doi:10.3772/j.issn.1002-0470.2017.11 – 12.003

基于顶点关联索引的最短路径查询算法研究①

余 靖②*,**,*** 杨清章*,**,***

(*燕山大学信息科学与工程学院 秦皇岛 066004)

(**河北省计算机虚拟技术与系统集成重点实验室 秦皇岛 066004)

(***河北省软件工程重点实验室 秦皇岛 066004)

摘 要 研究了图查询中的最短路径查询问题,针对现有的查询算法存在构建索引时间长和索引规模庞大所导致的低效性和扩展性问题,在索引构建方面提出了顶点关联索引策略。对度数为 1 的顶点构建顶点关联索引,对其他顶点构建 2-hop 标签索引,通过减少冗余数据存储和图的遍历次数,降低索引规模以减少构建索引时间。基于所提出的查询策略,给出了基于顶点关联关系和 2-hop 标签的最短路径查询算法。

关键词 图模型,最短路径查询,顶点关联索引,2-hop 标签索引

0 引 言

随着大数据和互联网迅速发展,越来越多的数据都用图模型来表示。图模型不仅可以应用在城市交通网络、生物信息网络、社交网络[1-3]中,而且可以应用在可扩展标记语言(XML)数据库[4]和语义网络中。近年来,在图模型上进行高效的查询和分析操作引起了数据库社区的重点关注。

最短路径查询在图查询领域中扮演着最重要也最基础的角色,与图相关的许多问题都应用到了最短路径查询操作,如最近邻查询[5,6]、k 步可达性查询[7,8]和 top-k 关键字查询[9,10]等。在生物信息网络中,最短路径查询可以应用于发现化合物之间的最佳路径,在计算机网络中,最短路径查询可以应用于计算机网络资源管理,在网络图上,最短路径查询可以应用于判断网页之间的相关性。与可达性查询相比,最短路径查询因为需在所有可达路径中计算一条最短路径,所以比可达性查询更为复杂。目前最短路径查询算法采用的是索引-查询策略,首先在原数据图上构建索引,随后在索引上回答顶点之间的最短路径或最短距离。这样的最短路径查询算法面临着索引规模庞大和构建索引时间长的问题,导致大多数算法在处理大规模图时,性能会急剧下降,甚至无法运行。

基于此问题,本文提出了顶点关联索引策略,并在此基础上给出了相应的最短路径查询算法,以解决图查询处理大规模图时所遇到的问题。

1 最短路径查询方法

1.1 基本的最短路径查询方法

基本的最短路径查询方法有广度优先搜索(breadth first search, BFS)算法、Dijstra 算法[11]、A-Star 算法[12]、Bellman-Ford 算法、Johnson 算法等,其中 BFS 算法解决无权图中的最短路径问题,Dijstra 算法、A-Star 算法、Bellman-Ford 算法以及 Johnson 算法解决加权图中的最短路径问题。本文简单介绍 BFS 算法。

BFS 算法是最简单的图查询算法之一,在许多

① 国家自然科学基金(61572421)资助项目。

② 女,1976 年生,博士,副教授;研究方向:空间数据库,数据挖掘等;联系人,E-mail: xyyj@ysu.edu.cn

(收稿日期:2017-07-22)

经典的图算法中都采用了与 BFS 算法相同的思想，例如，Prim 最小生成树算法和 Dijstra 算法。BFS 算法属于单源点最短路径算法，即，在数据图 G 中，若给定一个源点 s，BFS 算法可以解决从源点 s 到其他任意顶点之间的最短路径问题。BFS 算法系统地访问图中的所有边，找到从源点 s 可达的所有顶点，同时计算源点 s 到这些可达顶点的最短距离（即最少的边数）。该算法同时还构造一棵广度优先生成树，树的根节点是源点 s，非根节点由源点 s 可达的顶点组成。在广度优先生成树中，任意一条从源点 s 开始的路径，都是一条从源点 s 到其它顶点 v 的最短路径。BFS 算法不仅适用于无向图，同时也适用于有向图。BFS 算法总的运行时间为 $O(|V| + |E|)$，随着数据图的规模不断增大，查询时间成线性增长，这个查询时间对于现实生活中的许多应用是无法接受的。

