ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2014/08(11)-1324-10
doi: 10.3778/j.issn.1673-9418.1407045

E-mail: fcst@vip.163.com
http://www.ceaj.org
Tel: +86-10-89056056

时序图上动态子图查询优化算法*

朱 青[1,2+],李 红[1,2]
1. 中国人民大学 教育部数据工程与知识工程重点实验室,北京 100872
2. 中国人民大学 信息学院 计算机系,北京 100872

Querying Optimization Algorithm for Dynamic Subgraph on Time-Evolving Graph*

ZHU Qing[1,2+], LI Hong[1,2]
1. Key Laboratory of Data Engineering and Knowledge Engineering, Ministry of Education, Renmin University of China, Beijing 100872, China
2. Department of Computer Science, School of Information, Renmin University of China, Beijing 100872, China
+ Corresponding author: E-mail: zq@ruc.edu.cn

ZHU Qing, LI Hong. Querying optimization algorithm for dynamic subgraph on time-evolving graph. Journal of Frontiers of Computer Science and Technology, 2014, 8(11): 1324-1333.

Abstract: Finding specific patterns in the time-evolving graph can help people effectively get hidden information in the data, so the dynamic subgraph query and index optimization have become a hotspot in the research of the evolving graph. This paper selects dynamic subgraph query as a research point, and discusses index optimization emphatically. This paper firstly gives the definition of the query model and the basic query algorithm. Then, this paper provides two different indexing methods, waveform index and binary tree index. To test the applicability of the index, this paper designs the corresponding experiments and uses randomly generated datasets to test experiments, while analyzing the efficiency from time consumption and space utilization. The experiments show that waveform index has the advantage of simple storage structure, which is suitable for the situation of long edge length but small edge number. Binary tree index has good performance in query speed, which is suitable for the situation of short edge length and large edge number.

Key words: querying optimization algorithm; time-evolving graph; dynamic subgraph; index optimization

* The National Natural Science Foundation of China under Grant No. 61070053 (国家自然科学基金); the Open Project of Shanghai Key Laboratory of Trustworthy Computing of China under Grant No. 07dz22304 (上海市高可信计算重点实验室开放课题).
Received 2014-07, Accepted 2014-09.
CNKI网络优先出版:2014-09-26, http://www.cnki.net/kcms/doi/10.3778/j.issn.1673-9418.1407045.html

摘　要: 挖掘时序图中的特定模式,能够有效地发现有价值的信息,并进行预测与决策支持,因此动态子图的查询及索引优化成为时序图研究的一个热点。研究了聚焦在动态子图的快速查询,着重探讨了索引优化,给出了查询模型的定义及基本查询算法。针对查询算法进行索引优化,提出了两种不同的建立索引的方法,波形索引及二叉树索引。为了验证索引的适用条件,设计了相应的实验,并使用随机数据集对实验程序进行测试,从时间消耗和空间占用的角度对两种索引的运行效率进行了验证分析。波形索引的优势在于存储结构简单,适用于边长度较长边数量不多的情况。二叉树索引的查询速度快,适用于边长度较短边数目较多的情况。

关键词: 查询优化算法;时序图;动态子图;索引优化

文献标志码: A　　**中图分类号:** TP391

1　引言

随着计算机技术的不断发展,人们每天都面对着不断增加的海量数据。在这些历史数据中[1],有很多是基于图表示的数据。例如:社交网络[2-4]的关系结构,Facebook网站的主要数据就是由约8亿顶点1 040亿边组成的图数据[5]。还有网页之间的超链接结构,互联网实际的覆盖与连接结构等数据都可以基于图来表示。近年来图数据的研究呈现出两种趋势:一是由于数据量越来越大,越来越多的研究者希望找出在大数据背景下挖掘图数据的有效方法;二是由于在许多实际应用中图中的边通常是随时间变化的,越来越多的研究者在静态图的基础上引入了时间维度,从而演化出时序图。

时序图是边和顶点可以随时间变化的图,它可以用一系列静态图的快照表示。时序图在现实生活建模中非常有用,诸如通讯网络、社交网络、蛋白质相互作用、商业合作网络等实际应用都能使用时序图这种数据结构来表示。

在计算机网络的覆盖与连接的分析中,网络拓扑结构的变化可以用时序图来表示。一个顶点代表一个工作站,一条边的存在代表在某一时刻两个工作站之间存在一条通信路径。通过这样的数据表示,可以挖掘出许多事件。例如:是否存在某些空间区域的连接是不同步的?有哪些连接路径在一个特定的时间段不稳定?回答这些问题可以更快地帮助故障定位,发现不稳定的网络区域。

在社交网络的数据分析中,用户与用户之间的关系结构可以用时序图表示。一个顶点代表一位社交网络的用户,一条边的存在代表在某一时刻两位用户之间存在互动关系。通过分析这些数据,可以找到许多有价值的信息。比如:哪些用户具有相同的关注兴趣?哪一组用户在某一个时刻组成紧密的联系网?回答这些问题可以很好地在社交网络中进行好友推荐和社会商务等。

