doi:10.11887/j.cn.201405017 http://journal.nudt.edu.cn

递归划分的标签约束可达性计算方法*

吴 烨,钟志农,熊 伟,景 宁
(国防科技大学 电子科学与工程学院,湖南 长沙 410073)

摘 要:现实世界中的图往往在结点和边上包含描述信息,可达性查询是图数据管理和挖掘中的基本操作之一。针对图数据中标签约束的可达性计算问题,提出一种基于递归划分的可达性计算方法RP-Hop。该算法基于层次划分思想,利用独立集性质,在保持标签和可达性前提下对大规模图进行递归划分,并结合贪婪扩展思想和递归编码,为标签约束的可达性查询提供压缩索引。经过合成和真实数据集上的实验,结果表明,RP-Hop算法不仅降低了索引大小和构建时间,而且提高了查询效率。

关键词:标签约束可达性;递归划分;2-hop编码
中图分类号:TP391 **文献标志码**:A **文章编号**:1001-2486(2014)05-098-07

A label constraint reachability computation method on recursive partition

WU Ye, ZHONG Zhinong, XIONG Wei, JING Ning
(College of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China)

Abstract: Many of the real-world graphs are edge-labeled or node-labeled. A foundational operation on these labeled graphs is how to answer reachability queries fast. For the label-constraint reachability computation problem, a computation method called RP-Hop based on recursive partition was proposed. RP-Hop first utilized the hierarchical structure and independent set property to partition the origin large graph recursively, while keeping reachability and labels on paths between node pairs simultaneously. Combined with greedy and recursive labeling strategies, RP-Hop produced a compressed index for label-constraint reachability queries. Experiments on synthetic and real-world graph data sets demonstrate that RP-Hop can reduce index size and construction time, and guarantee the query efficiency.

Key words: label constraint reachability; recursive partition; 2-hop labeling

图数据模型是描述现实世界的普适模型,社会网络、生物信息、化学结构、电信网络、交通网络等都可以用图数据模型来表达。日益增长的图数据规模,使得图数据的管理和挖掘成为数据库领域的研究热点。可达性查询是图数据管理中的基础问题之一,用来回答两个结点之间是否存在一条可达路径。高效的可达性查询在软件工程、社会网络分析、XML、语义网等领域中有广泛的应用前景[1]。

近年来,可达性查询吸引了众多研究者的兴趣。简单来说,回答图上可达性查询的方法可以分为三类:传递闭包[2-3]、在线检索[4-5]以及Hop编码方法[6-7]。传递闭包方法能在$O(1)$时间回答可达性查询,但是由于其高昂的预计算代价($O(|V||E|)$)和存储代价($O(|V|^2)$),扩展性不强。在线查询方法能够回答大规模图上的可达性查询,但是查询速度有待提高($O(|V|+|E|)$)。Hop编码方法介于上述两者之间,在较低索引构建代价的同时能够保持较好的查询性能[1]。

事实上,现实世界中的图往往在结点和边上包含了属性信息[8-9],用以描述不同类型的实体或者实体间的关系,比如朋友关系、合作关系等。实际应用中,在回答两个实体是否可达时,只有考虑了可达性路径上的标签,结果才是有意义的。下面列举两个典型的应用实例:

交通网络:假设一家供货商计划通过卡车将一批货物运送到各个商店,考虑到公共健康和安全因素,某些地方是不允许卡车通行的。于是,标签约束的可达性查询可用来计算两个点之间的一条可达路径。

社会网络:在社会网络中,结点表示人,边表示人与人之间的关系,边上不同的属性值表示不

* **收稿日期**:2014-01-02
基金项目:国家自然科学基金资助项目(61070035);湖南省自然科学基金资助项目(11JJ4028)
作者简介:吴烨(1986—),男,湖南益阳人,博士研究生,E-mail:yewugfkd@nudt.edu.cn;
景宁(通信作者),男,教授,博士,博士生导师,E-mail:ningjing@nudt.edu.cn

同的关系,例如上下级关系、父子(女)、师生关系等。如果某个查询需要回答两个人是否是亲戚[8],则可以通过约束连接两个人的路径来实现。

为有效回答标签约束的可达性查询,研究者们提出了一些方法。Sampling Tree[8]方法为了在存储空间和时间效率之间做出平衡,采用树索引结构,通过构建最大权重生成树加上一个部分传递闭包,实现对传递闭包的压缩,并通过结合采样技术,进一步减小所依赖的传递闭包大小。然而,Sampling Tree 方法需要事先计算所有结点之间的传递闭包,且单个生成树和部分传递闭包方法对压缩传递闭包效果不是很明显,即使在小规模图上构建索引的时间代价都很高。Xu 等[9]通过去除单源传递闭包计算过程中的冗余计算,改善了传递闭包的计算代价。通过将图划分为较小的分块,在每个分块上计算传递闭包,在边界结点构成的框架图上计算是否可达。该方法同样需要对传递闭包进行物化,时间和空间代价过大。