1.2 基于图索引的最短路径查询方法

目前广泛使用 2-hop 标签索引来处理最短路径查询问题，基于 2-hop 标签索引的方法有 IS-Label 算法[13]和锁相环（phase locked loop，PLL）算法[14]。

IS-Label 算法需要在图中计算一个最大独立集，在图中计算最大独立集本身就是一个 NP 完全问题，使用近似方法在图中计算一个近似最大独立集，因为找一个近似最大独立集的时间代价也很高，所以该方法构建索引时间很长。在构建 2-hop 标签索引时，因为对每个顶点都需要求出它的所有标签，所以使用该方法构建的索引规模很大。

PLL 算法首先把图中每个顶点作为起始顶点，从每个起始顶点开始对图进行遍历操作，广度优先遍历计算出当前顶点 v 到顶点 u 的最短距离 d_1，通过 QUERY(v, u, H')计算出顶点 v 到顶点 u 的最短距离 d_2，其中 H' 表示已经构建的 2-hop 标签索引，若 d_1 小于 d_2，则在 $H(u)$ 标签中添加二元组(v, d_1)，若 d_1 大于等于 d_2，则不添加，按照上述过程广度优先遍历图，直到从顶点 v 开始遍历完整个图为止，算法结束的条件是图中每个顶点都进行过一次广度优先搜索，在处理大规模图时，由于 PLL 算法基于内存，对于每一个点的宽度优先搜索和剪枝过程需要暂时保存在内存中，无法在给定的内存空间中存储 2-hop 标签索引，因此该算法构建索引时间长和索引规模庞大，不适用于处理现在的大规模网络，这也会影响它的扩展性。

2 基于顶点关联索引的最短路径查询算法

针对 PLL 算法中存在构建索引时间长和索引规模庞大的问题，提出了一种索引构建策略，对度数为 1 的顶点构建顶点关联索引，对其它顶点构建 2-hop标签索引。顶点关联索引结合 2-hop 标签索引便可以查询图中任意顶点对之间的最短路径，同时提出构建索引的 One _ Degree 算法以及基于顶点关联索引的最短路径查询算法 One _ Degree _ Query 算法。

2.1 索引构建策略

定义 1（悬挂点）：在图 $G = (V, E)$ 中，把度数为 1 的顶点称作悬挂点，并把悬挂点唯一的邻接点称作悬挂点的关联顶点，把图中悬挂点的集合记为 V^*。

定义 2（非悬挂点）：在图 $G = (V, E)$ 中，把除悬挂点以外的顶点称作非悬挂点。

定义 3（顶点关联索引）：在图 $G = (V, E)$ 中，把 $R(v) = u$ 的集合称作顶点关联索引，其中 v 表示图中每一个悬挂点，u 表示悬挂点 v 的关联顶点。

根据定义 1 ~ 3，图 1 中顶点 v_6、v_7、v_8 和 v_9 是悬挂点，悬挂点 v_6 的关联顶点是 v_5，悬挂点 v_7 的关联顶点是 v_0，悬挂点 v_8 的关联顶点是 v_4，悬挂点 v_9 的关联顶点是 v_1。图中顶点 v_0、v_1、v_2、v_3、v_4、v_5 是非悬挂点。对图 1 中的悬挂点构建的顶点关联索引如表 1所示。

