重要的，由于动态单源问题最终的目标是要拓展到全动态图的问题中，也就意味着如果此算法只考虑增量图的话对于实际问题并没有什么实际作用。虽然目前更多的方法聚焦在解决减量图上，但是对减量图的解决远远没有达到最优的情况。本文中优化算法的方法在解决增量图上显示出了较大的优势，有一定的可能推广到减量图上。而且可以发现本文的优化算法中使用了较多的增量图的特性而完全没有考虑减量图，但是由于算法思想的存在以及减量图的某些性质事实上与增量图有一定的类似之处，那么可以考虑将此论文中的算法推广到减量图的应用中。

4.2 方法改进

回顾优化算法中的算法核心：将增量图中的结点的沉重度分为不同的等级，当 d(u)是 $2^{h(u)}$的倍数时，对 N(u)中的结点进行扫描，并根据需要减小结点的 d(v),同时决定是否要把结点 v 移出 N(u)。在减量图中事实上可以采用和这种方法极其类似的方法，总体的方法类似，在增量图中是在插入边的时候或是减少 d(u)的时候考察是否需要对 N(u)进行修改，优化算法的优势就在于将结点的沉重程度分成了多段，减少了更新 N(u)的次数。

故减量图也可以借鉴此种分类方法，增量图需要考量的是结点是否要从 N(u)中移出，而减量图需要考虑的是每次更新的时候是否需要将结点加入 N(u),如果能够证明仿照增量图设计的减量图的算法得到的估计最短距离在一定的误差范围内，那么由于更新方式基本类似，那么时间复杂度与增量图中的时间复杂度是一致的，由 4.3 中的证明可知，在减量图中这一算法也是可以证明其正确性的。那么这种算法是够可以直接推广到全动态单源有向图问题中呢？显然是不可以的，不妨假设极端情况在更新结点过程中采用一次增量图一次减量图的方式，那么更新结点的行为将永远不进行，而误差经过多次累积之后将达到极大的值，显然不会满足$d_0(u) \le d(u) \le (1+\xi)d_0$的条件，那就意味着这一算法无法直接推广到全动态单源有向图问题中。因此，这一算法只能作为全动态单源有向图中的一个子程序使用，且这一全动态也应该是受限的全动态，如果是经过设计的全动态问题那么将无法解决。

4.3 改进结果理论分析

仿照增量图中的结论，如果我们能证明$(1-\xi)d_0(u) \le d(u) \le d_0(u)$，那么就说明此算

法是可以直接推广到减量图中的。以下证明过程中，一些在增量图中证明的结论将不再证明直接写出。

在证明之前，先定义几个符号，为了保证证明的一致性，符号定义与增量图优化算法中的定义相同：定义τ为每一棵给定的 ES-tree 的深度，$d_\tau(v)$表示对于其中任意一个顶点 v 与原点 s 的距离估计，由于更新并不是每一次都发生，会因此为了更加符合定义，将初步设计中的 A_u 改称为 $Cache_u$。需要证明的命题是：$(1-\xi)d_0(u) \le d(u) \le d_0(u)$，其中 d(s,t)∈[$\tau$,2τ]。

引理 2.7. 在任何情况下，对任意τ和任何 t∈V，都有 $d_\tau(t) \le$ d(s,t)

证明：根据减量图的更新方式：每次可能出现的更新都是在删除边或者减少边的权重的时候发生的，而并不是每次进行这种操作都会进行减量图的更新，那就意味着在大多数时候满足 $d_\tau(t) \le$ d(s,t)，等号当且仅当进行了减量图更新的时候成立，因此引理 2.7 成立。

引理 2.8. 对于所有顶点$u \in V$，包含v的 $Cache_u$ 的索引只能随时间增大。

证明：根据减量图的更新方式可以很容易看出引理 2.8 成立，即无论是从 $Cache_u$ 中移除 v 或是向 $Cache_u$ 中加入 v，v 的位置都是在 $Cache_u$ [d(v)],因为 d(v)是单调递增的，因此这一引理成立。

由这一引理同样能给出以下两个不变式，证明方式与优化算法中的证明一致。

不变式 2.4. 在任何情况下，对所有 u∈V 以及所有正整数 i，有 h(u)<i≤logn，$|Cache_u$ [CacheInd(u, 2^i),$\tau_{\max}$]|≤$(2^i-1)^{\frac{12n\log n}{\zeta\tau}}$

则可知后续证明与优化算法中的证明一致，即可以证明$(1-\xi)d_0(u) \le d(u) \le d_0(u)$，即此算法经过一定的修改可以应用于减量图，且时间复杂度与在增量图中的使用一致，比现有的对减量图的操作有了明显的时间复杂度的提升。

综上所述，根据本文设计的算法的核心思想可以将这一种思想应用于增量图与减量图的动态单源最短路径问题的求解上，但是目前对于全动态图的单源最短路径问题暂无较好的解决方式。

4.4 改进结果实验验证

在理论上可以证明减量图的时间复杂度与增量图的时间复杂度是一致的，但是由于设计的实验中图的大小所限，在实际操作过程中与经典 ES-tree 算法相比并没有绝对的优势，因此实验结果在此不再展示。

5 附录

参考文献

[1] A. Schönhage. 1981. Partial and Total Matrix Multiplication. SIAM J. Comput. 10, 3 (1981), 434–455.

[2] Friedrich Eisenbrand and Fabrizio Grandoni. 2004. On the complexity of fixed parameter clique and dominating set. Theoretical Computer Science 326, 1-3 (2004), 57–67

[3] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. 2015. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In Proceedings of the fortyseventh annual ACM symposium on Theory of computing. ACM, 21–30.

[4] Monika Rauch Henzinger and Valerie King. 1995. Fully dynamic biconnectivity and transitive closure. In Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on. IEEE, 664–672

[5] Maximilian Probst Gutenberg and Christian Wulff-Nilsen. 2020. Decremental sssp in weighted digraphs: Faster and against an adaptive adversary. In Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms. SIAM, 2542–2561.