在蛋白质相互作用的数据分析中,使用时序图可以帮助人们快速地测量成千上万的分子相互作用。使用高通量技术,将产生的数据集用网络结构表示,其中一个顶点代表一个蛋白质,一条边的存在代表相应蛋白质之间具有相互作用。同时因为研究需要,高通量的测量是需要在不同条件下不同时间点进行的,时序图[6]可以很好地体现出时间上的变化。

这些例子都是使用时序图来表示现实数据的,而在这些数据分析中,常常是试图发现时序图中的某些特定模式,这就需要通过查询实现动态子图的查找,从而发现图中的特定模式。例如:在计算机网络的覆盖与连接的分析中,要发掘网络连接不稳定的区域就要通过查询特定的动态子图来实现。在社交网络的数据分析中,要知道哪组用户在某一时刻或者时间段组成紧密的联系网,就需要通过查询得到特定的子图,它们的边在某个时刻或者时间段具有相同或相似的变化规律。由此可见,在时序图的数据分析中通过动态子图的查询来挖掘数据中隐藏的信息是一种很重要的方法。

与标准的数据库查询相比,时序图查询更具探究性。当探究时序图的查询时,用户首先有一个需要查询的子图。用户会给出初始的查询,反复迭代,分析所得到的结果,并且修改查询。因为时序图数据量通常很大,有效地评估查询效率是非常重要的,

所以越来越多的研究者关注动态子图查询的效率及优化。

本文选取动态子图的查询作为研究点，着重探讨查询索引优化问题。主要贡献是：

(1)定义查询模型，包括动态子图的定义、边的表示等，并给出相应的查询算法思想。

(2)针对查询算法进行优化，提供两种不同的建立索引的方法，分别分析两种索引方法的空间复杂度和时间复杂度，以及两种索引方法各自的优缺点。

(3)设计实验并编写实验程序，对实验结果进行测试，从时间消耗和空间占用的角度对比两种索引的运行效率。

2 相关工作

首先，介绍时序图查询上的相关工作。Chan等人[7]定义了时序图的相关域概念，指出时序图是以区域相关的方式进化的，并给出了一种名为cSTAG的算法，用于发现相关域识别动态图变化的标识事件。Kan等人[8]着重研究有效的索引方法来进行查询优化，对时序图、子图等进行了定义，给出了高效的动态子图查询算法，并提出了使用哈希表建立索引的优化方法。

其次，在时序图中添加多属性进行深入的挖掘。Jin等人[9]在时序图的研究过程中给顶点、边添加了权重属性，提出了一种识别子图趋势的挖掘算法，发掘一组循环的子图，顶点与边在特定的时间段里有相似的动态变化，呈现出一种相似的周期型变化趋势。根据这种识别方法给出了频繁趋势子图的挖掘算法。在实际应用中，每组动态子图能够揭示一个复杂系统中的重大事件，频繁的动态子图组则可以揭示一些系统的变化规则。有研究者将静态图中的查询算法进行修改并应用到时序图查询中。其中，Borgwardt等人将静态图查询中的CFS算法应用到了时序图中[10]，关注图的动态行为，即图的边随着时间的改变而发生的有序的插入与删除操作，同时给出了频率的定义和寻找频繁动态子图的算法。

但是，时序图上动态子图查询的时空优化与查询效率是一个急需解决的问题。因此，时序图查询研究聚焦在特定模式的查询，即动态子图查询和趋势发现等方面。在动态子图查询的研究中，主要着重于查询效率优化这个方面，如索引优化。而在趋势发现的研究中，则着重于频繁动态子图挖掘、趋势预测、周期行为发现等。

3 基本模型

本章定义基本的查询模型，并对时序图和输出的动态子图、关联动态子图、关联边给出正式的定义。

3.1 查询模型定义

在查询模型中，首先需要输入时序图数据，因此首先给出时序图、动态子图的定义。

定义1(时序图) 一个时序图是一组连续不断的静态图快照 $G_{ts}\cdots G_{te}$。这些静态图有着相同的一组顶点 V，但是在不同的时刻可能有不同的边 E_t，$t=ts\cdots te$。图1表示了一个时序图在3个时间点的状态。

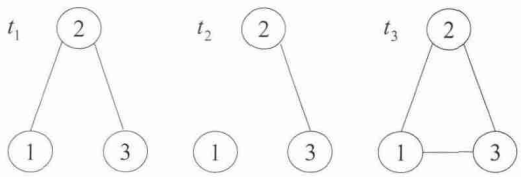

Fig.1 Evolution state of time-evolving graph

图1 时序图时间演变的状态

令 $E=\bigcup_{t=ts}^{te}E_t$ 为所有静态图快照边的并集，将时序图表示为 $eg=(V,E,ts,te,\varepsilon)$，$\varepsilon$ 是一组字符串，它用来表示每条边随时间的变化。对于 E 中的每条边 e，存在一个长度为 $te-ts+1$ 的字符串 $\varepsilon(e)$，如果边 e 在静态图快照 G_t 中存在，则记 $\varepsilon(e)[t]=$"1"，如果边 e 不在静态图快照 G_t 中，则记 $\varepsilon(e)[t]=$"0"。例如"1000100111"表示在0到9这个时间段里，在0、4、7、8、9这5个时刻图中存在边 e，在1、2、3、5、6这5个时刻图中不存在边 e。

