ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.3, March 2010, pp.401−414
doi: 10.3724/SP.J.1001.2010.03478

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

一种多到一子图同构检测方法*

张 硕+, 李建中, 高 宏, 邹兆年

(哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

Approach for Efficient Subgraph Isomorphism Testing for Multiple Graphs

ZHANG Shuo+, LI Jian-Zhong, GAO Hong, ZOU Zhao-Nian

(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

+ Corresponding author: E-mail: zhangshuocn@hit.edu.cn

Zhang S, Li JZ, Gao H, Zou ZN. Approach for efficient subgraph isomorphism testing for multiple graphs. *Journal of Software*, 2010,21(3):401−414. http://www.jos.org.cn/1000-9825/3478.htm

Abstract: This paper proposes an approach for subgraph isomorphism testing from a set of priori given small graphs to a large graph issued on-line. In order to reduce the computational cost, based on DFS Code, an elaborate organization of a set of graphs is presented, and a look-ahead-pruning based algorithm of subgraph isomorphism testing from multiple small graphs to a large graph is proposed. Moreover, an index technique based on data mining is introduced. Analytical and experimental results show the on-line computational cost of the proposed method is much less than the state-of-the-art method and it is about one order of magnitude faster than the existing method with more than one order of magnitude less off-line construction time.

Key words: graph matching; subgraph isomorphism; preprocessing; graph indexing

摘 要: 提出一种方法来解决从多个小图到一个大图的子图同构检测问题,其中多个小图是预先给定的,而大图是用户在线提交的.首先,基于 DFS 编码提出一种小图集合的压缩组织方法;其次,提出一种带有前向剪枝技术的从多个小图到一个大图的子图同构检测算法.另外,给出一种有效的基于数据挖掘的索引技术.分析和实验结果证实,所提出方法的在线计算代价远小于现有方法,在线执行时间比现有方法快约一个数量级,离线构造时间快一个数量级以上.

关键词: 图匹配;子图同构;预处理;图索引

中图法分类号: TP301 **文献标识码**: A

图被普遍用于表示具有内部拓扑特征的复杂结构化数据,以及对涉及实体及其联系的现象进行建模.科学和工程进展中产生的大量数据都可以用图来建模和表示.比如,图可以表示化学中化合物的分子结构、图像中的实体以及其间的联系、机械工程领域技术图纸中的基本对象及对社会网络进行建模等等.对图数据的诸多处理,如比较、对齐、包含关系判定等都会涉及到一个基本操作,即子图同构检测.子图同构检测是要在两个图之

* Supported by the National Basic Research Program of China under Grant No.2006CB303005 (国家重点基础研究发展计划(973)); the Key Program of the National Natural Science Foundation of China under Grant Nos.60533110, 60773063 (国家自然科学基金); the Program for New Century Excellent Talents in University of China under Grant No.NCET-05-0333 (新世纪优秀人才支持计划)

Received 2008-07-17; Accepted 2008-10-09; Published online 2009-04-07

间找出保持边结构的顶点的单射,其已被证明是一个 NP 完全问题.随着图数据的积聚,现实应用中经常会遇到从一组而非一个预先存好的小图中分别找出每个小图到用户在线输入图的子图同构的问题,即从一组小图(模型图:model)到一个大图(输入图:input graph)的子图同构检测,其中多个小图是预先给定的,而大图是用户在线提交的.例如,化学中的描述子是在化学反应中表现出特定性质的分子结构的子结构,它由一组原子(顶点)及其间特定的化合键(边)构成,它可以建模为图.当给出一个新的分子(输入图)时,识别出其内部包含的描述子(模型图)及其所在位置可以辅助化学工作者预测该分子可能的性质.在计算机视觉中,空间中的点、线等实体以及它们之间的距离等关系被建模为图.基于预先存储的人、动物和交通工具等基本视觉对象图集合,当给定一幅图像(输入图)时,自动识别和定位存在于其中的基本对象(模型图)有助于计算机对图像的自动理解,这同样在视频检索和机器人视觉中有很强的应用需求.由于两个图间的子图同构检测(以下称为一到一的子图同构检测)已经是一个 NP 完全问题,因此,在大量图数据上进行子图同构检测是一项非常耗时且具有研究意义的问题.

对子图同构检测问题的研究得到了研究者的关注[1,2].经典且得到广泛使用的 Ullmann 算法[3]涉及一个可前向剪枝的带有回溯的树搜索过程.同样基于搜索方法,VF2[2]算法利用一个快速计算的启发式规则进行剪枝,这使其性能得到显著提升.这些子图同构检测算法均是针对从一个图到另一个图的检测,将其直接用于从一组小图到一个大图的情况,即逐个检测每一个小图-大图对效率不高.

