软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(11):2964−2980 [doi: 10.13328/j.cnki.jos.004891]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向模式图变化的增量图模式匹配*

张丽霞[1,2], 王伟平[1], 高建良[1], 王建新[1]

[1](中南大学 信息科学与工程学院,湖南 长沙 410083)

[2](湖南师范大学 数学与计算机科学学院,湖南 长沙 410081)

通讯作者: 王伟平, E-mail: wpwang@mail.csu.edu.cn

摘 要: 在大数据时代,数据图的规模急剧增长,增量图模式匹配算法能够在数据图或模式图发生变化时避免重新在整个数据图上进行匹配、减少响应时间,因此成为了研究的热点.针对实际应用中数据图不变而模式图发生变化的情况,提出了一种面向模式图变化的增量图模式匹配算法 PGC_IncGPM,在模式图匹配的过程中记录适当的中间结果作为索引,用于后续的模式匹配.提出了增强的图模式匹配算法 GPMS,用于首次整个数据图上的模式匹配.该算法一方面能够建立后续增量匹配所需的索引,另一方面减少了整个数据图匹配的执行时间.设计实现了面向模式图增边和减边的两个核心子算法,通过子算法的组合,能够支持在模式图发生各种变化时进行增量图模式匹配.在真实数据集和合成数据集上进行实验,结果表明:与重新在整个数据图上进行匹配的 ReComputing 算法相比,当模式图中变化的边的数目不超过不变的边的数目时,PGC_IncGPM 算法能够有效减少图模式匹配的执行时间;随着数据图规模的增大,PGC_IncGPM 算法相对于 ReComputing 算法的执行时间的减少程度更加明显,对于大规模数据图具有更好的适用性.

关键词: 图模式匹配;增量算法;动态图;大数据

中图法分类号: TP311

中文引用格式: 张丽霞,王伟平,高建良,王建新.面向模式图变化的增量图模式匹配.软件学报,2015,26(11):2964−2980. http://www.jos.org.cn/1000-9825/4891.htm

英文引用格式: Zhang LX, Wang WP, Gao JL, Wang JX. Pattern graph change oriented incremental graph pattern matching. Ruan Jian Xue Bao/Journal of Software, 2015,26(11):2964−2980 (in Chinese). http://www.jos.org.cn/1000-9825/4891.htm

Pattern Graph Change Oriented Incremental Graph Pattern Matching

ZHANG Li-Xia[1,2], WANG Wei-Ping[1], GAO Jian-Liang[1], WANG Jian-Xin[1]

[1](School of Information Science and Engineering, Central South University, Changsha 410083, China)

[2](School of Mathematics and Computer Science, Hu'nan Normal University, Changsha 410081, China)

Abstract: In the age of big data, with the scales of data graphs growing rapidly, incremental graph pattern matching algorithm has become the research focus since it can avoid re-computing matches in the whole data graph and reduce the response time when the data graph or the pattern graph update. Considering the scenario where the data graph is constant while the pattern graph is changing in practical application, a pattern graph change oriented incremental graph pattern matching algorithm named PGC_IncGPM is proposed. The appropriate intermediate results generated in the process of graph pattern matching are recorded as indexes for subsequent pattern matching. An enhanced graph pattern matching algorithm named GPMS is presented for the first time for graph matching in the whole data graph. On one hand, the algorithm can build indexes for the subsequent incremental matching; on the other hand, it reduces the execution time of matching in the data graph. Two core subalgorithms designated to adding and reducing edges in the patter graph are designed and realized. No matter what mode changes in the pattern graph, incremental graph pattern matching can be carried out by combining these two subalgorithms. Experiments on real datasets and synthetic data show that PGC_IncGPM can effectively reduce the

* 基金项目: 国家自然科学基金(61232001, 61173169); 湖南省教育厅项目(15C0824)
收稿时间: 2015-02-28; 修改时间: 2015-05-11, 2015-07-14; 定稿时间: 2015-08-26

graph pattern matching execution time. Compared with the ReComputing algorithm which re-computes matches starting from scratch in the whole data graph, if the number of changed edges does not exceed the number of unchanged edges, the proposed algorithm can reduce execution time effectively. With the scale of the data graph increases, the reduction in execution time of PGC_IncGPM algorithm is more obvious, demonstrating the algorithm's applicability for large-scale data graph.

Key words: graph pattern matching; incremental algorithm; dynamic graph; big data

来自互联网及生活中的海量数据之间存在紧密的关联性,图作为一种被广泛应用的数据结构,非常适合刻画这种具有关联性的数据,图中的每个顶点代表现实世界中的实体对象,顶点之间的边表示实体之间的关系.图模式匹配是在一个数据图 G 中找到所有和给定模式图 P 相同或相似的子图.图模式匹配被广泛地应用于社交网络分析、Web 网络分析、知识发现和生物信息学等领域,如在万维网中用于进行 Web 文档分类[1],在软件工程领域用于软件代码剽窃检测[2]等.

在大数据时代,图数据规模急剧增长,例如,Facebook 中的好友网络包含 16 亿个节点和 1 500 亿条边,而万维网包含超过 500 亿个网页以及万亿量级的链接.对如此大规模的数据图进行一次完整的图模式匹配会耗费大量的时间和资源,因此,部分研究人员研究了动态图模式匹配[3−6],主要研究方向是模式图不变、数据图动态变化时的增量图模式匹配算法.通过引入增量处理技术,当数据图更新时,仅对图中更新的数据进行分析和重新匹配,避免对整体数据的重复计算.然而在实际生活中,模式图发生变化的情况也非常普遍,例如,在生物信息学领域,代表蛋白质和流感病毒的图通常被作为模式图,蛋白质变性、流感病毒变异时有发生;在网络安全领域,描述计算机病毒和网络攻击模式的图经常被作为模式图,病毒经常发生变种,网络攻击模式也不断变化.先前的增量图模式匹配算法不适用于模式图发生变化时的图模式匹配,随着图数据规模的增加,当代表蛋白质、病毒等的模式图 P 发生变化时,若重新在整个数据图 G 中查找 P,则将耗费大量的时间和资源,因此,研究支持模式图变化的增量图模式匹配算法具有非常重要的意义.与传统的重新进行匹配的重计算算法(ReComuputing)不同,面向模式图变化的增量图模式匹配算法将能够根据 P 的变化ΔP,利用原来的匹配结果 M,得到匹配结果的变化ΔM.

本文研究在固定不变的数据图上对发生变化的模式图进行匹配的增量图模式匹配问题.首先,提出了面向模式图变化的增量图模式匹配算法(pattern graph change oriented incremental graph pattern matching,简称 PGC_IncGPM)的基本框架;其次,提出了一种增强的图模式匹配算法(graph pattern matching strengthened,简称 GPMS),该算法除了能够在模式图匹配的过程中更高效地计算出当前的匹配结果以外,还能记录适当的中间结果作为索引,用于后续的模式匹配,减少后续匹配中不必要的重复计算;接着,分别针对模式图的两种基本变化情况(增加边和减少边),提出如何利用之前的匹配结果和索引进行增量图模式匹配的算法;最后,在真实数据和合成数据上对本文提出的算法进行实验,分析和验证了本文所提出算法的效率.

1 相关工作

1.1 图模式匹配

近年来,图模式匹配成为大数据研究的热点.典型的图模式匹配定义为子图同构,通常采用基于连接的方法或基于探测的方法实现.基于连接的方法的思路是:首先,根据顶点标签,为模式图中的每个顶点产生候选匹配点;然后,不断连接这些候选匹配点,得到一个更大的局部匹配结果;最后,找到准确的匹配结果.在该方法中,连接顺序影响算法效率[7,8].基于探测的方法的思路是:从数据图的一个顶点出发,根据模式图中的结构关系对数据图进行探测,检测被访问的点是否在一个匹配的子图中[9].

由于子图同构是 NPC 问题并且应用于社交网络等数据图时过于严格,因此现实生活中图模式匹配通常定义为近似匹配.匹配的方法通常是先根据节点的标签为模式图中的每一个节点产生一个候选匹配集,再根据对模式图中点的前驱和后继的不同近似程度要求过滤掉不匹配的节点.按照实际应用对匹配近似程度的不同要求,图模式匹配可以定义为图模拟(graph simulation)[10]、受限模拟(bounded simulation)[11]、强模拟(strong simulation)[12]、双向模拟(dual simulation)[13]和严格模拟(strict simulation)[13]等.与子图同构相比,图模拟只要求

匹配点保持模式图中对应点的后继关系;受限模拟把模式图中的边对应到数据图中的一条路径,数据图中的一个点只要存在一个子孙节点能与模式图中对应点的后继节点匹配,就认为该点与对应点匹配;双向模拟在图模拟的基础上,还要求匹配点保持模式图中对应点的前驱关系;强模拟在双向模拟的基础上,进一步要求匹配点所在子图(子图中包含非匹配点)的半径不大于模式图的直径;严格模拟比强模拟要求更严格,要求完全由匹配点构成的子图的半径不大于模式图的直径.按约束关系由弱到强的顺序对各种图模式匹配进行排序,其结果是:受限模拟、图模拟、双向模拟、强模拟、严格模拟.由于受限模拟和图模拟要求太松,在实际应用中会产生大量没有实际利用价值的匹配点,淹没有用信息;而强模拟和严格模拟的要求仍然较严格导致匹配算法时间复杂度较高,在实际应用中响应时间长.因此,双向模拟在模拟结果的有效性和响应及时性方面具有更好的平衡性,具有更高的实用价值.综上,本文中图模式匹配定义为双向模拟.

