计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(10):164−170 [doi: 10.15888/j.cnki.csa.008176]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

大图上跳数受限的可达查询算法研究与优化①

邢耕源[1], 徐　云[2]

[1](中国科学技术大学 大数据学院, 合肥 230026)
[2](安徽省高性能计算重点实验室, 合肥 230026)
通讯作者: 徐　云, E-mail: xuyun@ustc.edu.cn

摘　要: 判断有向图上两个顶点之间是否存在一条路径是一个经典问题, 而对于一些路由规划和图分析等实际应用, 要求查找是否存在跳数受限的可达路径, 这是一个变种的图可达查询问题. 对于大图上跳数受限的查询算法, 不仅仅要对大图查询的时间效率和空间效率进行权衡, 而且还要利用跳数受限的特性进行优化. 普通的可达查询算法存在小度数顶点索引项占用空间过多的问题, 造成空间浪费严重. 为此我们提出了一种面向跳数受限的 2-hop 部分索引方法, 采用改进的索引方法并结合局部搜索, 实现跳数受限的有效可达性查询. 实验结果表明, 在 Orkut 社交网络数据集上与已有算法相比, 该算法索引空间节省了 32%, 同时查询时间略微增加, 使得我们算法可以计算更大规模图的跳数受限可达问题.

关键词: 图论算法; 可达性查询; 索引优化; 网络流优化

引用格式: 邢耕源,徐云.大图上跳数受限的可达查询算法研究与优化.计算机系统应用,2021,30(10):164–170. http://www.c-s-a.org.cn/1003-3254/8176.html

Research and Optimization of Reachability Query Algorithm with Limited Hop on Big Graph

XING Geng-Yuan[1], XU Yun[2]

[1](School of Data Science, University of Science and Technology of China, Hefei 230026, China)
[2](Key Laboratory of High Performance Computing of Anhui Province, Hefei 230026, China)

Abstract: It is a classic problem to judge whether there is a path between two vertices in a directed graph. For some practical applications such as routing and graph analysis, it is required to find whether there is a reachable path with limited hops, which is a variant of reachability query in graphs. For the query algorithm with limited hops on a large graph, it is necessary to balance the time and space efficiency of large-graph query and optimize the algorithm with the characteristics of limited hops. The common reachability query algorithm takes up too much space for small-degree vertex index entries, which leads to serious space waste. Therefore, we propose a hop-limited 2-hop partial index method, which combines an improved index method with local search to achieve hop-limited effective reachability query. The experimental results show that, compared with the existing algorithms, the proposed algorithm can save 32% index space and slightly increase query time on the Orkut social network dataset. Thus, the proposed algorithm can calculate the hop-limited reachability problem of larger graphs.

Key words: graph algorithm; reachability query; index optimization; network flow optimization

① 基金项目: 国家自然科学基金面上项目 (61672480)
Foundation item: General Program of the National Natural Science Foundation of China (61672480)
收稿时间: 2021-01-11; 修改时间: 2021-02-23; 采用时间: 2021-03-16

图网络能够有效表达不同类型事物之间复杂的关系, 在现实生活不同领域中有着广泛的应用[1]. 随着信息技术的不断发展和互联网不断渗入生活, 各类图数据规模发生爆炸式增长, 以社交网络为例, 2011 年微信月活跃用户数有 0.5 亿人, 而 2019 年微信月活跃用户数达到了 11 亿, 是中国用户量最大的 APP. 面对如此大规模的数据量, 传统的图搜索方案已经无法胜任.

图可达查询是图数据管理领域的研究热点问题, 然而在现实生活中还需要对可达顶点间的距离加以限制[2]. 例如在社交网络中, 根据小世界理论[3], 任意两个人之间总存在着 6 跳的通路, 单纯的可达性查询失去了意义. 用户节点间的距离反映着用户之间联系的紧密程度, 限定距离下的可达性查询可以帮助分析用户社区间的关系或搜索与自身相近的其他用户[4]; 在数据中心网络中, 设计部署整体网络结构时, 网络图中服务器间的距离信息反映着信息传递的时效性[5]. 对于限定距离下的可达性查询, 在学术界被称为 k 跳可达问题[6].

