计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2020.20190569
57(9): 1949-1960, 2020

标签约束可达查询的高效处理方法

杜 明[1] 杨 云[1] 周军锋[1] 陈子阳[2] 杨安平[1]
[1](东华大学计算机科学与技术学院 上海 201620)
[2](上海立信会计金融学院信息管理学院 上海 201620)
(duming@dhu.edu.cn)

Efficient Methods for Label-Constraint Reachability Query

Du Ming[1], Yang Yun[1], Zhou Junfeng[1], Chen Ziyang[2], and Yang Anping[1]
[1](*School of Computer Science and Technology, Donghua University, Shanghai* 201620)
[2](*School of Information Management, Shanghai Lixin Institute of Accounting and Finance, Shanghai* 201620)

Abstract The label-constraint reachability query $s \rightarrow_L t$ is used to answer whether there is a directed path from s to t in the given graph, such that every label on the path belongs to L. Considering that existing approaches suffer from long index construct time, large index size and long query time, we first propose to construct a bidirectional path label index based on k nodes with large degrees, such that we can efficiently construct a smaller index, while at the same time covering a large portion of reachable information. For this large nodes based index, we propose several optimization techniques to reduce the index size, which can in turn speed up the processing of reachable queries. Further, even though the index size is small for large nodes based bidirectional path label, it does not cover all the reachable queries, and cannot avoid the graph traversal operation during query processing. To this end, we further propose a bidirectional path label index which is constructed based on all the nodes and therefore covers all the reachable information. Based on the bidirectional path label index, a label-constraint reachability query $s \rightarrow_L t$ can be answered by comparing the labels of the two query nodes s and t, and the traversal operation on the graph can be completely avoided during query processing. Finally, we conduct rich experiment study. In terms of index size, index construction time and query response time, the experimental results on multiple real datasets show that our approaches achieve smaller index size, and can construct the index and answer label-constraint reachability queries more efficiently compared with the state-of-the-art approaches.

Key words graph data management; directed graph; reachability query processing; label-constraint reachability; bidirectional path label index

摘 要 基于标签约束的可达性查询 $s \rightarrow_L t$ 用于回答给定图中顶点 s 到顶点 t 是否存在路径标签属于 L 的有向路径.针对现有方法索引构建时间长、索引规模大、查询效率低的问题,首先基于 k 个点构建双向路径标签索引,并提出相应的优化措施减小索引规模,以此来加速可达查询的处理速度.由于其索引

收稿日期:2019-08-19;**修回日期**:2020-02-13
基金项目:国家重点研发计划项目(2017YFB0309800);国家自然科学基金项目(61472339,61572421,61873337)
This work was supported by the National Key Research and Development Program of China (2017YFB0309800) and the National Natural Science Foundation of China (61472339, 61572421, 61873337).
通信作者:周军锋(zhoujf@dhu.edu.cn)

没有完全覆盖可达查询,虽然索引规模小,但仍然无法避免查询过程中的图遍历操作.为此,进一步提出覆盖所有可达信息的双向路径标签索引,基于该索引,查询处理时可以完全避免图上的遍历操作.最后,基于多个真实数据集进行测试,实验结果从索引大小、索引构建时间和查询响应时间方面验证了所提方法相对现有方法具有索引规模小、索引时间短且查询响应快的优势.

关键词 图数据管理;有向图;可达性查询处理;标签约束可达性;双向路径标签索引

中图法分类号 TP311

可达性查询是数据图上的基本操作之一,用于回答有向图中指定的2个顶点间是否存在有向路径,在过去一段时间得到了深入研究[1-16].可达性查询可用于社交网络、生物网络、语义网络、交通网络等检测指定的两点间是否存在某种特定联系,也可作为基本的原子操作,协助回答图上的结构化查询,如XQuery或者SPARQL(SPARQL protocol and RDF query language)表达式.随着实际应用中图规模不断增大和查询约束条件的增加,可达性查询的处理方法也需要支持更多的查询约束条件.例如,在实际的数据图中,每条边除了表示两点之间直接可达的关系外,还常常附带有表示关系特征的文本标签.用户在查询时除了关心两点之间是否可达外,可能想更进一步知道两点之间可达路径上的文本标签是否满足指定的标签约束[17-22].类似的查询有:从哈尔滨出发仅通过高速路是否能到达广州,用户A和用户B的联系是否与工程或者学术相关,用户A和用户B是否存在师生或者血缘上的关联等.这些问题被称为标签约束可达查询的处理问题.

为了回答标签约束的可达性查询,研究者提出了多种索引技术来加速查询响应的速度[17-22],其中最新工作是文献[21]提出的基于地标索引的方法.该方法选择k个地标点,然后建立2种索引:1)每个地标点的可达点及其最小路径标签集;2)非地标点的部分可达点及其路径标签集.查询处理时,如果索引可回答查询,则返回结果;否则需要通过图遍历操作得到结果.实际中,该方法建立的索引的可达信息覆盖率较低,需要执行昂贵的图遍历操作.

针对以上问题,本文首先提出一种基于部分点的双向路径标签索引方法,不仅显著提升可达信息的覆盖率,而且索引规模显著降低.该方法的提出基于以下观察:假设顶点v可达的顶点集是N,M是可达v的顶点集,则基于本文的方法,处理v记录的可达顶点对数量是$|M|\times|N|+|M|+|N|$,而文献[21]构建的是单向索引,仅记录$|N|$个可达顶点对信息.基于本文的双向索引处理查询可显著减少图遍历的代价,提升系统性能.其次,本文提出一种基于标签集合包含关系的压缩优化策略,进一步减少索引规模.在此基础上,本文提出一种完全索引技术,用以记录所有顶点对的可达信息,从而在查询处理时完全避免图遍历操作.最后,基于真实数据集进行对比实验,实验结果证明了本文所提方法的高效性.

1 背景知识及相关工作

1.1 相关概念

给定有向无环标签图$G=(V,E,\mathcal{L},\lambda)$,$V$表示$G$中的顶点集合,$E\subseteq V\times V\times\mathcal{L}$表示$G$中有向边的集合,$\mathcal{L}$表示$G$中边标签的集合,$\lambda$函数定义映射$E\to\mathcal{L}$.从顶点$s$到顶点$t(s,t\in V)$存在1条标签为$l$的有向边,将其记为$(s,t,l)\in E$.

本文处理的是有向无环标签图,对于有环的情况,可通过借鉴文献[20]的方法进行处理,将其转换为无环图.一个有向无环图中的有向边体现了顶点间的可达偏序关系,通过拓扑排序可将顶点的偏序关系全序化,拓扑号即为全序关系的序号.拓扑排序过程中,对于每个被处理顶点s建立1个拓扑号$X(s)$,则对图中任意从s到t的有向边(s,t)来说,有$X(s)<X(t)$.根据可达的传递特性,对s可达的任意顶点t来说,同样存在$X(s)<X(t)$.因此,若$X(s)>X(t)$成立,则s不可达t.

定义1. 路径标签.给定$G=(V,E,\mathcal{L},\lambda)$中的任意2个顶点$(s,t\in V)$,假设从顶点$s$到顶点$t$有1条路径$p:s\xrightarrow{e_1}v_1\xrightarrow{e_2}v_2\cdots v_{p-1}\xrightarrow{e_p}t$,则该路径上的标签可表示为$L_p=L_{\langle s,v_1,\cdots,v_{p-1},t\rangle}=\bigcup\limits_{e\in p}\lambda(e)$,$s$到$t$的路径标签集表示为$L(s,t)=\{L_p\mid p$为从$s$到$t$的路径$\}$.