1.2 数据图变化的增量图算法

对于数据图发生变化的情况,Ramalingam 等人[14]研究了针对最短路径问题的增量算法,对由于插入和减少边而导致最短路径有可能变化的点的最短路径进行重新计算.Fan 等人[3]通过借鉴 Ramalingam 的解决最短路径问题的算法提出了针对图模拟的增量算法:先根据数据图的变化确定受影响区域(即匹配状态会改变的节点集);然后,按照图模式匹配的定义对受影响节点重新进行匹配.文献[4]提出了子图同构的增量匹配算法.文献[5]提出了快速不精确图模式匹配算法及增量算法:利用一个指数级的索引,近似确定在一组图流中是否包含一组给定的模式图.该算法把近似子图搜索问题转化为向量空间的关系检测,当图中增加边或减少边时,会修改相关节点的向量,重新检查新的向量是否依然包含模式图的向量即重新确定匹配状态.文献[15]提出了增量 simrank 算法:首先,根据矩阵操作的特点采用一些技巧,使得链路更新时,相似矩阵的更新操作转变为部分行和列的更新操作;在此基础上又提出有效的剪枝技术,以获得受影响区域的相似性变化,在不影响准确性的情况下减少不必要的计算.文献[16,17]提出了一种在数据图上的基于等价关系的增量互模拟算法,当数据图发生变化时,先标识已有的划分中需要改变的部分,然后对其进行重新计算.上述增量算法的思路基本相同:在模式图不变的前提下,根据数据图的变化分析并确定可能受影响的区域,对动态变化部分、受影响的子图进行重新计算,得到变化的匹配结果.

先前的增量算法面向的是数据图动态变化的情况,不适用于模式图动态变化的情形.在实际生活中,模式图动态变化的情况也非常普遍.针对这种应用场景,本文提出一种面向模式图变化的增量图模式匹配算法.

2 模式图变化的增量图模式匹配的形式化描述

首先,我们将给出图模式匹配的基本定义;然后,在此基础上提出面向模式图变化的增量图模式匹配的形式化描述.对于图模式匹配,模式图和数据图都是带标签的有向图,图中的每个节点有且仅有一个标签,该标签定义了节点的属性(如关键词、技能、等级、姓名、公司等).

定义 1(图). $G=(V,E,L)$是一个图,V 是节点集,$E\subseteq V\times V$ 是边集,L 是一个标签函数,V 中的每个节点都有一个标签 att,即 $L(v)=att$,att 是 v 的属性.

定义 2(图模式匹配). 假设有一个模式图 $P=(V_p,E_p,L_p)$和一个数据图 $G=(V,E,L)$,u 和 v 分别是 P 和 G 中的节点.如果存在一个二元关系 $R\subseteq V_p\times V$,满足以下条件,则说 G 匹配 P,R 是一个匹配:

(1) 如果$(u,v)\in R$,那么 u 和 v 有相同的标签,即 $L_p(u)=L(v)$.

(2) 对 V_p 中的任意一个节点 u,V 中都存在一个节点 v,使得$(u,v)\in R$,且

(a) 对 E_p 中的任意一条边(u,u'),在 E 中都存在一条边(v,v'),使得$(u',v')\in R$;

(b) 对 E_p 中的任意一条边(u'',u),在 E 中都存在一条边(v'',v),使得$(u'',v'')\in R$;

条件(a)保证了匹配点 v 保持了 u 的后继关系,条件(b)保证匹配点 v 保持了 u 的前驱关系.显然,对于任意 P 和 G,如果 G 匹配 P,则一定存在一个最大的匹配关系 R_M.图模式匹配问题就是找出 R_M,根据 R_M 建立匹配结果图 G_r.

定义 3(模式图变化的增量图模式匹配). 给定一个数据图 G,一个模式图 P,模式图变化前 G 匹配 P 的结果

为 $M(P,G)$.当模式图发生变化ΔP 以后,增量图模式匹配能够找到匹配结果的变化ΔM,使得增量图模式匹配结果 $M(P\oplus\Delta P,G)$满足如下关系:$M(P\oplus\Delta P,G)=M(P,G)\oplus\Delta M$,即,当 P 变化时,增量图模式匹配算法能够利用以前的计算结果 $M(P,G)$,通过获得ΔM 且与ΔM 进行运算,获得新的匹配结果,达到减少不必要的重复计算的目的.

3 面向模式图变化的增量图模式匹配算法 PGC_IncGPM

根据上一节中的定义3,为了在模式图 P 发生变化后通过增量式的算法获得新的匹配结果,除了需要使用原来的匹配结果 $M(P,G)$以外,还需要获得匹配结果的变化ΔM,其中,如何获得ΔM 以及如何根据 $M(P,G)$和ΔM 得出新的匹配结果 $M(P\oplus\Delta P,G)$,成为整个算法的关键.为了快速获得ΔM,可以预先选择数据图中的特征建立索引,使得增量模式匹配时可以根据索引快速缩小搜索空间.建立的索引越多,增量匹配的搜索空间越少,获得ΔM 所需的时间越少,但是保存索引所占存储空间也越大.对于大规模图来说,一方面需要减少匹配时间,另一方面也需要尽量减少存储开销.我们在平衡考虑索引对存储开销和时间开销的影响之后,提出在原来匹配过程中记录以下 3 类集合作为索引:

(1) $cand(\cdot)$集合:对 P 中的任一节点 u,$cand(u)$包含 G 中所有仅仅标签和 u 相同的节点;

(2) $sim(\cdot)$集合:对 P 中的任一节点 u,$sim(u)$包含 G 中所有保持 u 的后继关系的节点;

(3) $mat(\cdot)$集合:对 P 中的任一节点 u,$mat(u)$包含 G 中所有与 u 匹配的节点,$mat(\cdot)$集合中的所有点都在匹配结果图中.

3.1 PGC_IncGPM算法基本框架

面向图模式变化的增量图模式匹配算法 PGC_IncGPM 的基本框架如图 1 所示.

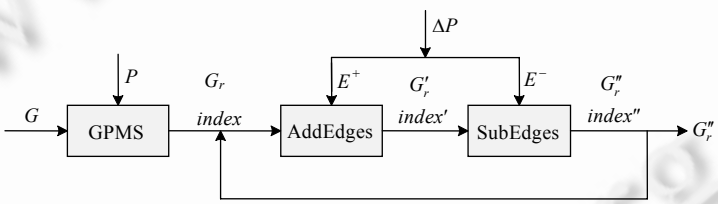


Fig.1 Basic framework of PGC_IncGPM algorithm

图 1 PGC_IncGPM 算法的基本框架

首先使用增强的图模式匹配算法 GPMS,用于第一次在整个数据图 G 上对模式图 P 进行图模式匹配,一方面获得匹配结果图 G_r,另一方面建立后续增量匹配所需的索引 $index$.由于模式图的变化可以分解为由“增加边”(+e)操作或“减少边”(–e)操作或两种操作的联合作用导致的变化,因此,模式图的变化ΔP 中可能有增加的边,也可能有减少的边,我们分别用 E^+和 E^-表示所有增加的边的集合和所有减少的边的集合.增量图模式匹配算法将分解为依次根据 E^+和 E^-进行增量式匹配的过程:先对 E^+,调用子算法 AddEdges,获得匹配结果 G'_r 和索引 $index'$;再对 E^-,调用子算法 SubEdges,获得匹配结果图 G''_r 和索引 $index''$. G''_r 就是模式图变化ΔP 后的新匹配结果图;$index''$是目前状态下的索引,可以继续用于模式图再次发生变化后进行后续的增量匹配.如果模式图的变化仅包含一种操作,即,仅“增加边”或仅“减少边”,则以上两个步骤可以省略一个.若模式图继续发生变化,则继续调用 AddEdges 和 SubEdges 子算法,产生下一轮匹配结果.下面依次详细阐述增强的图模式匹配算法 GPMS、面向模式图增边的子算法 AddEdges 和面向模式图减边的子算法 SubEdges.

3.2 增强的图模式匹配算法GPMS

由于 PGC_IncGPM 算法中 AddEdges 和 SubEdges 子算法的执行依赖于在整个数据图 G 上对 P 进行图模式匹配产生的匹配结果图 G_r 和建立的索引 $index$,因此,整个数据图上的图模式匹配算法是增量图匹配算法的基础.为了高效实现 PGC_IncGPM 算法,我们对传统的图模式匹配算法 GPM 进行了改进:

- 一方面,GPMS 算法在执行图模式匹配过程中增加了建立用于后续匹配的索引的过程.

使用 GPM 算法时,在整个数据图 G 上进行图模式匹配仅输出匹配结果图 G_r,而使用 GPMS 算法除了能够得到 G_r 以外,还能对 P 中的每个节点 u,产生 $cand(u)$,$sim(u)$和 $mat(u)$这 3 个互斥的集合作为索引,用于支持后续的增量图模式匹配.3 个集合中,$cand(u)$保存了 G 中所有仅仅标签和 u 相同的节点,$sim(u)$保存了 G 中所有只保持 u 的后继关系的节点,$mat(u)$保存了 G 中所有与 u 匹配的节点.

- 另一方面,GPMS 算法对 GPM 算法中的候选点筛选顺序进行了优化,减少了执行时间.