大图上各类算法, 都会通过构建索引信息, 帮助在查询时快速求解. 大图索引的建立和利用索引的查询策略, 是算法设计的关键. 索引的构建是为了根据图局部结构归纳信息, 进行状态压缩; 查询策略需要在快速求解的同时保证问题的完备性, 大图上现有的查询算法, 都需要对时间效率和空间效率进行平衡[7]. 现有的 2-hop 类算法通过构建由中介顶点构成的 2-hop 类索引, 加速搜索. 该算法查询时间虽然迅速, 但是受限于存储空间, 不适合更大规模的图上运算.

为此本文提出了一种基于部分索引与局部搜索结合的跳数受限可达查询算法. 通过改变原索引结构, 提高索引利用率, 从而在减少空间占用同时保持查询时间尽量低. 本方法首先构建了基础的 2-hop 可达索引; 然后采用了近邻节点和大度数热点顶点对索引进行约简; 最后采用索引表和局部搜索结合的方法进行可达性查询, 能够节省更多的索引空间, 从而适用于解决更大规模的图可达查询问题.

1 已有大图查询类算法分析

目前, 已有的图查询算法包括一般可达性查询、最短路径查询、k 跳可达性查询等.

1.1 一般可达性查询算法

大图上的可达性查询算法可以根据索引完整的程度和对在线搜索的依赖性, 分为传递闭包类[8]、hop 索引类[9]、索引+类算法[10].

(1) 传递闭包类算法

该类算法利用设计的索引结构对大图可达性信息进行预计录, 利用自身设计的各类索引结构能够快速查找顶点间的可达性关系. 在索引设计上, 这类算法利用了图上局部结构 (如树[11]、链[12]) 的拓扑性进行剪枝并记录, 这一类算法在响应时间上是最快的, 如图 1 所示. 但是其自身的索引结构会随着图规模的增大而快速增大, 因此存储空间是这类问题的桎梏. 随着图规模达到百万甚至千万顶点级别, 这类算法已经不能完成求解.

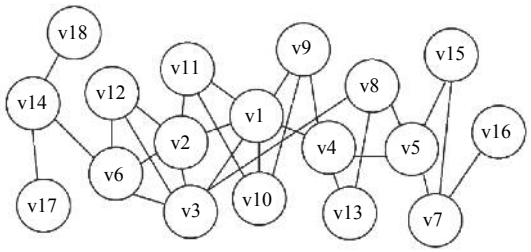

图 1 简单图网络

(2) hop 索引类算法

该类算法利用一些选定的中继节点作为中介, 进而考虑目标顶点与中继顶点间的可达性. 随着研究的不断深入, 该类算法利用的中继结构不再限于单顶点, 开始利用链路、树等图结构作为中间结构; 演化出了 2-hop[13] 类, 演化成 3-hop[14]、path-hop[15] 类算法, 如图 2 所示, 为简单图网络对应的 2-hop 索引.

(1, 0)					
(2, 0)	(1, 1)				
(3, 0)	(1, 1) (2, 1)				
…	…				
(16, 0)	(7, 1)	(5, 2) (15, 2)	…		
(17, 0)	(14, 1)	(6, 2)	…	(7, 6) (15, 6)	(6, 7)
(18, 0)	(14, 1)	(6, 2)	…	(7, 6) (15, 6)	(6, 7)

图 2 2-hop 索引结构

这一类算法在回答可达性查询时, 只需要在对应行索引上进行搜索, 寻找公共中介顶点的表项. 该类方法的关键是 hop 索引的建立, 即选取合适的中介结构辅助索引建立.

(3) 索引+类算法

该类算法发现在实际搜索中, 有向图点对间的不

可达查询占了绝大多数[16]; 相比于发现可达性, 他们更期望快速发现点对间的不可达关系, 因此他们利用自身索引对可达性进行快速筛选.

如果凭借索引不足以证明点对间的不可达关系, 这类算法利用在线搜索用于判定顶点间的可达性. 这类算法的空间占用是最小的, 但相应其查询时间最长.

