# 大规模图上标签集约束路径的集合查询

包佳佳 田 伟
(东南大学计算机科学与工程学院 南京 211189)

**摘 要** 图数据模型被广泛用于社交网络、生物技术、语义网络等开放、异构环境下的数据建模。标签集约束路径查询是基本路径查询问题之一,因其具有路径描述的灵活性而受到目前研究的重视。目前重点研究布尔查询问题:判断给定顶点对间是否有满足标签集约束的路径,返回是或否。现研究布尔查询问题的正交问题,称为集合查询问题:给定标签约束集,返回满足标签集约束可达的顶点对。集合查询问题面临两个困难:1)简单地将集合查询问题简化为布尔查询问题的迭代会陷入穷举困境;2)压缩传递闭包的生成树结构虽然能够有效地回答布尔查询问题,但是,这种压缩结构不能有效支持集合查询,因为集合查询需要搜索满足约束连通的所有顶点对。为此,继续采用生成树来压缩标签路径传递闭包,用倒排索引表来加快集合查询所导致的搜索,并进一步给出两个优化算法。在大规模的数据集上的测试表明,本方法在时间和空间效率方面都具有优势。

**关键词** 图,标签集约束路径查询,标签集约束路径的集合查询,倒排索引
**中图法分类号** TP392 **文献标识码** A

## All Pairs Label-constraint Path Query in Large Graph

BAO Jia-jia TIAN Wei
(School of Computer Science and Engineering, Southeast University, Nanjing 211189, China)

**Abstract** Graph data has been used to model open and heterogeneous data such as social network, biological network and semantic Web. The edge-labeled graphs are drawing the attention of researchers for its scalability to describe the path reachability. Its fundamental problem is about returning true or false of the label-constraint path query. Based on this, we put forward all pairs label-constraint path query problem. There are two kinds of difficulty to solve this problem: 1) It needs to enumerate all pairs of vertices exhaustively if taking the label-constraint path query to solve it; 2) The spanning tree method can't support the all pairs path query problem even though it can answer the path query efficiently. In this work, we compressed the label path transitive closure through spanning tree and quickened the query time by inverted index technique. We also gave two optimal algorithms for the query when searching answers on the spanning tree. The extensive experiments value the effectiveness and efficiency of our approach both on computing time and storage space.

**Keywords** Graph, Label-constraint path query, All pairs label-constraint path query, Inverted index

## 1 引言

图数据模型广泛应用于生物信息、化学结构、语义互联网以及社交网络等开放、异构领域的数据建模。目前,图结构数据规模日益增大[10],使得大规模图数据管理成为数据库领域研究的重点问题之一。而路径查询(path query)[14]是图数据管理的基本问题之一。在带标签的图上,通常通过标签对路径进行约束,主要约束方法有基于语言的路径约束,如正则路径或上下文无关路径约束,即路径上的标签满足正则文法[12]或上下文无关文法[10]。近年来,标签集路径约束[1]引起了广泛的研究兴趣:对于标签集合 $A$,给定两个顶点,其是否存在连通路径使得该路径上的标签均属于 $A$。如果存在这样的路径,这两顶点被称为 $A$-相通,这样的查询称为布尔查询。举例来说,在社交网络中,成员被映射为图顶点,成员间关系被映射为带标签的边,如父子关系可以将边的标签标为"父子",表示该边的两个端点之间是父子关系。给定标签约束集 $A=$ {父子,兄弟,姐妹},如果两个成员 $m_1$ 和 $m_2$ 之间存在一条路径,并且该路径的标签都属于 $A$,则可以判断 $m_1$ 和 $m_2$ 具有近血亲关系。这种布尔查询的结果要么是"真",要么是"假"。

从这个例子中不难看出,标签集约束可以灵活表达近血亲关系,但是正则路径或上下文无关路径并不方便表达这种近血亲关系。其关键是,基于语言的路径约束是非常"精细"的约束,像近血亲这样标签集约束需要分拆成语言描述的多个近血亲类型,如,"父子|兄弟|姐妹"的正则约束表达的是直接血亲关系,"父子·父子|兄弟·兄弟|姐妹·姐妹"表达两代之间的近血亲关系,如此等等,标签集约束需要将这些正则

到稿日期:2012-06-16 返修日期:2012-10-12 本文受国家自然科学基金(60973023,61003057)资助。
包佳佳(1987-),女,硕士生,主要研究领域为图结构数据管理;田 伟(1971-),男,高级工程师,主要研究领域为电力系统自动化、计算机应用技术。