为提高大规模图上的查询效率,一种常用的处理方式对图进行划分,形成层次结构。通过递归抽取关键结点,形成骨干网络[10],使得非骨干网上的结点到骨干网的距离小于设定阈值。基于独立集[11-12],将大图"折叠",并在"折叠"过程中保持结点间的可达性,但没有考虑结点或边上的标签。另一种构建层次结构的方法是,将图划分为小的子图,或者抽取边界结点,事先计算每个子图边界结点之间的最短距离[13]。通过层次结构,计算任意给定两个结点之间的距离。类似这样的方法存在以下不足:图的划分本身就是一个难题,在何种准则下对图进行快速划分;对非平面图而言,子图相互之间的边可能非常多,导致边界结点数量很大。

本文在已有研究基础上,基于递归划分结构,研究一种标签约束的可达性计算方法。

1　问题描述

为描述方便,本节首先给出问题的基本描述和文中用到的定义。

定义 1　边约束标签图

边约束标签图定义为 4 元组 $G=(V,E,\Sigma,f_e)$,其中 V 表示结点集合,E 表示边的集合,Σ 表示边上的标签集合,函数 $f_e:E\mapsto\Sigma$ 定义了从边到标签的映射关系,将每条边 $e\in E$ 映射为标签 $f_e(e)\in\Sigma$。

定义 2　边标签约束的可达性查询

给定结点 $u,v\in G$ 和标签约束 $A\subset\Sigma$,如果存在一条路径 p 从结点 u 到 v,且 p 上的标签 $L(p)\subseteq A$,则称结点 u 和 v 在标签 A 约束下可达,表示为 $u\overset{A}{\rightsquigarrow}v$。

例 1　如图 1 所示,整数表示结点,小写字母表示边上的标签约束。举例来说,结点 10 能到达 13,其标签是 $\{a,c\}$ 和 $\{a,b\}$,即 $u\overset{A}{\rightsquigarrow}v$,其中 $A=\{a,c\}\cup\{a,b\}$。进一步,有(10,7,8,12,13)是结点 10 和 13 之间的 $\{a,b\}$ 路径。有(10,7,9,11,13)是结点 10 和 13 之间的 $\{a,c\}$ 路径。

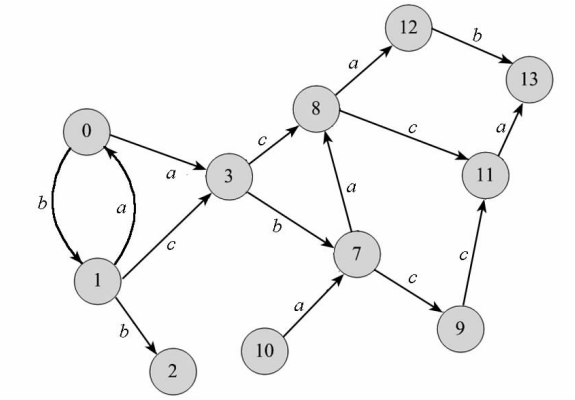


图 1　示例

Fig. 1　Example

在实际应用中,结点上也可能存在标签约束,用来表示实体的属性。与边约束标签图类似,结点约束标签图定义为 4 元组 $G=\{V,E,\Sigma,f_v\}$,其中 $f_v:V\mapsto\Sigma$ 表示每个结点 $u\in V$ 具有标签 $f_v(u)\in\Sigma$。通常地,可以将一个结点标签约束的图转化为一个边标签约束的图。如图 2 所示,路径 p 经过结点 v_i 和 v_j,结点的相应标签为 A 和 B。如果将其转化为边标签图,则路径 p' 应该以 v_i 和 v_j 为端点,且边上标签为 A 和 B。所以,对任意的以边 v_i 和 v_j 为端点的边 $e\in E$,增加一个结点 v_k,使得 $f_e(v_i,v_k)=f_v(v_i)$ 且 $f_e(v_k,v_j)=f_v(v_j)$。因此,本文主要讨论边标签约束的图。

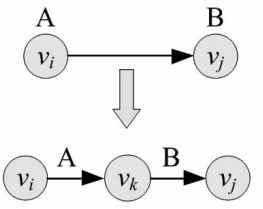


图 2　结点标签转化为边标签

Fig. 2　Convert node-label to edge-label

2　递归划分模型

在线遍历查询方法和基于传递闭包的方法是计算可达性查询的两个极端方法,前者需要最少的内存但需要高计算代价回答可达性查询;后者

需要高昂的时间和空间代价构建索引,但能在常数时间内回答查询。因此,本文的出发点是在上述两种方法中做出合适的折中,在降低索引构建代价和存储空间的同时,仍然保持查询效率。