GPM 算法在模式图匹配时,先根据标签匹配为 P 中每个节点产生候选匹配集,然后对候选匹配集中的各点根据其前驱和后继关系进行筛选过滤,直到候选匹配集不再变化为止.这种方法会造成筛选过程的反复迭代,导致算法执行时间较长.由于图一般采用邻接表的方式存储,访问节点后继比较容易,而访问节点前驱的时间复杂度很高,所以我们按如下方法对 GPM 算法中的候选点筛选顺序进行优化:首先,按照 P 的逆拓扑序列的次序,根据后继关系依次筛选 P 中各节点的候选匹配集;其次,按照 P 的拓扑序列的次序,根据节点前驱进一步筛选各节点的候选匹配集;最后,在规模已经很小的候选匹配集上对各节点根据后继、前驱分别筛选,直到候选匹配集不再变化为止,得到最终的匹配集.模式图 P 中有可能存在强连通分量(strong connected component,简称 SCC),对于这种情况,先把 P 转化为一个有向无环图 P',方法为:a) 找出 P 中所有强连通分量;b) 把每个强连通分量汇聚成一个节点,得到一个图 P';c) 对 P'进行拓扑排序,然后再把每个强连通分量汇聚点用原来的节点集替换,得到 P 的类拓扑序列.

综上,GPMS 算法的描述如算法 1 所示,其步骤如下:

- 第 1 步,根据标签匹配为 P 中每个节点 u 产生候选匹配集 $sim(u)$(第 1 行、第 2 行).
- 第 2 步,按 P 的逆拓扑序列,依次根据节点的后继筛选 P 中各节点的候选集 $sim(\cdot)$,得到满足后继匹配条件的结果 $sim(\cdot)$(SCC 内的节点的候选集要筛选多遍,直至 SCC 内各节点的 $sim(\cdot)$无变化为止),从 $sim(\cdot)$中删除的节点加入 $cand(\cdot)$(第 3 行~第 6 行).
- 第 3 步,按 P 的拓扑序列,依次根据节点的前驱筛选 P 中各节点的 $sim(\cdot)$,满足前驱匹配条件的候选点加入 $mat(\cdot)$(第 7 行~第 10 行).
- 第 4 步,反复对各节点的 $mat(\cdot)$集根据后继和前驱进行筛选,直至无变化,得出最终的匹配集 $mat(\cdot)$(第 11 行、第 12 行).
- 第 5 步,根据 $mat(\cdot)$组合得出最大匹配集 R_M,由 $mat(\cdot)$中节点及它们之间的边构建出匹配结果图 $G_r=(E_r, V_r)$(第 13 行、第 14 行).

算法 1. 增强的图模式匹配算法 GPMS.

输入:模式图 P,数据图 G.

输出:结果图 G_r,$cand(\cdot)$,$sim(\cdot)$和 $mat(\cdot)$.

1) **for each** $u\in V_P$ **do**
2) $sim(u)=\{v|v\in V\wedge L(v)=L_p(v)\}$; $cand(u)=\varnothing$; $mat(u)=\varnothing$;
3) **for each** ordered $u\in V_P$ **do** //according to the inverse topological sort order of P
4) **for each** $w\in sim(u)$ **do**
5) **if** there exist $u'\in succ(u)$ such that $succ(w)\cap sim(u')=\varnothing$ **then** //$succ(u)$ is the set of succsor nodes of u
6) $sim(u)=sim(u)\backslash\{w\}$; $cand(u)=cand(u)\cup\{w\}$;
7) **for each** ordered $u\in V_P$ **do** //according to the topological sort order of P
8) **for each** $w\in sim(u)$ **do**
9) **if for** any $u''\in pre(u)$ such that $pre(w)\cap mat(u'')\neq\varnothing$ **then** //$pre(u)$ is the set of precursor nodes of u
10) $mat(u)=mat(u)\cup\{w\}$; $sim(u)=sim(u)\backslash\{w\}$;
11) **while** there exist $u\in V_P$, $w\in mat(u)$ and $u''\in pre(u)$ such that $pre(w)\cap mat(u'')=\varnothing$ or $u'\in succ(u)$ such that $succ(w)\cap mt(u')=\varnothing$ **do**
12) $mat(u)=mat(u)\backslash\{w\}$; $sim(u)=sim(u)\cup\{w\}$;

13) $R_M=\{(u,w)|u\in V_p,w\in mat(u)\}\}$;

14) Construct G_r according to the nodes in $mat(\cdot)$ and the relations between them;

下面以一个图模式匹配的实例解释 GPMS 算法的工作过程,如图 2 所示.图 2 左边是模式图 P,中间是数据图 G.P 中的 D 和 E 构成一个强连通分量,把 D 和 E 看成一个汇聚点,对 P 进行拓扑排序得到 P 的类拓扑序列为 $\{A,B,C,\{D,E\}\}$.按照 GPMS 算法:

- 第 1 步产生各节点的初始 $sim(\cdot)$集,见表 1(a).
- 第 2 步,按 P 的逆拓扑序列$\{\{D,E\},C,B,A\}$对各节点的 $sim(\cdot)$集根据后继进行筛选:先筛选 $sim(D)$,此时,$sim(D)$中各节点已满足后继匹配,对其筛选完成;再筛选 $sim(E)$,由于 E_4 没有后继匹配 D,所以从 $sim(E)$中删除 E_4,并将其加入 $cand(E)$,由于 D 和 E 构成一个强连通分量,因此,$sim(E)$发生变化后还要继续筛选 $sim(D)$,将 D_4 从 $sim(D)$中删除,并将其加入 $cand(E)$,然后再筛选 $sim(E)$,最终,E_3,D_3 也依次从 $sim(E)$和 $sim(D)$中删除,并加入到 $cand(E)$和 $cand(D)$中,至此,$sim(E)$和 $sim(D)$不再变化,对它们的筛选完成;接下来依次筛选 $sim(C)$,$sim(B)$,$sim(A)$,C_3,A_3 依次从对应的 $sim(\cdot)$中删除,并加入到 $cand(\cdot)$中.至此,索引状态见表 1(b).
- 第 3 步,按 P 的类拓扑序列$\{A,B,C,\{D,E\}\}$对各节点的 $sim(\cdot)$集根据前驱进行筛选:$sim(A)$中的所有节点都满足前驱匹配,因此全部从 $sim(A)$中移到 $mat(A)$中;由于 B_3 没有能够匹配 A 的前驱,所以只能留在 $sim(B)$中而不能加入 $mat(B)$.依此类推,对其余 $sim(\cdot)$集按照相同的方法进行筛选.至此,索引状态见表 1(c).
- 第 4 步,对 $mat(\cdot)$中的点根据前驱和后继分别筛选,将不满足前驱或后继匹配的节点从 $mat(\cdot)$移动到 $sim(\cdot)$中.在该实例中,本步骤未引起索引状态变化.
- 第 5 步,把 $mat(\cdot)$中的各点用它们之间的边连起来,得到结果图 G_r,如图 2 中最右边的 G_r 所示.

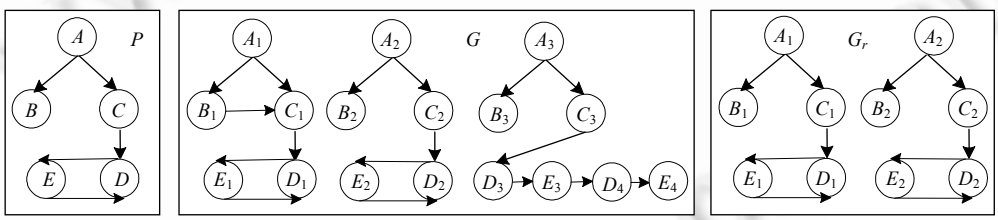


Fig.2 An example of graph pattern matching using GPMS algorithm

图 2 使用 GPMS 算法进行图模式匹配的一个实例

Table 1 Index built during the process of graph pattern matching for the example in Fig.2

表 1 对图 2 实例进行图模式匹配建立的索引

	(a)			(b)			(c)		
u	$cand(u)$	$sim(u)$	$mat(u)$	$cand(u)$	$sim(u)$	$mat(u)$	$cand(u)$	$sim(u)$	$mat(u)$
A		A_1, A_2, A_3		A_3	A_1, A_2		A_3		A_1, A_2
B		B_1, B_2, B_3			B_1, B_2, B_3			B_3	B_1, B_2
C		C_1, C_2, C_3		C_3	C_1, C_2		C_3		C_1, C_2
D		D_1, D_2, D_3, D_4		D_3, D_4	D_1, D_2		D_3, D_4		D_1, D_2
E		E_1, E_2, E_3, E_4		E_3, E_4	E_1, E_2		E_3, E_4		E_1, E_2

3.3 面向模式图增边的子算法AddEdges

如果模式图 P 增加一条或者多条边,那么匹配结果图 G_r 中有些点可能就不再匹配,ΔM 即为所有这些不再匹配部分的集合,此时,需要对原来的匹配结果进行筛选,增量模式匹配将从原来的匹配结果中去除ΔM,是一个对原匹配结果的精简过程.

对于增加边后的模式图 P^+,增量匹配算法的思路是对每一条增加的边(u,u')执行如下操作:

- 首先,对任一 $v\in mat(u)$,检查 v 是否有后继节点匹配 u',如果没有,则 v 不再匹配 u;对任一 $v'\in mat(u')$,检

查其是否有前驱节点匹配 u,如果没有,则 v 不再匹配 u;如果 v 原来匹配 u,但在新的模式图 P^+下 v 不再匹配 u,那么原来匹配结果中 v 的前驱和后继也可能不再匹配,因此还要继续检查 v 的前驱和后继是否匹配.以上操作会使 $mat(\cdot)$中的某些节点加入 $cand(\cdot)$或者 $sim(\cdot)$,在得到新的匹配结果图的同时,也会更新索引.