对于从一组模型小图到一个输入大图的子图同构检测问题,其中模型图集预先给定而输入图在线提交,文献[4]提出了一种层次组织方法.该方法在离线阶段构建一个层次组织,其中根节点关联一个包含模型图的不同子图的超图.在线给定一个输入图时,根节点超图首先被检测,然后算法基于检测结果来访问层次结构.这种方法的缺点是根节点超图可能比模型图大很多,从而对该超图检测的时间消耗可能比逐个检测模型图的总和还要大.文献[5]基于在预处理阶段构建的决策树,可以在关于输入图大小的二次项时间内完成检测,但决策树的空间消耗及其构造时间均是指数的,即它无法适用于模型图数量比较大的情况.一种基于分解的方法[6]在关于模型图数量次线性的时间内返回结果.但鉴于所用分解策略的机理,它总是产生模型图到输入图的所有子图同构,而不能根据某些应用需求关于每个被包含的模型图只找到一个子图同构即停止考察该模型图,造成了不必要的计算消耗.文献[7]提出一种快速的基于过滤-验证机制的方法,具有非常高的在线处理性能.该过滤-验证机制包括两部分:首先在预处理阶段,离线地构造索引;其次在线处理输入图时,先根据索引过滤掉若干不被包含的模型图来得到候选结果集,再通过子图同构检测算法验证候选图以得到最终结果.但该方法索引构造时间较长;且其在构建索引时需要历史输入图记录,也使该方法的输入不仅只是模型图集和输入图,还有历史输入图记录.

过滤-验证机制在处理子图同构检测相关的问题上,如图搜索问题[8]等,展示了良好的高效性[9,10].在这些方法中,索引的构建方法被广泛研究,它们证实了基于图的数据挖掘(简称图挖掘)是构建索引的一个很有效的手段.然而,由于处理的问题有所不同,这些方法(关联于一到多检测)构造的索引不能直接用于本文研究的多到一的子图同构检测.另外,文献[11]给出了图挖掘的导引.关于挖掘频繁(诱导)子图,有多种方法被提出[12−15].后文将会看到,这些图挖掘方法为本文的离线构造奠定了基础.

综上所述,关联于多到一的子图同构检测问题,据我们所知,采用高效的过滤-验证机制的方法只有文献[7],即 cIndex.它在预处理阶段离线地生成模型图集合的特征子图,即一些稀少出现于历史输入图中的模型图的子图.通过过滤和验证步骤,特征子图和候选图的数量总和通常远小于模型图的数量,由此,所执行子图同构检测的数量大为减少致使处理非常高效.然而,cIndex 将模型图逐个地独立存放,没有采用一种有效的组织来提高性能.另外,随着新提交的输入图不断到来,历史输入图记录相继改变,这使得已建索引过期失效(即使模型图集合并未改变),而其索引动态维护方法需要进行大量子图同构检测,代价昂贵.因此,构造有效的模型图组织结构并支持高效的过滤-验证机制,同时不依赖于历史提交的输入图记录的方法有待探索.

本文从模型图集合的有效组织以加快总体检测效率出发,采用高效的过滤-验证机制来处理多到一的子图同构检测问题.并且不依赖于历史提交的输入图的信息,避免了多次提交输入图后不必要的昂贵更新操作.(1) 不同于已有采用过滤-验证机制的方法,其中模型图集中的图逐个地独立存放,提出一种图集的压缩组织方法 COPE(common prefix tree),将多个图结构化地组织起来以提高检测效率.(2) 提出一种带有前向剪枝技术的

从多个小图到一个大图的子图同构检测算法 COPESI.(3) 给出一个基于图挖掘的索引特征生成方法,其不依赖于历史输入图记录.(4) 分析和实验结果证实,所提出方法的计算代价远小于现有方法,处理响应时间比现有方法快约一个数量级.

本文第 1 节给出相关术语与问题定义.第 2 节给出多到一子图同构检测方法(按图集合的压缩组织方法、子图同构检测算法、索引技术的顺序给出).第 3 节给出实验结果.第 4 节总结全文.

1 基本概念

本文主要考虑连通的无向标号简单图.通过简单修改,我们提出的方法也适用于有向、无标号或不连通的伪图.后文中将我们集中考虑的这类图简称为图.一个图 g 定义为一个四元组(V,E,Σ,l),其中,非空集合 V 代表顶点集合,$E\subseteq V\times V$ 代表边的集合,Σ代表标签的集合,$l:V\cup E\to\Sigma$代表为顶点和边分配标签的标签函数.$|S|$表示集合 S 的基数.图 g 的大小定义为 $size(g)=|E(g)|$,其中,$E(g)$表示图 g 的边集.若无特别说明,图 g 的顶点集被记为 $V(g)$.在后文子图同构检测中,小图也被称为模型图,而大图也被称为输入图.