约束进行逻辑或运算列举。可见，即使是简单标签集约束路径查询问题，若采用正则表达式或无关文法描述查询路径信息，都需要复杂的表达式结构，使得求解变得复杂。

上述提出的标签集约束路径的布尔查询问题，是给定两个顶点和标签集约束 $A$，回答这两顶点是否 $A$-相通。而在一些应用中，往往没有给定顶点信息，只是给出约束标签集 $A$，搜索 $A$-相通的所有顶点对，返回 $A$-相通的顶点对集合。这种集合查询有广泛的应用，如顺应互联网而产生的针对网络安全监控的内容监视，监视过程中，监视方并不能事先获得异常事件的发生点，而是通过事先给定的违法以及异常事件集，实时地对网络中的各种信息包括博客、微博、日志以及邮件等的公开的信息进行扫描，来获得满足异常事件集的异常信息源点，从而达到对潜在异常事件的网络监控。同样，如今在社交网络中交友的用户不再满足于在自己熟识的交友圈中寻找朋友，更多是到互联的社交网中找寻满足用户自己提出的交友条件，如性别、年龄、同城等的网络成员。沿用上述血亲关系图来说，血亲关系查询也会有类似的集合查询，即给出近血亲标签集约束 $A$，返回图中所有近血亲关系的成员，从而获得社交网络中成员间的血缘关系。

当然，集合查询可以直接简化为布尔查询问题的反复迭代，如近血亲约束路径的集合查询可以简化为布尔查询，穷举每对成员，判断他们是否 $A$-相通。这种迭代时间消耗成为应用障碍。而布尔查询方面，通常采用生成树结构来压缩存储路径传递闭包。这种压缩方法能够有效地处理布尔查询问题，因为路径的两个端点是事先给定的。但是，在集合查询中，路径的两个端点没有事先给定，因此，集合查询需要在压缩结构上进行搜索，导致时间开销增加。

为此，本文提出标签集约束路径的集合查询问题，拓展标签集约束路径的布尔查询。在解决方法上，避开将集合查询问题简化为布尔查询，因此避开反复迭代布尔查询。同时，也避开两个极端的解决方案：要么高计算开销的在线搜索(DFS/BFS)方法，要么高存储开销的物化路径传递闭包方法。

本文采用高计算开销和高存储开销间的折中方法。技术的关键是基于生成树的路径压缩结构搜索 $A$-相通顶点对，而这个问题简化为集合上的包含查询，即返回所有的路径，其上的标签被 $A$ 包含。最后，在集合包含查询的过程中，应用倒排索引进一步压缩存储开销，最终提高约束路径集合查询效率。现将本文的贡献描述如下：

1. 提出标签集约束路径的集合查询问题，拓展标签集约束路径的布尔查询；

2. 在基于生成树方法的压缩结构上，通过倒排表加快 $A$-相通顶点对搜索，并能够进一步压缩存储开销；

3. 针对标签集约束路径的集合查询问题，提出在生成树压缩结构上搜索 $A$-相通分枝的两个优化方法，从而避开将集合问题简化为布尔查询，并且加快查询效率。

实验部分实现了生成树算法、倒排索引技术，以及优化搜索算法，并在测试数据集上将本文的方法与 DFS、物化路径传递闭包方法进行对比。测试表明，本文的方法在时间开销与空间开销上都具有优势。

## 2 相关工作

路径查询问题，有两种简单处理方法：(1)采用 DFS/BFS 等图遍历算法，直至搜索到目标顶点为止；(2)物化路径传递闭包，回答查询问题。这两个极端的方法都存在利弊：DFS/BFS 算法，无需消耗存储空间，但是时间开销为 $O(V)$，在处理大规模图查询时，时间代价难以忍受；物化路径传递闭包的方法无需在线搜索，但需事先存储全部的路径传递闭包，空间代价为 $O(V^2)$，用来处理大图空间的代价过高。目前，许多算法[3-5,8,9]都介于这两个方法之间：采用压缩结构压缩路径传递闭包，压缩的路径通过在线搜索获得。但上述算法都没有考虑路径中的标签信息，传递闭包中存储的路径也只是顶点信息，并不能直接应用到标签图上带有标签约束的路径连通性问题。文献[1,2]提出了各自的解决方案：文献[1]提出了标签集约束路径查询问题，并给出生成树压缩结构压缩标签图上的路径传递闭包；文献[2]中通过重新定义路径距离，提出求解标签路径传递闭包的改进算法。

文献[1,2]提出的算法解决的是路经查询判断性问题，如果用来处理集合查询问题，在没有事先给定顶点对的情况下，只能先将集合查询简化为布尔查询，再通过穷举迭代列举出满足集合查询条件的所有顶点对，显然大规模图上的穷举会陷入困境。针对路径描述信息的查询，目前的解决方法有正则路径表达或上下文无关路径表达，若采用该方法，集合查询问题需要用严格的正则表达式或上下文无关文法表示，而正则路径查询能够“精细”地描述路径信息，也是 NPC 问题，从而不能有效解决“灵活”的标签集约束集合查询问题。

因此，本文利用生成树压缩传递闭包，并用倒排索引加快集合查询。第 3 节进行问题定义，并给出解决集合查询的两个基本方法；第 4 节提出倒排索引以及优化算法；第 5 节通过实验验证算法效率；最后进行总结与展望。

## 3 问题定义

带标签的有向图 $G$ 可以表示成四元组 $G(V,E,\Sigma,\lambda)$，其中 $V$、$E$、$\Sigma$ 分别为图 $G$ 的顶点集、边集和标签集，映射 $\lambda:E\to\Sigma$ 表示对边 $e$ 加上标签 $\lambda(e)\in\Sigma$，其中 $e\in E$。图中一条从 $u$ 到 $v$ 的路径 $p$ 是一系列的点边相间隔的序列，即 $p=(u,e_1,v_1,\cdots,v_{i-1},e_i,v_i,\cdots,e_n,v)$，$L(p)$ 表示路径 $p$ 的标签集合，即 $L(p)=\{\lambda(e_1)\}\cup\{\lambda(e_2)\}\cup\cdots\cup\{\lambda(e_n)\}$。