- 其次,由于增加边(u,u')后 $sim(u)$中的节点 w 也有可能不再满足 u 的后继关系,为了保证索引数据一致性,还需要把 w 从 $sim(u)$中移出加入 $cand(u)$;另外,w 状态的变化可能会影响到它的所有前驱节点,因此对其前驱也需进行同样的处理.以上操作会进一步更新索引.

综上,面向模式图增边的子算法 AddEdges 如算法 2 所示:

- 依次对插入的每一条边(u,u')进行后面的处理,队列 Q 中保存原来匹配结果中在新的模式图下不再匹配的节点,其初始化为空(第 2 行).
- $mat(u)$中没有后继匹配 u'的节点放入 Q 中(第 3 行~第 5 行),$mat(u')$中没有前驱匹配 u 的节点也放入 Q 中(第 6 行~第 8 行).
- 当 Q 不空时,我们从 Q 中删除队头元素(u,v),将 v 从 V_r 和 $mat(u)$中删除(第 11 行).
- 然后检查 v 是否保持 u 的后继关系:如果满足,则 v 加入 $sim(u)$;否则,v 加入 $cand(u)$(第 12~16 行).
- 接下来,对结果图 G_r 中 v 的每个前驱 v_1,首先从 G_r 中减少边(v_1,v),然后根据后继关系检查 v_1 的匹配状态是否改变,若改变,则把 v_1 加入 Q 中(第 17 行~第 21 行);对 G_r 中 v 的每个后继 v_2,根据前驱关系检查 v_2 的匹配状态是否改变,若改变,则把 v_2 加入 Q 中(第 22 行~第 26 行).
- 对 $sim(u)$中的任一节点 w,检查其是否保持新模式图下 u 的后继关系,若不满足,则将 w 从 $sim(u)$中移出加入 $cand(u)$;同时,对 w 的所有前驱继续进行同样的处理(第 27 行~第 30 行).
- 最后,返回新的匹配结果图(第 31 行).

算法 2. 面向模式图增边的子算法 AddEdges.

输入:模式图 P,结果图 $G_r=(V_r,E_r)$,$mat(\cdot)$,$sim(\cdot)$,$cand(\cdot)$,P 中插入的边的集合 E^+;

输出:新的匹配结果图 G_r,更新后的索引 $cand(\cdot)$,$sim(\cdot)$和 $mat(\cdot)$.

```
1)  for each (u,u') in E+ do
2)    Queue Q=∅
3)    for each v∈mat(u) do
4)      if succ(v)∩mat(u')=∅ then
5)        push(Q,(u,v));
6)    for each v'∈mat(u') do
7)      if (pre(v')∩mat(u)=∅) then
8)        push(Q,(u',v'));
9)    while (Q is not empty) do
10)     (u,v)=pop(Q);
11)     Vr=Vr\{v}; mat(u)=mat(u)\{v};
12)     check if v can match u according to the subsequence relationships of u;
13)     if v can match u according to subsequence relationships of u then
14)       sim(u)=sim(u)∪{v}
15)     else
16)       cand(u)=cand(u)∪{v}
17)     for each ep=(u1,u) that er=(v1,v) can match do
18)       Er=Er\(v1,v);
19)       check if v1 can match u1 according to subsequence relationships of u1;
```

20) **if** v_1 can't match u_1 **then**
21) $push(Q,(u_1,v_1))$;
22) **for each** $e_p=(u,u_2)$ that $e_r=(v,v_2)$ can match **do**
23) $E_r=E_r\backslash(v,v_2)$;
24) check if v_2 can match u_2 according to the precursor relationships of u_2;
25) **if** v_2 can't match u_2 **then**
26) $push(Q,(u_2,v_2))$;
27) **for each** $w\in sim(u)$ **do**
28) check if w can match u according to subsequence relationships of u;
29) **if** w can not match u **then**
30) remove w from $sim(u)$ to $cand(u)$ and continue to handle the precursor nodes of w;
31) **return** G_r;

仍然以图 2 中的图模式匹配为例,如果在图 2 中 P 上插入一条边(B,C),首先对原来的匹配结果 G_r 进行检查,B_2 和 C_2 不再匹配,B_2 加入 $cand(B)$,C_2 加入 $sim(C)$;对 B_2 和 C_2 的前驱和后继进行检查和处理,A_2~E_2 构成的子图从 G_r 中移出.A_1~E_1 构成的子图中加入一条边(B_2,C_2)即为新的 G_r,如图 3 所示.更新后的索引见表 2.

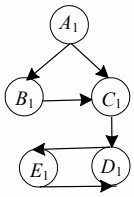


Fig.3 Match result of AddEdges when the edge (B,C) is added to P
图 3 P 增加边(B,C)后,执行 AddEdges 得到的匹配结果

Table 2 Index updated by AddEdges when the edge (B,C) is added to P
表 2 P 增加边(B,C)后,执行 AddEdges 得到的索引

u	$cand(u)$	$sim(u)$	$mat(u)$
A	A_3, A_2	–	A_1
B	B_2, B_3	–	B_1
C	C_3	C_2	C_1
D	D_3, D_4	D_2	D_1
E	E_3, E_4	E_2	E_1

3.4 面向模式图减边的子算法SubEdges

如果 P 中删除一条或多条边,原来的匹配结果依然是匹配的,原来不匹配的节点可能会匹配,ΔM 即为所有新增的匹配节点的集合,增量模式匹配将在原来的匹配结果中增加ΔM,是一个扩充匹配结果的过程.对减少边后的模式图 P^-,增量匹配算法更新匹配结果;同时,为了能够支持模式图的再次变化,继续执行后续的增量图模式匹配,还需要对索引进行同步更新.

对于 P^-,增量匹配算法的思路是对每一条删除的边(u,u')执行如下操作:由于边(u,u')被删除,$cand(u)$中有些节点 w 可能满足了 u 的后继关系,需要加入 $sim(u)$中;w 匹配状态的改变会影响到 w 的前驱节点,继而又影响到该前驱节点的前驱节点.依此类推,影响会逐级扩散下去.因此,对所有受影响的节点的匹配状态进行重新检测.

对所有删除的边进行上述操作后,再在更新的 $sim(\cdot)$上采用和 GPMS 第 4 步一样的方法,根据前驱和后继进行筛选,同时满足前驱和后继关系的点从 $sim(\cdot)$中移到 $mat(\cdot)$中.当 $sim(\cdot)$和 $mat(\cdot)$不再变化时,就获得了新的匹配结果图和更新后的索引.

面向模式图减边的子算法 SubEdges 如算法 3 所示:

- 依次对删除的每一条边(u,u')进行处理,以更新由于该边的删除而引起的 $cand(\cdot)$,$sim(\cdot)$和 $mat(\cdot)$的变化.队列 Q 中保存 $cand(\cdot)$中那些由于模式图中(u,u')的删除而有可能满足节点后继匹配条件的节点,其初始化为空(第 2 行).
- $cand(u)$中的任一节点匹配状态都有可能改变,因此把它们都放入 Q 中(第 3 行、第 4 行).
- 当 Q 不空时,从 Q 中删除队头元素(u,v),检查 v 是否满足 u 的后继关系匹配(第 6 行、第 7 行):如果满足,则 v 从 $cand(u)$中移出加入 $sim(u)$;同时,v 的前驱的匹配状态也有可能改变,因此,v 的前驱也都插入到 Q 中(第 8 行~第 12 行).
- 当所有边处理完毕后,对 P 中的每个节点 u 和 $sim(u)$中的每个节点 v,根据 u 的前驱和后继检查 v 是否匹配 u:若匹配,则将 v 从 $sim(u)$中移出加入 $mat(u)$;同时,将 v 和 v 相关联的边加入 G_r(第 13 行~第 17 行).
- 最后,返回新的匹配结果图(第 18 行).

算法 3. 面向模式图减边的子算法 SubEdges.

输入:模式图 P,结果图 $G_r=(V_r,E_r)$,$mat(\cdot)$,$sim(\cdot)$,$cand(\cdot)$,P 中删除的边的集合 E^+;

输出:新的结果图 G_r,更新后的索引 $cand(\cdot)$,$sim(\cdot)$和 $mat(\cdot)$.

```
1)  for each (u,u') in E⁻ do
2)     Queue Q=∅
3)     for each v∈cand(u) do
4)        push(Q,(u,v));
5)     while (Q is not empty) do
6)        (u,v)=pop(Q);
7)        check if v can match u according to according to subsequence relationships of u;
8)        if v can match u according to according to subsequence relationships of u then
9)           sim(u)=sim(u)∪{v}; cand(u)=cand(u)\{v};
10)          for each e_p=(u_1,u) do
11)             for each v_1∈cand(u_1)∩pre(v) do
12)                push(Q,(u_1,v_1));
13) for each u∈V_P and each w∈sim(u) do
14)    check if w can match u according to precursor and subsequence relationships of u;
15)    if w can match u then
16)       mat(u)=mat(u)∪{w}; sim(u)=sim(u)\{w};
17)       add w and the edges associated with it to G_r;
18) return G_r;
```

继续以图 2 中的图模式匹配为例,如果从 P 上删除边(E,D),则 E_4,D_4,E_3,D_3,C_3,A_3 依次从对应的 $cand(\cdot)$中移出并加入 $sim(\cdot)$.在更新的 $sim(\cdot)$上进行前驱后继检查的结果是 A_3~E_3 也匹配,需加入 $mat(\cdot)$,把这些点及相关的边加入原来的匹配结果图中,即得到新的匹配结果图,如图 4 所示.更新后的索引见表 3.