定义 1.1(子图同构和诱导子图同构). 对于两个图$g=(V,E,\Sigma,l)$,$g'=(V',E',\Sigma',l')$,一个从 g 到 g'的子图同构是一个单射函数 $f:V\to V'$,满足:(1) $\forall u\in V,l(u)=l'(f(u))$;(2) $\forall(u,v)\in E,(f(u),f(v))\in E'$且 $l((u,v))=l'((f(u),f(v)))$.一个从 g 到 g'的诱导子图同构是一个单射函数 $f^I:V\to V'$,满足:(1) $\forall u\in V,l(u)=l'(f^I(u))$;(2) $\forall(u,v)\in E,(f^I(u),f^I(v))\in E'$且 $l((u,v))=l'((f^I(u),f^I(v)))$;(3) $\nexists u,v\in V,(u,v)\notin E$ 且$(f^I(u),f^I(v))\in E'$.

如果存在一个从 g 到 g'的子图同构,则 g 称为 g'的子图,记为 $g\subseteq g'$,称 g'为 g 的超图,称 g'包含 g.如果存在一个从 g 到 g'的诱导子图同构,则 g 称为 g'的诱导子图,记为 $g\subseteq^I g'$,称 g'诱导包含 g.

给定一个图集合 $D=\{g_1,g_2,\ldots,g_n\}$和一个图 g,g 在 D 中的支持集是以 g 为子图的 D 中图的集合,记为 $sup_D(g)=\{g_i|g\subseteq g_i,g_i\in D\}$.$|sup_D(g)|$称为 g 在 D 中的支持度.对于一个用户给定的最小相对支持度阈值$\sigma(0\le\sigma\le1)$,称 g 在 D 中频繁,如果$|sup_D(g)|\ge\sigma\times|D|$.类似地,$g$ 在 D 中的诱导支持集记为 $\sup_D^I(g)$,是以 g 为诱导子图的 D 中图的集合.诱导支持度是诱导支持集的基数.给定最小阈值$\sigma'(0\le\sigma'\le1)$,称 g 在 D 中诱导频繁,如果$|\sup_D^I(g)|\ge\sigma'\times|D|$.

定义 1.2(从多到一的子图同构检测). 输入:一个模型图集合 $D=\{g_1,g_2,\ldots,g_n\}$和一个输入图 g_{IN}.输出:集合$Answer_D(g_{IN})=\{g_i|g_i\subseteq g_{IN},g_i\in D\}$,其中对于$\forall g_i\in Answer_D(g_{IN})$,其输出附带所有(或任意一个)从 g_i 到 g_{IN} 的子图同构.

CIndex[7]在预处理阶段将一些带有特征子图和模型图包含关系的特征子图进行记录和存储;在运行时,利用存储的特征子图来去掉若干假正结果之后,进行一到一子图同构检测以达到减少在线处理计算代价的目的.具体地说,对于模型图集合 D 和特征集 F,子图同构检测的在线处理过程可分为两步:(1) 过滤:取得所有索引特征子图 $f\not\subseteq g_{IN}$,并计算候选结果集 $C(g_{IN})=D-\cup_{f\not\subseteq g_{IN},f\in F}\sup_D(f)$;(2) 验证:对每个图 $g\in C(g_{IN})$进行一到一的子图同构检测,从 $C(g_{IN})$中去除假正结果并返回结果 $Answer_D(g_{IN})$,其中每个结果模型图附带全部(或任意一个)子图同构.

在这个两步骤处理框架中,给定图集 D、输入图 g_{IN} 和特征集合 F,检测执行时间 $T_{on\text{-}line}=T_{filtering}+T_{verification}$,其中,$T_{filtering}$ 是过滤阶段所用时间,$T_{verification}$ 是验证阶段的时间.其可以进一步描述为

$$T_{on\text{-}line}=T_{filtering}+|C(g_{IN})|\times T_{iso\_cand} \tag{1}$$

其中,T_{iso_cand} 为从候选集中一个模型图到输入图 g_{IN} 的子图同构检测的平均时间.由于子图同构检测是 NP 完全的,故$|C(g_{IN})|\times T_{iso_cand}$ 通常占据了 $T_{on\text{-}line}$ 的主要部分.如果特征集很大以致无法被内存容纳,则 $T_{filtering}$ 将会很大.