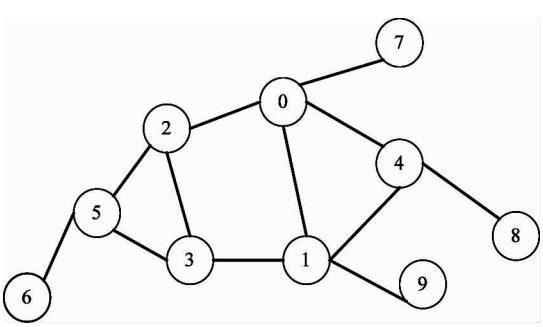

图 1 无向无权图 G_1

表 1　图 G_1 的顶点关联索引

v	$R(v)$
v_6	v_5
v_7	v_0
v_8	v_4
v_9	v_1

数据图 G'_1 是数据图 G_1 删除悬挂点和其相应边之后的图,如图 2 所示。

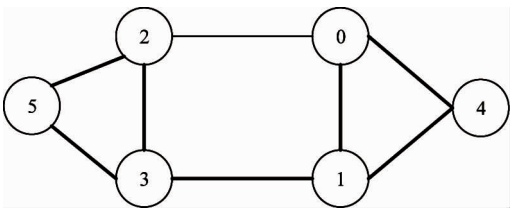

图 2　无向无权图 G'_1

createIndex 算法(算法 1)是对不包含悬挂点的数据图 G'_1 构建 2-hop 标签索引的算法,该算法的输入是数据图 G'_1,输出是数据图 G'_1 的 2-hop 标签索引。

算法 1: createIndex (G)
1. $H \leftarrow$ empty 2-hop index
2. sort in $V(G)$ by degree
3. for each $v \in V(G)$ do
4. $Q \leftarrow$ a queue with only one element v
5. $d[v] \leftarrow 0$ and $d[u] \leftarrow \infty$ for all $u \in V(G) \setminus \{v\}$
6. while Q is not empty do
7. Dequeue u from Q
8. if QUERY (v, u, H') $\leqslant d[u]$ then continue
9. $H[u] \leftarrow H[u] \cup \{(v, d[u])\}$
10. for all $w \in N(u)$ and $d[w] = \infty$ do
11. $d[w] \leftarrow d[u] + 1$
12. Enqueue w in Q
13. return H

首先把 2-hop 标签索引 H 初始化为空,在数据图 G 中,按照度数由大到小的顺序遍历图中的每一个顶点 v,首先初始化一个队列 Q 并把顶点 v 入队列,初始化一个距离数组d,数组中除了$d[v]$的值是零外,其他每个元素的值都是无穷大,数组中元素的值表示当前顶点 v 到 u 的最短距离。判断队列 Q 是否为空,如果不为空,则让队头元素出队列,并把该元素记录为 u,判断 QUERY (v, u, H') 值和$d[u]$值的大小,其中 QUERY(u, v, H')值是通过已经构建的2-hop标签索引计算出的 u 和 v 之间的最短距离,如果 QUERY(u, v, H')值小于等于 $d[u]$值,则退出本次循环,继续判断下一个顶点,表示顶点 u 和 v 之间的最短路径值可以通过已经构建的 2-hop 标签索引来计算,并不需要向 $H(u)$标签中添加二元组(v, $d[u]$),否则继续执行,接下来向 $H[u]$中添加二元组(v, $d[u]$),其中 v 表示图遍历的起始顶点,$d[u]$表示顶点 v 和 u 之间的最短距离。在第 10 行中访问顶点 u 的未访问的邻接点 w,未访问邻接点的 $d[w]$值初始化为无穷大,在访问 w 时,把$d[w]$值更新为 $d[u]$值加 1,同时将 w 入队列 Q,最后返回 2-hop 标签索引,为图 G'_1 构建的 2-hop 标签索引如表 2 所示。

表 2　图 G'_1 的 2-hop 标签索引

$H(v_0)$	$H(v_1)$	$H(v_2)$	$H(v_3)$	$H(v_4)$	$H(v_5)$
$(v_0,0)$	$(v_0,1)$	$(v_0,1)$	$(v_0,2)$	$(v_0,1)$	$(v_0,2)$
	$(v_1,0)$	$(v_2,0)$	$(v_1,1)$	$(v_1,1)$	$(v_1,2)$
			$(v_2,1)$	$(v_4,0)$	$(v_2,1)$
			$(v_3,0)$		$(v_3,1)$
					$(v_5,0)$