Table 3 Index updated by SubEdges when the edge (E,D) is deleted from P

表 3 P 减少边(E,D)后,执行 SubEdges 得到的索引

u	$cand(u)$	$sim(u)$	$mat(u)$
A	–	–	A_1, A_2, A_3
B	–	–	B_1, B_2, B_3
C	–	–	C_1, C_2, C_3
D	–	D_4	D_1, D_2, D_3
E	–	E_4	E_1, E_2, E_3

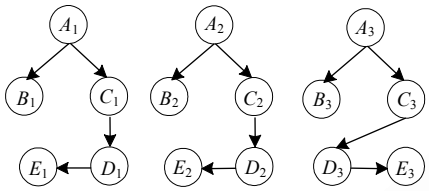


Fig.4 Match result of SubEdges when the edge (*E*,*D*) is deleted from *P*

图 4 *P* 减少边(*E*,*D*)后,执行 SubEdges 得到的匹配结果

4 实验与分析

本节将通过实验评估我们提出的 PGC_IncGPM 算法,使用执行时间作为考察图模式匹配算法的关键技术指标,同时定义改进比率(improved ratio,简称 IR)指标,即,提出的算法与传统算法相比执行时间缩短的比率,用于直观反映提出算法的有效性.首先,通过比较 GPMS 算法和传统的 GPM 算法,对 GPMS 算法进行对比评估;然后,对 PGC_IncGPM 算法和基于 GPMS 的重新计算(ReComputing)算法进行评估,比较和分析两种算法的执行时间,评估 PGC_IncGPM 算法的改进比率.由于算法执行时间与模式图和数据图的规模有关,因此我们还考察了模式图规模和数据图规模分别发生变化对 PGC_IncGPM 算法的执行时间的影响.实验使用两个真实的数据集:Epinions 和 Slashdot,它们均来自于斯坦福大学的共享数据集(http://snap.stanford.edu/data/),前者是一个信任网络,有 75 879 个节点和 508 837 条边;后者是一个社交网络,有 82 168 个节点和 948 464 条边.实验使用的合成数据集为 6 个不同规模的数据集.所有的数据图上的每个节点都被随机设置一个标签,标签共 10 种(A~J).在实际应用中,模式图通常不大,在本领域先前的相关研究中,当模式图不变而数据图发生变化时,增量图匹配算法中模式图的规模一般设置为 3~8 个节点和 3~8 条边[3].在模式图规模的选取上,我们首先参考之前的相关研究选取了通常规模的模式图;另外,为了验证本文所提出算法的有效性和适用范围,我们还构造了较大规模的模式图进行实验.每次实验中,使用 3 个不同的模式图,每个模式图边的变化选取 5 种不同的方式,通过多次图模式匹配实验,取平均结果作为最终的实验结果.

4.1 GPMS算法的执行时间和改进比率

为了评估 GPMS 算法对传统的 GPM 的改进,我们在 Epinions 上对不同规模的模式图(节点个数$|V_p|$分别为 5,6,7,8,初始边数$|E_p|$按照公式$|E_p|=|V_p|^{1.2}$选取)使用两种算法分别进行图模式匹配,结果如图 5 所示:随着模式图规模的增加,两种算法的执行时间都会增加;但是无论对于哪种情况,GPMS 算法的执行时间都比 GPM 算法的执行时间短;当模式图节点个数分别为 5,6,7 和 8 时,GPMS 算法的改进比率 IR 分别达到了 27.97%,29.04%,27.27%和 23.08%,平均为 26.84%.

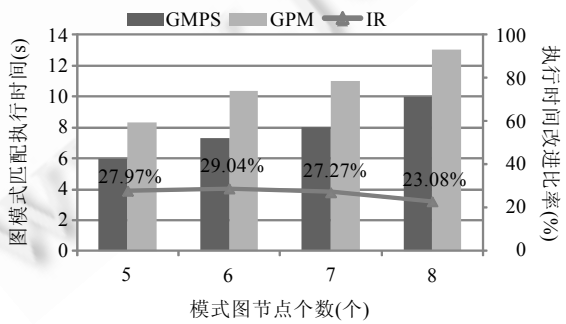
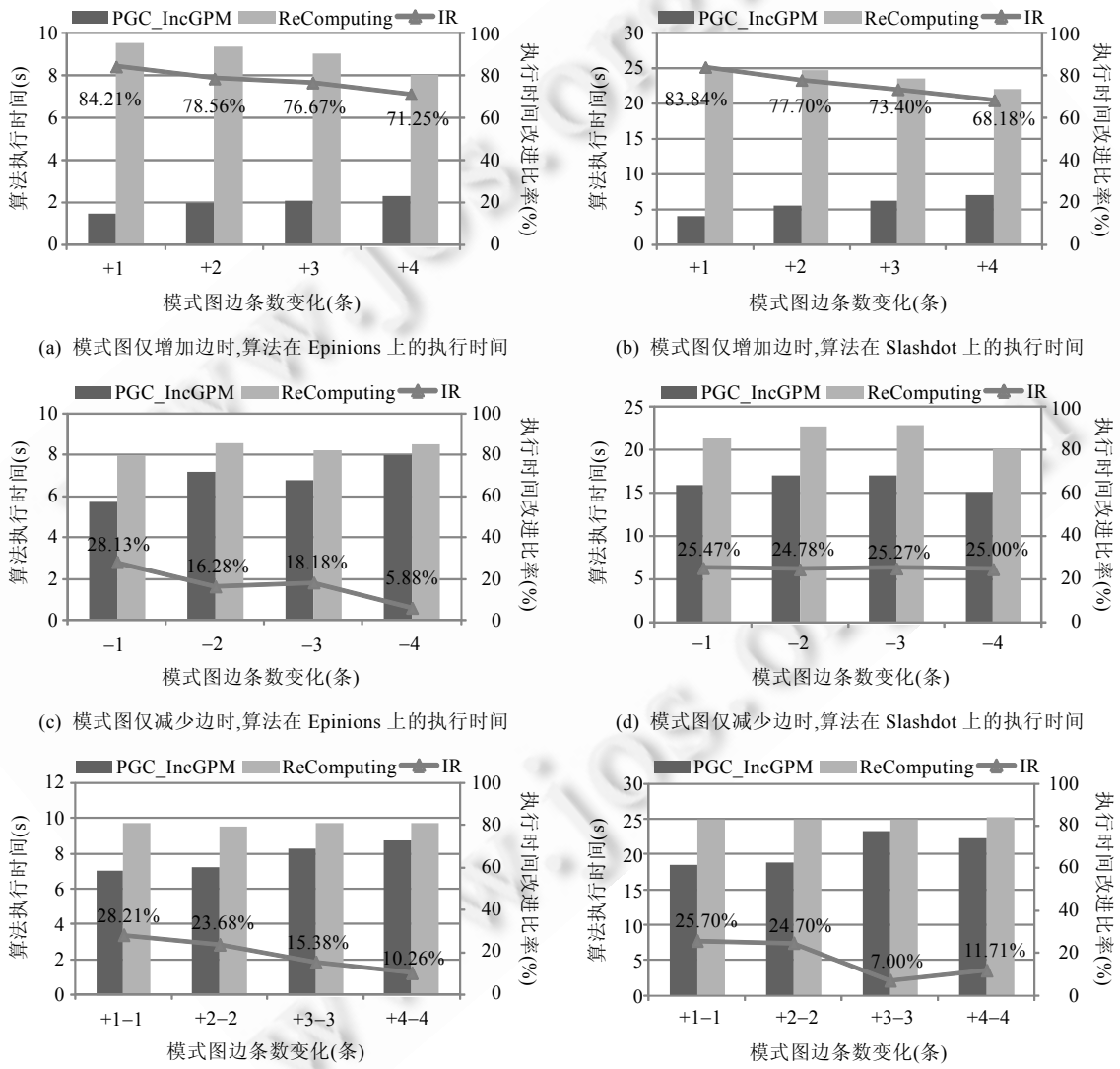


Fig.5 Execution time and improved ratio of GPMS algorithm

图 5 GPMS 算法的执行时间与改进比率

4.2 PGC_IncGPM算法的执行时间和改进比率

为了评估 PGC_IncGPM 算法的执行时间和改进比率,分别假设模式图仅增加边、仅减少边和既增加边又减少边这 3 种情况.首先对通常规模的模式图进行实验,实验中,仅增加边时,模式图 P 的节点个数$|V_p|$为 6,边数$|E_p|$为 5,分别增加 1~4 条边后的模式图 P^+的边数$|E_p|$为 6~9(新增加边数最多达到新模式图总边数的 44.4%).仅减少边时,模式图 P 的节点个数$|V_p|$为 6,边数$|E_p|$为 9,分别减少 1~4 条边后的模式图 P^-的边数$|E_p|$为 8~5(减少边数最多达到原模式图总边数的 44.4%).既增加边又减少边时,模式图 P 的节点个数$|V_p|$为 6,边数$|E_p|$为 6.为了平衡考虑增加边和减少边对算法执行时间的影响,假设增加边的数目与减少边的数目相同,因此这种情况下.新模式图边数与原模式图的边数相同,$|E_p|$维持 6 不变.实验结果如图 6 所示.