本文提出的方法减少代价模型(1)中的全部 3 项,即$|C(g_{IN})|$,T_{iso_cand} 和 $T_{filtering}$.首先,提出一种图集合的压缩组织方法 COPE 来提高在线处理效率.其次,给出一个采用轻量级且具有更强过滤能力的前向剪枝策略的一到一子图同构检测算法,然后提出一种新的从多个图到一个图的高效子图同构检测算法 COPESI.再次,提出显著有效的子图即索引特征子图的生成方法.在过滤-验证框架下,提出的压缩组织方法 COPE 和多到一的子图同构检测算法 COPESI 可分别应用于过滤和验证两个阶段,因为它们均属于从多个预先给定的小图到一个在线提交的大图的子图同构检测的范畴.T_{iso_cand} 和 $T_{filtering}$ 被前两项技术减少;$|C(g_{IN})|$的降低受益于全部 3 项技术.

2 多到一的子图同构检测

多到一的子图同构检测过程是:(1) 离线构造阶段:首先,以 COPE 方式压缩组织模型图集合 D,记为 DTr (与此同时生成初始特征集);其次,构造索引结构,即先从初始特征集中选择显著有效的子图作为特征子图,再将选择得到的特征子图集合 F 以 COPE 方式压缩组织为 FTr.(2) 在线处理阶段:给定一个输入图 g_{IN},首先以 FTr 上的所有特征子图(F)和 g_{IN} 为输入,执行 COPESI 来构造候选集 $C(g_{IN})=D-\cup_{f\subseteq g_{IN},f\in F}\sup_D(f)$,其中,对于每个特征子图要求至多只找到一个子图同构;其次,用 COPESI 作用于以 DTr 形式组织的候选集上来得出最终结果.

2.1 图集合的压缩组织

为了减少在线处理代价,本节给出图集合的压缩组织方法.首先将每个模型图转化为一个序列,然后由所有序列构造一个前缀树.多个序列的公共前缀对应于原来多个图的公共子图.下面分别介绍图序列化方法以及 COPE 组织结构和构造方法.

2.1.1 图的序列化

基于搜索的子图同构检测方法要逐个考察模型图的每个顶点,现有方法尚未考虑模型图顶点的考察顺序对整体性能的影响问题[2,3].我们发现,特定的顺序可以提高算法性能.

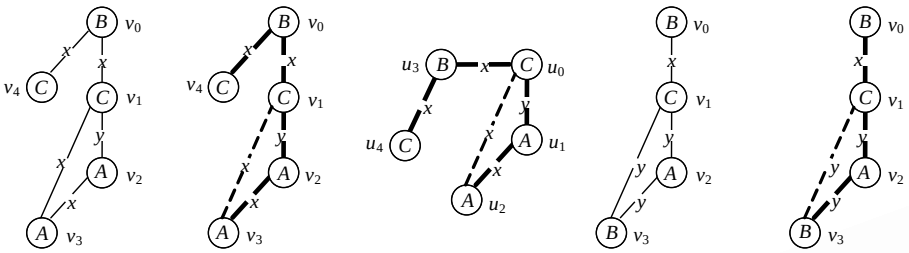
首先,回顾一种有效的图序列化方法,即基于先深搜索(DFS)技术的 DFS 编码[12].一个图的 DFS 编码是根据在图上进行一次先深搜索而确定并生成的边的序列.显然,一个 DFS 编码不是图的任意边序列,在该序列中存在一定的约束.其中,DFS 编码邻接约束阐述了 DFS 编码生成时的一种特定的边增长方式,即最右扩展.对于图 g 及其先深搜索树 T(其 DFS 编码记为 $dfsc_T(g)$),根据先深搜索的访问顺序,为 g 的每个顶点标出下标.第 1 个访问的顶点称为根,最后一个访问的称为最右顶点.在搜索树 T 上,从根到最右顶点的路径称为关于 T 的最右路径.给定 g 和 T,图的所有边分为两类,在 T 上的边称为前向边,记为 $E_{f,T}$;其余边称为后向边,记为 $E_{b,T}$.在 DFS 编码中,边的顺序也可以叙述为:前向边的位置一致于其指向顶点的下标的升序,所有源于同一顶点的后向边出现且仅出现在紧随指向该顶点的前向边的位置;源于同一顶点的多个后向边的顺序一致于它们指向顶点的下标升序.DFS 编码可以通过如下构造得到:基于一个先深搜索树添加指向下一个下标顶点的前向边到当前编码中,然后添加所有源于该顶点的后向边;重复此过程直至所有边被包括进来.在图 1 中,g_1 是一个例图,图 1(b)和图 1(c)是其两个不同的先深搜索树,其中粗实线表示前向边,虚线表示后向边.对应于各自的先深搜索树,g_1 的 DFS 编码分别为 $(v_0,v_1,B,x,C)(v_1,v_2,C,y,A)(v_2,v_3,A,x,A)(v_3,v_1,A,x,C)(v_0,v_4,B,x,C)$ 和 $(u_0,u_1,C,y,A)(u_1,u_2,A,x,A)(u_2,u_0,A,x,C)(u_0,u_3,C,x,B)(u_3,u_4,B,x,C)$.同样,图 1(e)是例图 g_2 的一个先深搜索树,g_2 对应于图 1(e)的 DFS 编码为 $(v_0,v_1,B,x,C)(v_1,v_2,C,y,A)(v_2,v_3,A,y,B)(v_3,v_1,B,y,C)$.