定义2. 标签约束可达查询.给定图G中的2个顶点s,t以及标签集L,标签约束可达查询就是确定从s到t是否存在有向路径p满足$L_p\subseteq L$,用

$s\rightarrow_L t$ 表示.如果 s 到 t 存在一条满足标签集 L 的路径,记作$s\rightarrow_L t$,查询返回 true,否则记作 $s\not\rightarrow_L t$,查询返回 false.

假设 $L_p=\{a_1,a_2,\cdots,a_n\}$,后续讨论中,为了简化表示,在上下文允许的情况下,本文以字符串的形式表示 L_p,即 $L_p=a_1a_2\cdots a_n$.假设 $L_{p_i}=a_1a_2\cdots a_i$ 且 $L_{p_i}\subseteq L_p$,则该关系也可在上下文允许的情况下表示为 $a_1a_2\cdots a_i\subseteq a_1a_2\cdots a_n$.表 1 列出了本文经常使用的符号及其意义.

Table 1　The Symbols Used in This Paper and Description

表 1　本文所用符号及其意义

Symbols	Description
$G=(V,E,\mathcal{L},\lambda)$	Labeled directed acyclic graph
$children(v)$	Outer node set of node v
$d_{\text{in}}(v),d_{\text{out}}(v)$	Indegree and outdegree of node v
L_p	Path label of p
$L(u,v)$	Path label set from node u to node v
$inLabel(v)$	$inLabel(v)=\{(u,L)\mid L=L(u,v)\}$
$outLabel(v)$	$outLabel(v)=\{(w,L)\mid L=L(v,w)\}$
$X(v),Y(v)$	Two topological numbers of node v

1.2　相关工作

1.2.1　无约束的可达性查询

该类查询主要用于判断从源点到目标点是否存在路径.在最新的研究中[14-16],文献[14]讨论了2hop label 的并行构建策略,主要用于回答最短路径和一般无约束的可达查询.文献[15]讨论了大图上的近似可达查询处理问题,并提供了近似解决方案.文献[16]改进了 2hop label 以便加速时态图上的最快路径查询.以上文献均无法用来回答标签约束的可达查询,和本文方法没有可比性.

1.2.2　对点和边都进行标签约束的可达性查询

该类查询对路径上的点和边都进行约束,并且点和边具有多个标签,主要用于判断从源点到目标点是否存在路径,并且路径上所有点和边的标签都满足给定的标签约束.针对该类查询,文献[17]中提出在二级存储框架中开发,将图形拓扑存储在主存储中,标签信息存储在辅助存储中.Yung 等人[17]提出一种基于散列映射的算法,其基本思想是利用"完美"散列函数将顶点和边的多个标签映射成散列值,并将散列值存储在主存储器中.查询时,从源点出发进行广度优先遍历(breadth first search, BFS)/深度优先遍历(depth first search, DFS),遍历的过程中验证散列值.对于可能存在散列冲突的顶点或边,则需访问辅助存储器中存储的标签,通过比对标签决定是否继续遍历该分支.最终筛选满足约束的点和边进行遍历,直至到达终点返回可达或者中途就停止了遍历返回不可达.

1.2.3　仅对边进行标签约束的可达性查询

该类查询主要用于判断给定 2 个顶点是否存在有向路径,且路径中每条边的标签仅使用预定义标签集中的标签.针对该类查询,研究者们提出多种不同的解决方法,主要有 4 种:

1) 基于生成树索引方法

文献[19]提出一个基于树状结构的索引框架.首先定义了 2 个顶点间的最小路径标签集,然后利用有向最大权生成树算法和采样技术为图 G 生成一棵有向生成树 T,同时对非树边建立一个包含标签集压缩的部分传递闭包 NT.基于 T,图中的所有路径被划分为 3 组 $P_{\text{s}},P_{\text{e}},P_{\text{n}}$.$P_{\text{s}}$ 包含所有成对路径,其中第 1 条边是 T 的一部分;P_{e} 包含所有成对路径,其中最后 1 条边是 T 的一部分;P_{n} 包含所有成对路径,其中第 1 条和最后 1 条边都不在 T 中.$NT(u,v)$包含 u 和 v 之间的 P_{n} 中路径的所有路径标签.实际上,该方法是在无约束可达查询的树状索引基础上,增加了标签集约束实现的,因此该算法存在很多冗余路径并且具有树状索引对稠密图处理效率较低问题.

2) 基于二分图索引方法

文献[20]提出利用强连通分量(strong connected component, SCC)将图转换为二分图来实现标签约束可达性查询.首先将图 G 分解为强连通分量 C_1,$C_2,\cdots,C_k$,并计算每个 C_i 的局部传递闭包,即 C_i 中每个顶点到其他点的最短路径标签集.其次,用二分图 B_i 替代每个 C_i,将其转化成带有标签的增强有向无环图(augmented directed acyclic graph, ADAG) $G_{\text{a,d,a}}$,并根据逆拓扑排序的顺序依次计算 $G_{\text{a,d,a}}$ 中的每个顶点 u 的最短路径标签集.最后,计算每个 u ($u\in C_i,u\notin G_{\text{a,d,a}}$)的最短路径标签集.该方法得到的索引保存了图中每个顶点的完整可达性信息,即标签图的完全传递闭包.因此查询应答相当于索引中的简单查找,但索引构建时间过长,索引规模过大.

3) 基于地标点索引方法

文献[21]提出选择地标点建立索引,并维护其传递闭包中顶点可达信息的地标索引(landmark index, LI^+)方法.具体来说,将图中顶点按出度与入度总和降序排列,并选择前 k 个点作为地标顶点.对于地标点,计算其到其他点的最小路径标签集,并利用辅助索引重新组织部分地标索引;对于非地标点,

计算部分路径标签集.查询时,如果该条查询的源点是地标点,则查找地标索引进行回答,否则查找非地标索引.若也不能通过非地标点索引回答,则执行遍历,遍历过程中可以通过辅助索引进行剪枝.该方法的主要缺点是索引的可达信息覆盖率低,图遍历代价高.

4) 基于随机游走的采样算法

文献[22]基于随机游走的采样算法来回答具有全范围正则表达式约束的可达性查询,且可扩展到大图,但该算法提供的是非精确解的近似解决方案,和本文方法不具有可比性.

2 基于部分点的双向索引

文献[21]通过构建 k 个地标点的单向可达索引来回答查询.假设顶点 v 可达的顶点集是 N,则 1 个地标点对应的索引仅能回答 v 与 $|N|$ 个点的可达关系.由于 k 值与图中顶点数量相比较小,通常情况下文献[21]提出的单向地标索引的可达信息覆盖率低,查询过程中不可避免需要执行大量昂贵的遍历操作.针对该问题,本文提出通过构建 v 的双向路径标签索引来提升索引的可达信息覆盖率.假设 M 是可达 v 的顶点集,则基于本文的方法,处理 v 记录的可达顶点对数量是 $|M|\times|N|+|M|+|N|$,其中 M 中的顶点通过 v 可达 N 中顶点,且可达顶点对数量为 $|M|\times|N|$,$|M|$ 表示 M 中顶点和 v 的可达顶点对数量,$|N|$ 表示 v 和 N 中顶点的可达顶点对数量.下面具体介绍该索引的构建方法.

2.1 索引构建

2.1.1 基本思想

本文为每个点 v 设定 2 个标签,分别是 $inLabel(v)$ 和 $outLabel(v)$.其中,$inLabel(v)=\{(u,L)\mid L=L(u,v)\}$,用于记录可达 v 的顶点及该顶点到达 v 的路径标签;$outLabel(v)=\{(w,L)\mid L=L(v,w)\}$,用于记录 v 可达的顶点及 v 到达该顶点的路径标签.给定查询 $s\nrightarrow_L t$,可通过求解 $outLabel(s)$ 和 $inLabel(t)$ 的顶点交集来判断 s 是否可达 t.如果交集为空,说明 s 不可达 t;否则,需要进一步检测路径标签是否满足要求,具体检测方法是对于交集中的每个顶点 v,将对应的路径标签进行合并,并判断合并的结果是否为 L 的子集.若是,则说明标签可达,否则还需进一步判断.