(a) 模式图仅增加边时,算法在 Epinions 上的执行时间　(b) 模式图仅增加边时,算法在 Slashdot 上的执行时间

(c) 模式图仅减少边时,算法在 Epinions 上的执行时间　(d) 模式图仅减少边时,算法在 Slashdot 上的执行时间

(e) 模式图既增加边又减少边时,算法在 Epinions 上的执行时间　(f) 模式图既增加边又减少边时,算法在 Slashdot 上的执行时间

Fig 6　Execution time of different algorithms when the edges in pattern graph with normal size are changed

图 6　通常规模的模式图的边变化时算法的执行时间

当模式图仅增加边时,PGC_IncGPM 算法只需调用 AddEdges 子算法.图 6(a)和图 6(b)表示了模式图仅增加

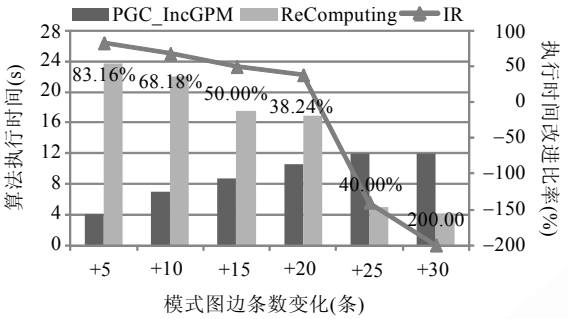
边时,PGC_IncGPM 算法和 ReComputing 算法分别在 Epinions 和 Slashdot 数据集上的执行时间,图中横坐标为模式图边数的变化,“+1”表示模式图中增加了一条边,“+2”表示模式图增加了两条边,依此类推;柱状图表示两个算法在不同情况下的实际执行时间,折线表示 PGC_IncGPM 算法相对于 ReComputing 算法的改进比率.如图所示:对于模式图增加 1 到 4 条边的情况,与 ReComputing 算法相比,PGC_IncGPM 算法的执行时间显著减少.例如,对于 Epinions 数据集,当模式图增加 1 条边时,ReComputing 算法的执行时间为 9.5s,PGC_IncGPM 算法的执行时间为 1.5s,改进比率为 84.21%.通过对两个数据集上的实验数据进行统计分析,当模式图仅增加边时,PGC_IncGPM 算法的平均改进比率为 76.73%.改进的原因分析如下:当模式图中仅增加边时,只需调用 AddEdges 子算法,算法执行时间分为两部分:第 1 部分时间用于在 GPMS 的匹配结果中根据模式图增加的边进行过滤,精简结果图,修改 $mat(\cdot)$,因为结果图相对于数据图规模很小,所以精简结果图的时间相对很短;第 2 部分时间用于对匹配状态可能受影响的节点根据后继关系重新检测,以更新 $sim(\cdot)$和 $cand(\cdot)$的操作,与 Recomputing 算法需要在整个数据图中对所有节点计算 $sim(\cdot)$以及在 $sim(\cdot)$上根据前驱和后继反复筛选节点相比,AddEdges 子算法需要处理的节点数显著减少,并且不需要根据前驱进行筛选,因此这部分时间也很短.总体来看,PGC_IncGPM 算法获得了明显的执行时间改进比率.

当模式图仅减少边时,PGC_IncGPM 算法只需调用 SubEdges 子算法.图 6(c)和图 6(d)表示了模式图仅减少边时,PGC_IncGPM 算法和 ReComputing 算法分别在 Epinions 和 Slashdot 数据集上的执行时间,图中横坐标中“–1”表示模式图中减少了一条边,“–2”表示模式图中减少了两条边,依此类推.如图 6 所示,对于删除 1 到 4 条边的情形,无论是在 Epinions 还是 Slashdot 数据集上,PGC_IncGPM 算法在执行时间上均获得了一定程度的优化,平均改进比率为 21.13%.与模式图增加边相比,当减少边时,算法对执行时间的改进相对较小,主要是因为:减少边时,除了需要在数据图中对匹配状态可能发生改变的点根据后继关系重新进行检测以更新 $sim(\cdot)$和 $cand(\cdot)$外,还需要在更新的 $sim(\cdot)$上对所有点根据前驱和后继反复筛选,以产生新的 $mat(\cdot)$,这些操作所需时间相当于在一个较小规模数据图上进行模式匹配,与在完整的数据图上进行模式匹配的 Recomputing 算法相比,尽管数据图规模的减小能够获得执行时间的改进,但是由于数据图规模减小幅度有限,因此执行时间改进比率有限.进一步分析数据可以发现以下现象:PGC_IncGPM 算法的执行时间并不是随着减少边条数的增加而线性增加.造成该现象的原因是:随着模式图减少边条数的增加,一方面,算法用于在数据图中修改 $sim(\cdot)$和 $cand(\cdot)$的时间逐渐增加;另一方面,减少边数越多,则 P^-中边越少,P^-中各节点的前驱和后继也越少,因此,对 $sim(\cdot)$中各点根据前驱和后继筛选产生 $mat(\cdot)$的时间越少.综合作用下,PGC_IncGPM 算法的总执行时间取决于以上两方面中哪个方面占主导.

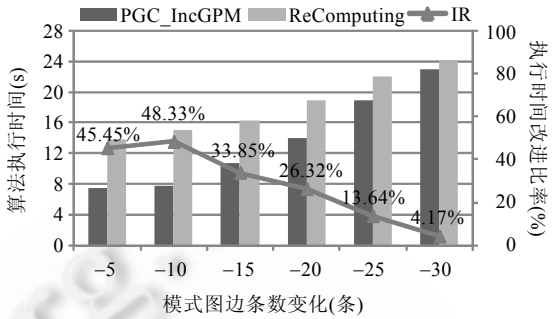
图 6(e)和图 6(f)表示了模式图既增加边又减少边时,两种算法分别在 Epinions 和 Slashdot 数据集上的执行时间.此时,PGC_IncGPM 算法将先调用 AddEdges 子算法,再调用 SubEdges 子算法.横坐标中“+n–n”表示模式图增加了 n 条边,减少了另外 n 条边,其中,n 为 1~4.如图所示,对于模式图既增加边又减少边的情形,PGC_IncGPM 算法在执行时间上仍获得了一定程度的优化,通过统计在 Epinions 和 Slashdot 数据集上的实验数据,算法的平均改进比率为 18.33%.模式图既增加边又减少边时,PGC_IncGPM 算法的执行时间由 AddEdges 子算法和 SubEdges 子算法各自的执行时间共同决定.由于 SubEdges 子算法执行时间占总执行时间的比例更大,因此,总执行时间的改善比率主要取决于模式图减少边时的改进比率.

从图 6(a)、图 6(b)的实验结果也可以看出,对于仅增加边的情形,随着模式图增加的边的数目的增多,传统的 Recomputing 算法的耗时逐渐降低,而 PGC_IncGPM 算法的执行时间逐渐增加.因此,PGC_IncGPM 算法的有效性和模式图的变化幅度密切相关.当模式图的变化幅度达到一定的程度时,PGC_IncGPM 算法是否仍然能够比 Recomputing 算法具有更短的执行时间是一个非常重要的问题.为了充分考量 PGC_IncGPM 算法的有效性和适用范围,我们使用规模较大的模式图做进一步的实验.实验环境设置如下:使用 Epinions 数据集,固定模式图中节点个数$|V_p|$为 20,逐步增大模式图上增加、减少边的条数,测试 PGC_IncGPM 算法和 Recomputing 算法的执行时间,并计算改进比率.实验中,仅增加边时,模式图边数$|E_p|$为 19;仅减少边时,模式图边数$|E_p|$为 39;既增加边又减少边时,假定增加边的条数和减少边的条数相同,模式图边数$|E_p|$为 20.实验中,对于增加边和减少边的条数均

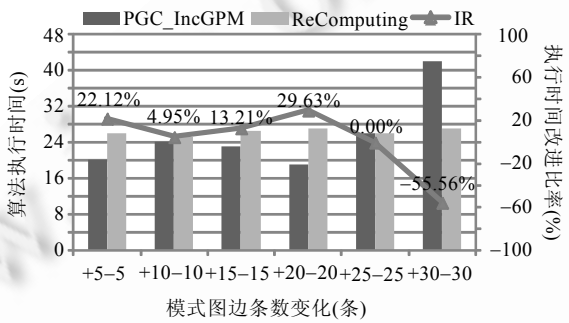
从 5 开始以 5 为间隔逐步增加直到 30,实验结果如图 7 所示.



(a) 模式图仅增加边时,算法的执行时间



(b) 模式图仅减少边时,算法的执行时间



(c) 模式图既增加边又减少边时,算法的执行时间

Fig.7 Execution time of different algorithms when pattern graphs with different sizes changed

图 7 较大规模的模式图的边变化时算法的执行时间

从图 7(a)中可以看出:

- 当增加边的条数从 5,10,15 依次变化到 20 时,新增加边的数目分别占新模式图总边数的 20.83%,34.48%,44.12%和 51.28%;PGC_IncGPM 算法的执行时间均小于 Recomputing 算法的执行时间,改进比率分别为 83.16%,68.18%,50%和 38.24%,平均为 59.89%,改进效果仍很明显.
- 当增加边的条数达到 25 时,新增加边数达到新模式图总边数的 56.82%,此时,PGC_IncGPM 算法的有效性达到边界点,其执行时间开始大于 Recomputing 算法的执行时间,不再适合进行模式图边数继续增加情况下的图模式匹配.