定义2(动态子图) 给定时序图 $eg_1=(V_1,E_1,ts_1,te_1,\varepsilon_1)$ 和 $eg_2=(V_2,E_2,ts_2,te_2,\varepsilon_2)$，如果 $V_1\subseteq V_2$，$E_1\subseteq E_2$，$[ts_1,te_1]\subseteq[ts_2,te_2]$，并且在 $[ts_1,te_1]$ 这个时间段里，对于每一个 $e\in E_1$，ε_1 都包含 ε_2 的子串，则说 eg_1 是 eg_2 的动态子图，记为 $eg_1\subseteq eg_2$。一个子串必须是一个

连续匹配的序列，中间不可以有缺口。例如：对于“abcde”，“abc”、“cd”都是它的子串，而“ace”、“cba”就不是它的子串[11]。

查询模型是要寻找能够满足查询问题的特定类型的动态子图，将其定义为关联动态子图。即在一定的时间段里，动态子图所有的边都随着时间表现出相同的变化特征。希望能使用关联动态子图回答一个查询问题，即在一个时序图中要找到与某一种特定的模式匹配的关联动态子图。在本文的模型里，查询结果就是一系列的动态子图，它们满足每一条边在特定时刻的波形都与提供的查询波形相同，即每一条边在特定时刻的子串都与提供的查询字符串相同。

定义3（关联边）给定边 e 及时间点 t，对于任意的波形 W，如果从 t 时刻起长度与 W 相同的 e 的子串 $W_e=W$，则称 e 为波形 W 的关联边。

例如：e=“1000010001”，$t=5$，W=“10001”时，e 为波形 W 的关联边。e=“1000010001”，$t=3$，W=“10001”时，e 不是波形 W 的关联边。e=“1000010001”，$t=3$，W=“00100”时，e 为波形 W 的关联边。

定义4（关联动态图）给定一个动态图 $eg=(V,E,te,ts,\varepsilon)$，如果所有的 $e\in E$ 都为 W 的关联边，且 $G=(V,E)$ 为连通图，则称 eg 为 W 的关联动态图。

3.2　改进的查询算法

在传统的查询方法中，通常先找出与波形 W 匹配的所有动态子图，然后对所有动态子图进行筛选，找出若干个最大动态子图。这种算法的缺陷在于会产生许多中间动态子图，并且需要很多时间去检验它们，从而只留下最大动态子图。为了避免产生不符合条件的中间动态子图，节省筛选时间，给出一种新的算法。

改进的查询算法的核心思想是从一开始先寻找 W 的关联边，然后将所有的关联边组合成为若干最大连通图，得到若干最大动态子图。算法主要分为两步。

步骤1 找出所有 W 的关联边：

$sg_t \leftarrow CorrEdge(e,t,W)$

首先使用查询目标 W 对时序图中的每一条边逐一进行子串匹配的判断，然后将符合条件的边加入到关联边的集合中。这一步是算法最重要的步骤，因为查询关联边的效率很大程度影响了动态子图查询的效率。寻找关联边集合的时间复杂度主要取决于时序图边的数量及查询目标 W 的长度。若查询目标 W 的长度为 m，时序图边的数量为 n，则时间复杂度为 $O(mn)$。

步骤2 将找出的关联边划分为若干最大连通图 $P_t=Partition(sg_t)$，这一步在找出了关联边集合的基础上进行，将这些边划分为若干连通图。

```
QuerySubGraph (eg, W, t)
Input: eg, 待查询的时序图
       W, 查询目标字符串
       t, 查询的开始时刻
Description:
1. For each e in E do
2. 找到e从t时刻开始长度与W相同的子串et
3. if et == W
4.          //将符合条件的关联边加入集合
        sg_t ← CorrEdge(e, t, W)
5. else
              //继续匹配下一条边
              continue
6. //将符合条件的关联边的集合划分为若干最大连通图
   P_t = Partition(sg_t)
7. Output (P_t)
```

改进查询算法中，步骤1占据了大部分的时间消耗，算法的优化主要在这部分进行。因此，考虑在存储边时建立相应的索引从而进行查询优化。下面给出两种建立索引的方法，并对两种索引分别对算法运行效率产生的影响进行比较分析。

4　查询索引优化

图数据索引方法常常可以归纳为两个类：一类是基于子图的索引，例如Yan等人[12]提出的gIndex索引。使用算法gIndex选择那些具有较高过滤能力的频繁子图作为索引项，这些被选定的索引项被称为特征。文中选择一部分频繁子图作为索引项可以有效地提高过滤能力。另一类是基于边的索引，例如