而使用 PLL 算法构建 2-hop 标签索引如表 3 所示。

悬挂点的顶点关联索引只存储它的关联顶点,而 PLL 算法需要为图中每个顶点 v 都分配一个 $H(v)$标签,根据顶点之间的关联关系,没必要为悬挂点 v 分配 $H(v)$标签,因此使用顶点关联索引结合 2-hop标签索引可以降低索引规模。例如在表 3 中,$H(v_6)$标签由 6 个二元组构成,若每个二元组由 2 个整型数表示,则 $H(v_6)$标签由 12 个整型数组成,而悬挂点 v_6 的顶点关联索引 $R(v_6)$只存储一个整型数。因此由顶点关联索引和 2-hop 标签索引组成的索引比 PLL 算法构建的 2-hop 标签索引规模要小。

表 3 PLL 算法的 2-hop 标签索引

$H(v_1)$	$H(v_2)$	$H(v_3)$	$H(v_4)$	$H(v_5)$	$H(v_6)$	$H(v_7)$	$H(v_8)$	$H(v_9)$
$(v_0,1)$	$(v_0,1)$	$(v_0,2)$	$(v_0,1)$	$(v_0,2)$	$(v_0,3)$	$(v_0,1)$	$(v_0,2)$	$(v_0,2)$
$(v_1,0)$	$(v_2,0)$	$(v_1,1)$	$(v_1,1)$	$(v_1,2)$	$(v_1,3)$	$(v_7,0)$	$(v_1,2)$	$(v_1,1)$
		$(v_2,1)$	$(v_4,0)$	$(v_2,1)$	$(v_2,2)$		$(v_4,1)$	$(v_9,0)$
		$(v_3,0)$		$(v_3,1)$	$(v_3,2)$		$(v_8,0)$	
				$(v_5,0)$	$(v_5,1)$			
					$(v_6,0)$			

One _ Degree 算法的伪代码如算法 2 所示。该算法的输入是数据图 G,输出是数据图 G 的索引。将顶点关联索引 R 初始化为空,依次遍历图中的每一个顶点 v,判断当前顶点 v 的度数是否为 1,如果顶点 v 的度数为 1,则把顶点 v 的邻接点赋值给顶点关联索引 $R[v]$,删除顶点 v 以及包含顶点 v 的边;如果顶点的度数不为 1,则退出本次循环,继续判断下一个顶点;接下来生成一个新的数据图 G',数据图 G'是数据图 G 删除悬挂点以及悬挂点相邻边之后得到的图,即数据图 G'中不包含任何悬挂点。调用 createIndex 算法,为数据图 G'构建 2-hop 标签索引,合并顶点关联索引 R 和 2-hop 标签索引 H。返回 One _ Degree 算法构建的索引。

算法 2: One _ Degree (G)
1. $R \leftarrow$ empty relation index
2. for each $v \in V(G)$ do
3. if degree$[v] = 1$
4. $R[v] = Adj[v][0]$
5. delete v and edge $(v, Adj[v][0])$
6. $G' \leftarrow$ delete vertex and edge in G
7. $H \leftarrow$ create _ index (G')
8. $I \leftarrow R \cup H$
9. return I

One _ Degree 算法构建的索引分为两部分,第一部分是在悬挂点上构建的顶点关联索引,第二部分是在非悬挂点上构建的 2-hop 标签索引。PLL 算法需要对图中的每个顶点构建 2-hop 标签索引,而 One _ Degree 算法采用顶点关联索引策略,使得悬挂点 v 的顶点关联索引必定小于其 2-hop 标签索引的大小,因为顶点关联索引只存储一个整型数,而 2-hop 标签索引至少存储两个整型数,由此可得,One _ Degree 算法减少了冗余数据的存储从而降低了索引规模。One _ Degree 算法在构建 2-hop 标签索引时,需进行 $|V - V^*|$ 次图遍历操作,而 PLL 算法需进行 $|V|$ 次图遍历操作,其中 $|V - V^*|$ 表示图中非悬挂点的数目,因此 One _ Degree 算法减少了图遍历次数从而缩短了索引构建时间。