然后,分析并提出 DFS 编码的一系列性质.先深搜索的多样性导致了一个图具有多种 DFS 编码.但给定一个 DFS 编码 $dfsc$,其对应的图是唯一确定的,记为 $g(dfsc)$,因为先深搜索过程记录了图的完整信息.因此,对于两个相同的 DFS 编码,其对应的图是彼此同构的;反之却不然.在对图 g 进行先深搜索的同时增量地构建其 DFS 编码的边序列过程中的任意时刻,已经构造的子序列即前缀 P 对应着由这些访问过的边组成的 g 的一个子图,记为 $g(P)$.其中间子图 $g(P)$是 g 的一个连通子图.另外,根据 DFS 编码的最右扩展性质可知,对任一 DFS 编码序列的前缀,如果其在序列中的下一条边是前向边,则该前缀对应于其涉及的节点集合的诱导子图.

然而,对于图 g 及其任一子图 sg,并不总是分别存在它们的 DFS 编码使得 sg 的编码是 g 的编码的前缀.下面的引理指出了一个具有此性质的充分条件.

引理 2.1(前缀充分性). 给定图 $g=(V,E,\Sigma,l)$和它的一个具有 Hamilton 路的诱导子图 $hig=(V',E',\Sigma',l')$,在 hig 的所有 DFS 编码和 g 的所有编码中,分别存在编码 $dfsc(hig)$和 $dfsc(g)$满足 $dfsc(hig)$是 $dfsc(g)$的一个前缀.

综上所述,图 g 的 DFS 编码的任一前缀均为一个合法的 DFS 编码,它所对应的图是 g 的一个连通子图.对于 g 的任一具有 Hamilton 路的诱导子图 hig,总存在 hig 的一个 DFS 编码,其为 g 的某个编码的前缀.不连通图的编码即为其逐个连通分支编码的连接,显然也满足上述性质.后文将具有 Hamilton 路的诱导子图简称为 H 诱导子图.



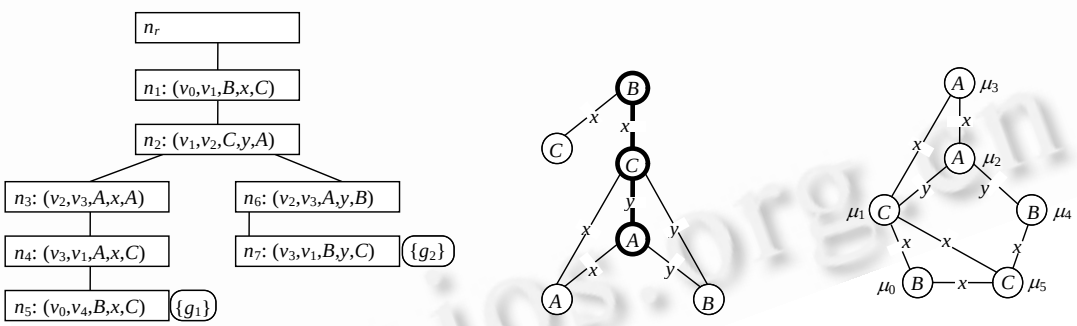
(a) Graph g_1 (b) DFS tree (c) DFS tree (d) Graph g_2 (e) DFS tree
(a) 例图 g_1 (b) DFS 树 (c) DFS 树 (d) 例图 g_2 (e) DFS 树

Fig.1 Two sample graphs and their DFS trees

图 1 两个例图及其先深搜索(DFS)树

2.1.2 COPE 组织结构

采用上述图序列化方法,每个图序列化为一个 DFS 编码.COPE 组织结构是一个由所有图的 DFS 编码构成的前缀树,树根节点不对应任何边,它的多个儿子节点分别对应所有不同 DFS 编码边序列的第 1 条边.在前缀树中,某些节点关联着非空的图 ID 集合,表示从根到该节点的路径对应的边集构成该 ID 对应的模型图;否则,图 ID 集合为空的节点表示相应路径的边集不构成任一模型图.在图 2 中,图 2(a)是一个由图 1 的 g_1 和 g_2 构造而成的 COPE 组织结构,此处的 g_1 和 g_2 分别按图 1(b)和图 1(e)先深搜索树序列化.在该前缀树的组织结构中,树根 n_r 不对应任何边,n_1 对应于边(v_0,v_1,B,x,C),n_2 对应于边(v_1,v_2,C,y,A),等等.n_5 关联着图 ID 集合$\{g_1\}$,n_7 关联$\{g_2\}$,其他节点的图 ID 集合均为空(略去显示).图 2(b)示意了以图 2(a)的方式组织时,g_1 和 g_2 的直观压缩组织情况.粗实线表示被合并在一起并仅存储一次的 g_1 和 g_2 的公共(诱导)子图.