Kan提出了使用哈希表对边的存储建立索引，加快时序图查询过程中边的定位。算法主要是对关联边的查询进行索引优化，索引基于边建立。在查询关联边时，需要使用查询目标 W 对时序图中的每条边都进行匹配判断。

索引优化方法可以从两个角度来考虑：一是考虑事先过滤掉明显不符合要求，不需要进行匹配的边，这样可以大幅度减少边匹配判定的次数；二是寻找能够在众多边信息中快速定位到查询目标的方法，那么关联边的查询效率就可以大幅度提高。接下来将根据这两个优化角度给出具体的索引方法。

4.1 波形索引

在介绍这种索引之前，首先给出转变序列的概念。转变序列可以概括整个波形的变化过程，因为在波形中只有“1”和“0”两种状态，所以当波形由“0”变为“1”时，可以记为“+”，当波形由“1”变为“0”时，可以记为“-”。

例如：波形“11111111”的转变序列为空；波形“01111110”的转变序列为“+-”；波形“00001111”的转变序列为“+”。

建立索引将边按照转变序列进行排序，而这个索引的建立会在对图进行存储的时候进行，即在存储边的时候就进行计算和顺序插入操作。

```
WaveIndexSearch (eg, t, W)
Input: eg, 待查询的时序图
       W, 查询目标字符串
       t, 查询的开始时刻
Description:
1. For each e in E do
2.   //计算每一条边的转变序列
     cal=CalTrans(e)
3.   if cal in Outertable
     //将边加入转变序列相对应的内表中
       Innertable(cal)←e
4.   else //将新出现的转变序列加入外表
           Outertable←cal
           Innertable(cal)←e
5.   Input W
6.   //计算查询目标的转变序列
     calW=CalTrans(W)
7.   for each trans in Outertable
//查询目标的转变序列为转变序列的子串
       if calW in trans(i)
8.   //遍历转变序列，找出符合条件的关联边
         loop Innertable(trans(i))
           sg_t ← CorrEdge(e, t, W)
9.       else
           Continue
```

波形索引的基本思想是将边的“波形”（即转变序列）作为一个特征过滤项，通过事先过滤掉明显不符合条件的转变序列而减少查询目标 W 与边的匹配次数。

对波形索引的具体步骤通过实例来进行解释。例如：对于以下这组边，在一边存储时一边就会进行转变序列的计算。当计算出转变序列后，将转变序列进行排序，使用转变序列对所有的边进行索引，每一个转变序列后面存储若干对应相应转变序列的边的波形。这样索引由两张表组成，将存储转变序列的表称为外表，存储所有边的波形的表称为内表。

Table 1 Internal and external tables corresponding

表1 内外对应关系表

外表	内表
	11111111111
-	11100000000
-+	10000000001
	10000011111
	10000111111
	11100000011
-+-	11100011000
	11110111100
-+-+	10011001111
	11000100011

(1)给定查询波形 W=“011110”，此时波形 W 的转变序列为“+-”。

(2)使用 W 的转变序列“+-”在外表中进行转变序列匹配，通过匹配得到在表1中只有”-+-”和“-+-+”中包含”+-”这个转变序列。

(3)在“-+-”和“-+-+”后面所指的边上对 W 进行匹配判定操作，得到边“11110111100”为符合条件的关联边。

在没有波形索引的情况下，在这个实例中关联边集合的寻找需要进行10次字符串匹配，而加入了波形索引后只需要进行4次字符串匹配。

可见波形索引是通过转变序列来过滤掉明显不符合条件的波形及其后面存储的相应的边，大幅度减少了字符串匹配的次数，从而提高了寻找关联边的效率。在没有使用波形索引的情况下，给定波形 W，在查询关联边时，需要对所有边的波形进行一次字符串匹配操作。但是当使用波形索引存储之后，给定波形 W，先计算其转变序列，则外表中所有与 W 的转变序列不匹配项之后所存储的边都会被过滤掉。

一般情况下波形索引能够表现出比较好的性能，但是在某些特殊情况下，波形索引会带来额外的时间消耗或是较为薄弱的过滤能力。当边数据的规律性较弱，具有相同转变序列的边不多时，外表中每一个转变序列后面所跟的边数据就会很少。这意味着计算出的转变序列较多，此时波形索引会为算法带来额外的时间消耗，因为产生了过多的 W 的转变序列与外表转变序列之间的匹配。同时每次筛选掉一个不符合条件的转变序列时过滤掉的边很少，波形索引的过滤能力会大大降低。所以波形索引尤其适用于边数据规律性较强的时序图。因为它的基本思想是事先过滤转变序列不符合查询目标的边，当一条转变序列可以指向的边的数量较大时，能够展现出比较好的过滤能力。

4.2　二叉树索引

二叉树索引虽然是使用树存储的，但是与以往的一些基于树存储的索引不同。Zhang等人提出的TreePi索引[13]以及Zhao等人提出的频繁子树索引[14]都是基于图数据的拓扑结构而建立的索引。而本文所给出的二叉树索引是基于时序图的边所建立的索引，目的是提高关联边查询时的定位效率。