本文提出一种基于递归划分的索引方法 RP-Hop,其基本观点是:结合标签信息,扩展 2-hop 编码,通过对原图结构递归划分进行压缩,形成结点的自顶而下编码信息,并基于扩展 2-hop 编码计算结点之间标签约束的可达性。通过递归将上层结点广播到下层结点,每个结点可以记录其2-hop 编码来压缩传递闭包。

2.1 层次结构构建

在开始讨论递归划分模型之前,首先给出边界结点以及路径偏序关系定义,作为 RP-Hop 的基础。

定义 4 给定图的 2 个分块 G_i 和 G_j,G_i 中的某个结点 v 称为边界结点,当且仅当满足至少在 G_j 中存在结点 v 的一个入边结点 u 和一个出边结点 w,且(u,w)称为路径对。

定义 5 给定结点 $u,v\in V$,若 p_i,p_j 表示 u 和 v 之间的 2 条路径,$L(p_i)$ 表示路径 p_i 上的标签。如果满足 $L(p_i)\subset L(p_j)$,则称路径 p_i 优先于路径 p_j,表示为 $p_i\preceq p_j$。

给定图 $G=(V,E,\Sigma,f_e)$,$h\in\mathbb{N}^+$ 表示层次结构的高度,RP-Hop 结构定义为映射关系 $G\mapsto\{G_1,G_2,\cdots,G_h\}$,其中 $V=V(G_0)\supset V(G_1)\supset V(G_2)\supset\cdots\supset V(G_h)$,且相应的边为 $E=E(G_0)$,$E(G_1),\cdots,E(G_h)$。

TF-label[11] 和 IS-label[12] 方法通过每次从 G_i 中计算独立结点集合并保留当前路径距离来实现大规模图中的最短路径和可达性计算,然而,边上的标签与最短路径不同。根据定义 5,由于可能存在多个优先标签,每条路径上的标签可能不止一个。

基于独立集[12] 思想,递归将图 $G_i(0\leqslant i<h)$ 划分为 G_i^l 和 G_i^r 两个部分,其中 G_i^r 中所有结点为独立集,满足 $\forall u,v\in G_i^r,(u,v)\notin E_i$ 且 $(v,u)\notin E_i$。然后,将 G_i^l 中所有结点作为 G_{i+1},并按如下方式计算结点之间的连接关系及其标签:

1)若 $(v_i,v_j)\in G_i^l$,在 v_i 和 v_j 添加边,其标签为 $f_e(v_i,v_j)$;

2)求解 G_i^l 中所有边界结点,记为 V_b;

3)$\forall v\in V_b$,在路径对 $(u,w)\in(G_i^l\times G_i^l)$ 之间添加边,其标签为 $\min\{f_e(u,v)+f_e(v,w),f_e(u,w)\}$,其中 min 比较关系为定义 5 优先关系。

2.2 结点编码

为限制层次结构高度,引进阈值 ζ,当 $|V_{i+1}|/|V_i|>\zeta$ 时,停止递归划分,并记 G_{i+1} 为 G_h。基于“贪婪”思想,求解 G_h 中所有结点的 2-hop 编码。

采用 2-hop 编码方法,由于现有 2-hop 编码方法都没有考虑边上的标签,不能直接采用。传统的传递闭包计算过程中,从某个结点出发的单源传递闭包,其他结点只需要访问一次。而在边上具有标签的图中,不同的路径可能具有不同的标签,每个结点可能访问多次,由此也引发计算过程中的冗余。为避免计算顶层图 G_h 的传递闭包,扩展 2-hop 编码方法,采取某种方式确定“重要”结点,将这些结点加入到 L_{out} 或 L_{in} 中。根据某个预定顺序,只有在 $u\in V$ 覆盖了新的结点时,将结点加入到 L_{out} 或 L_{in} 中,从而避免冗余结点的加入,最小化编码代价。

用 $N_{\text{out}}(v)$ 和 $N_{\text{in}}(v)$ 分别表示结点 $v\in V$ 的出边邻居结点和入边邻居结点,则 v 的覆盖定义为 $cover(v)=S(A_v,v,D_v)$,满足 $A_v=\{(u_1,L(u_1,v)),\cdots,(u_{|N_{\text{in}}(v)|},L(u_{|N_{\text{in}}(v)|},v))\}$,$D_v=\{(w_1,L(v,w_1)),\cdots,(w_{|N_{\text{out}}(v)|},L(v,w_{|N_{\text{out}}(v)|}))\}$,其中 $u_i\in N_{\text{in}}(v)$,$w_j\in N_{\text{out}}(v)$。覆盖 $cover(v)$ 表示 $\forall u\in S_v$,u 能够通过$L(u,v)$到达 v,且 v 通过 $L(v,w)$ 能到达 w。此外,$u\in S_v$ 和 $w\in D_v$ 通过 $A=L(u,v)\cup L(v,w)$ 可达,即 $u\overset{A}{\rightsquigarrow}w$。