例 1. 针对图 1,假设顶点的处理顺序是 $v_2\rightarrow v_4\rightarrow v_3\rightarrow v_5\rightarrow v_1\rightarrow v_0$,根据以上思路建立的索引如表 2 所示.显然,这种索引虽然可回答所有查询,但存在很多冗余路径,索引规模过大.例如,$inLabel(v_4)$ 中同时包含 (v_0,c),(v_0,ac),(v_0,bc).当给定查询 $v_0\rightarrow_b v_5$,$outLabel(v_0)$ 与 $inLabel(v_5)$ 的交集为 $\{v_0,v_1,v_2,v_3,v_5\}$,将交集顶点对应的标签取并集.对于顶点 v_0:$\varnothing\cup ab=ab$,$\varnothing\cup ac=ac$,$\varnothing\cup abc=abc$;对于顶点 v_1:$c\cup a=ac$,$c\cup ab=abc$;对于顶点 v_2:$b\cup a=ab$,$ac\cup a=ac$;对于顶点 v_3:$ac\cup b=abc$;对于顶点 v_5:$ab\cup\varnothing=ab$,$ac\cup\varnothing=ac$,$abc\cup\varnothing=abc$.这些并集标签均不是查询给定约束标签 b 的子集,因此 $v_0\nrightarrow_b v_5$.显然,如果查询不可达,处理效率低.

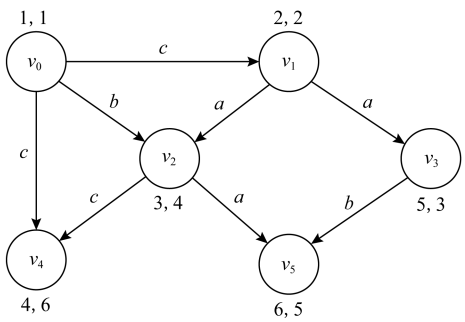


Fig. 1 Labeled DAG G

图 1 有向无环标签图 G

Table 2 The Initial Path Label Index for Fig. 1

表 2 初始路径标签索引

Node	*inLabel*	*outLabel*
v_0	$\{(v_0,\varnothing)\}$	$\{(v_0,\varnothing),(v_1,c),(v_2,b),(v_2,ac),(v_3,ac),(v_4,c),(v_4,ac),(v_4,bc),(v_5,ab),(v_5,ac),(v_5,abc)\}$
v_1	$\{(v_0,c),(v_1,\varnothing)\}$	$\{(v_1,\varnothing),(v_2,a),(v_3,a),(v_4,ac),(v_5,a),(v_5,ab)\}$
v_2	$\{(v_0,b),(v_0,ac),(v_1,a),(v_2,\varnothing)\}$	$\{(v_2,\varnothing),(v_4,c),(v_5,a)\}$
v_3	$\{(v_0,ac),(v_1,a),(v_3,\varnothing)\}$	$\{(v_3,\varnothing),(v_5,b)\}$
v_4	$\{(v_0,c),(v_0,ac),(v_0,bc),(v_1,ac),(v_2,c),(v_4,\varnothing)\}$	$\{(v_4,\varnothing)\}$
v_5	$\{(v_0,ab),(v_0,ac),(v_0,abc),(v_1,a),(v_1,ab),(v_2,a),(v_3,b),(v_5,\varnothing)\}$	$\{(v_5,\varnothing)\}$

Note: $(v,\{L_{p1},L_{p2},\cdots,L_{pn}\})$ is organized into $\{(v,L_{p1}),(v,L_{p2}),\cdots,(v,L_{pn})\}$.

针对以上问题,本文仅构建基于 k 个顶点的标签索引来减小索引规模,同时提出 2 种优化方法来降低索引规模.下面具体介绍.

定理 1. 给定顶点 v,在基于 v 建立标签索引的过程中,如果遇到已处理过的地标点 u,则无需从 u 继续遍历.

证明. 首先,在处理 u 的过程中,已经记录了所有 u 到 u 可达顶点的路径标签以及可达 u 的顶点到 u 之间的路径标签.其次,在处理 v 时,如果遍历过程中遇到了 u,一方面,在处理 u 时已经得到 v 和 u 之间路径的标签,另一方面,如图 2 所示,对 u 可达的任意顶点 x,处理 u 时已经得到了两者之间的路径标签.处理 v 时,如果沿着 u 继续遍历到 x,则得到的标签等价于 u 到 x 已经得到的标签与 v 到 u 之间路径标签的并集.由于这些信息已经具备,因此无需专门记录 v 到 x 的路径标签.　　证毕.

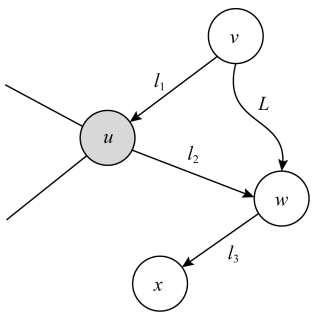


Fig. 2　Illustration of landmark nodes

图 2　地标点作用示意图

定理 2. 给定顶点 v,在基于 v 建立标签索引的过程中,对于遍历到的顶点 w 满足:1)v 可达 w,路径标签是 L_1;2)v 和 w 之间通过遍历非地标点得到的路径标签是 L_2.如果 $L_2 \supseteq L_1$,则无需从 w 出发继续遍历.

证明. 如图 2 所示,从 v 出发建立标签索引.当遍历到 w 时,发现 v 可达 w,其路径标签是 $L_1 = l_1 \cup l_2$,且 v 通过非地标点遍历到 w 的路径标签是 L,如果 $L_1 \subseteq L$ 成立,说明对于 w 之后遍历到的任一点 x,v 通过直接遍历得到的路径标签是 $L \cup l_3$,而通过 u 计算得到的路径标签是 $L_1 \cup l_3$,可知通过非地标点得到的标签始终是通过地标点的超集,因此无需从 w 继续遍历.　　证毕.

针对图 1,假设选择 2 个地标点 v_2 和 v_1,处理顺序是先 v_2 后 v_1,则基于上述 2 种优化方法构建的路径标签索引如表 3 所示.例如,处理完 v_2 再处理 v_1 时,通过 BFS 遍历到 v_2,由于 v_2 已被处理过,由定理 1 可得,不更新索引且停止该条分支的遍历,并开始下一条分支的遍历.当 BFS 通过路径 $\langle v_1, v_3, v_5\rangle$ 遍历到 v_5 时,其路径标签 $L_{\langle v_1,v_3,v_5\rangle} = ab$.由已构建的索引可知,$v_1$ 通过地标点 v_2 可达 v_5,对应的路径标签为 $L_{\langle v_1,v_2,v_5\rangle} = a$.因为 $L_{\langle v_1,v_3,v_5\rangle} \supseteq L_{\langle v_1,v_2,v_5\rangle}$,由定理 2 得,不更新索引且无需从 v_5 继续遍历.

Table 3　Optimized Path Label Index

表 3　优化的路径标签索引

Node	*inLabel*	*outLabel*
v_0		$\{(v_1,c),(v_2,b),(v_2,ac)\}$
v_1	$\{(v_1,\varnothing)\}$	$\{(v_1,\varnothing),(v_2,a)\}$
v_2	$\{(v_2,\varnothing)\}$	$\{(v_2,\varnothing)\}$
v_3	$\{(v_1,a)\}$	
v_4	$\{(v_2,c)\}$	
v_5	$\{(v_2,a)\}$	

2.1.2　索引构建算法

算法 1 基于以上 2 个定理构建双向路径标签索引(bidirectional path label index, BPLI).