二叉树索引是指用二叉树这个数据结构来存储时序图的边。因为在任意时刻，本文模型的时序图中边只有两种状态，即存在记为“1”，不存在记为“0”，所以在二叉树中可以令左儿子表示边存在(即波形索引中的字符“1”)，右儿子表示边不存在(即波形索引中的字符“0”)。

二叉树索引的核心思想在于并不存储每一条边对应的波形，而是使用一个二叉树表示所有可能的波形，每当需要存储一条边时，在二叉树上搜索相应的路径，将其存储在相应的终点节点上。通过树来存储边信息，相当于为每条边都提供了相应的路径，且路径相同的边会被存储在同一个叶子节点上。这样无需将 W 与所有的边进行匹配，只需定位 W 的路径，寻找相同路径的边即可。二叉树索引是直接对查询目标进行路径定位，其查询时间不受边数目的影响，时间复杂度为 $O(1)$。这是通过快速定位方法来提高查询效率。具体算法如下：

TreeIndexSearch(eg, t, W)

Input: eg, 待查询的时序图

　　W, 查询目标字符串

　　t, 查询的开始时刻

Description:

1. //获取时序图的时间长度(边字符串长度)

　n = GetElength()

2. 　　//创建一个 n 层的二叉树

　CreateTree(n)

3. 　　for each e in E

4. //将每条边加到相应路径的终点节点上

　　addeToTree()

5. Input W, t

6. //找到第 t 层的节点，即从时间点 t 开始

　$Nodestart \leftarrow findtreeN(t)$

7. 　　for each Nodestart

8. //按照 W 的路径找到路径终点

　　SearchPath(W)

9. //路径终点出发得到的所有叶子节点中所存的边即为 W 的关联边

　$sg_i = getFinalNode()$

使用一个简单的实例对算法进行解释。例如：W_1 = “111”，W_2 = “101”，W_3 = “100”，W_4 = “011”，W_5 = “010”，W_6 = “101”，W_7 = “011”。二叉树索引实例如图2所示，7条边的路径如表2所示。

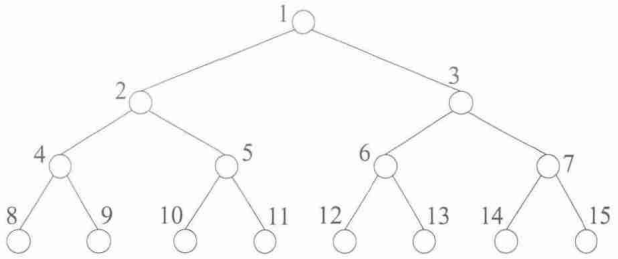

Fig.2 Index instance of binary tree

图2 二叉树索引实例

Table 2 Paths and edge sets table

表2 路径与边对照表

终点节点	路径	字符串	存储的边
8	1-2-4-8	111	W_1
9	1-2-4-9	110	—
10	1-2-5-10	101	W_2, W_6
11	1-2-5-11	100	W_3
12	1-3-6-12	011	W_4, W_7
13	1-3-6-13	010	W_5
14	1-3-7-14	001	—

在二叉树中的每一层代表一个时刻，假如需要查询时刻0~2之间波形W="01"的边，只需要从时刻0的节点（时刻0为节点1）开始搜索"01"这个路径，即1-3-6，然后在这个路径下的所有叶子节点里所存储的边即为查询结果。如图2，路径1-3-6的叶子节点为节点12和节点13，因此查询符合条件的边为节点12、节点13后面分别存储的W_4、W_7和W_5。

二叉树索引的特点为：当时序图的时间长度一定，即时序图边的长度一定时，二叉树索引占用的存储空间相近。因为若时间长度为n，则二叉树索引就为一个n层的二叉树。针对这个特点，可以得知二叉树索引适用于时间长度较短、边数目较多的时序图查询。因为若时序图的时间长度太长，二叉树索引的空间复杂度为$O(2^n)$，时间长度大的情况下会造成空间存储的负担；同时当边数目较多时，二叉树同一个路径下存储的边数目才会较多，存储路径而非边才会显示出优势，否则当边数目较少时，二叉树较为稀疏，不仅不会显示出路径定位的优势，还会造成空间的浪费。

5 实验分析

5.1 实验设计

为了对比波形索引和二叉树索引这两种索引方法对算法效率的影响，使用C++语言实现了这两种索引方法，并对多种动态子图查询的约束条件进行了实验测试。本文使用不同数据规模的数据集对两种索引方法的效率进行验证。