1.2 路径查询算法

路径查询算法按照索引规模、索引种类可以分为在线搜索[17]、地标算法[18]、树分解算法[19] 等.

(1) 在线搜索算法

在线搜索算法不利用索引信息, 在图上直接搜索两点之间的路径信息, 在线搜索包括经典的 Dijkstra 算法和 Floyd-Warshall 算法等. 在后续发展中有了 A*算法、蚁群算法等改进的算法, 该类算法在图顶点规模达到百万及以上时, 即使在并行下, 算法时间效率仍旧较为低下.

(2) 地标算法

地标搜索是搜索最短路径的一类近似算法. 这类算法事先选定一些地标点, 以地标点作为中介, 依次计算所有经过地标下的最短路径信息; 将其中最短的路径作为备选路径[20]. 这类算法的效果取决于最短路径是否经过一个真实的地标点, 因此合适的地标点选取策略是这类算法的关键. 目前常见的地标选取策略通常是基于平均顶点度数和中心距离的启发式算法.

树分解类算法构建了一棵新的索引树[21], 利用构建的索引树, 树分解算法可以按级对小规模节点团进行搜索, 最终构建出最短路径. 该类算法将原图中的一团临近节点缩为索引树中的一个节点, 真实图中的一条路径对应团内路径和团间路径, 利用搜索节点对的最小共同祖先和局部最短路径搜索算法可以分别解决两个子问题, 该类策略的关键是在树高和团内节点数间平衡, 并构建出一棵合适的搜索树.

1.3 *k* 跳可达性查询

关于 *k* 跳可达算法, 学术界已有算法相对较少, 已有的 *k* 跳可达算法大多是有向图上算法. 基于顶点覆盖集的 *k* 跳可达查询算法通过求解原图上的一个顶点覆盖集, 然后在顶点覆盖集之上根据顶点之间的距离关系构建关系闭包作为图索引, 在查询时可以通过查询顶点所在闭包索引, 直接得到顶点间 *k* 跳可达关系. 但是这个方法的空间复杂度较高, 因为顶点覆盖集中的顶点之间的关系越稠密, 带来的索引开销越大, *k*-reach 的索引空间和索引时间都随着图规模的增长呈指数上涨.

为了改善 *k* 跳可达性的扩展性问题, Cheng 等给出了一种基于控制集的索引方法[22], 并且将索引组织成两级索引的结构, 一定程度上提高了算法的扩展性, 但在扩展性得到了提高的同时, 却牺牲了查询的时间效率.

近年国内也有 *k* 跳可达查询相关研究, 基于双向搜索的 *k* 跳查询算法通过优化双向搜索过程提高了 *k* 跳可达性查询的时间效率[23]. 基于图压缩的 *k* 跳可达性查询方法[24], 通过对原图进行等价类的压缩来降低图的规模, 提高算法的可拓展性. 基于 BFS 树的 *k* 跳查询算法通过计算索引交集的计算方法, 提高了算法的时间效率.

2 基于 hop 索引的 *k* 跳可达算法

k 跳可达性查询相较于一般的可达性查询约束更为严格, 因为一般可达性查询可以视为 $k=\infty$下的情况; *k* 跳可达相较于最短路径查询约束较为宽松, 因为 *k* 跳可达在求解到一条跳数不大于 *k* 的路径后, 不关心点对间的最短路径.

我们解决索引空间和查询时间之间平衡的主要方法是: 首先对于原索引中不常被利用缺占据大量空间的小度数顶点索引项, 按一定比率选取其中部分进行“约简”操作. “约简”操作指: 保留其近邻顶点索引和经过的大度数顶点索引, 剔除其他索引项. 对于优化后的索引表无法确定的查询, 转用使用与索引表相结合使用 LRU 策略的局部搜索进行补充搜索.

为了求解 *k* 跳可达问题并优化原索引结构, 设计了算法 1.