如果结点 $v\in G_i\setminus G_{i+1}$,则 v 的层次定义为 $\ell(v)=i$。$\forall v\in G$,如果 $\ell(v)=i$,则 v 的标签由 $G_j(i\leqslant j\leqslant h)$ 中的结点组成。首先定义

$$\psi_{\text{out}}(v)=\{u\mid u\in G_{i+1}\wedge u\in N_{\text{out}}(v,G_i)\}\quad(1)$$

$$\psi_{\text{in}}(v)=\{u\mid u\in G_{i+1}\wedge u\in N_{\text{in}}(v,G_i)\}\quad(2)$$

给定结点 $v\in G$,必然属于某个层次 G_i,需要计算该结点的标签。具体地,首先计算 G_h 中结点的 2-hop 编码,然后递归将标签依次从层次 $h-1$ 扩展至层次 0。结点 $v\in G_h$ 的标签为 $L_{\text{out(in)}}(v)=(v,null)$。$\forall u\in G_i(0\leqslant i<h)$,设 $NL_{\text{out}}^i(v)=\{(w,L(v,w))\mid w\in N_{\text{out}}(v,G_i)\}$和 $NL_{\text{in}}^i(v)=\{(u,L(u,v))\mid u\in N_{\text{in}}(v,G_i)\}$ 分别表示 G_i中(出边邻居,标签)和(入边邻居,标签)对。假设 $G_j(j>i)$ 中所有结点都已编码(L_{in} 和 L_{out}),用以下规则计算结点 $v\in G_i$ 的编码:

$$L_{\text{out}}(v)=\{NL_{\text{out}}^i(v)\}\bigcup_{w'\in\psi_{\text{out}}(v,G_i)}\{L_{\text{out}}(w')\}\quad(3)$$

$$L_{\text{in}}(v)=\{NL_{\text{in}}^i(v)\}\bigcup_{u'\in\psi_{\text{in}}(v,G_i)}\{L_{\text{in}}(u')\}\quad(4)$$

对图 G 中的结点进行递归 2-hop 编码的算法流程见算法 1。

算法 1:递归 2-hop 编码

Alg1:Recursive 2-hop

输入:图 $G=(V,E,\Sigma,f_e)$,阈值 ζ

过程:

1. 对图 G 进行递归划分,形成层次结构;
2. 对最顶层子图 G_h 中的结点 v 进行编码初始化,令其编码为 $L_{\text{out}}(v)=L_{\text{in}}(v)=(v,null)$;
3. 按照自上至下的顺序,依次获取第 i 层子图,对所有满足 $v\in\{G_i\setminus G_{i+1}\}$ 条件的结点,将其编码初始化为 $L_{\text{in}}(v)=(v,null)\bigcup\limits_{u_i\in N_{\text{in}}(v)}\{u_i,f_e(v,u_i)\}$ 和 $L_{\text{out}}(v)=(v,null)\bigcup\limits_{w_i\in N_{\text{in}}(v)}\{w_i,f_e(v,w_i)\}$,直至达到最底层原始图;
4. 计算 G_h 中所有结点的 2-hop 编码;
5. 从第 $h-1$ 层开始,基于式(3)和式(4)自顶向下计算所有结点 $v\in\{G_i\setminus G_{i+1}\}$ 的 2-hop 编码 $L_{\text{out}}(v)$ 和 $L_{\text{in}}(v)$,直至第 0 层

输出:所有结点的 2-hop 编码

2.3　基于编码的可达性计算

设 $I(s,t)=L_{\text{out}}(s)\cap L_{\text{in}}(t)$,给定 2 个结点 s,$t\in V$ 和标签约束 $A\in\Sigma$,通过 $L_{\text{out}}(s)$ 和 $L_{\text{in}}(t)$ 是否与 G_h 相交,可以分为 2 种情况回答 s 和 t 之间的标签约束可达性查询:

Case 1:$L_{\text{out}}(s)\cap G_h=\varnothing$ 或 $L_{\text{in}}(t)\cap G_h=\varnothing$,则

$$s\overset{A}{\rightsquigarrow}t=\begin{cases}\text{true} & I(s,t)\neq\varnothing\wedge L(s,w_s)\cup L(w_t,v)\subset A\\ \text{false} & \text{otherwise}\end{cases}\tag{5}$$

其中 $s\rightsquigarrow w_s$ 且 $w_t\rightsquigarrow t,w_u,w_v\in I(s,t)$。

Case 2:$L_{\text{out}}(s)\cap G_h\neq\varnothing$ 且 $L_{\text{in}}(t)\cap G_h\neq\varnothing$.