**定义 1**(标签集约束路径查询) 在带标签的有向图 $G(V,E,\Sigma,\lambda)$ 中，给定顶点 $u$、$v\in V$，以及约束标签集 $A\subseteq\Sigma$，如果 $u$ 和 $v$ 存在一条路径 $p$，使得 $L(p)\subseteq A$，则 $u$ 和 $v$ 是 $A$-相通的，记为 $u\xrightarrow{A}v$。

**定义 2**(标签集约束路径的集合查询) 在带标签的有向图 $G$ 中，给定约束标签集 $A$，返回所有 $A$-相通的顶点对，查询结果表示为 $[A]$，$[A]=\{(u,v)\mid u\xrightarrow{A}v\}$。

全文采用图 1 作为示例用图，下面给出图 1 上标签集约束路径查询和标签集约束路径集合查询实例。给定约束标签集 $A=\{e,c\}$，以及顶点 0 和 2，$0\xrightarrow{\{e\}}2$，$\{e\}\subseteq A$，可知 $0\xrightarrow{A}2$；对于标签集约束路径集合查询，有如下点对 $A$-相通：$0\xrightarrow{\{e\}}1$，

$0 \xrightarrow{\{e\}} 2,0 \xrightarrow{\{c\}} 3,0 \xrightarrow{\{e\}} 5,0 \xrightarrow{\{e,c\}} 8,5 \xrightarrow{\{c\}} 8,10 \xrightarrow{\{c\}} 13,15 \xrightarrow{\{c\}} 6$，所以标签集 $A$ 约束路径的集合查询结果为：$[\{e,c\}]=\{(0,1),(0,2),(0,3),(0,5),(0,8),(5,8),(10,13),(15,6)\}$。

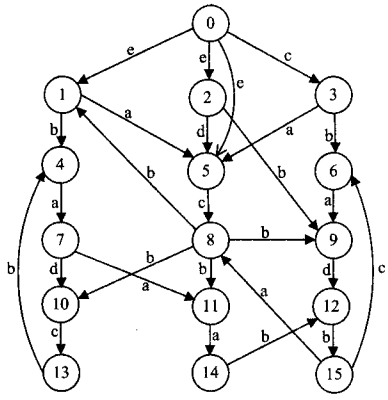


图 1　例图

下面两节分别介绍解决标签集约束路径集合查询的两种不同方法。其中，3.1 节给出 BFS/DFS 算法，3.2 节给出标签路径传递闭包存储的方法。

### 3.1　基于图遍历的集合查询

标签集约束路径集合查询可通过图遍历方法求解，图遍历方法中又以 DFS/BFS 方法最为常见，下面以 DFS 为例描述集合求解问题。

从图中任意顶点 $u$ 开始，以 DFS 方法搜索 $u$ 能够 $A$ 约束可达的所有顶点，搜索结果构成一个 $A$-连通分枝。从图中删去由 $u$ 导出的 $A$ 连通分枝的边，重复 DFS 操作，直至图中没有 $A$ 连通分枝。该算法的时间复杂度为 $O(|E|)$，面对大规模图数据时，会导致计算时间代价过高。

### 3.2　基于传递闭包的集合查询

另一种方法是事先计算出图中所有顶点对的连通路径标签信息，从而得到标签路径传递闭包（label path transitive closure，LPTC），存储开销为 $O(|V|^2 2^{|\Sigma|})$，同样在遇到图数据很大时，存储开销难以忍受。

目前，解决标签集约束路径查询问题的文献[1,2]中提到，全部的 LPTC 中存在大量冗余信息。举例来说，图 1 中的点对(0,5)有(0,2,5)、(0,5)两条路径，相应标签集为$\{e,d\}$、$\{e\}$，对于任意给定的 $A$，如果$\{e,d\}\subset A$，必然有$\{e\}\subset A$，路径标签信息$\{e\}$完全能够回答(0,5)的约束查询问题。下面给出用以减少冗余路径标签信息的相关定义。

**定义 3**（完备路径标签集）　设 $S$ 为点对 $u,v$ 之间的路径标签集的集合，如果 $S$ 满足这样的条件：对于任意给定的标签集合 $A$，有 $u \xrightarrow{A} v$ 当且仅当 $S$ 中存在一个路径标签 $s\in S$、$s\subset A$，则称这样的 $S$ 为点对 $u,v$ 的完备路径标签集合。

点对之间的所有可达的路径标签集合当然是一个完备路径标签集合，但没有减少任何冗余。如果 $S$ 的元素有如下性质：$\forall s_1、s_2\in S$，不存在 $s_1\subset s_2$ 或者 $s_1\supset s_2$，则称之为最小完备路径标签集合，记为 $M(u,v)$，且所有点对的最小完备路径标签集合记为 $M$。目前，求解 $M(u,v)$ 的算法的时间复杂度为 $O(|V|^3 2^{|\Sigma|})$，空间复杂度为 $O(|V|^2\binom{|\Sigma|}{\lfloor|\Sigma|/2\rfloor})$。