2.2 最短路径查询算法

定理 1:设图 G 是无权图,$s \in V - V^*$ 为图 G 中的任意一个非悬挂点,$t \in V^*$ 为图 G 中任意一个悬挂点,则对于边 $(u, t) \in E$,有:

$$\delta(s, t) = \delta(s, u) + 1$$

证明:如果从顶点 s 可达顶点 u,则从顶点 s 也可达顶点 t。在这种情况下,从顶点 s 到 t 的最短路径必须经过顶点 u,而从顶点 u 到顶点 t 的距离是 1,因此等式成立;如果从顶点 s 不可达顶点 u,那么从顶点 s 也不可达顶点 t,则 $\delta(s, t) = \infty$,因此等式也成立。

定理 2:设图 G 是无权图,$s \in V^*$ 为图 G 中的任意一个悬挂点,$t \in V - V^*$ 为图 G 中任意一个非悬挂点,则对于边 $(s,v) \in E$,有:

$$\delta(s, t) = \delta(v, t) + 1$$

定理 3:设图 G 是无权图,$s \in V^*$ 为图 G 中的任意一个悬挂点,$t \in V - V^*$ 为图 G 中任意一个悬挂点,则对于边 $(s, v) \in E$ 和 $(u, t) \in E$ 有:

$$\delta(s, t) = \delta(v, u) + 2$$

定理 2、3 证明同定理 1。

算法基本思想是当求顶点 s 到 t 的最短路径时,根据顶点 s 和 t 是否属于悬挂点,把最短路径查询分为 4 种情况。

(1) 顶点 s 和 t 属于非悬挂点,在 2-hop 标签索

引中找到顶点 s 所对应的 $H(s)$ 标签和顶点 t 所对应的 $H(t)$ 标签,即可求出两点之间的最短路径值。

(2) 顶点 s 属于非悬挂点而顶点 t 属于悬挂点,由定理 1 可知,顶点 s 到 t 的最短路径值等于顶点 s 到 u 的最短路径值加上 1,其中 u 是顶点 t 的关联顶点,由情况 1 可知顶点 s 到 u 的最短路径值。

(3) 顶点 s 属于悬挂点而顶点 t 属于非悬挂点,由定理 2 可知,顶点 s 到 t 的最短路径值等于顶点 v 到 t 的最短路径值加上 1,其中 v 是顶点 s 的关联顶点,由情况 1 可知顶点 v 到 t 的最短路径值。

(4) 顶点 s 和顶点 t 属于悬挂点,由定理 3 可知,顶点 s 到 t 的最短路径值等于顶点 v 到 u 的最短路径值加上 2,其中 v 是顶点 s 的关联顶点,u 是顶点 t 的关联顶点,由情况 1 可知顶点 v 到 u 的最短路径值。算法正确性证明根据定理 1 ~ 3 可证。

在执行最短路径查询之前,需将数据图 G 和使用 One _ Degree 算法对数据图 G 构建的索引加载到内存中,输入是顶点 s 和顶点 t,输出是顶点 s 到 t 的最短路径值。如算法 3 所示,首先判断顶点 s 和 t 是否属于悬挂点,若顶点 s 和顶点 t 属于非悬挂点,则直接在 2-hop 标签索引 H 中找到顶点 s 所对应的 $H(s)$ 标签和顶点 t 所对应的 $H(t)$ 标签,便可求出两点之间的最短路径值。若顶点 s 是非悬挂点而顶点 t 是悬挂点,则在顶点关联索引 R 中找到顶点 t 的关联顶点 u,返回顶点 s 到 t 的最短路径值,最短路径值等于顶点 s 到 u 的最短路径值加上 1。若顶点 s 是悬挂点而顶点 t 是非悬挂点,则在顶点关联索引 R 中找到顶点 s 的关联顶点 v,返回顶点 s 到 t 的最短路径值,最短路径值等于顶点 v 到 t 的最短路径值加上 1。若顶点 s 和顶点 t 都是悬挂点,则在顶点关联索引 R 中找到顶点 s 的关联顶点 v 和顶点 t 的关联顶点 u,返回顶点 s 到 t 的最短路径值,最短路径值等于顶点 v 到 u 的最短路径值加 2。