实验输入：节点数目，边数目，时间长度（组成时序图的静态图的数量），查询目标（需要查询的边的模式），查询时刻（查询的起始时间）。

实验输出：生成的边信息，查询结果，波形索引输入下的运行时间及占用空间，二叉树索引在输入下的生成时间、运行时间及占用空间。

5.2 结果分析

从时间消耗及空间占用这两个方面对实验结果进行分析。使用几组不同的数据分别对两种索引方法进行实验，对比它们的时间消耗和空间占用。

在实验数据的变量中，关联边的查询主要是以边匹配的方式进行的，因此边的数目成为影响查询效率的重要衡量指标。首先研究随着边数目的增加，两种索引的时间效率及空间占用的变化。对两种索引方法随边数目的变化而产生的时间消耗与空间占用进行了对比分析。为方便对比，将节点数目、边长度、查询目标、查询时刻都设为固定值。

约束条件1 节点数目=1 000，边长度=10，查询目标="10001"，查询时刻=3。

约束条件2 节点数目=1 000，边长度=16，查询目标="10001"；查询时刻=3。

两种索引方法随边数目的变化而产生的时间消耗与空间占用的对比分析实验结果如图3~图6所示。

图3是约束条件1下索引的时间效率对比结果。从实验结果中可以看出，波形索引的运行时间及二叉树索引的生成时间与边数目之间都呈现出线性关系。而二叉树索引是直接对查询目标进行路径定位，因此其查询时间不受边数目的影响，时间复杂度为$O(1)$。但这并不意味着二叉树索引相对于波形索引具有绝对的优势。

图4是约束条件2下索引的时间效率对比结果。从实验结果中可以看出，二叉树索引的生成时间远远高于波形索引的查询时间。由于边的长度决定了二叉树索引的层数，二叉树索引的生成时间随着边长度的变化呈现指数增长，边长度较长时，二叉树索引的生成时间会成为主要的瓶颈。

从以上的对比中得出以下结论：边长度较短且查询次数较为频繁的情况下，适合使用二叉树索引来进行查询优化。边长度较长的情况下，若查询次数不频繁，此时生成二叉树就需要占用很大的时间消耗，因此选择波形索引进行查询优化更为合适。

从图5和图6的实验结果中可以看出，波形索引的空间占用主要取决于边的数量，二叉树索引的空间占用主要取决于边的长度。当边的数量不断增加时，波形索引的空间占用呈现出线性增长，但是二叉树索引的空间占用基本不受边数量的影响，因为一旦边的长度确定，二叉树的层数就确定了，其空间占用会随着边长度的增长呈现出指数形式的变化。

通过对比可以得出以下结论：

(1)当边长度较小、边数量较多时，二叉树索引具有比较大的存储优势。

(2)当边数目较少、边长度较长时，不适合使用二叉树索引。因为二叉树索引的空间占用会随着边长度的增长呈现出指数形式的变化，且在同样的时间长度设定下，即便边数目较少，二叉树索引相对于波形索引仍然需要占用较大的存储空间。