算法 1. *genBPLI*(G).

输入:$G=(V,E,\mathcal{L},\lambda)$;

输出:k 个顶点的双向路径标签索引.

① 根据$(d_{\text{out}}(v)+1)\times(d_{\text{in}}(v)+1)$对顶点降序排序,并选择前 k 个顶点;

② for each node v do

③　*BiBFS*(v);

④ end for

Function *BiBFS*(s)

/* 执行正向 BFS 求路径标签 */

① $(s,\varnothing)$入队列 Q;

② *inLabel*(s),*outLabel*(s)分别初始化为$(s,\varnothing)$;

③ while $Q\neq\varnothing$ do

④　队列 Q 的队头元素出队并记作(v,L);

⑤　if *isSuperset*(s,v,L)=true then

⑥　　continue;

⑦　end if

⑧　*inLabel*(v)中插入(s,L);

⑨　for each $w\in children(v)$ do

⑩　　if *marked*(w)=1 then

⑪　　　continue;

⑫　　end if

⑬　　$(w,L\cup l)$入队列 Q;

⑭ end for
⑮ end while
⑯ 基于行①～⑪执行反向 BFS 求路径标签;
⑰ $marked(s)\leftarrow 1$;
end Function
Function $isSuperset(s,t,L)$
① if s can reach t then
② 由已经构建的索引计算得到顶点 s 到顶点 t 的路径标签集为 L';
③ if $L'\subseteq L$ then
④ return true;
⑤ end if
⑥ end if
⑦ return false;
end Function

本文采用文献[23]中的策略,即按照$(d_{out}(v)+1)\times(d_{in}(v)+1)$递减的顺序对图中顶点排序,选择前 k 个地标点(行①).对每个地标点 v,调用 $BiBFS(v)$ 从正反 2 个方向进行遍历,分别建立被访问顶点的 $inLabel$ 和 $outLabel$(行②③).在基于地标点构建路径标签索引的过程中,同时应用定理 1 和定理 2 进行剪枝来减小索引规模.对于每个遍历到的点,函数 $isSuperset(s,t,L)$ 用于根据定理 2 以及定理 1 判断是否还继续遍历该分支.

算法 1 的时间复杂度是 $O(|V|+|E|)$,空间复杂度是 $O(|V|)$.因为一个顶点进行 BFS 操作的最坏代价是 $O(|V|+|E|)$,而算法 1 仅仅处理了 k 个顶点(k 是自定义的常数),因此,时间复杂度是 $O(|V|+|E|)$.此外,BFS 操作需要借助一个辅助队列,总的大小为 $O(|V|)$,又每个顶点包含 2 个拓扑号,因此总的空间复杂度是 $O(|V|)$.

例 2. 对于图 1,根据$(d_{out}(v)+1)\times(d_{in}(v)+1)$对图中顶点降序排列,得到的顶点顺序为:$v_2\rightarrow v_1\rightarrow v_0\rightarrow v_3\rightarrow v_4\rightarrow v_5$.假设选择 2 个地标点,则地标点的处理顺序是 $v_2\rightarrow v_1$.

1) 在处理 v_2 时.初始化 $inLabel(v_2)=\{(v_2,\varnothing)\}$和 $outLabel(v_2)=\{(v_2,\varnothing)\}$.从 v_2 出发执行 BFS 遍历到 v_4 和 v_5,由于 $marked(v_4)=0$,需判断 $outLabel(v_2)$与 $inLabel(v_4)$是否存在顶点交集.发现 $outLabel(v_2)$与 $inLabel(v_4)$没有交集,因此将(v_2,c)插入到 $inLabel(v_4)$中.同理,将(v_2,a)插入到 $inLabel(v_5)$中.此时 v_2 已没有出邻居,从 v_2 出发执行反向 BFS 遍历到 v_0 和 v_1.因为 v_0 未被标记且 $outLabel(v_0)$与 $inLabel(v_2)$没有交集,将(v_2,b)插入到 $outLabel(v_0)$中.同理,将(v_2,a)插入到 $outLabel(v_1)$中.继续反向 BFS 遍历到 v_0,由已构建的 $outLabel(v_0)$和 $inLabel(v_2)$可得 $L_{\langle v_0,v_2\rangle}=b$,而遍历路径$\langle v_2,v_1,v_0\rangle$的标签为 $L_{\langle v_2,v_1,v_0\rangle}=ac$.根据集合关系将$(v_2,ac)$插入到 $outLabel(v_0)$中.至此,v_2 正反 2 个方向的遍历结束,将其标记为 $marked(v_2)=1$.

2) 在处理 v_1 时.从 v_1 出发执行 BFS 遍历到 v_2,由于 v_2 已被标记过,由定理 1 得,不更新索引并停止该分支的遍历,开始下一分支的遍历,即遍历到 v_3.因为 $outLabel(v_1)$和 $inLabel(v_3)$没有交集,将(v_1,a)插入到 $inLabel(v_3)$中.继续执行 BFS 遍历,经过路径$\langle v_1,v_3,v_5\rangle$遍历到 v_5,对应的路径标签 $L_{\langle v_1,v_3,v_5\rangle}=ab$,由已构建的索引知 $outLabel(v_1)$与 $inLabel(v_5)$的顶点交集为 v_2,对应的标签取并集可得 $L_{\langle v_1,v_2,v_5\rangle}=a$.又 $L_{\langle v_1,v_3,v_5\rangle}\supseteq L_{\langle v_1,v_2,v_5\rangle}$,由定理 2 得,不更新索引且无需从 v_5 继续遍历.接下来,从 v_1 出发执行反向 BFS 遍历到 v_0,因为 v_0 未被标记且 $outLabel(v_0)$与 $inLabel(v_1)$没有交集,将(v_1,c)插入到 $outLabel(v_0)$中.至此,v_1 正反 2 个方向的遍历结束并标记 $marked(v_1)=1$.因此,针对图 1 建立的索引如表 3 所示.

2.2 查询策略

给定查询 $s\rightarrow_L t$,可通过求解 $outLabel(s)$和 $inLabel(t)$的交集来判断 s 是否可达 t.如果交集为空,说明 s 通过地标点不可达 t,需要进行遍历判断是否可达;如果交集不为空,说明 s 可达 t,需要进一步检测路径标签是否满足要求,具体检测方法是对于交集中的每个点 v,将 $L(s,v)$与 $L(v,t)$进行合并,判断合并的结果是否是 L 的子集.若是,则说明标签可达,否则需要进行遍历.

例如,当给定查询 $v_1\rightarrow_{abc}v_5$ 时,$outLabel(v_1)$与 $inLabel(v_5)$的顶点交集为 v_2,对应的路径标签 $L_{\langle v_1,v_2,v_5\rangle}=a\subseteq abc$,所以 $v_1\rightarrow_{abc}v_5$.当给定查询 $v_3\rightarrow_{ac}v_4$ 时,$outLabel(v_3)$与 $inLabel(v_4)$没有交集,不能直接通过索引判断,需要执行遍历得出 $v_3\not\rightarrow_{ac}v_4$.

可以看出,基于部分地标点构建的路径索引不能高效处理不可达查询,为此本文进一步利用文献[13]中提出的基于栈的互逆拓扑顺序方法来快速处理不可达查询.简而言之,得出拓扑顺序 X 后,在构建逆序拓扑顺序 Y 时,始终优先访问拓扑顺序 X 中拓扑号最大且所有入邻居均被访问的顶点.具体操作步骤见文献[13],此处不再赘述.