例如,在数据图 G_1 中,若计算顶点 v_6 到 v_7 的最短路径值,首先判断顶点 v_6 和顶点 v_7 是否属于悬挂点,通过判断可知顶点 v_6 和顶点 v_7 是悬挂点,然后在顶点关联索引 R 中找到顶点 v_6 的关联顶点 v_5 和顶点 v_7 的关联顶点 v_0,在 2-hop 标签索引 H 中找到顶点 v_5 所对应的 $H(v_5)$ 标签和顶点 v_0 所对应的 $H(v_0)$ 标签,即可求出顶点 v_5 到 v_0 的最短路径值,最后,由算法第 11 行可知顶点 v_6 到 v_7 的最短路径值是 4。若计算顶点 v_9 到 v_2 的最短路径值,首先判断顶点 v_9 和顶点 v_2 是否属于悬挂点,通过判断可知顶点 v_9 是悬挂点而顶点 v_2 是非悬挂点,然后在顶点关联索引 R 中找到顶点 v_9 的关联顶点 v_1,在 2-hop 标签索引 H 中找到顶点 v_1 所对应的 $H(v_1)$ 标签和顶点 v_2 所对应的 $H(v_2)$ 标签,即可求出顶点 v_1 到 v_2 的最短路径值,最后,由算法第 8 行可知顶点 v_9 到 v_2 的最短路径值是 3。若计算顶点 v_3 到 v_8 的最短路径值,首先判断顶点 v_3 和顶点 v_8 是否属于悬挂点,通过判断可知顶点 v_3 是非悬挂点而顶点 v_8 是悬挂点,然后在顶点关联索引 R 中找到顶点 v_8 的关联顶点 v_4,在 2-hop 标签索引 H 中找到顶点 v_3 所对应的 $H(v_3)$ 标签和顶点 v_4 所对应的 $H(v_4)$ 标签,即可求出顶点 v_3 到 v_4 的最短路径值,最后,由算法第 5 行可知顶点 v_3 到 v_8 的最短路径值是 3。若计算顶点 v_4 到 v_5 的最短路径值,首先判断顶点 v_4 和顶点 v_5 是否属于悬挂点,通过判断可知顶点 v_4 和顶点 v_5 是非悬挂点,然后在 2-hop 标签索引 H 中找到顶点 v_4 所对应的 $H(v_4)$ 标签和顶点 v_5 所对应的 $H(v_5)$ 标签,最后,由算法第 2 行可知顶点 v_4 到 v_5 的最短路径值是 3。

算法 3: One _ Degree _ Query (G, H, R, s, t)

```
1.  if s ∈ V − V* and t ∈ V − V* do
2.      return QUERY (s, t, H)
3.  else if s ∈ V − V* and t ∈ V* do
4.      u = R[t]
5.      return QUERY (s, u, H) +1
6.  else if s ∈ V* and t ∈ V − V* do
7.      v = R[s]
8.      return QUERY (v, t, H) +1
9.  else if s ∈ V* and t ∈ V* do
10.     v = R[s] and u = R[t]
11.     return QUERY(v, u, H) +2
```

3　实验分析

本文实验所使用的硬件环境为 Intel Pentium

(R) Dual-Core E5300 主频为2.60GHz的 CPU,4GB 的内存和 500GB 的硬盘,操作系统为 Windows7,实验运行环境为 Microsoft Visual Studio 2013,编程语言为C ++。实验是将本文提出的索引构建算法和查询策略与目前最高效的 PLL 算法在索引构建时间和索引规模方面以及查询时间上进行比较。