为了研究边长度(时序图时间长度)对索引优化产生的影响，以边长度为变量设置以下实验，对两种索引方法随边长度的变化而产生的时间消耗进行了对比分析。

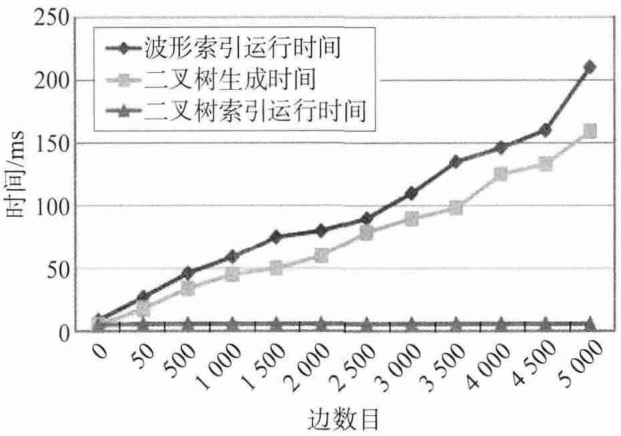

Fig.3　Time efficiency comparison under constraint 1

图3　约束条件1下索引的时间效率对比

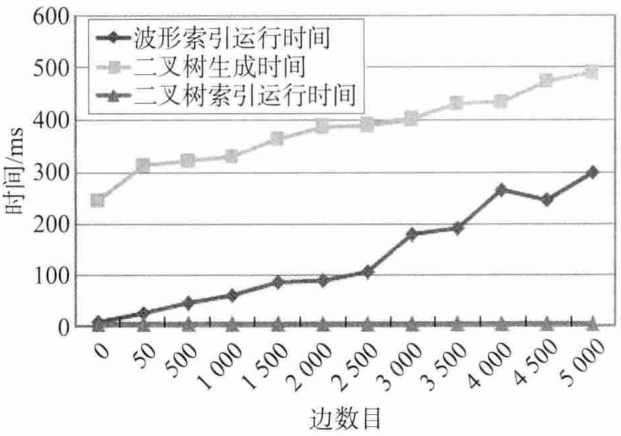

Fig.4　Time efficiency comparison under constraint 2

图4　约束条件2下索引的时间效率对比

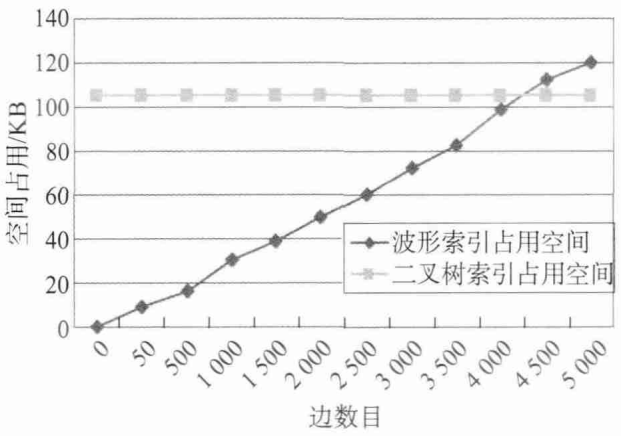

Fig.5　Space efficiency comparison under constraint 1

图5　约束条件1下索引的空间效率对比

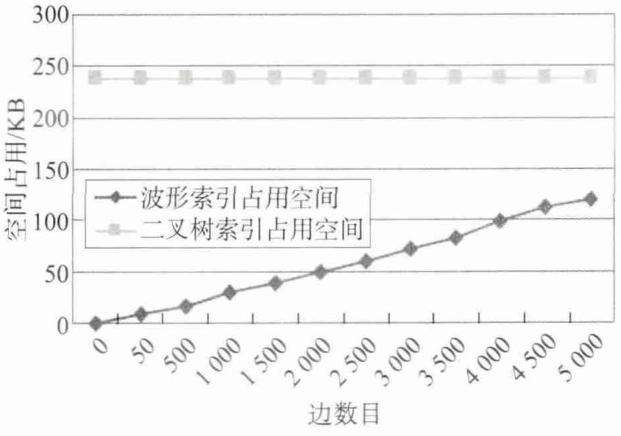

Fig.6　Space efficiency comparison under constraint 2

图6　约束条件2下索引的空间效率对比

约束条件3 为方便对比，将节点数目、边数目、查询目标、查询时刻都设为固定值，节点数目=1 000，边数目=3 000，查询目标="10001"，查询时刻=3。

实验结果如图7所示，基于时序图时间长度的对比，可以明显地看出，时序图时间长度对二叉树索引的效率影响较大。它对二叉树索引的时间效率影响主要体现在生成二叉树的阶段，该阶段随着时间长度的增长呈现出指数增长趋势，这与之前所分析的时间复杂度是一致的。因此当边长度较长时，二叉树索引会在生成时间上遇到较大的瓶颈，此时选择波形索引较为合适。

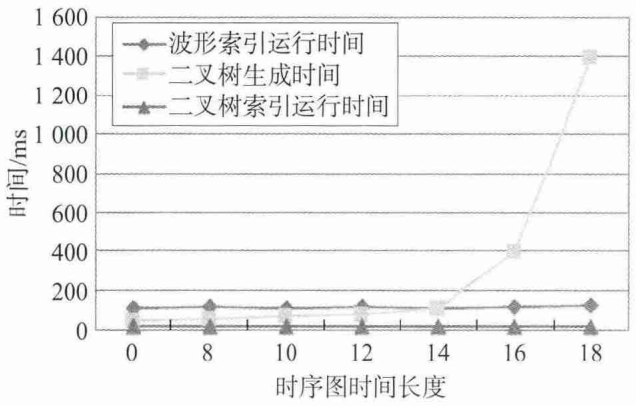

Fig.7 Time efficiency comparison under constraint 3

图7 约束条件3下索引的时间效率对比

综上所述，两种索引方法的特点及适用情况对比：波形索引主要的原理是通过事先过滤转变序列中明显不符合条件的边，从而减少匹配判定的次数。波形索引的优势在于存储结构简单，边信息由字符串结构进行存储，因此即使边长度（时序图的时间长度）较长的情况下，也不会为波形索引的存储带来问题。因此波形索引更适用于边长度较长但是边数量不多的情况。二叉树索引的主要特点是通过二叉树的结构来存储边的路径。二叉树索引并不存储每条由字符串表示的边信息，而是通过查询目标 W 的路径来进行查询定位，因此它的查询速度非常快。二叉树索引的主要问题在于二叉树的生成及存储，当边长度过长时，二叉树索引所需要的存储空间远远高于波形索引。因此二叉树索引适用于边长度较短、边数目较多的情况。

6 总结与展望

本文研究了时序图动态子图查询算法的索引优化，给出了新的查询算法，提出了先寻找关联边集合后生成动态子图的方法。同时针对寻找关联边这一最耗时的步骤提出了波形索引和二叉树索引两种索引优化方法。波形索引是通过使用转变序列对查询边进行过滤，减少匹配判定的次数而达到优化的效果。二叉树索引是通过二叉树的结构来将同一路径的边存储在一起，然后查询目标通过路径定位找到相应的关联边。通过实验，采用多组数据对时间消耗和空间占用进行了对比分析，用折线图的形式展现其变化趋势，分析了两种索引的特点及其适用情况。