设 $w_s\in L_{\text{out}}(s)\cap G_h$,$w_t\in L_{\text{in}}(t)\cap G_h$,$C=L(s,w_s)\cup L(w_s,w_t)\cup L(w_t,t)$,则

$$s\overset{A}{\rightsquigarrow}t=\begin{cases}\text{true} & I(w_s,w_t)\neq\varnothing\wedge C\subset A\\ \text{false} & \text{otherwise}\end{cases}\tag{6}$$

2.4　复杂度分析

RP-Hop 的算法复杂度包括 3 个部分:1)层次结构构建复杂度;2)G_h 编码计算;3)自上而下编码计算。在层次结构构建过程中,设图 G_i 中的结点数和边数分别为 $|V_i|$ 和 $|E_i|$,初始化图 $G_0=G$,阈值控制参数为 ζ。则每次迭代中将结点分为两部分的代价为 $O(|V_i|)$,在每层图 G_i 中,若边已经存在 G_{i-1} 中,则添加一条边的代价为 $O(1)$;若边不存在,则需要遍历其在 G_i 中的邻居结点,设每个结点的平均度为 $\bar{d}$,则这些结点之间添加一条边的代价为 $O(\bar{d}^2)$。于是,每次迭代过程中计算层次结构的总时间复杂度为 $O(|V_i|+\bar{d}^2|E_i|)\leqslant O(|V|+\bar{d}^2|E|)$,$h$ 次迭代的最差时间复杂度为

$$O[(1+(\zeta)+\cdots+(\zeta)^h)\cdot(|V|+\bar{d}^2|E|)]<\frac{\zeta}{1-\zeta}(|V|+\bar{d}^2|E|)$$

实际情况中,$|E_i|<|E|$,且随着 i 增大,两者之间的差距会越来越大。

G_h 编码计算中,其边数 $|E_h|<(|V_h|)^2/2$,采用 C++ STL 中的最小优先队列,G_h 中结点编码计算复杂度为 $O[|V_h||E_h|\log(|V_h|)]$。自顶向下编码计算中,层次数为 h,初始化代价为 $h|E|$。编码过程需要 $|V|$ 次迭代,设编码集合平均大小为 $\bar{L}$,则编码代价为 $O[h(|E|+\bar{L}|V|)]$。

于是,计算 RP-Hop 总时间复杂度为 $O[|V_h||E_h|\log(|V_h|)+(\bar{d}^2+h)|E|+(\bar{L}+1)h|V|]$。

根据文献[8],Sampling Tree(ST)索引构建时间复杂度为 $O\left[|V||E|\binom{|\Sigma|}{|\Sigma|/2}+(|E|+|V|\log|V|)\right]$,查询时间复杂度为 $O(\log n+\sum_{i=1}^{k}|NT(u_i,v_i)|)$,其中 n 表示辅助结构 NT 上的实体数量,k 表示 NT 中非空实体对数量。而传递闭包方法构建传递闭包的时间复杂度为 $O(|V|^3 2^{|\Sigma|})$,查询时间复杂度为 $O(1)$。

从理论上分析,RP-Hop 方法的索引构建时间复杂度要远远低于 Sampling Tree 和传递闭包方法,而查询时间复杂度介于上述两者之间。

3　实验与分析

本节给出在真实数据集和合成数据集上的实验,并与典型方法进行比较。本文方法与其他 3 个方法进行比较:ST[8]、压缩传递闭包方法(简称 TC)[9]、宽度优先遍历(Breath First Search,BFS)方法。算法用 C++语言实现。

在实验中,收集 3 个主要参数实验结果,索引构建时间(CT)、索引大小(IS)和查询时间(QT)。每个实验在数据集上执行 10 000 次查询并收集总体计算时间。合成数据集中所有的标签从 Σ 随机选取并符合长尾定理,即大部分结点标签出现概率较小,少量标签出现概率很大。

实验 1　真实数据集上的实验。采用 Yeast 和 Yago 数据集,由文献[8]作者提供。Yeast 是生成酵母的蛋白质交互网络,每个结点代表一种蛋白质,边表示 2 种蛋白质之间的反应。Yeast 数据集包含 3063 个结点,密度 $|E|/|V|=2.4$,包含 5 个类型的边标签,表示不同类型的反应。Yago