算法 1. 基于 hop 索引的 *k* 跳可达算法

1. 从原图文件中使用 PSL 算法建立 2-hop 类索引;
2. 在原图中确定代表顶点度数范围;
3. 原索引图上小度数顶点按照约简比率进行约简处理: 保留原行索引上的近邻顶点以及经过的步骤 2 筛选后的大度数代表顶点;
4. 搜索对应查询顶点对所在的行索引, 检测其对应行是否存在中介顶点满足对应跳数之和在 *k* 跳之内, 若存在则算法中止;
5. 查询 LRU 索引是否存在顶点的完备信息, 若存在则在 LRU 上进行检测, 算法中止;
6. 在原索引表上, 迭代式建立完整的行索引, 在完整索引上进行检测, 并将建立好的完整索引更新至 LRU 索引上.

2.1 图索引的建立

为了求解距离限制的可达查询问题, 选用带有距

离信息的 2-hop 类索引, 2-hop 类索引维护着所有顶点对应的 2-hop 表. 当两个不同的顶点存在着一个公共的中介节点时, 说明两者可达. 此时查询两者对应的距离标签, 如果所有距离标签中最小的在限制范围内, 则说明这两个顶点是在限制范围内可达的.

构建完整表的方法, 依赖于顶点对间的距离信息. 算法最开始将每个顶点距自身距离为 0 的顶点加入自己对应的表单中 (自身); 然后检索所有的边, 构建距离为 1 的项, 按照优先度, 将每条边加入优先度较高的顶点表单中, 然后依照距离信息, 递归的依次构建基于距离的索引, 直到构建到距离为 k 的索引.

2.2 索引结构的约简

2-hop 类索引构建时, 表单中索引项的选取都是单条最短路径上优先度最高的顶点; 因此大度数顶点的索引项会较短, 而小度数顶点的完整索引项会较繁琐. 回答完整性的可达性查询, 因保存这些边缘顶点全部的信息牺牲掉了很大的存储空间; 而真正查询时针对这些边缘顶点的查询频度也较低. 这也造成了因存储不能被及时利用的信息所造成的存储空间浪费, 此需要调整存储空间和查询时间之间的平衡.

为了弥补保留部分索引造成的信息丢失, 在查询时需要进行局部搜索, 检测限制距离内两个目标顶点之间的可达性关系, 在社交图中, 除了边缘顶点临近的顶点外, 利用其他高度数的顶点作为索引可以加速搜索. 针对小度数边缘顶点采用保留近邻节点信息以及高度数代表顶点作为索引的策略, 因社交图上的顶点倾向于发现与自身相近的顶点或与热点顶点间关系, 所以保留两种不同类型的顶点作为部分索引, 使其加速整体索引搜索过程.

2.3 基于约简后索引表的查询算法

2.3.1 可达性查询算法

针对社交图上 2-hop 索引信息不平衡的情况, 本文通过保留近邻节点和大度数代表顶点的手段约简了索引结构, 整体查询流程如图 3 所示.

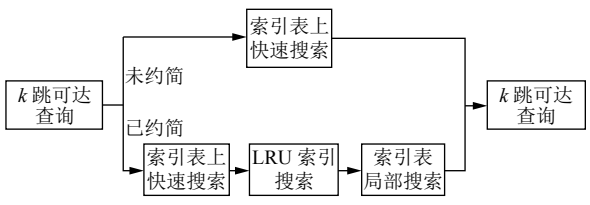

图 3 可达查询算法流程图

在查询点对间可达性时, 首先在约简后的部分索引上进行查询, 若查询顶点并非小度数顶点, 则其对应的表项是完整的, 基于索引表的可达性查询结果是完备的. 若待查询的顶点是小度数顶点, 则优先在其约简后的部分索引上进行 2-hop 索引查询; 若经过查询后存在着 k 跳关系, 则可以直接回答可达性, 若未搜索到 k 跳可达关系, 则需在 LRU 索引上查询已有的可达性索引, 查看是否近期已经补全过对应行索引.

若 LRU 索引上仍未有对应的结果, 则需在索引表上进行对应行的局部搜索; 并将补全后的行索引存入 LRU 索引, 并将 LRU 索引进行更新, 在原索引表上的局部搜索过程与建立 2-hop 索引时的检索过程一致, 依据距离信息从小到大依次检查约简的小度数顶点不同跳数下的 2-hop 关系, 将其依次补全.