## 4　索引和集合查询

对给定图上求解出的 $M$，在进行集合查询时，需要依次搜索点对的路径信息，并进行包含关系判断，穷搜的方法还是不能加快查询。为此，本文首先研究在 $M$ 上建立倒排索引，随后在考虑到树状压缩结构时，对因无法压缩而必须存储的路径信息采用与 $M$ 上相同的索引方法。

### 4.1　$M$ 上的倒排索引结构

$M$ 倒排索引记为 $M'$，构建过程为：$M$ 中的元素为标签集合，对每一个标签集建立一个反向索引，指向所有能够通过该标签集合相通的点对，索引项数总量为 $|\bigcup_{(u,v)\in V\times V} M(u,v)|$。表 1 给出了图 1 的 $M'$ 部分内容以作参考。

表 1　$M$ 的倒排表

| 标签集 | 顶点对 |
|---|---|
| {a} | (1,5),(3,5),(4,7),(6,9),(7,11),(11,14),(15,8) |
| {b} | (1,4),(2,9),(3,6),(8,9),(8,1),(8,10),(8,11),(12,15),(13,4),(14,12) |
| {c} | (0,3),(5,8),(10,13),(15,6) |
| {d} | (2,5),(7,10),(9,12) |
| …… | …… |

接下来，通过对 $M$ 与 $M'$ 之间压缩量的分析研究，进一步阐述倒排索引的压缩优势。已知 $|M|=\Sigma_{(u,v)\in V\times V}|M(u,v)|$，$|M'|=|\bigcup_{(u,v)\in V\times V} M(u,v)|$。从 $M$ 到 $M'$，增量记为 $\Delta=|M|-|M'|$。

**引理 1**　$|M'|<=|M|$。

图 1 中，$|M|=210$，$|M'|<2^{|\Sigma|}$，$|\Sigma|=5$，于是 $|M'|<32$，有 $|M'|<|M|$。因为图 1 的 $|\Sigma|$ 远小于 $|E|$，顶点对中的路径标签集重复的比较多，所以 $\Delta$ 较大。极端情况下，当 $|\Sigma|=|E|$ 即每条边上的标签都不重复时，会得到 $|M'|=|M|$。

### 4.2　倒排索引结构上的集合查询

由定义 1 可知，如果点对路径标签集被给定标签集 $A$ 包含，则该顶点对 $A$-相通。同样，在标签集约束集合查询问题中仍需满足该性质。根据这一性质，给出引理 2，描述利用倒排索引技术求解集合查询的过程。

**引理 2**　给定约束标签集 $A$，路径的集合查询的解空间组成为：$[A]=\bigcup_{L_i\subseteq A}[L_i]$。

可以看出，搜索倒排表中用到集合包含关系判断，这与一般的集合包含关系查询类似，可以应用已有的集合属性索引的方法来加快查询。

### 4.3　优化

基于 $M$ 上的倒排索引技术，虽然压缩了冗余的路径标签信息，但是查询时间代价还是过高。本节利用树状结构优化存储，并利用 $M$ 上的倒排技术处理因不能被压缩而被存储下来的剩余路径标签信息，同时证明在树状结构下的倒排索引技术具有良好的压缩优势。

#### 4.3.1　压缩结构下的倒排表

虽然 $M$ 为标签图路径信息的最小完备路径标签集，但是对于规模很大的图来说，$M$ 的存储开销还是很大。文献[1]中利用生成树结构进一步压缩 $M$，将 $M$ 转化为树上连通关系 $T$ 和非树上的连通关系 $NT$。本文在此压缩结构基础上，利用上节的倒排技术对 $NT$ 表建立倒排索引，记为 $NT'$。图 1 的 $NT$ 表对应的 $NT'$ 如表 2 所列。

表 2　示例图 $NT$ 倒排表 $NT'$

| NT 中标签集 | 顶点对 |
|---|---|
| {a} | (3,5),(6,9),(7,11) |
| {a,b} | (3,9),(7,9),(7,12),(13,9),(13,12),(14,9) |
| {a,b,c} | (3,12),(5,12) |
| {a,c} | (3,8), |
| {a,b,e} | (0,12) |
| {c,e} | (0,8) |
| {d} | (2,5),(7,10) |
| {b} | (3,6),(8,9),(13,4),(14,12) |
| {b,c} | (5,9) |
| {b,c,d} | (7,4) |
| {c} | (5,8) |
| {c,d} | (2,8) |
| {e} | (0,1),(0,5) |

简略介绍 $T$ 的生成过程：为图中每条边赋予权值，记为 $w(u,v)=|\bigcup_{u'\in V}((M(u',u)\times\{L(u,v)\})\cap M(u',v))|$，$\times$ 表示集合笛卡尔乘积，利用生成树算法求得权值最大的生成树，其值记为 $W_T=\Sigma_{(u,v)\in T}w(u,v)$。得到 $T$ 后，获得 $NT$ 的过程描述如下：$M(u,v')$ 中存在一条路径 $p'$，$M(u,v)$ 中存在一条路径 $p$，使得 $L(p')\bigcup L(p_T'(v',v))=L(p)$，则从 $M$ 表中的 $M(u,v)$ 中删除该路径的标签信息，依次操作，直至不能再删除标签信息为止，即得到 $NT$。图 2 为例图的生成树。