数据集是 RDF 数据集的抽样,包含 5000 个结点,66 种类型的标签,密度为 5.7。实验数据见表 1。

表 1 真实数据索引构建性能

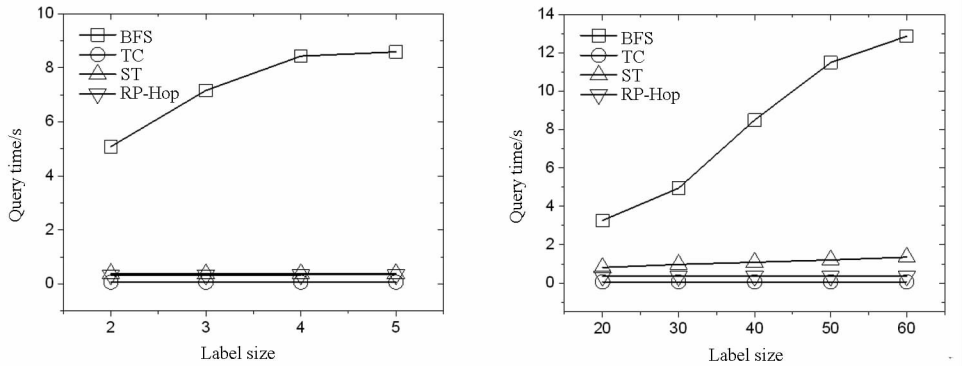
(时间:ms,索引大小:kB)

Tab. 1 Index construction performance of real-world data

数据集	TC		ST		RP-Hop	
	CT	IS	CT	IS	CT	IS
Yeast	6700	14 100	738 100	158	196.2	202.1
Yago	3400	6100	689 300	198	568.3	277.9

实验 1 中,对比分析 ST 方法、TC 方法和 RP-Hop 方法的离线构建时间以及在线查询时间。表 1 展示了每种方法的离线构建时间、索引大小。显然,RP-Hop 在构建时间上,优于其他算法 2 ~ 3 个数量级。特别地,ST 在 Yeast 数据集上的构建时间是 RP-Hop 在其上构建时间的 3000 多倍,甚至比 TC 方法还慢。在索引大小上,RP-Hop 的索引大小明显小于 TC 方法,并且与 ST 方法相当。这是由于 RP-Hop 方法只保留了图中每个结点 v 可以到达的结点以及可以到达 v 的结点,不需要存储所有其他结点信息,而 TC 方法需要记录许多冗余的可达信息。ST 方法由于只记录了最大生成树和部分可达信息,故其索引能保持较小。

在查询性能上,4 种方法在 Yeast 和 Yago 数据集上的查询时间随着约束标签大小的变化如图 3 所示,BFS 的查询时间随着标签大小变化成线性增长,与理论分析 $O(|V|+|E|)$ 相符。ST、TC 和 RP-Hop 方法的查询时间基本与标签大小无关,这是 TC 方法能在常数时间回答查询,ST 方法构建的索引由生成树和部分传递闭包构成,标签越多,部分传递闭包之间满足条件的结点会越多,所以查询时间随约束标签增加而缓慢增长。RP-Hop 采用 2-hop 编码方法,在计算过程中虽然需要计算标签的归属关系,但其编码大小固定,故查询时间与标签大小基本无关。从实验可得,RP-Hop 的查询时间介于传递闭包方法以及 ST 方法之间,与理论分析相符。