2.3.2 基于约简后索引表的查询算法优劣势分析

使用约简的 2-hop 索引可以有效地减小索引空间, 并保持可达性查询时间与完备索引上的查询时间在同一数量级下, 保持时间性能并优化空间性能的关键在于代表索引, 即近邻顶点和高度数顶点的选取. 但本文的约简方法对原图类型有较强要求, 对于非社交图处理效果较差, 因其不再具有选取近邻顶点和高度数顶点作为代表点的特征, 同时对于待查询点对也有一定要求, 对于完全随机的点对间查询效果也不明显.

3 实验结果与分析

实验分别测定了不同约简比率下、不同局部搜索方法、不同跳数下索引空间与查询时间之间的关系.

3.1 数据集介绍与实验配置

本文选用的数据集来自斯坦福网络分析项目提供的开源数据集, YouTube[25] 和 Orkut 数据集均为社交网络数据集. 其中 YouTube 数据集规模较小, 该数据集顶点数为 1134 890 个, 边数为 2987 624 条; Orkut 数据集规模较大, 该数据集顶点数为 3072 441 个, 边数为 117 185 083 条, 所有的实验都在单机 Ubuntu 18.04.5 LTS 环境下运行. 内存为 512 GB, 实验环境的线程数最高为 56 个.

3.2 评估标准与结果分析

由于本文提出的基于方法是基于 hop 类算法的 k 跳可达查询算法, 主要贡献在于优化 hop 类索引结构, 能够胜任规模更大的图上快速查询的任务. 因此在实验评估标准的选择上, 本文主要选择索引空间、查

询时间两个评估标准. 同时在对比方法选取上, 本文选择了 2019 年 SIGMOD 会议上提出的 PSL 索引算法[26], 该算法已是最先进的 hop 类算法.

整体实验方法分为两部分: 确定局部搜索策略和选用某一实验参数. 其中, 不同局部搜索算法实验部分用于确定局部搜索算法, 优化比率实验部分用于确定实验的一个可变参数: 低度数顶点索引优化比率. 索引优化实验部分用于测定在选用参数下索引空间和查询时间之间的关系.

3.2.1 不同局部搜索算法实验

本文设计了多种不同的局部搜索算法, 实验表中展示的是优化后的基于启发式搜索的双向局部搜索与基于索引表的搜索方法间对比, 本实验数据集采用 YouTube 数据集, 对比了不同跳数 (k) 下两种采纳不同局部搜索算法与原算法 (PSL) 间索引空间和查询时间的关系.

图 4 和图 5 中 PSL 算法为对比算法, PSL_pru 为双向启发式搜索算法, PSL*算法为索引表局部搜索算法. 实验具体数据见表 1, 表 2. 由图可见, 两种改进的索引优化算法均在索引空间上效率超过了原算法, 而在查询时间上高于原算法. 其中基于索引的局部搜索算法, 由于采用了 LRU 机制, 因此其索引空间略高于双向搜索查询算法; 但其对应的查询时间则大大减少, 与未经优化的 PSL 算法更为相近. 保证了查询时间在同一数量级下, 同时索引空间得到了优化.

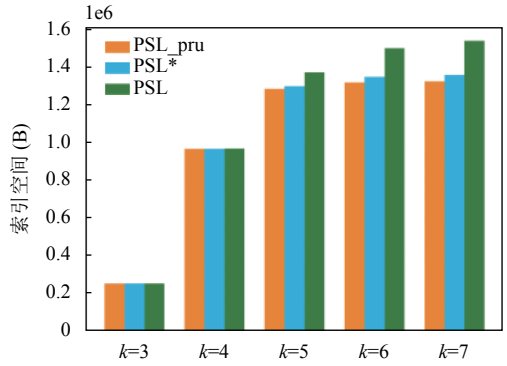

图 4 不同搜索算法空间性能比较

3.2.2 优化比率实验

在确定了局部搜索算法的种类后, 为了得到更好的空间优化效果, 同时保证时间性能在同一数量级下我们测试了不同优化比率下查询时间和索引空间的关系. 优化比率作为平衡索引时的一个重要参数, 影响的是约简小度数顶点的比例, 优化比率越低, 被选中修改的小度数顶点项越少, 实验测试数据集为 YouTube 数据集, 测试了不同跳数下, 不同优化比率下的索引空间和查询时间.