从图 7(b)中可以看出:

- 当减少边的条数逐渐增加到 20 条时,PGC_IncGPM 算法的执行时间均小于 Recomputing 算法的执行时间,改进比率平均为 38.49%.
- 当减少边的条数从 20 开始继续增加到 30 时,新减少边数达到原模式图总边数的 76.92%,此时,PGC_IncGPM 算法的执行时间虽然仍逐渐增加,但是均小于 Recomputing 算法的执行时间,因此,PGC_IncGPM 算法的有效性还未达到边界点.

从图 7(c)中可以看出:

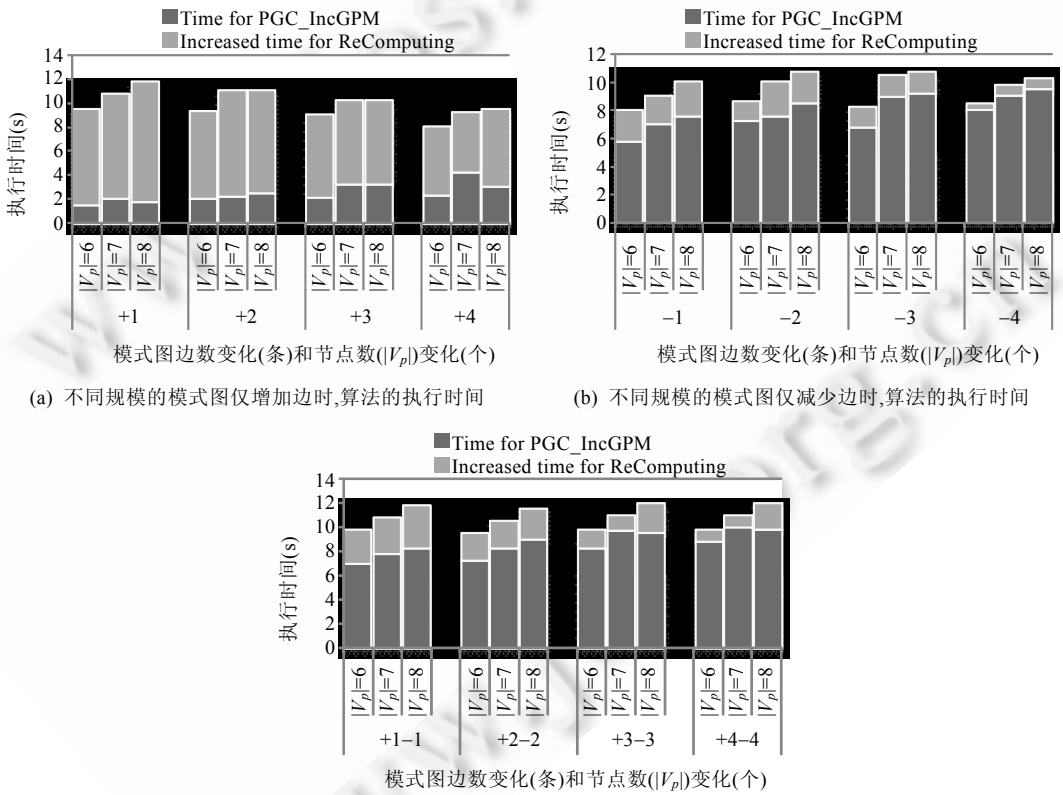
- 当增加的边的条数和减少的边的条数均从 5 变化到 20 时,PGC_IncGPM 算法的执行时间均小于 Recomputing 算法的执行时间,改进比率平均为 17.48%.
- 当增加的边的条数和减少的边的条数达到并超过 25 时,PGC_IncGPM 算法的有效性达到边界点,其执行时间开始大于 Recomputing 算法的执行时间.

分析以上结果可知,当模式图变化的边的数目不超过不变的边的数目时,PGC_IncGPM 算法均有效.

综上,我们提出的 PGC_IncGPM 增量算法的应用场景是模式图发生一定幅度变化且变化的边的数目不超过不变的边的数目时的图模式匹配,该类应用场景在现实生活中非常常见,通常发生变化的边的条数比不变的条数少,否则,模式图已经“面目全非”,模式图的根本特性已经被淹没.对于我们的算法,可以接受“模式图中变化的边的数目不超过不变的边的数目”这样的变化幅度,该幅度已经达到了模式图继承原有特性前提下的极限,因此,PGC_IncGPM 算法具有很好的实用性和适应性.当模式图发生较大变化(模式图中变化的边的数目超过不变的边的数目)时,新的模式图中从原始模式图中继承的特性已不占主导,这种情况下不再适合使用增量算法.

4.3 模式图规模对算法的影响

为了考察模式图规模对算法执行时间的影响,我们在 Epinions 上用不同规模的模式图(模式图节点个数$|V_p|$为 6~8)分别进行仅增加边、仅减少边、既增加边又减少边的图模式匹配.实验结果如图 8 所示,柱状图中每一个图柱分为两部分:深色部分为 PGC_IncGPM 算法的执行时间,浅色部分为 ReComputing 算法比 PGC_IncGPM 算法多出的执行时间,两部分的和即为 ReComputing 算法的执行时间.



(a) 不同规模的模式图仅增加边时,算法的执行时间

(b) 不同规模的模式图仅减少边时,算法的执行时间

(c) 不同规模的模式图既增加边又减少边时,算法的执行时间

Fig.8 Execution time of different algorithms for pattern graphs with different sizes

图 8 不同规模的模式图变化时算法的执行时间

图 8(a)是模式图中仅增加边时,不同规模模式图下的实验结果,模式图初始边数$|E_p|$按照$|E_p|=|V_p|-1$ 设置.如图所示,随着模式图规模的增大,ReComputing 算法的执行时间逐渐增加.因为当模式图规模增大时,ReComputing 算法会产生更多的候选匹配集,增加对候选点根据边筛选过滤的次数,导致执行时间变长.PGC_IncGPM 算法的执行时间随着模式图规模的增加可能增加也可能下降,原因如下:当模式图增加边(u,u')时,执行

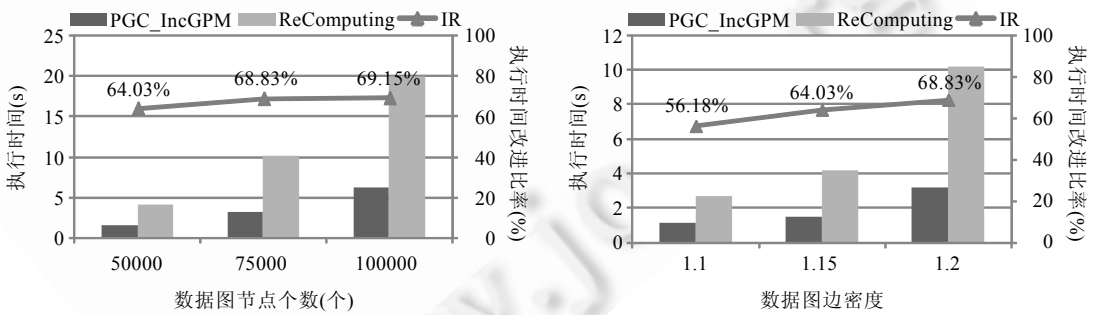
时间取决于两个方面:一个方面是对 $mat(u)$和 $sim(u)$集合中所有点的匹配状态进行检测的时间,模式图规模增大后,$mat(u)$和 $sim(u)$集合变小,检测的时间变短;另一方面是检查由于模式图变化而导致匹配状态发生变化的节点的祖先节点,这部分的时间取决于 u 的祖先节点的多少,数目越多,则检查所需的时间越多.最终,执行时间的变化取决于哪个方面占主导.由于在实验中采用常规的树形模式图,增加的边通常在树中的深度较大,图规模越大,则 u 的祖先节点越多,所以 PGC_IncGPM 算法的执行时间取决于上述第 2 方面的影响,随着模式图规模的增大而增加.但是对于增加的边在树中的深度较浅的情况,算法执行时间取决于上述第 1 个方面的影响,此时,执行时间会随着模式图规模增大而减少.例如图 7(a)中模式图发生"+4"变化,当模式图节点数$|V_p|$从 7 变化到 8 时,由于增加的边深度的较浅,所以 PGC_IncGPM 算法执行时间反而减小.

图 8(b)是模式图中仅减少边时,不同规模模式图下的实验结果,模式图初始边数$|E_p|$按照$|E_p|=|V_p|^{1.2}$ 设置.如图所示,随着模式图中节点个数的增大,PGC_IncGPM 算法的执行时间也逐渐增加.在仅减少边的情况下,PGC_IncGPM 调用 SubEdges 子算法,先在数据图中对 $cand(\cdot)$中的节点进行检测,判断其是否满足后继匹配,再在更新的 $sim(\cdot)$上,根据前驱和后继进行筛选,以产生 $mat(\cdot)$.由于随着模式图规模的增加,$sim(\cdot)$中节点个数增加,在$sim(\cdot)$上,根据前驱和后继进行筛选的时间增加,从而导致算法执行时间增加.