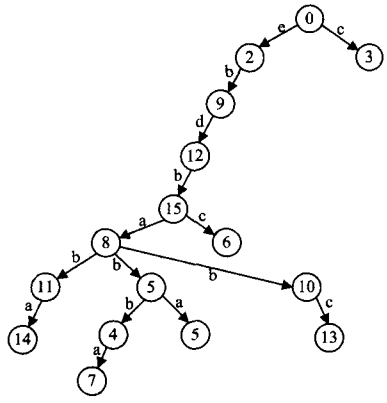


图 2　例图的生成树

将 $M$ 中因 $T$ 被压缩的路径标签用 $M_T$ 来表示。本文研究发现，$W_T$ 和 $|M_T|$ 并不相等。举例来说，如图 3 所示，边 $(v',v)$、$(u,u_4)$ 为 $T$ 中的边，假设 $w(v',v)=3$，则 $M(u_1,v)$、$M(u_2,v)$ 和 $M(u_3,v)$ 中存在路径标签，其可以由 $NT(u_1,v')$、$NT(u_2,v')$ 和 $NT(u_3,v')$ 分别加上边 $(v',v)$ 中的标签恢复出。但是 $M(u,v)$ 中存在一条路径，它的始边经过 $(u,u_4)$，可由 $NT(u_4,v)$ 中的一条路径标签加上 $(u,u_4)$ 的标签得到，但是生成 $NT(u,v)$ 过程中这一路径标签并没有删除。这是因为，$W_T$ 计算的是路径中终边在树上的路径标签，而此例中的该条路径恰恰是始边在树上。同时，路径中存在连续多条边存在于 $T$ 上的路径也没有被算在 $W_T$ 中。于是，得到下面的引理。

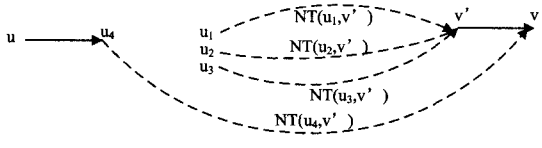


图 3　$NT$ 末端在 $T$ 中展开

**引理 3**　$W_T<|M_T|$。

由引理 3 可以看出，$T$ 中存储的少量边能够带来超过 $W_T$ 的压缩量。基于引理 1、引理 3，可以得到 $NT'<=NT$。同时，$T$ 中存储的边和标签又不会超过 $|M_T|$。进一步，得到如下定理。

**定理 1**　$W_T+|NT'|<|M'|<|M_T|+|NT'|<|M_T|+|NT|=|M|$。

证明：对于最左边的小于式，由引理 3，$W_T<|M_T|$，得到 $W_T+|NT'|<|M_T|+|NT'|$。对于 $M'$，可将它分为 $NT'$ 和 $M_T'$，即 $|M'|=|M_T'|+|NT'|$。再由引理 1，$|M'|<=|M|$，从而进一步能够得到 $|M_T'|<|M_T|$。

所以 $W_T+|NT'|<|M'|=|M_T'|+|NT'|<|M_T|+|NT'|$，于是有 $W_T+|NT'|<|M'|$。由引理 1，我们得到 $|M_T'|<|M_T|$，所以有 $|M'|=|M_T'|+|NT'|<|M_T|+|NT'|$。

于是，得到 $W_T+|NT'|<|M'|<|M_T|+|NT'|$。同理，有 $|NT'|<|NT|$，得到 $|M_T|+|NT'|<|M_T|+|NT|$。而 $|M_T|=|M|-|NT|$，定理得证。

定理 1 说明，$NT$ 上的倒排索引能够进一步增加压缩存储量。下面通过定理 2 给出集合查询解的构成。

**定理 2**　$\forall(u,v)\in V\times V$，$\exists s\in M(u,v)$，$s'\in NT'(u',v')$，使得 $s=L(p_T(u,u'))\bigcup s'(u',v')\bigcup L(p_T(v',v))$，其中 $u'\in Succ(u)$、$v'\in Pred(u)$，并且 $L(p_T(u,u'))$、$L(p_T(v',v))\subseteq A$，$s'\subseteq A$，则 $(u,v)$ 为集合查询的解之一。

由定理 2 可知，在树状压缩结构之下求解标签集约束路径集合查询时，分别到 $T$ 与 $NT'$ 中搜索获得满足标签集限制的部分解，余下的解需要通过 $T$ 与 $NT'$ 连接而得。对于例图，给定标签集 $A=\{a,b\}$，从 $NT'$ 中得到的解如下：(3,5)，(6,9)，(7,11)，(3,9)，(7,9)，(7,12)，(13,9)，(13,12)，(14,9)。$T$ 中的解如图 4 中加粗部分所示。$T$ 与 $NT'$ 连接的解如图 4 中虚线连接所示。