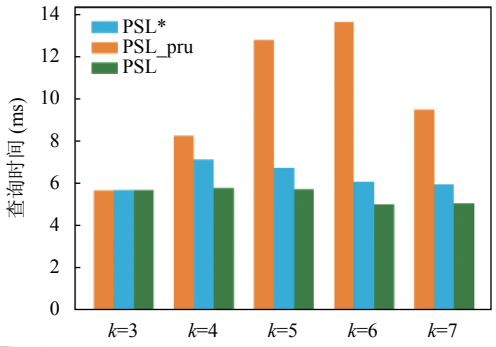

图 5 不同搜索算法时间性能比较

表 1 不同跳数下搜索策略的索引空间和跳数间关系 (B)

算法	k=3	k=4	k=5	k=6	k=7
PSL_pru	254 568	964 384	1283 821	1315 918	1322 170
PSL	254568	967260	1371 620	1497 324	1535 412
PSL*	254568	965 763	1295 593	1347 226	1354 289

表 2 不同跳数下搜索策略的查询时间和跳数间关系 (ms)

算法	k=3	k=4	k=5	k=6	k=7
PSL_pru	5.7386	8.3129	12.8272	13.703	9.5526
PSL	5.7491	5.84903	5.7824	5.08302	5.13664
PSL*	5.7486	7.18248	6.79591	6.13341	6.02741

此处的优化比率作为实验的一个参数, 可以根据实际需求人工配置. 该比率越高保留的顶点数目越少, 算法空间优化比率越高; 比率越低, 保留的原索引项越多, 算法查询时间消耗越低. 本实验为了突出空间优化效果, 选用了查询时间不发生显著改变下的并有着较优空间优化效果的比率进行测定. 实际应用时, 可以根据需求选取适当的参数配置.

由表 3 可见, 30% 至 70% 的比率下, 索引空间减小的速度较为均匀, 但查询时间方面, 50% 至 60% 的优化比率出现了较为明显的查询时间上升. 60% 至 70% 的优化比率下出现了更为剧烈的时间上升, 因此考虑到使查询时间保持在原数量级, 本文后续实验选用 50% 下的约简比率进行实验.

表 3 不同优化比率下查询时间与索引空间关系

优化比率(%)	30	40	50	60	70
查询时间 (ms)	6.68372	7.0287	7.67301	14.32041	53.4981
索引空间 (B)	1014735	902873	801909	699446	591672

3.2.3 索引优化实验

确定了局部搜索算法和优化比率后, 我们在规模更大的社交网络数据集下进行了与原算法相比较新的对比实验, 实验的性能评估指标, 我们仍旧选定为索引空间和查询时间. 测定了不同跳数变化下, 我们的算法与原算法相比性能的优劣,

通过实验结果, 由图 6 和图 7 可见, 从跳数为 4 开始, 本文的算法空间性能开始逐步优化, 同时保证了时间效率上略高于原算法. 具体实验数据见表 4, 表 5. 其中索引空间可以优化为原算法的 68.18%, 同时保证查询时间为原算法 1.38 倍, 保证了时间性能在原算法同一数量级下.

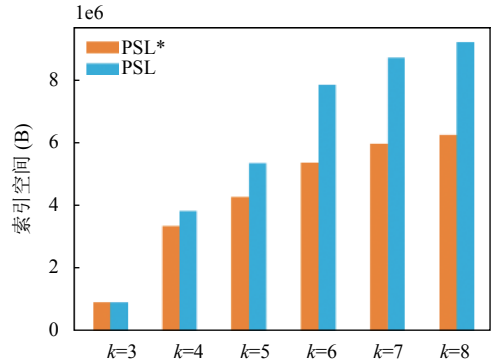

图 6 索引优化前后空间性能比较

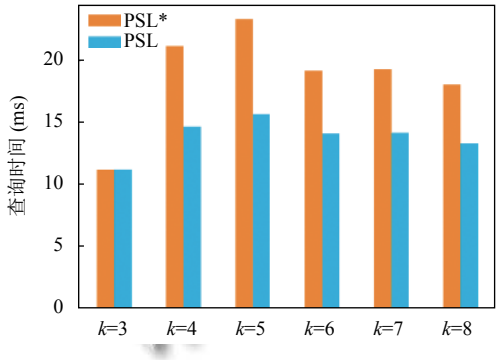

图 7 索引优化前后时间性能比较

表 4 索引优化前后空间性能比较 (B)