(a) Yeast (b) Yago

图 3 Yeast 和 Yago 数据集中的约束变化

Fig. 3 Performance comparison varying constraint size

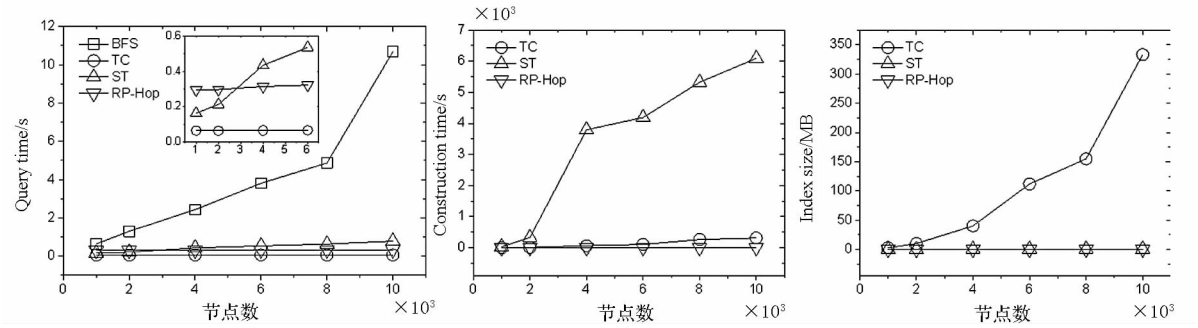
表 2 不同数据规模索引构建性能

Tab. 2 Index construction performance of different date scale $|V|$

$\|V\|$	BFS	TC			ST			RP-Hop		
	QT(s)	CT(s)	IS(MB)	QT(ms)	CT(10^3s)	IS(MB)	QT(ms)	CT(s)	IS(MB)	QT(ms)
10k	19.1	562.4	486.5	66.6	9.9	3.5	593.3	8.4	2.8	341.3
20k	37.1	1489	2134	70.9	20.3	8.3	685.2	19.5	5.6	351.7
40k	77.1	–	–	–	–	–	–	75.0	10.1	363.4
60k	92.3	–	–	–	–	–	–	135.5	16.1	362.7
80k	169.1	–	–	–	–	–	–	192.5	22.3	372.2
100k	185.3	–	–	–	–	–	–	202.1	27.4	370.5
200k	403.7	–	–	–	–	–	–	553.1	54.5	379.2

为测试不同因素,包括数据集大小、数据密度以及标签大小对 RP-Hop 算法的影响,采用合成数据集来测试算法性能。有 2 种生成图模型,Erd-s Rényi(ER)模型和 Scale-Free(SF)模型[14]。由于 SF 模型更接近于现实世界中的图,在该实验中,产生基于 SF 模型的图。在实现上,基于 GTgraph① 来产生随机图,并且设置 $|E|/|V|$ 和 $|V|$ 进行变化。在产生随机图的过程中,设置 $a=0.45$, $b=0.15$, $c=0.15$ 和 $d=0.25$。

实验 2　算法扩展性。固定 $|\Sigma|=10$, $|E|/|V|=1.5$,标签大小 $30\% \times |\Sigma|=3$,结点数量 $|V|$ 从 1k 变化到 10k。对于较大规模图,固定 $|E|/|V|=2$,且结点数量从 10k 变化到 200k。由于内存限制,TC 方法无法处理大规模图。而 ST 方法由于构建索引时间太长,若构建时间超过 8h,在实验中予以省略。表 2 中的符号“-”表示由于资源限制,该方法无法完成。表 2 展示了 ST、TC 以及 RP-Hop 的构建时间随结点数量变化的趋势。图 4 表示 4 种方法的查询时间、索引构建时间以及索引大小随结点数量增长的变化趋势,其中图 4(a)为对数纵坐标。



(a) 查询时间　　(b) 构建时间　　(c) 索引大小

图 4　结点数量变化性能比较

Fig. 4　Performance comparison varying the number of nodes $|V|$

1)RP-Hop 的构建时间较之 ST 和传递闭包方法增长缓慢,且保持在较短时间。当 $|V|=10$k 时,ST 的构建时间是 RP-Hop 的 4500 倍,而索引大小是 RP-Hop 的 238 倍。

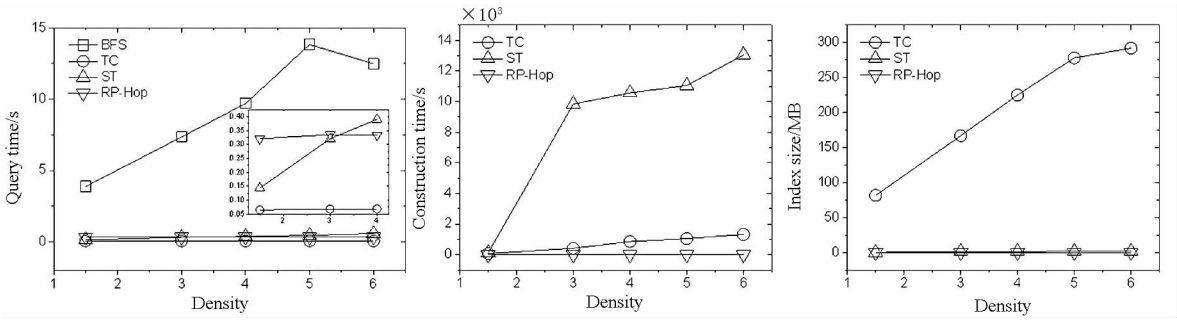
2)RP-Hop 的索引大小不仅比 ST 和传递闭包方法小,而且增速较小。

3)随着结点数量的增加,RP-Hop 和传递闭包方法的查询时间基本保持在几百毫秒,而 ST 方法和 BFS 的查询时间线性增长,但 ST 方法查询时间增长缓慢。

从 RP-Hop 的索引构建时间复杂度分析,其复杂度与图的规模,即结点和边的数量成正比,故随着结点数量增多,构建时间成线性增长。而索引大小与每个结点记录的必要可达结点相关,当结点数量增加时,每个结点的 2-hop 编码大小能基本保持不变,故索引大小与结点数量成正比。另一方面,最大生成树和部分传递闭包上的搜索空间大于 2-hop 编码,且随结点数量增加,传递闭包上的非空对也会增长,故 ST 方法的查询时间增长速度大于 RP-Hop,当结点数达到一定规模时,其查询时间会超过 RP-Hop。但由于在最大生成树和部分传递闭包上的查询相对较快,故查询时间保持在较低数值。