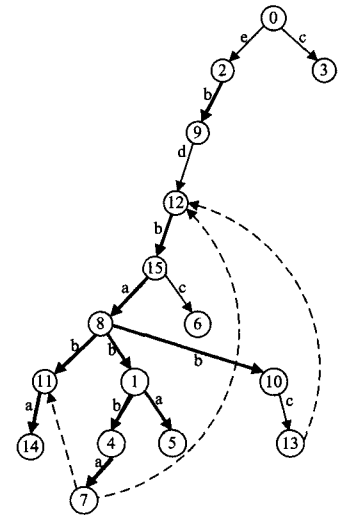


图 4　标签集 $A=\{a,b\}$ 的树中求解

而由标签集约束路径查询问题可知，只有在获知源点信息的情况下，才能到 $T$ 中去搜索。但是，本文的集合查询问题并没有获知任何的源点信息，所以需要 $T$ 来给出的解并不能用已有的方法轻易获得。下面重点讨论 $T$ 中求解。

4.3.2　树上求解算法

记 $[A]_T$ 为 $T$ 中能够 $A$ 连通的分枝，本文给出两种求解 $[A]_T$ 的方法，分别为 $ST_1$ 和 $ST_2$。$ST_1$ 建立 $T$ 的倒排表 $T'$，得到 1-$A$ 连通的顶点对，即通过一条边 $A$-相通的顶点对，得到的结果记为 $[A]_1$，对 $[A]_1$ 表进行自连接，通过对 $T$ 中的节

点区间标号，将自连接得到的解中满足祖先关系的顶点对做适当的删除；$ST_2$ 深度优先遍历 $T$，从根节点开始，将每一步遍历到的节点都存入栈中，对于满足 $A$-连通的节点，将其所在分枝的最大连通路径存入结果中。区间标记的具体技术参见文献[7]，具体应用可参见文献[1]中提及的查找 $NT$ 的方法。

算法 $ST_1(T,A)$描述如下：

$ST_1(T,A)$

1. 先序遍历 T，并进行区间标记，记为[pre(u),index(u)]；
2. 搜索 T′，得到$[A]_1$；
3. (k−1)-A 连通与$[A]_1$ 进行连接，得到 k-A 连通的解，$[A]_k$；
4. 利用区间标记，删除$[A]_1$，$[A]_2$，…，$[A]_k$ 中互相包含的顶点对。

算法 $ST_2(T,A)$描述如下：

$ST_2(T,A)$

1. 深度优先遍历 T，遍历的同时将节点存入栈中；
2. 遍历过程中，将满足 A 连通的分枝存入结果中；
3. 堆栈中的元素，利用递归算法，即不断地调用上述深度优先遍历算法，即重复步骤 1 至 2，直至栈为空。

其中，$ST_1$ 的主要计算代价是不断地进行连接，而 $ST_2$ 需要栈来存储需递归处理的子树，栈的空间代价为 $O(\log(n))$。我们将通过实验来比较二者的求解效率。

## 5 实验

本节通过实验验证标签集约束路径的集合查询问题。分别在真实数据集和人造数据集上(均来自文献[1]提供的数据集)测试。实验通过存储开销和时间开销两方面，对 DFS、传递闭包存储(Transitive Closure，TC)以及生成树方法(Spanning Tree，ST)进行对比测试。实验编写语言为C++，并在 Ubuntu 10.04LTS 上测试，测试用的主机 CPU 为：Intel® Core™ 2 CPU 4300@1.80GHz，内存为 2GB。

### 5.1 存储代价对比

实验中，因 DFS 方法未涉及到传递闭包存储，所以只考察传递闭包和生成树方法的存储代价。而 $ST_1$ 和 $ST_2$ 也只在查询时间上存在差异，存储上不做比较。本节树中搜索算法均采用 $ST_1$。传递闭包存储的方法中，生成的传递闭包 $M$ 为主要存储代价，而生成树方法中，$T$ 和 $NT'$ 为主要存储代价。

在人造数据集上，分别通过改变图的密度以及规模，来观察上述两种方法的存储开销变化，参数变化情况如下：

1. 固定$|V|=5000$ 以及 $A$ 的大小，其中设置 $A$ 为 20，改变图的密度，使得$|E|/|V|$从 1.5 到 5.5 之间变化，分别观察传 $TC$ 和 $ST$ 的存储开销；

2. 固定 $A$ 的大小以及密度$|E|/|V|=1.5$ 的情况下，其中设置 $A$ 为 20，改变$|V|$的大小，使得$|V|$从 2000 到 10000 变化，分别观察传 $TC$ 和 $ST$ 的存储开销。

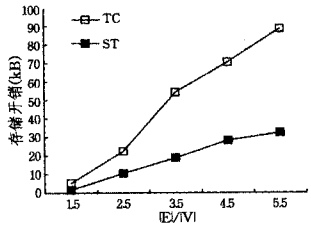


图 5 图密度对存储开销的影响

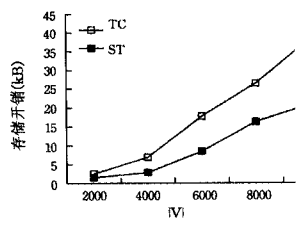


图 6 图顶点数$|V|$对存储开销的影响

从图 5、图 6 中可以看出，图密度和顶点数的变化使得 $TC$ 和 $ST$ 的存储开销均增大，但 $ST$ 相对 $TC$ 增长较缓慢，相同条件下 $ST$ 的存储开销少于 $TC$。