图 8(c)是模式图中既增加边又减少边时,不同规模模式图下的实验结果,模式图初始边数$|E_p|$按照$|E_p|=|V_p|$设置.如图所示,随着模式图规模的增加,PGC_IncGPM 算法的执行时间有增加也有减少的情况.原因是,此时,PGC_IncGPM 算法的执行时间由 AddEdges 子算法和 SubEdges 子算法各自的执行时间共同决定.随着模式图规模的增大,AddEdges 子算法的执行时间可能增加也可能减少,SubEdges 子算法的执行时间增加.最终,PGC_IncGPM 算法执行时间的变化取决于哪个子算法对执行时间的影响占主导.

4.4 数据图规模对算法的影响

为了测试数据图的规模对算法执行时间的影响,我们固定模式图的变化情况,通过分别修改数据图的节点个数$|V|$和边密度α($|V|^{\alpha}$等于图的边数)两个关键参数改变数据图的规模,考察模式图变化情况下,增量算法在不同规模的数据图上进行图模式匹配的执行时间.设置模式图初始的的节点数和边数都为 5,模式图的变化固定为,在模式图中增加两条边并减少另两条边.图 9 为数据图规模变化时的算法执行时间和改进比率.



(a) 数据图不同节点数下的算法执行时间 (b) 数据图不同边密度下的算法执行时间

Fig.9 Execution time of different algorithms over data graphs with different sizes

图 9 不同规模的数据图上算法的执行时间

图 9(a)给出了数据图的边密度α为 1.2、节点个数$|V|$分别为 50 000,75 000 和 100 000 时的算法执行时间.如图所示,随着数据图中节点个数的增加,PGC_IncGPM 算法的执行时间随之增加.因为当数据图中节点个数增加时,候选匹配集($cand(\cdot)$,$sim(\cdot)$)和匹配集 $mat(\cdot)$的规模也增加,使得对已有匹配集和候选匹配集的筛选时间增加,从而算法的执行时间增加.图 9(b)给出了数据图节点个数$|V|$为 75 000,边密度α分别为 1.1,1.15 和 1.2 时的算法执行时间.如图所示,随着数据图边密度的增加,PGC_IncGPM 算法的执行时间也随之增加.因为数据图边密度增加时,数据图中满足模式图中前驱后继关系的点增加,使得候选匹配集和匹配集规模都随之增加,对已有匹

配集和候选匹配集的处理时间相应增加,从而算法的执行时间增加.ReComputing 算法的执行时间同样随数据图规模的增大而增加,但增加的幅度大于 PGC_IncGPM 算法执行时间增加的幅度.随着数据图节点个数和边密度的增加,图 9(a)中,PGC_IncGPM 算法的改进比率从 64.03%增加到 69.15%;图 9(b)中,PGC_IncGPM 算法的改进比率从 56.18%增加到 68.83%.可见,PGC_IncGPM 算法在更大规模的数据图上能够获得更加显著的改进比率,因此在大数据图上具有更好的适用性.

5 总 结

本文研究了面向模式图变化的增量图模式匹配算法,提出了 PGC_IncGPM 算法基本框架.该算法能够在数据图固定不变、模式图发生各种变化时进行增量图模式匹配.在改进传统的图模式匹配算法 GPM 的基础上,提出了 GPMS 算法用于在整个数据图上进行图模式匹配,一方面能够建立后续增量匹配所需的索引,另一方面减少了整个数据图匹配的执行时间.设计实现了面向增边的子算法 AddEdges 和面向减边的子算法 SubEdges 两个核心子算法,用于增量图模式匹配.选取真实数据集和合成数据集两类数据集进行实验,结果表明:

(1) 与 GPM 算法相比,GPMS 算法能够平均减少 26.84%的执行时间.

(2) 与基于 GPMS 的 ReComputing 算法相比,当模式图中变化的边的数目不超过不变的边的数目时,PGC_IncGPM 算法都能有效减少图模式匹配的执行时间.可见,PGC_IncGPM 算法的适用场景是模式图发生一定幅度变化且模式图中变化的边的数目不超过不变的边的数目时的图模式匹配.此时,新的模式图中从原始模式图中继承的特性占主导,PGC_IncGPM 算法具有较好的加速效果.

(3) PGC_IncGPM 算法在更大规模的数据图上能够获得更加显著的执行时间改进比率,在大数据图上具有更好的适用性.

下一步,我们将研究面向模式图变化的分布式图模式匹配算法.另外,在数据图和模式图同时发生变化的情况下,研究如何进行增量式图模式匹配也是一个非常有意义且具有挑战性的课题,是我们未来的研究方向.

致谢 感谢中南大学信息科学与工程学院计算机软件与理论研究所提供实验平台,感谢国家自然科学基金委员会和湖南省教育厅对本研究的资助.

References:

[1] Schenker A, Last M, Bunke H, Kandel A. Classification of Web documents using a graph model. In: Proc. of the 7th Int'l Conf. on Document Analysis and Recognition (ICDAR 2003). Edinburgh: IEEE Computer Society, 2003. 240−244. [doi: 10.1109/ICDAR.2003.1227666]

[2] Liu C, Chen C, Han JW, Philip S. Yu. GPLAG: Detection of software plagiarism by program dependence graph analysis. In: Proc. of the 12th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD 2006). Philadelphia: ACM Press, 2006. 872−881. [doi: 10.1145/1150402.1150522]

[3] Fan WF, Li JZ, Luo JZ, Tan ZJ, Wang X, Wu YH. Incremental graph pattern matching. In: Proc. of the 2011 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2011). Athens: ACM Press, 2011. 925−936. [doi: 10.1145/2489791]

[4] Stotz A, Nagi R, Sudit M. Incremental graph matching for situation awareness. In: Proc. of the 12th Int'l Conf. on Information Fusion (FUSION 2009). Seattle: IEEE, 2009. 452−459.

[5] Wang C, Chen L. Continuous subgraph pattern search over graph streams. In: Proc. of the 25th Int'l Conf. on Data Engineering (ICDE 2009). Shanghai: IEEE Computer Society, 2009. 393−404. [doi: 10.1109/ICDE.2009.132]

[6] Gao J, Zhou C, Zhou JS, Yu J. Continuous pattern detection over billion-edge graph using distributed framework. In: Proc. of the 30th Int'l Conf. on Data Engineering (ICDE 2014). Chicago: IEEE Computer Society, 2014. 556−567. [doi: 10.1109/ICDE.2014.6816681]

[7] Shang H, Zhang Y, Lin X, Yu J. Taming verification hardness: An efficient algorithm for testing subgraph isomorphism. Proc. of the VLDB Endowment, 2008,1(1):364−375. [doi: 10.14778/1453856.1453899]

[8] Zhao P, Han J. On graph query optimization in large networks. Proc. of the VLDB Endowment, 2010,3(1):340−351. [doi: 10.14778/1920841.1920887]

[9] Sun Z, Shao B, Wang HZ, Li JZ, Wang HX. Efficient subgraph matching on billion node graphs. Proc. of the VLDB Endowment, 2012,5(9):788−799. [doi: 10.14778/2311906.2311907]

[10] Fan WF, Li JZ, Ma S, Wang HZ, Wu YH. Graph homomorphism revisited for graph matching. Proc. of the VLDB Endowment, 2010,3(1):1161−1172. [doi: 10.14778/1920841.1920986]

[11] Fan WF, Li JZ, Ma S, Tang N, Wu YH, Wu YP. Graph pattern matching: From intractable to polynomial time. Proc. of the VLDB Endowment, 2010,3(1):264−275. [doi: 10.14778/1920841.1920878]

[12] Ma S, Cao Y, Fan WF, Wo T. Capturing topology in graph pattern matching. Proc. of the VLDB Endowment, 2012,5(4):310−321. [doi: 10.14778/2095686.2095690]

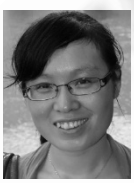
[13] Frad A, Nisar MU, Ramaswamy L, Miller J, Saltz M. A distributed vertex-centric approach for pattern matching in massive graphs. In: Proc. of the IEEE Int'l Conf. on Big Data. Santa Clara Marriott: IEEE Computer Society, 2013. 403−411. [doi: 10.1109/BigData.2013.6691601]

[14] Ramalingam G, Andreps T. An incremental algorithm for a generalization of the shortest-path problem. Journal of Algorithms, 1996,21(2):267−305. [doi: 10.1006/jagm.1996.0046]

[15] Yu WR, Lin XM, Zhang WJ. Fast incremental simrank on link-evolving graphs. In: Proc. of the 30th Int'l Conf. on Data Engineering (ICDE 2014). Chicago: IEEE Computer Society, 2014. 304−315. [doi: 10.1109/ICDE.2014.6816660]

[16] Diptikalyan SH. An incremental bisimulation algorithm. In: Proc. of the 27th Int'l Conf. on Foundations of Software Technology and Theoretical Computer Science. New Delhi: Springer-Verlag, 2007. 204−215. [doi: 10.1007/978-3-540-77050-3_17]

[17] Yi K, He H, Stanoi I, Yang J. Incremental maintenance of XML structural indexes. In: Proc. of the 2004 ACM SIGMOD Int'l Conf. on Management of Data. Paris: ACM Press, 2004. 491−502. [doi: 10.1145/1007568.1007624]



张丽霞(1979－),女,河南周口人,博士生,讲师,主要研究领域为大数据,图算法.

高建良(1979－),男,博士,副教授,CCF 会员,主要研究领域为大数据,图算法.

王伟平(1969－),女,博士,教授,博士生导师,主要研究领域为大数据,网络安全.

王建新(1969－),男,博士,教授,博士生导师,主要研究领域为计算机算法,生物信息学.