(a) COPE (b) Interpretation
(a) COPE (b) 直观示意

Fig.2 A sample COPE and its interpretation

图 2 COPE 组织结构及其直观示意

Fig.3 A sample input graph g_{IN}

图 3 输入图 g_{IN}

2.1.3 COPE 结构构建

一个图具有多种 DFS 编码,COPE 构造的关键就是为每个图指定一个 DFS 编码,以达到节省对多个图进行子图同构检测开销的目的.也就是说,通过有效合并多个图的公共子图来减少总在线处理代价.

首先,示例阐释 COPE 结构减少总在线处理代价的机理.考虑图 2 中 g_1 和 g_2 的 COPE 结构及图 3 中的输入图 g_{IN}.在从该 COPE 到 g_{IN} 的子图同构检测过程中,一旦我们找到了从粗实线表示的子图到输入图的一个子图同构 f 后,我们可以进一步检测 g_1,并得出其为输入图的子图;然后回溯到 f(而非回溯到初始状态),并进一步检测 g_2,得出其不是输入图的子图.这样,由于未回溯到初始状态,使得以这样方式的检测比逐个依次检测节省了一次子图同构 f 的检测代价,即节省了从粗实线子图到输入图的一次子图同构检测.因此,越大程度地合并公共子图,将会带来越大的代价节省空间.

其次,用示例来说明需要合并公共诱导子图而非任意的公共子图.一到一的子图同构检测(参见文献[2,3])涉及一个顶点级的带有回溯的树搜索过程.首先考虑从 g_1 到 g_{IN} 的子图同构检测过程.在检测过程中,在验证了 g_1 中由(*A*,*x*,*A*)(*A*,*y*,*C*)和(*C*,*x*,*A*)这 3 条边构成的三角形子图是 g_{IN} 的子图后,算法添加顶点 v_0 到该三角形子图,并检查该增长的子图(g_1 的关于$\{v_0,v_1,v_2,v_3\}$诱导子图)是否为 g_{IN} 的子图,来继续搜索过程(最终得出 g_1 是 g_{IN} 的子图).其次考虑从 g_2 到 g_{IN} 的子图同构检测过程.在检测过程中,倘若验证了 g_2 中由(*B*,*y*,*A*)(*A*,*y*,*C*)两条边构成的子图 *nig* 是 g_{IN} 的子图,之后不能添加顶点 v_0 并考察增长的子图来继续算法运行,因为基于子图 *nig* 的顶点增长的子图无法成为 g_2(缺少(*B*,*y*,*C*)边).这样,即使知道 *nig* 是 g_{IN} 的子图,仍然需要回溯至初始状态并重新检查 g_2 的所有顶点来考察 g_2 是否为 g_{IN} 的子图.归根结底是因为缺少(*B*,*y*,*C*)边,即 *nig* 不是 g_2 的诱导子图.综上所述,要想再次利用搜索过程中已经得到的子图同构而免去回溯到初始状态,就要求搜索过程中所考察的子图是模型图的诱导子图.因此准确地说,当有多个模型图时,通过一并考察多个模型图的公共诱导子图,可以节省在线计算代价.可以看到,上一段实例中的粗实线子图恰好是 g_1 和 g_2 的公共诱导子图.

因此,以尽可能多地合并尽可能大的公共诱导子图为目标来构造 COPE 组织结构.一个模型图可以开始于它的任何 H 诱导子图来产生顶点顺序,我们按照上述目标为每个模型图选择最合适的 H 诱导子图.另外,由于一个图具有指数数量的诱导子图,故找出每一个模型图的所有诱导子图是困难的,本文将挖掘频繁诱导子图作为有效 H 诱导子图选择之前的工作.