在大图研究中，时序图上动态子图查询算法的时空优化是最具有挑战性的。图查询算法中的精确匹配查询与近似匹配检索的选取是一个值得研究的问题。例如，查询过程中的字符串子串匹配不一定需要是完全精确匹配，在实际应用中可以引入欧几里得空间距离进行近似匹配。因为在实际应用中"10000001"和"100001"有时候可以表示相同的状态。比如在互联网连接与覆盖数据中，它们都可以表示原有的连接出现了故障并在一段时间后恢复。连续的"0"串就可以表示故障，不一定需要满足"0"的个数完全相同。另外，关联边的查询不仅可以停留在查询的层面，还可以深入挖掘频繁边的波形变化，在进行查询操作时，可以让频繁边优先进行匹配，改进已有的FM索引[15-16]，提高查询效率。

References:

[1] Leskovec J, Kleinberg J M, Faloutsos C. Graphs over time: densification laws, shrinking diameters and possible explanations[C]//Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD '05). New York, NY, USA: ACM, 2005: 177-187.

[2] Lahiri M, Berger-Wolf T. Mining periodic behavior in dynamic social networks[C]//Proceedings of the 8th IEEE International Conference on Data Mining (ICDM '08). Washington, DC, USA: IEEE Computer Society, 2008: 373-382.

[3] Backstrom L, Huttenlocher D P, Kleinberg J M, et al. Group

formation in large social networks: membership, growth, and evolution[C]//Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD '06). New York, NY, USA: ACM, 2006: 44-54.

[4] Palla G, Barabasi A-L, Vicsek T. Quantifying social group evolution[J]. Nature, 2007, 446(7136): 664-667.

[5] Kuramochi M, Karypis G. Grew—a scalable frequent subgraph discovery algorithm[C]//Proceedings of the 4th IEEE International Conference on Data Mining (ICDM '04). Washington, DC, USA: IEEE Computer Society, 2004: 439-442.

[6] Khurana U, Deshpande A. Efficient snapshot retrieval over historical graph data[C]//Proceedings of the 2013 IEEE 29th International Conference on Data Engineering (ICDE '13). Piscataway, NJ, USA: IEEE, 2013: 997-1008.

[7] Chan J, Bailey J, Leckie C. Discovering correlated spatio-temporal changes in evolving graphs[J]. Knowledge and Information Systems, 2008, 16(1): 53-96.

[8] Kan A, Chan J, Bailey J, et al. A query based approach for mining evolving graphs[C]//Proceedings of the 8th Australasian Data Mining Conference (AusDM '09). Melbourne: Australian Computer Society, 2009, 101: 139-150.

[9] Jin R, McCallen S, Almaas E. Trend motif: a graph mining approach for analysis of dynamic complex networks[C]// Proceedings of the 7th IEEE International Conference on Data Mining (ICDM '07). Washington, DC, USA: IEEE Computer Society, 2007: 541-546.

[10] Borgwardt K M, Wackersreuther P. Pattern mining in frequent dynamic subgraphs[C]//Proceedings of the 6th International Conference on Data Mining (ICDM '06). Washington, DC, USA: IEEE Computer Society, 2006: 818-822.

[11] Gusfield D. Algorithms on strings, trees and sequences: computer science and computational biology[M]. Cambridge: Cambridge University Press, 1997.

[12] Yan Xifeng, Yu P S, Han Jiawei. Graph indexing: a frequent structure-based approach[C]//Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD '04). New York, NY, USA: ACM, 2004: 335-346.

[13] Zhang Shijie, Hu Meng, Yang Jiong. TreePi: a novel graph indexing method[C]//Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering (ICDE '07). Piscataway, NJ, USA: IEEE, 2007: 966-975.

[14] Zhao Peixiang, Yu J X, Yu P S. Graph indexing: tree+delta>= graph[C]//Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB '07). [S.l.]: VLDB Endowment, 2007: 938-949.

[15] Simpson J T, Durbin R. Efficient construction of an assembly string graph using the FM-index[J]. Bioinformatics, 2010, 26(12): i367-i373.

[16] Fernandez E, Najjar W, Harris E, et al. String matching in hardware using the FM-index[C]//Proceedings of the 2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM '11). Piscataway, NJ, USA: IEEE, 2011: 218-225.

ZHU Qing was born in 1963. She received the Ph.D. degree in computer science and technology from Renmin University of China. Now she is an associate professor at Renmin University of China, and the senior member of CCF. Her research interests include cloud computing and big data, subgraph query computing, high-performance database, information retrieval and graph mining, etc.

朱青(1963—)，女，江苏无锡人，中国人民大学博士，中国人民大学信息学院计算机系副教授，CCF高级会员，主要研究领域为云计算与大数据，子图查询计算，高性能数据库，信息检索，图挖掘等。

LI Hong was born in 1990. Her research interests include subgraph query computing and querying optimization algorithm of time-evolving graph, etc.

李红(1990—)，女，上海人，主要研究领域为子图查询算法，时序图查询算法等。