由文献[13]可知,求解 2 个互逆拓扑顺序的时间代价为 $O(|V|+|E|)$,且后文中索引大小包含了拓扑号的存储开销.

算法 2. $BPLI(s,t,L)$.

输入:$s\rightarrow_L t$;

输出:true 或者 false.

① if $s=t$ then

② 　return true;

③ end if

④ if $L=\varnothing$ then

⑤ 　return false;

⑥ end if

⑦ if $X(s)>X(t)\ \|\ Y(s)>Y(t)$ then

⑧ 　return false;

⑨ end if

⑩ if $isSuperset(s,t,L)$=true then

⑪ 　return true;

⑫ end if

⑬ for each $v\in children(s)$ do

⑭ 　$BPLI(v,t,L)$;

⑮ end for

给定查询 $s\rightarrow_L t$ 时,首先利用 2 个拓扑序号 X 和 Y 过滤不可达的查询.如果 $X(s)>X(t)$或 $Y(s)>Y(t)$,那么 $s\nrightarrow t$(行⑦～⑨);否则,根据双向路径标签索引来判断标签是否满足约束(行⑩～⑫).如果不能通过索引判断则需要遍历,并在遍历的过程中根据拓扑序号及标签做进一步判断(行⑬～⑮).具体的查询处理过程如算法 2 所示.

例 3. 根据文献[13]中提出的基于栈的互逆拓扑顺序方法,为图 1 中每个顶点建立2 个拓扑序号,如顶点旁的数字所示.其中,左边数字代表 X,右边数字代表 Y.给定查询 $v_3\rightarrow_{bc}v_2$ 时,$X(v_3)>X(v_2)$,即 $v_3\nrightarrow v_2$,则 $v_3\nrightarrow_{bc}v_2$.给定查询 $v_1\rightarrow_{acd}v_5$ 时,$X(v_1)<X(v_5)$,$Y(v_1)<Y(v_5)$,则 v_1 不一定可达 v_5;又因为 $outLabel(v_1)$与 $inLabel(v_5)$的交集为 v_2,对应的标签取并集得 $L_{\langle v_1,v_2,v_5\rangle}=a\subseteq acd$,所以 $v_1\rightarrow_{acd}v_5$.

3　基于所有点的双向索引

由于算法 1 选择 k 个顶点建立索引,对某些数据集而言,可能存在索引覆盖率低的问题,并且当索引判断不了时还需要执行遍历操作.基于第 2 节给出的定理 1 和定理 2,本文进一步提出对所有点建立双向路径标签索引.由于这种索引可用于回答不可达查询,因此基于所有点建立路径标签索引时,不再需要拓扑序号进行过滤,且无需执行图上的遍历操作.

基于第 2 节构建路径标签的思路为所有点构建的路径标签索引如表 4 所示.查询算法如算法 3 所示.

Table 4　Path Label Index for All Nodes

表 4　所有点的路径标签索引

Node	*inLabel*	*outLabel*
v_0	$\{(v_0,\varnothing)\}$	$\{(v_0,\varnothing),(v_1,c)$ $(v_2,b),(v_2,ac)\}$
v_1	$\{(v_1,\varnothing)\}$	$\{(v_1,\varnothing),(v_2,a)\}$
v_2	$\{(v_2,\varnothing)\}$	$\{(v_2,\varnothing)\}$
v_3	$\{(v_1,a),(v_3,\varnothing)\}$	$\{(v_3,\varnothing)\}$
v_4	$\{(v_0,c),(v_2,c),(v_4,\varnothing)\}$	$\{(v_4,\varnothing)\}$
v_5	$\{(v_2,a),(v_3,b),(v_5,\varnothing)\}$	$\{(v_5,\varnothing)\}$

该方法的时间复杂度为 $O(|V|\times(|V|+|E|)\times|L_{\max}|)$,空间复杂度为 $O(|V|\times|L_{\max}|)$,这里$|L_{\max}|$表示所有顶点中最长的标签长度.虽然看起来时间复杂度较高,但实验结果显示,由于剪枝技术的有效性,索引时间比现有方法更快.

算法 3. $BPLI^{+}(s,t,L)$.

输入:$s\rightarrow_L t$;

输出:true 或者 false.

① if $s=t$ then

② 　return true;

③ end if

④ if $L=\varnothing$ then

⑤ 　return false;

⑥ end if

⑦ if $isSuperset(s,t,L)$=true then

⑧ 　return true;

⑨ end if

⑩ return false.

例如,给定查询 $v_0\rightarrow_{abc}v_3$,由 $outLabel(v_0)$与 $inLabel(v_3)$可得 $L_{\langle v_0,v_1,v_3\rangle}=c\cup a=ac\subseteq abc$,因此 $v_0\rightarrow_{abc}v_3$.当查询 $v_4\rightarrow_{bc}v_3$ 时,$outLabel(v_4)$和 $inLabel(v_3)$不存在顶点交集,所以 $v_4\nrightarrow_{bc}v_3$.

4　实　　验

4.1　实验环境

本文实验所用机器的基本配置为 Intel® Core™

i5-6600 CPU@3.30 GHz,4 GB 内存,1 TB 硬盘,操作系统为 Linux Ubuntu 16.04.用于比较的算法包括基于遍历的 BFS 算法、LI$^+$ 算法[21]以及本文提出的 2 种算法.所有算法均采用 C++语言实现,通过 G++7.3.0 编译器进行编译.所有算法的源代码由本文作者实现.

4.2 数据集

本文所用数据集为 23 个真实数据集,相关统计信息如表 5 所示.其中,d 表示平均出度,约定 $d \geqslant 4$ 时为稠密图,$|V| \geqslant 100\,000$ 时为大图.

Table 5 The Information of Dataset Statistics

表 5 数据集统计信息

Dataset	$\|V\|$	$\|E\|$	d
Amaze[10]	3 710	3 600	0.97
Human①	38 811	39 576	1.02
Anthra①	12 499	13 104	1.05
Ecoo①	12 620	13 350	1.06
Vchocyc①	9 491	10 143	1.07
Kegg[10]	3 617	3 908	1.08
Xmark[7]	6 080	7 025	1.16
Mtbrv①	9 602	10 245	1.07
Nasa[7]	5 605	6 537	1.17
Go②	6 793	13 361	1.97
Citeseer③	9 000	40 028	4.45
Pubmed④	10 720	44 258	4.13
Yago⑤	6 642	42 392	6.38
Email-EuAll⑥	231 000	223 004	0.97
WikiTalk⑥	2 281 879	2 311 570	1.01
Soc-LiveJournal⑥	971 232	1 024 140	1.05
Web-Google⑥	371 764	517 805	1.39
10cit-Patent⑥	1 097 775	1 651 894	1.50
05cit-Patent⑥	1 671 488	3 303 789	1.98
Cit-Patents⑥	3 774 768	16 518 947	4.38
Dbpedia⑦	3 365 623	7 989 191	2.37
Uniprot22m⑧	1 595 444	1 595 442	1.00
10go-Uniprot⑧	469 526	3 476 397	7.40

4.3 查询集

为了比较不同算法对标签可达查询处理的整体性能,本文生成 100 万个可达查询和 100 万个不可达查询来进行测试,并且 $|L| \leqslant 8$.

针对上述的查询集,所有的运行时间都是执行 10 次 100 万个查询的总时间取平均值.所有结果均保留到小数点后 2 位.本文实验的评估指标包括:

1) 索引构建时间;

2) 索引大小;

3) 查询时间.

实验中,BPLI 算法的参数 $k=32$,LI$^+$ 算法的参数 $k=1\,250+\sqrt{|V|}$,该值为文献[21]建议的参数值.