### 5.2 时间代价对比

本节实验中给出 DFS 和 $ST_{(1,2)}$ 在 $A$、图密度以及图规模改变的情况下查询时间代价的对比。首先在人造数据集上测试对比。

1. 固定$|V|=5000$ 以及密度$|E|/|V|=1.5$ 的情况下，改变$|A|$的大小，$|A|$从占整个$|\Sigma|$的 10%到 80%变化，分别观察 DFS 和 $ST_{(1,2)}$ 的存储开销；

2. 固定$|V|=5000$ 以及 $A$ 的大小，改变图的密度使得$|E|/|V|$从 1.5 到 5.5 之间变化，分别观察 DFS 和 $ST_{(1,2)}$ 的存储开销。

表 3 标签数对查询时间(ms)的影响

| $\|A\|$ | DFS | $ST_1$ | $ST_2$ |
|---|---|---|---|
| 10% | 121.33 | 20.87 | 29.60 |
| 20% | 129.45 | 28.62 | 33.56 |
| 30% | 136.22 | 40.98 | 50.27 |
| 40% | 150.25 | 55.21 | 60.49 |
| 50% | 175.96 | 70.98 | 83.26 |
| 60% | 220.63 | 80.14 | 100.59 |
| 70% | 383.21 | 100.56 | 143.60 |
| 80% | 499.52 | 175.23 | 245.65 |

表 4 图密度对查询时间(ms)的影响

| $\|E\|/\|V\|$ | DFS | $ST_1$ | $ST_2$ |
|---|---|---|---|
| 1.5 | 135.77 | 8.50 | 10.46 |
| 2.0 | 203.25 | 16.26 | 21.63 |
| 2.5 | 269.52 | 19.34 | 25.32 |
| 3.0 | 311.96 | 22.78 | 30.85 |
| 3.5 | 397.65 | 31.65 | 40.87 |
| 4.0 | 462.83 | 37.46 | 45.12 |
| 4.5 | 522.74 | 45.68 | 53.89 |
| 5.0 | 634.86 | 51.79 | 60.78 |

从表 3、表 4 中可以看出，DFS 查询时间逐渐增大，$ST$ 也随之逐渐增大。在相同条件下，$ST$ 的查询时间仍低于 DFS。同样，$ST_1$ 优于 $ST_2$。

在真实数据集 Yeast 和 Yago 上，通过$|A|$的改变，来测试上述 3 种方法的完成查询的时间开销，对 Yeast 和 Yago 的处理参照文献[1]。Yeast 数据中抽取出的图的节点为 3063，密度为 2.4，标签数为 5，所以该实验数据中，设置$|A|$从 2 变化到 5，观察查询时间的变化。Yago 上抽取的图有 5000 个节点，标签数为 66，图密度为 5.7，实验中$|A|$从 20 改变到 60 进行测验。

由图 7、图 8 可以看出，标签集约束路径集合查询，随着$|A|$的增大，DFS 的查询时间逐渐增大，$ST$ 无明显增大，并且容易看出 $ST$ 方法比 DFS 的要快两倍。$ST_1$ 和 $ST_2$ 之间，$|A|$逐渐增大，$ST_1$ 比 $ST_2$ 快，这与人造集上的测试结果一致。

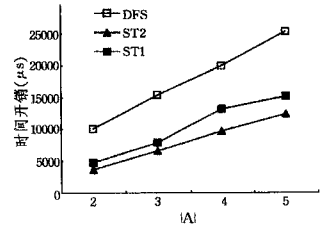


图 7 Yeast 数据集$|A|$对查询时间的影响

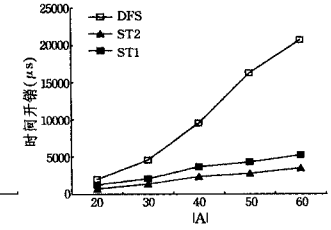


图 8 Yago 数据集上$|A|$对查询时间的影响

(下转第 192 页)

[6] Zhu Yao, Zang Liang-jun, Cao Ya-nan, et al. A Manual Experiment On Commonsense Knowledge Acquisition From Web Corpora[C]// International Conference on Machine Learning and Cybernetics. Kunming, China, 2008

[7] Cao Ya-nan, Cao Cun-gen, Cao Liang-jun, et al. Acquiring Commonsense Knowledge about Properties of Concepts from Text [C]// 5th International Conference on Fuzzy Systems and Knowledge Discovery. 2008

[8] Cao Ya-nan, Cao Cun-gen, Cao Liang-jun, et al. Extracting Comparative Commonsense from the Web[C]// 6th International Conference on Intelligent Information Processing. 2010

[9] 彭会良，曹存根. 相关事件挖掘与角色联系发现的研究[J]. 计算机科学，2010，37(12)：149-155

[10] 曹亚男. 面向 Web 语料的因果知识获取研究[D]. 北京：中国科学院研究生院，2012

[11] Lenat D. CYC: A Large-scale Investment in knowledge Infrastructure[J]. Communications of the ACM, 1995, 38(11): 33-38