实验 3　密度变化对算法的影响。在实验 3 中,设置 $|V|=5000$, $|\Sigma|=20$,以及标签大小 $30\% \times |\Sigma|=6$,使得密度 $|E|/|V|$ 从 1.5 到 5 发生变化。图 5 表示不同方法的索引构建时间、索引大小以及查询时间。当密度大小发生变化时,尽管 ST 方法的查询时间和索引大小是 3 种方法中最优的,但是其索引构建时间随密度增长而快速增加。另一方面,RP-Hop 方法的索引构建时间、索引大小以及查询时间都能够保持较低,而且与密度大小无关。在查询时间方面,由于 RP-Hop 的查询时间只与每个结点的标签集合大小相关,而标签集合大小与图的密度无关,因此,RP-Hop 的查询时间基本保持不变。

① www.cse.psu.edu/~madduri/software/GTgraph



(a) 查询时间 (b) 构建时间 (c) 索引大小

图5 密度变化对算法性能影响

Fig. 5 Performance comparison varying density $|E|/|V|$

4 结论

针对目前图中的标签约束可达性查询中的不足,本文提出一种基于递归划分结构的标签约束可达性计算方法 RP-Hop。RP-Hop 的构建时间、索引大小和查询时间与图的密度$|E|/|V|$和约束大小$|A|$基本无关;且图的规模较大时,RP-Hop 的索引构建时间和索引大小比 ST 方法和传递闭包方法快 2 ~ 3 个数量级,而且随着结点数量大小增长呈线性变化。实验表明,RP-Hop 能在保持查询性能的同时,有效地压缩索引规模和减少索引构建时间。

参考文献(References)

[1] Yu J X, Cheng J F. Graph reachability queries: q survey[M]. Managing and Mining Graph Data, 2010,40: 181 -215.

[2] Nuutila E. Efficient transitive closure computation in large digraphs[J]. Acta Polytechnica Scandinavia: Mathematics of Computation Engineering,1995, 74: 1 -124.

[3] Van Schaik S J, De Moor O. A memory efficient reachability data structure through bit vector compression[C]//Proceedings of the ACM SIGMOD International Conference on Management of data, 2011: 913 -924.

[4] Yıldırım H, Chaoji V, Zaki M J. GRAIL: a scalable index for reachability queries in very large graphs [J]. The VLDB Journal. 2012, 21(4): 509 -534.

[5] Trißl S, Leser U. Fast and practical indexing and querying of very large graphs [C]//Proceedings of the ACM SIGMOD International Conference on Management of Data, 2007: 845 -856.

[6] Cai J, Poon C K. Path-hop: efficiently indexing large graphs for reachability queries[C]//Proceedings of ACM SIGMOD, 2010: 119 -128.

[7] Cheng J, Shang Z, Cheng H, et al. K-reach: who is in your small world[J]. PVLDB. 2012, 5(11): 1292 -1303.

[8] Jin R M, Hong H, Wang H X, et al. Computing label-constraint reachability in graph databases[C]//Proceedings of the ACM SIGMOD International Conference on Management of Data, 2010: 123 -134.

[9] Xu K, Zou L, Yu J X, et al. Answering label-constraint reachability in large graphs [C]//Proceedings of ACM SIGMOD, 2011: 1595 -1600.

[10] Jin R M, Ruan N, Dey S, et al. SCARAB: scaling reachability computation on large graphs[C]//Proceedings of the ACM SIGMOD International Conference on Management of Data, 2012: 169 -180.

[11] Cheng J, Huang S, Wu H, et al. TF-Label: a topological-folding labeling scheme for reachability querying in a large graph[C]//Proceedings of the International Conference on Management of Data, 2013: 193 -204.

[12] Fu A W, Wu H, Cheng J, et al. IS-Label: an independent-set based labeling scheme for point-to-point distance querying[J]. PVLDB,2013, 6(6): 457 -468.

[13] Jing N, Huang Y W, Rundensteiner E A. Hierarchical encoded path views for path query processing: an optimal model and its performance evaluation[J]. IEEE Transactions on Knowledge and Data Engineering. 1998, 10(3): 409 -432.

[14] Chakrabarti D, Faloutsos C, Mcglohon M. Graph mining: laws and generators[M]//Managing and Mining Graph Data, Aggarwal C C, Wang H X, Springer US, 2010, 40: 69 -123.