4.4 实验结果

4.4.1 索引构建时间

图 3 比较了不同规模数据集上的索引构建时间.从图 3 可以看出,在索引构建时间方面,BPLI 和 BPLI$^+$ 优于 LI$^+$.对于规模较小的图,如 Kegg 数据集,BPLI 比 LI$^+$ 快 1372 倍,BPLI$^+$ 比 LI$^+$ 快 891 倍.在 Pubmed 数据集上,BPLI 比 LI$^+$ 快 21.6 倍,BPLI$^+$ 比 LI$^+$ 快 4.4 倍.在规模较大的图上,如 Email-EuAll 数据集,BPLI 比 LI$^+$ 快 1042 倍,BPLI$^+$ 比 LI$^+$ 快 623 倍.在 Cit-Patents 数据集上,BPLI 比 LI$^+$ 快 111 倍,BPLI$^+$ 比 LI$^+$ 快 46 倍.原因在于,LI$^+$ 除了构建地标点的所有单向索引和非地标点的部分单向索引外,还需要根据路径标签对地标点的索引进行重新组织,因此会耗费大量时间.

4.4.2 索引大小

图 4 展示了不同规模数据集上的索引大小比较.可以看出,由于稀疏小图中 LI$^+$ 索引的地标点个数几乎是顶点个数的一半,构建地标索引的传递闭包较大,如数据集 Amaze 的索引大小是本文方法的 200 多倍.对于大图来说,BPLI 仅选择了 32 个顶点建立索引,而 LI$^+$ 中选择了 $1\,250+\sqrt{|V|}$ 个地标点建立索引外还建立了非地标点的部分索引,因此,BPLI 的索引比 LI$^+$ 小.而 BPLI$^+$ 对图中的每个顶点都进行了剪枝操作,在小图上的索引规模比 LI$^+$ 小.在大图上,BPLI$^+$ 构建的是所有顶点的路径标签索引,但基于定理 1 和定理 2 进行了优化,总的索引规模增长并不明显,甚至在某些数据集上,索引规模远小于 LI$^+$.

① ecocyc.org
② www.geneontology.org
③ citeseerx.ist.psu.edu
④ pubmedcentral.nih.gov
⑤ mpi-inf.mpg.de/yago-naga/yago
⑥ snap.stanford.edu
⑦ pubmedcentral.nih.gov
⑧ www.uniprot.org

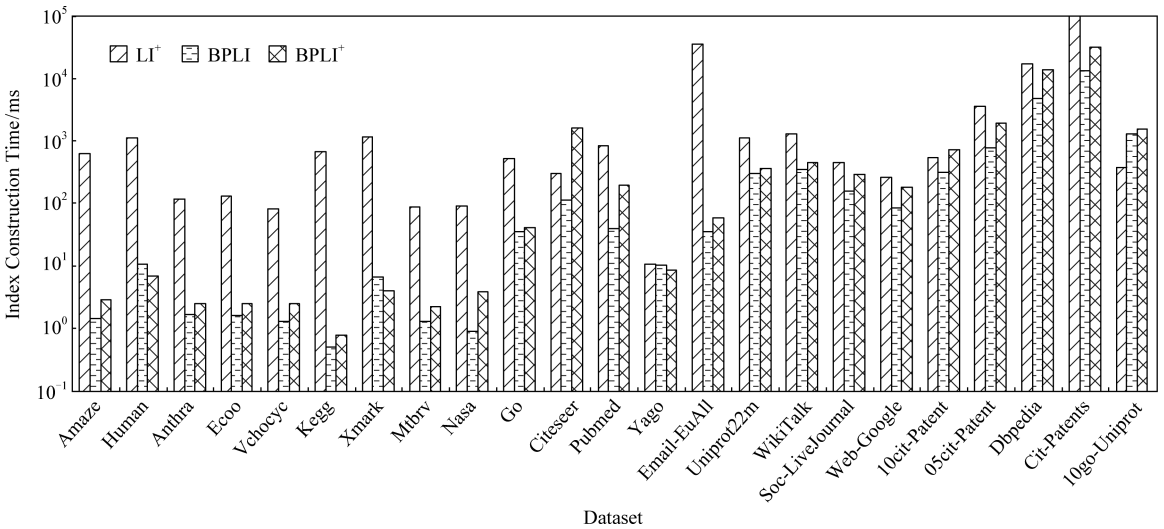


Fig. 3　Comparison of index construction time on different datasets

图 3　不同规模数据集上的索引构建时间对比

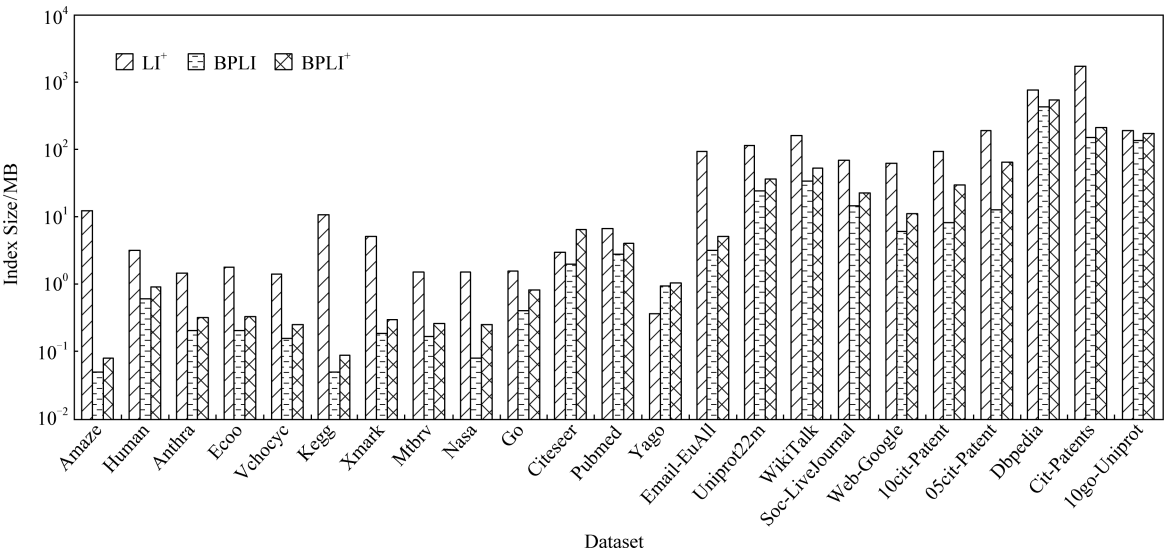


Fig. 4　Comparison of index size on different datasets

图 4　不同规模数据集上的索引大小对比

4.4.3　查询时间

图 5 和图 6 展示了不同算法分别处理 100 万不可达查询及 100 万可达查询的查询时间对比.当查询时间超过 10 000 ms 时,均显示 10 000 ms.本文有如下观察:

1) BPLI 方法与 BPLI$^+$ 方法处理可达查询以及不可达查询的效率均高于 LI$^+$.原因在于:BPLI 使用了拓扑号处理不可达查询,且路径标签索引的可达覆盖率高,因此查询响应时间较短;BPLI$^+$ 方法无需遍历操作,且索引规模不大,因此查询响应时间最短.

2) BFS 没有索引协助,因此查询响应时间最长.

5　结束语

针对现有方法在处理大图上标签约束可达性查询时索引构建时间长、索引规模大且查询响应慢的问题,本文提出了 2 种基于地标点的双向路径标签索引,并基于所提出的优化方法来减小索引规模.实验结果从索引构建时间、索引大小以及查询时间 3 方面验证了本文所提方法的高效性.例如,本文方法在 Email-EuAll 数据集上的索引规模比 LI$^+$ 降低了 96.67%,索引构建时间比 LI$^+$ 降低了 99%,可达查询数据集上查询时间比 LI$^+$ 提高了 17 倍,不可达查询数据集上查询时间比 LI$^+$ 提高了 26 倍.