[12] Singh P. The Public Acquisition of Commonsense Knowledge [C]// AAAI Spring Symposium on Acquiring Linguistic Knowledge for Infoemation Acess. 2002

[13] Chklovski T A. Using Analogy to Acquire Commonsense Knowledge from Human Contributors[D]. Boston : Massachusetts Institute of Technology, 2003

[14] 刘东立，姚天顺. 自然语言处理中继承理论的分类和应用[J]. 东北大学学报，1997，18(5)：486-489

[15] WordNet. A lexical database for the English language [EB/OL]. http://wordnet.princeton.edu/

[16] Filatova E, Hatzivassiloglou V. Domain-independent detection, extraction, and labeling of actomic events[C]// Proceeding of RANLP. 2003: 145-152

[17] 刘宗田，黄美丽，周文，等. 面向事件的本体研究[J]. 计算机科学，2009，36(11)：189-192

[18] 鲁川. 知识工程语言学[M]. 北京：清华大学出版社，2010：245-248

[19] 鲁川，侯瑞隆，董丽萍. 现代汉语基本句模[J]. 世界汉语教学，2000(4)：11-24

[20] Nelson K, Gruendel J. Event Knowledge : Structure and Function in Develpoment[M]. Hillsdale: Erlbaum, 1986

[21] 周文. 基于概念的若干知识表示模型及相关方法研究[D]. 上海：上海大学，2007

[22] 王寅. 事件域认知模型及其解释力[J]. 现代汉语，2005，28(1)：17-26

[23] Lenat D. The Dimensions of Context-Space[EB/OL]. http://www.casbah.org/resources/cycContextSpace.shtml

[24] 彭会良. 人物相关事件的常识知识获取方法研究[D]. 北京：首都师范大学，2010

[25] 阮智富，郭志新，乐嘉民，等. 现代汉语大辞典[M]. 上海：汉语大词典出版社，2000：860-860

(上接第 176 页)

**结束语** 本文在标签集约束路径查询问题的基础上，提出标签集约束路径集合查询问题，给出求解的生成树压缩存储结构与倒排索引技术。同时，理论证明倒排索引存在良好的压缩效益。给出两个优化算法解决树结构上无源点信息的集合查询问题。实验证明，在大规模图上，生成树方法与在线的 DFS 和预存储标签路径传递闭包相比，在时间和空间开销上都有优势。下一步，在本文的基础上，继续研究如何改进压缩路径标签的存储结构以及查询集合包含关系的算法，以进一步减少时间和空间开销。

## 参 考 文 献

[1] Jin Ruo-ming, Hong Hui, Wang Hai-xun, et al. Computing Label-Constraint Reachability in Graph Database [C]// SIGMOD' 10. 2010: 123-134

[2] Zou Lei, Xu Kun, Yu J X. Answering Label-Constraint Reachability in Large Graphs[R]. TR-DB-ICST-PKU-2011-002. Institute of Computer Science and Techniloge

[3] Fang Wei. TEDI: Efficient Shortest Path Query Answering on Graphs [C]// SIGMOD '10. 2010: 99-110

[4] Jin Ruo-ming, Xiang Yang, Ruan Ning, et al. 3-HOP: A High-Compression Indexing Scheme for Reachability Query [C]// SIGMOD '09. 2009: 813-826

[5] Wang Hai-xun, He Hao, Yang Jun, et al. Dual labeling: Answering graph reachability queries in constant time [C]// ICDE '06. 2006: 75

[6] Yan Y, Wang C, Zhou A, et al. Efficiently querying RDF data in triple stores [R]// Tech report. 2008

[7] Gou Gang, Chirkova R. Efficiently querying large xml data repositories: A survey [J]. IEEE Trans. Knowl. Data Eng., 2007, 19(10): 1381-1403

[8] Jagadish H V. A compression technique to materialize transitive closure[J]. ACM Trans. Database Syst., 1990, 15(4): 558-598

[9] Cohen E, Halperin E, Kaplan H, et al. Reachability and distance queries via 2-hop labels [C]// Proc of the 13th annual ACM-SIAM Symp on Discrete Algorithms. 2002: 937-946

[10] Chomsky, Noam. Three Models for the Description of Language [J]. IRE Transactions on Information Theory, 1956, 2(3): 113-124

[11] Barabasi A L, Albert R. Emergence of scaling in random networks [J]. Science, 1999, 286(5439): 509-512

[12] Abiteboul S, Vianu V. Regular path queries with constraints [C]// PODS. 1997: 122-133

[13] Ramasamy K, Patel J M, Naughton J F. Set Containment Joins: The Good, The Bad and The Ugly [C]// Proc of the 26th VLDB Conf. Cairo, Egypt, 2000

[14] Mendelzon A O, Wood P T. Finding regular simple paths in graph databases[J]. SIAM J. Comput., 1995, 24(6): 1235-1258

[15] Leskovec J, Singh A, Kleinberg J M. Patterns of influence in a recommendation network [C]// PAKDD'06. 2006: 380-389