算法	k=3	k=4	k=5	k=6	k=7	k=8
PSL	954568	3867267	5371058	7849384	8725119	9214335
PSL*	954568	3380262	4296846	5394068	5994156	6287463

表 5 索引优化前后时间性能比较 (ms)

算法	k=3	k=4	k=5	k=6	k=7	k=8
PSL	11.2683	14.7251	15.7095	14.1580	14.2369	13.3616
PSL*	11.2461	21.1734	23.3159	19.1784	19.3230	18.0972

该实验结果中, 可见空间性能随跳数 k 有明显的变化, 但查询时间上出现了 PSL*整体查询时间先上升后下降的现象. 引发这样的现象的原因是: 当限制的跳数 k 较小时, 顶点间的 k 跳关系较难达成, 因此需要借助除原索引外的局部搜索进行测定; 而当跳数 k 上升至一定程度后, 顶点间的 k 跳约束得到了松弛, 此时仅凭借索引表可以得到肯定结果的顶点对比率得到了上升. 得到肯定查询后的顶点对不再需要借助局部搜索进一步查询, 因此算法的查询时间得到了下降.

4 结论与展望

本文针对 2-hop 类索引在 k 跳可达问题上出现的因索引分布不均匀造成的空间浪费的现象, 提出了人工可控的平衡索引空间与查询时间的优化算法. 该算法首先构筑一般的 2-hop 类索引表, 再根据约简比率对原索引表进行平衡处理, 并记录下被处理过的小度数顶点标号. 同时, 我们设计了与约简算法相应的查询算法, 保证查询的完备性, 同时利用 LRU 机制加速查询阶段. 在查询可达性关系时, 首先在约简后的索引表上进行查找, 若查询不到对应顶点信息, 则转入 LRU 索引上查询, 若仍得不到肯定结果, 再转入局部搜索, 并将结果更新至 LRU 索引上.

实验表明, 我们的方法相比于已有的最先进的索引方法, 能够有效地减小索引占用的空间, 同时保证查询时间增加的幅度可控. 下一步的工作重点, 是设计针对不同图类型和图半径等信息的参数自动优化算法, 增强算法对不同类型的图网络的适应性, 同时设计新的机制, 在优化查询时间上得到进一步的发展.

参考文献

1 杨晓冬. 大图上的 K 跳可达性查询算法 [硕士学位论文]. 武汉: 华中科技大学, 2016.

2 Cheng JF, Yu JX. On-line exact shortest distance query processing. Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology. Saint Petersburg: ACM, 2009. 481–492.

3 Cheng J, Shang ZC, Cheng H, *et al*. K-reach: Who is in your small world. Proceedings of the VLDB Endowment, 2012, 5(11): 1292–1303. [doi: 10.14778/2350229.2350247]

4 BIG Data Center Members. The BIG Data Center: From deposition to integration to translation. Nucleic Acids Research, 2017, 45(D1): D18–D24. [doi: 10.1093/nar/gkw