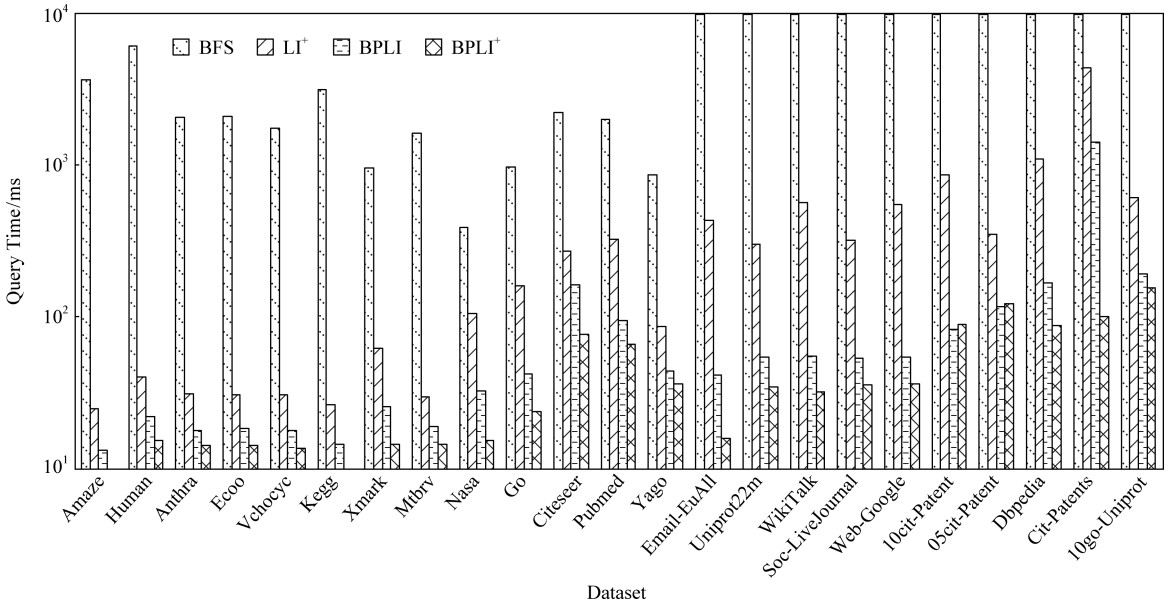


Fig. 5 Comparison of query time on unreachable query workload

图 5 不可达查询的查询时间对比

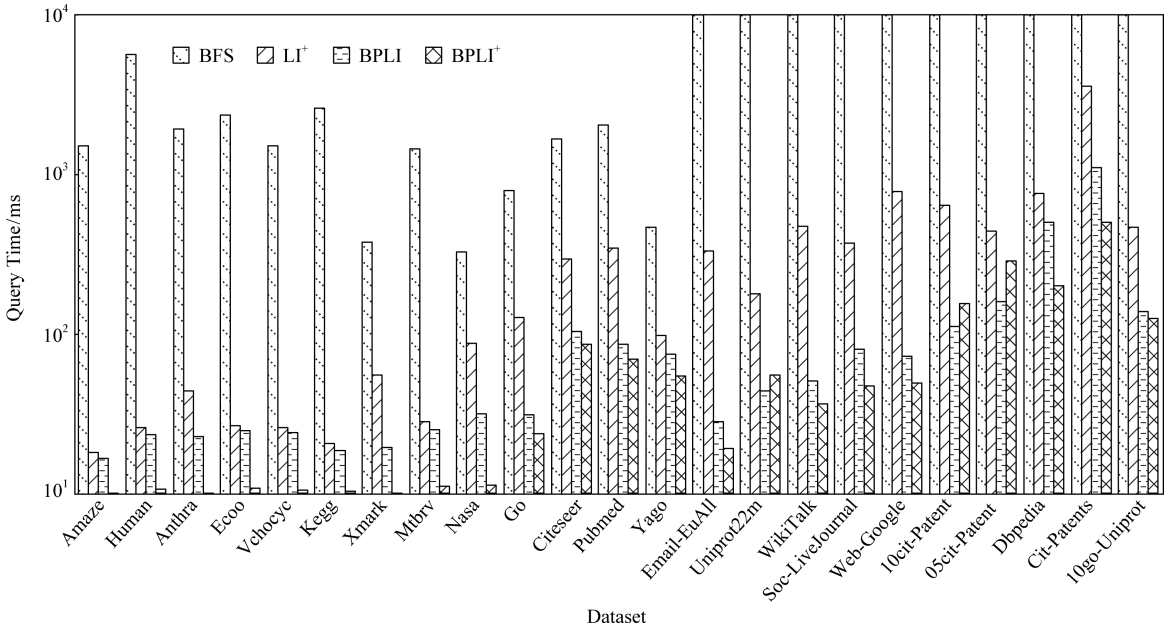


Fig. 6 Comparison of query time on reachable query workload

图 6 可达查询的查询时间对比

参 考 文 献

[1] Fu Lizhen, Meng Xiaofeng. Reachability indexing for largescale graphs: Studies and forecasts [J]. Journal of Computer Research and Development, 2015, 52(1): 116-129 (in Chinese)

(富丽贞, 孟小峰. 大规模图数据可达性索引技术: 现状与展望[J]. 计算机研究与发展, 2015, 52(1): 116-129)

[2] Cheng James, Huang Silu, Wu Huanhuan, et al. TF-Label: A topological-folding labeling scheme for reachability querying in a large graph [C] //Proc of the 2013 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2013: 193-204

[3] Yildirim H, Chaoji V, Zaki M J. GRAIL: A scalable index for reachability queries in very large graphs [J]. International Journal on Very Large Data Bases, 2012, 21(4): 509-534

[4] Chen Yangjun, Chen Yibin. An efficient algorithm for answering graph reachability queries [C] //Proc of the 24th Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2008: 893-902

[5] Zhu Linhong, Choi B, He Bingsheng, et al. A uniform framework for ad-hoc indexes to answer reachability queries on large graphs [C] //Proc of the 14th Int Conf on Database Systems for Advanced Applications. Berlin: Springer, 2009: 138-152

[6] Yano Y, Akiba T, Iwata Y, et al. Fast and scalable reachability queries on graphs by pruned labeling with landmarks and paths [C] //Proc of the 22nd ACM Int Conf on Information and Knowledge Management. New York: ACM, 2013: 1601-1606

[7] Jin Ruoming, Xiang Yang, Ruan Ning, et al. Efficiently answering reachability queries on very large directed graphs [C] //Proc of the 2008 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2008: 595-608

[8] Chen Yangjun, Chen Yibin. Decomposing DAGs into spanning trees: A new way to compress transitive closures [C] //Proc of the 27th Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2011: 1007-1018

[9] Jin Ruoming, Ruan Ning, Dey S, et al. SCARAB: Scaling reachability computation on large graphs [C] //Proc of the 2012 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2012: 169-180

[10] Trißl S, Leser U. Fast and practical indexing and querying of very large graphs [C] //Proc of the 2007 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2007: 845-856

[11] Yin Dan, Gao Hong, Zou Zhaonian, et al. Reachability query in heterogeneous information networks [J]. Journal of Computer Research and Development, 2016, 53(2): 479-491 (in Chinese)
(尹丹, 高宏, 邹兆年, 等. 异构信息网上的可达性查询[J]. 计算机研究与发展, 2016, 53(2): 479-491)