3.1 数据集

实验使用 15 个真实数据集:包括来自 EcoCyc (ecocyc. org)、代谢网络、基因本体信息, Uniprot 数据库、XML 文档以及社交网络的数据集,这些数据集都可以转换为无向图,数据集中顶点和边的信息如表 1 所示,其中 $|V|$ 表示无向无权图中的顶点数目,$|E|$ 为无向无权图中边的数目,从表 4 中可以看出,前 11 个数据集的规模较小,顶点数目在几千到几万之间,后 4 个数据集规模较大,顶点数目在百万到千万之间。为了比较在不同索引上最短路径查询策略的性能,实验中对每个数据集随机选取 100 万个顶点对进行测试。

表 4 数据集统计信息

数据集	$\|V\|$	$\|E\|$
Agrocyc	12684	13657
Amaze	3710	3947
Anthra	12499	13327
Ecoo	12620	13575
Go	6793	13361
Human	38811	39816
Kegg	3617	4395
Mtbrv	9602	10438
Nasa	5605	6538
Vchocyc	9491	10345
Xmark	6080	7051
Uniprot22m	1595444	1595442
Uniprot100m	16087295	16087293
Last	1117749	1135147
Twitter	18121168	18359487

3.2 索引构建时间和索引规模

表 5 中展示了 PLL 算法和本文提出的 One _ Degree 算法在索引构建时间方面的对比。在数据集 Agrocyc 中,One _ Degree 算法在索引构建时间上比 PLL 算法快 3 倍多;在大数据图上,本文提出的算法在索引构建时间上比 PLL 算法快 1 ~2 个数量级。

表 5 索引构建时间(ms)

数据集	PLL	One _ Degree
Agrocyc	186	50
Amaze	38	20
Anthra	181	55
Ecoo	184	53
Go	406	250
Human	1430	226
Kegg	51	18
Mtbrv	137	51
Nasa	306	143
Vchocyc	138	52
Xmark	91	59
Uniprot22m	58860	1620
Uniprot100m	---------	689040
Last	18000	7200
Twitter	--------	---------

表 6 中展示了 PLL 算法和本文提出的 One _ Degree 算法在索引规模方面的比较,One _ Degree 算法中的索引大小表示非悬挂点的 2-hop 标签索引大小与悬挂点的顶点关联索引大小之和。从表中可

表 6 索引大小

数据集	PLL	One _ Degree
Agrocyc	51067	10827
Amaze	8031	3463
Anthra	53033	10269
Ecoo	51007	10891
Go	417099	211688
Human	104894	24225
Kegg	8700	4293
Mtbrv	42519	9103
Nasa	432691	138128
Vchocyc	42942	9164
Xmark	56897	23315
Uniprot22m	3246441	856663
Uniprot100m	---------	10495800
Last	2296141	617697
Twitter	--------	-------

以看出 One _ Degree 算法在索引规模上比 PLL 算法构建的索引要小,如在数据图 Anthra 中,One _ Degree 算法在索引大小方面上比 PLL 算法小 5 倍多。

3.3　最短路径查询时间

表 7 中展示了两种算法的最短路径查询时间比较,在查询时间上,从表中可以看出本文提出的 One _ Degree 算法比 PLL 算法要快,例如在数据图 Agrocyc、Amaze、Anthra、Ecoo、Human、Kegg、Mtbrv 和 Uniprot22m 中,One _ Degree 算法比 PLL 算法的查询时间要快,其主要原因是本文提出的算法扫描 2-hop 标签的个数比 PLL 算法扫描 2-hop 标签的个数要少。

表 7　查询时间(ms)

数据集	PLL	One _ Degree
Agrocyc	99	94
Amaze	73	68
Anthra	108	90
Ecoo	103	88
Go	438	538
Human	84	65
Kegg	74	66
Mtbrv	99	98
Nasa	390	576
Vchocyc	101	85
Xmark	147	173
Uniprot22m	92	43
Uniprot100m	---------	83
Last	125	60
Twitter	-------	---------