1060]
5 Chen W, Sommer C, Teng SH, *et al*. A compact routing scheme and approximate distance oracle for power-law graphs. ACM Transactions on Algorithms, 2012, 9(1): 4.
6 Wang HX, He H, Yang J, *et al*. Dual labeling: Answering graph reachability queries in constant time. 22nd International Conference on Data Engineering. Atlanta: IEEE, 2006. 75.
7 Trißl S, Leser U. Fast and practical indexing and querying of very large graphs. Proceedings of The 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD'07). Beijing: ACM, 2007. 845–856.
8 Jagadish HV. A compression technique to materialize transitive closure. ACM Transactions on Database Systems, 1990, 15(4): 558–598. [doi: 10.1145/99935.99944]
9 Jin RM, Xiang Y, Ruan N, *et al*. 3-Hop: A high-compression indexing scheme for reachability query. Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. Rhode Island: Association for Computing Machinery, 2009. 813–826.
10 Yildirim H, Chaoji V, Zaki MJ. GRAIL: Scalable reachability index for large graphs. Proceedings of the VLDB Endowment, 2010, 3(1–2): 276–284. [doi: 10.14778/1920841.1920879]
11 Guo YF, Qin Z, Chang Y. A novel hybrid algorithm for the dynamic shortest path problem. Sixth International Conference on Natural Computation. Yantai: IEEE, 2010. 2545–2550.
12 Abraham I, Delling D, Goldberg AV, *et al*. Hierarchical hub labelings for shortest paths. 20th Annual European Symposium on Algorithms. Berlin Heidelberg: Springer, 2012. 24–35.
13 Piccardi M. Background subtraction techniques: A review. IEEE International Conference on Systems, Man and Cybernetics. The Hague: IEEE, 2004. 3099–3104.
14 Cohen E, Halperin E, Kaplan H, *et al*. Reachability and distance queries via 2-hop labels. SIAM Journal on Computing, 2003, 32(5): 1338–1355. [doi: 10.1137/S0097539702403098]
15 Cai J, Poon CK. Path-hop: Efficiently indexing large graphs for reachability queries. Proceedings of the 19th ACM International Conference on Information and Knowledge Management. Toronto: Association for Computing Machinery, 2010. 119–128.
16 Wei H, Yu JX, Lu C, *et al*. Reachability querying: An independent permutation labeling approach. The VLDB Journal, 2018, 27(1): 1–26. [doi: 10.1007/s00778-017-0468-3]
17 D'Andrea A, D'Emidio M, Frigioni D, *et al*. Experimental evaluation of dynamic shortest path tree algorithms on homogeneous batches. 13th International Symposium on Experimental Algorithms. Cham: Springer, 2014. 283–294.
18 Akiba T, Iwata Y, Yoshida Y. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. New York: Association for Computing Machinery, 2013. 349–360.
19 Akiba T, Sommer C, Kawarabayashi KI. Shortest-path queries for complex networks: Exploiting low tree-width outside the core. Proceedings of the 15th International Conference on Extending Database Technology. Berlin: Association for Computing Machinery, 2012. 144–155.
20 Cheng J, Huang SL, Wu HH, *et al*. TF-Label: A topological-folding labeling scheme for reachability querying in a large graph. ACM SIGMOD International Conference on Management of Data. New York: ACM, 2013. 193–204.
21 Hay M, Miklau G, Jensen D, *et al*. Resisting structural re-identification in anonymized social networks. Proceedings of the VLDB Endowment, 2008, 1(1): 102–114. [doi: 10.14778/1453856.1453873]
22 Cheng J, Shang ZC, Cheng H, *et al*. Efficient processing of *k*-hop reachability queries. The VLDB Journal, 2014, 23(2): 227–252. [doi: 10.1007/s00778-013-0346-6]
23 周军锋, 陈伟, 费春苹, 等. BiRch: 一种处理 *k* 步可达性查询的双向搜索算法. 通信学报, 2015, 36(8): 50–60. [doi: 10.11959/j.issn.1000-436x.2015230]
24 李鸣鹏, 高宏, 邹兆年. 基于图压缩的 *k* 可达查询处理. 软件学报, 2014, 25(4): 797–812. [doi: 10.13328/j.cnki.jos.004567]
25 Yang J, Leskovec J. Defining and evaluating network communities based on ground-truth. Proceedings of the 2012 IEEE 12th International Conference on Data Mining. Brussels: IEEE, 2012. 745–754.
26 Li WT, Qiao M, Qin L, *et al*. Scaling distance labeling on small-world networks. Proceedings of the 2019 International Conference on Management of Data. Amsterdam: Association for Computing Machinery, 2019. 1060–1077.