[12] Van S, Sebastiaan J, Oege D M. A memory efficient reachability data structure through bit vector compression [C] //Proc of the 2011 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2011: 913-924

[13] Chen Ziyang, Chen wei, Li Na, et al. Reachability query processing algorithm for large graphs [J]. Chinese Journal of Computers, 2019, 42(3): 582-595 (in Chinese)
(陈子阳, 陈伟, 李娜, 等. 面向大图的可达性查询处理算法[J]. 计算机学报, 2019, 42(3): 582-595)

[14] Li Wentao, Qiao Miao, Qin Lu, et al. Scaling distance labeling on small-world networks [C] //Proc of the 2019 Int Conf on Management of Data. New York: ACM, 2019: 1060-1077

[15] Sengupta N, Bagchi A, Ramanath M, et al. ARROW: Approximating reachability using random walks over web-scale graphs [C] //Proc of the 35th IEEE Int Conf on Data Engineering(ICDE). Piscataway, NJ: IEEE, 2019: 470-481

[16] Li Lei, Wang Sibo, Zhou Xiaofang. Time-dependent hop labeling on road network [C] //Proc of the 35th IEEE Int Conf on Data Engineering(ICDE). Piscataway, NJ: IEEE, 2019: 902-913

[17] Yung D, Chang Shikuo. Fast reachability query computation on big attributed graphs [C] //Proc of the IEEE Int Conf on Big Data. Los Alamitos, CA: IEEE Computer Society, 2016: 3370-3380

[18] Fan Weifei, Li Jianzhong, Ma Shuai, et al. Adding regular expressions to graph reachability and pattern queries [J]. Frontiers of Computer Science, 2012, 6(3): 313-338

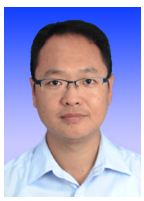
[19] Jin Ruoming, Hong Hui, Wang Haixun, et al. Computing label-constraint reachability in graph databases [C] //Proc of the 2010 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2010: 123-134

[20] Zou Lei, Xu Kun, Yu J X, et al. Efficient processing of label-constraint reachability queries in large graphs [J]. Information Systems, 2014, 40(1): 47-66

[21] Valstar L D J, Fletcher G H L, Yoshida Y. Landmark indexing for evaluation of label-constrained reachability queries [C] //Proc of the 2017 ACM Int Conf on Management of Data. New York: ACM, 2017: 345-358

[22] Wadhwa S, Prasad A, Ranu S, et al. Efficiently answering regular simple path queries on large labeled networks [C] //Proc of the 2019 ACMSIGMOD Int Conf on Management of Data. New York: ACM, 2019: 1463-1480

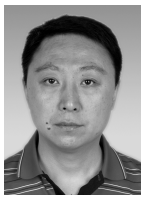
[23] Akiba T, Iwata Y, Yoshida Y. Fastexact shortest-path distance queries on large networks by pruned landmark labeling [C] //Proc of the 2013 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2013: 349-360



Du Ming, born in 1975. PhD, associate professor. His main research interests include information retrieval techniques, data analysis and mining, and natural language processing.

Yang Yun, born in 1995. Master. Her main research interests include query processing and optimization on graph data.

Zhou Junfeng, born in 1977. PhD, professor. His main research interests include information retrieval techniques, query processing on semi-structured data and graph data, and optimization.



Chen Ziyang, born in 1973. PhD, professor. His main research interests include computation theory, query processing and optimization on graph data.

Yang Anping, born in 1994. Master. His main research interests include query processing and optimization on graph data.

2018 年《计算机研究与发展》高被引论文 TOP10

排名	论文信息
1	李然，林政，林海伦，王伟平，孟丹. **文本情绪分析综述**[J]. 计算机研究与发展，2018，55(1)：30-52 Li Ran, Lin Zheng, Lin Hailun, Wang Weiping, Meng Dan. Text Emotion Analysis: A Survey [J]. Journal of Computer Research and Development, 2018, 55(1): 30-52
2	吕华章，陈丹，范斌，王友祥，乌云霄. **边缘计算标准化进展与案例分析**[J]. 计算机研究与发展，2018，55(3)：487-511 Lü Huazhang, Chen Dan, Fan Bin, Wang Youxiang, Wu Yunxiao. Standardization Progress and Case Analysis of Edge Computing [J]. Journal of Computer Research and Development, 2018, 55(3): 487-511
3	赵梓铭，刘芳，蔡志平，肖侬. **边缘计算：平台、应用与挑战**[J]. 计算机研究与发展，2018，55(2)：327-337 Zhao Ziming, Liu Fang, Cai Zhingping, Xiao Nong. Edge Computing: Platforms, Applications and Challenges [J]. Journal of Computer Research and Development, 2018, 55(2): 327-337/
4	陈珂，梁斌，柯文德，许波，曾国超. **基于多通道卷积神经网络的中文微博情感分析**[J]. 计算机研究与发展，2018，55(5)：945-957 Chen Ke, Liang Bin, Ke Wende, Xu Bo, Zeng Guochao. Chinese Micro-Blog Sentiment Analysis Based on Multi-Channels Convolutional Neural Networks [J]. Journal of Computer Research and Development, 2018, 55(5): 945-957
5	余永红，高阳，王皓，孙栓柱. **融合用户社会地位和矩阵分解的推荐算法**[J]. 计算机研究与发展，2018，55(1)：113-124 Yu Yonghong, Gao Yang, Wang Hao, Sun Shuanzhu. Integrating User Social Status and Matrix Factorization for Item Recommendation [J]. Journal of Computer Research and Development, 2018, 55(1): 113-124
6	贺海武，延安，陈泽华. **基于区块链的智能合约技术与应用综述**[J]. 计算机研究与发展，2018，55(11)：2452-2466 He Haiwu, Yan An, Chen Zehua. Survey of Smart Contract Technology and Application Based on Blockchain [J]. Journal of Computer Research and Development, 2018, 55(11): 2452-2466
7	齐彦丽，周一青，刘玲，田霖，石晶林. **融合移动边缘计算的未来 5G 移动通信网络**[J]. 计算机研究与发展，2018，55(3)：478-486 Qi Yanli, Zhou Yiqing, Liu Ling, Tian Lin, Shi Jinglin. MEC Coordinated Future 5G Mobile Wireless Networks [J]. Journal of Computer Research and Development, 2018, 55(3): 478-486
8	谢振平，金晨，刘渊. **基于建构主义学习理论的个性化知识推荐模型**[J]. 计算机研究与发展，2018，55(1)：125-138 Xie Zhenping, Jin Chen, Liu Yuan. Personalized Knowledge Recommendation Model Based on Constructivist Learning Theory [J]. Journal of Computer Research and Development, 2018, 55(1): 125-138
9	陈伟利，郑子彬. **区块链数据分析：现状、趋势与挑战**[J]. 计算机研究与发展，2018，55(9)：1853-1870 Chen Weili, Zheng Zibin. Blockchain Data Analysis: A Review of Status, Trends and Challenges [J]. Journal of Computer Research and Development, 2018, 55(9): 1853-1870
10	邓晓衡，关培源，万志文，刘恩陆，罗杰，赵智慧，刘亚军，张洪刚. **基于综合信任的边缘计算资源协同研究**[J]. 计算机研究与发展，2018，55(3)：449-477 Deng Xiaoheng, Guan Peiyuan, Wan Zhiwen, Liu Enlu, Luo Jie, Zhao Zhihui, Liu Yajun, Zhang Honggang. Integrated Trust Based Resource Cooperation in Edge Computing [J]. Journal of Computer Research and Development, 2018, 55(3): 449-477

数据来源：CSCD，中国知网；统计日期：2019 年 12 月 23 日