算法 1 给出了 COPE 的构建算法,其分为 3 部分:挖掘频繁 H 诱导子图、对模型图的 H 诱导子图分配以及前缀树的形成.具体地说,构建算法首先挖掘找出满足诱导支持度约束的频繁 H 诱导子图.每个诱导子图关联着其诱导支持集,支持集中的每个模型图还记录着一个关于其部分顶点的序列,这些顶点对应于该诱导子图到此模型图的一个诱导子图同构(第 1 行).然后,采用选择具有最大$(2^{|V(hig)|-1})\times(|\sup^I_{D-A}(hig)|-1)$值的启发式策略为每个模型图分配一个 H 诱导子图,其中,$V(hig)$表示图 *hig* 的顶点集(第 2~10 行). $\sup^I_{D-A}(hig)=\{g|hig\subseteq^I g,g\in D-A\}$. 再次,基于上述分配情况,为每个模型图构造 DFS 编码来构造前缀树 COPE.即先将赋予得到 H 诱导子图的模型图按照挖掘时得到的部分顶点的序列顺序生成其 DFS 编码的前半部分,继续随机、顺序地先深搜索生成编码的剩余部分(第 11 行、第 12 行);再对未分配到诱导子图的模型图进行随机顺序的先深搜索以生成其 DFS 编码(第 13 行).构建前缀树时,在对应每个编码最末边的节点关联的图 ID 集合中,添加该编码对应的模型图的 ID.

算法 1. COPE 构建.

输入:一个图集 *D*,一个最小支持度阈值 σ^I_T;

输出:COPE 压缩组织结构.

1 $HIG\leftarrow HIGMining(D,\sigma^I_T)$

2 $A\leftarrow\varnothing$; $assignSeq\leftarrow\varepsilon$

3 while $\exists hig\in HIG$ s.t. $|\sup^I_{D-A}(hig)|>1$ do

4 select $hig\in HIG$ with largest $(2^{|V(hig)|-1})\times(|\sup^I_{D-A}(hig)|-1)$ s.t. $|\sup^I_{D-A}(hig)|>1$

5 *assignSeq* 在对应于集合 $\sup^I_{D-A}(hig)$ 中的所有模型图的位置上赋值为 *hig*

6 $A\leftarrow A\cup\sup^I_{D-A}(hig)$

7 while $\exists hig\in HIG$ s.t. $|\sup^I_{D-A}(hig)|=1$ do

8 select $hig\in HIG$ with largest $|V(hig)|$ s.t. $|\sup^I_{D-A}(hig)|=1$

9 *assignSeq* 在对应于集合 $\sup^I_{D-A}(hig)$ 中的所有模型图的位置上赋值为 *hig*

10 $A\leftarrow A\cup\sup^I_{D-A}(hig)$

11 for *assignSeq* 的每一个位置 *pos* do

12 根据 *pos.hig*(由上述步骤在 *pos* 位置上赋予的 H 诱导子图)得到 *pos.g*(*pos* 位置关联的模型图)的部分顶点序列 *vseq*(其值本身由 HIGMining 过程得到),遵循 *vseq* 生成 *pos.g* 的 DFS 编码的前半段,继续随机、顺序地先深搜索来生成其后半段

13 为未分配到 H 诱导子图的模型图进行随机、顺序先深搜索,构造一个随机的 DFS 编码.

14 用所有的 DFS 编码构造前缀树,在树上对应每个编码最末边的节点关联的图 ID 集合中,添加该编码对应的模型图的 ID,树根节点 r 关联的图 ID 集合 $r.IDset\leftarrow\varnothing$;

Hamilton 路径的判定具有很高的复杂度.为了避免指数时间的搜索,我们采用最小化度数生成树(MDST)问题的上界为Δ^*+1,即$\Delta'\leq\Delta^*+1$ 的多项式时间算法[16]取而代之,其中Δ'和Δ^*分别表示输出生成树的顶点最大度数和最小化度数生成树的顶点最大度数.如果$\Delta'\leq2$,则该输出的生成树(即 Hamilton 路)及对应的子图作为这一步的结果.由于版面所限,这里不显式地对所采用的特制的频繁诱导子图挖掘算法 HIGMining 给予描述.

下面进行 COPE 构造复杂度的分析.除了从模型图集中进行诱导子图挖掘外,COPE 构造涉及的主要计算代价是诱导子图分配.对于每个诱导子图,其诱导支持度和诱导支持集都被分别记录,在更新 A 时,一方面删除所有诱导子图的诱导支持集中的 A 中元素即模型图 ID 集合,一方面更新诱导子图的诱导支持度,故对于每个模型图,计算次数以$|HIG|$为上界.第 4 行的最大值选择可以在更新支持度的同时完成,故鉴于共有$|D|$个模型图,在第 3~6 行维护所有诱导子图的诱导支持度和诱导支持集需$O(|HIG|\times|D|)$时间.同理,第 7 行~第 10 行也消耗$O(|HIG|\times|D|)$时间.故诱导子图分配时间复杂度为$O(|HIG|\times|D|)$.由顶点序列生成 DFS 编码的时间为$O(|D|\times\max\{|V(g)|+size(g)|g\in D\})$.故除去诱导子图挖掘,COPE 构造的时间复杂度为$O(|D|\times(\max\{|V(g)|+size(g)|g\in D\}+|HIG|))$.