4　结　论

本文针对现有方法在处理最短路径查询时索引构建时间长、索引规模庞大的问题展开研究。在索引构建方面,提出了索引构建策略以及相应的算法,解决了构建索引时间长和索引规模庞大所导致的低效性和扩展性问题。实验结果表明,One _ Degree 算法通过减少冗余数据存储和图的遍历次数,降低了索引规模,减少了构建索引时间。

参考文献

[1] Abraham I, Delling D, Goldberg A V, et al. A hub-based labeling algorithm for shortest paths in road networks[C]. In: Proceedings of the 10th International Conference on Experimental Algorithms, Crete, Greece, 2011. 230-241

[2] Bader J S, Chaudhuri A, Rothberg J M, et al. Gaining confidence in high-throughput protein interaction networks[J]. *Nature Biotechnology*, 2004, 22(1):78-85

[3] Swamynathan G, Wilson C, Boe B, et al. Do social networks improve e-commerce?: a study on social marketplaces[C]. In: Proceedings of the Workshop on Online Social Networks, Seattle, USA, 2008. 1-6

[4] Xu Y, Papakonstantinou Y. Efficient keyword search for smallest LCAs in XML databases[C]. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, USA, 2005. 527-538

[5] Jensen C S, Pedersen T B, Timko I. Nearest neighbor queries in road networks[C]. In: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems, New Orleans, USA, 2003. 1-8

[6] Yu X, Pu K Q, Koudas N. Monitoring k-nearest neighbor queries over moving objects[C]. In: Proceedings of the IEEE International Conference on Data Engineering, Tokyo, Japan, 2005. 631-642

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

[8] Cheng J, Shang Z, Cheng H, et al. Efficient processing of k-hop reachability queries[J]. *VLDB Journal*, 2014, 23(2):227-252

[9] Guo L, Shao J, Aung H H, et al. Efficient continuous top- k, spatial keyword queries on road networks. *Geoinformatica*, 2015, 19(1):29-60

[10] Qiao M, Qin L, Cheng H, et al. Top-k nearest keyword search on large graphs[J]. *Proceedings of the VLDB Endowment*, 2013, 6(10):901-912

[11] Dijkstra E W. A note on two problems in connexion with graphs[J]. *Numerische Mathematik*, 1959, 1(1):269-271

[12] Hart P E, Nilsson N J, Raphael B. A formal basis for the heuristic determination of minimum cost paths in graphs [J]. *IEEE Trans actions on Systems Sciiences and Cyber-*

netics,1968, 4(2): 100-107

[13] Fu W C, Wu H, Cheng J, et al. IS-label: an independent-set based labeling scheme for point-to-point distance querying[J]. *Proceedings of the VLDB Endowment*, 2012, 6(6):457-468

[14] Akiba T, Iwata Y, Yoshida Y, et al. Fast exact shortest-path distance queries on large networks by pruned landmark labeling[C]. In: Proceedings of 2013 ACM SIGMOD International Conference on Management of Data, New York, USA, 2013. 349-360

Research on a shortest path query algorithm based on vertex-related index

Yu Jing[*,**,***], Yang Qingzhang[*,**,***]

([*]School of Information Science and Engineering, Yanshan University, Qinhuangdao 066004)

([**]Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province, Qinhuangdao 066004)

([***]Key Laboratory for Software Engineering of Hebei Province, Qinhuangdao 066004)

Abstract

The shortest path query problem in graph query is studied. Aiming at existing algorithms' problem of low efficiency and expansibility caused by the long time of the construction of index and the large scale of indexes, a vertex-related index strategy is proposed in index construction: constructing a vertex-related index on the vertex of 1, constructing a 2-hop tag index for other vertices, reducing the index size by reducing the number of traversal times of redundant data storage and graphs to reduce the build index time. Based on the proposed query strategy, the shortest path query algorithm based on vertex- related index and 2-hop tag is given.

Key words: graphical model, shortest path query, vertex-related index, 2-hop label index