综上所述,所有预先存好的模型图采用 COPE 方法形成一个压缩组织结构,称为 *DTr*.

2.2 子图同构检测算法

本节首先描述一种基于搜索的可前向剪枝的一到一子图同构检测算法,它包括若干具有更强剪枝能力的剪枝条件.然后以此为基础,并基于 COPE 压缩组织,给出从 COPE 中多个模型图到输入图的多到一子图同构检测算法 COPESI.

基于搜索的子图同构检测方法要逐个考察模型图的每个顶点,并在一系列剪枝条件均失败的情况下枚举从该顶点到输入图中顶点的所有可能映射.但现有方法尚未考虑模型图顶点的考察顺序[2,3].后文给出的算法以特定顺序考察模型图顶点,以此来提高算法的整体性能.

从模型图 $g_m=(V_m,E_m,\Sigma_m,l_m)$到输入图 $g_{IN}=(V_{IN},E_{IN},\Sigma_{IN},l_{IN})$的子图同构检测涉及一个搜索子图同构单射函数(或称完全映射)的过程.其中,单射函数映射 V_m 的所有顶点到 V_{IN} 的一部分互不相同顶点.算法涉及的搜索树用状态空间表示(SSR),其输出是一个完全映射 CM,或为空,如果任何完全映射无法找到.CM 是 V_m 的所有顶点到 V_{IN} 的一部分互不相同顶点的对应匹配的顶点对的集合,例如,对于图 1 的 g_1(对应于 g_m)和图 3 的 g_{IN},一个 $CM=\{(v_0,\mu_0),(v_1,\mu_1)(v_2,\mu_2),(v_3,\mu_3),(v_4,\mu_5)\}$.状态空间以如下方式构造:$g_m$ 的顶点按在其 DFS 编码 $dfsc(g_m)$中的顺序被考察,g_{IN} 根据 $dfsc(g_m)$以顶点增长的方式逐个顶点地被比较.在状态空间中,每个状态 t 关联一个部分映射 $M(t)$,它是 CM 的一个子集.状态间的转换对应一个新的匹配顶点对的添加.算法运用带有回溯的先深搜索作用于状态空间.对于一个顶点 $v\in V_m$,在经过若干条件考察后,g_{IN} 中一个局部匹配的顶点μ被选择,进而顶点对(v,μ)被添加到当前部分映射中,形成当前状态的一个儿子状态,并递归执行下去;当 g_{IN} 中没有这样的顶点μ时,算法回溯.具体地说,对于 SSR 中的一个状态 t,算法首先计算出潜在匹配的顶点对集合 $PS(t)$:$dfsc(g_m)$中下一条前向边所指向的顶点为 g_m 的下一个待考察顶点 v,$PS(t)$中顶点对的第 1 个分量相同,均为 v;借助于 $M(t)$,找到该前向边的源顶点在 g_{IN} 中对应的顶点ξ,ξ在图 g_{IN} 中的邻居顶点除去已存在 $M(t)$中的顶点外,构成了 $PS(t)$中互不相同顶点对的第 2 个分量.继续以 g_1 和 g_{IN} 为例,并采用图 1(b)确定的 $dfsc_b(g_1)$,对于部分映射 $M(t)$为$\{(v_0,\mu_0)\}$的状态 t,此时,上述的 v 为 v_1,ξ为μ_0,故 $PS(t)=\{(v_1,\mu_1),(v_1,\mu_5)\}$.在算法执行过程中,为了避免不必要的搜索,集合 $PS(t)$应该被精炼以去掉那些不能后续扩展出 CM 的顶点对.用于精炼 $PS(t)$的条件成为剪枝条件.同时,用如下标记:$V_m(t)=\{v|(v,\mu)\in M(t)\}$, $V_{IN}(t)=\{\mu|(v,\mu)\in M(t)\}$;关于 $V_m(t)$和 $V_{IN}(t)$的邻接顶点集合分别为 $A_m(t)=\{u|(v,u)\in E_m, v\in V_m(t)\}$和$A_{IN}(t)=\{\zeta|(\mu,\zeta)\in E_{IN},\mu\in V_{IN}(t)\}$;$N(v)$为顶点 v 在相应图上的邻居集合,$deg(v)$为顶点 v 在相应图上的度.

首先,V_m 中的顶点不能匹配 V_{IN} 中不同标号的顶点.

对于顶点对(v,μ),其中 $v\in V_m,\mu\in V_{IN}$,有常数时间的剪枝条件:

剪枝条件1. $l_m(v)\neq l_{IN}(\mu)\vee deg_m(v)>deg_{IN}(\mu)$.

其次,对于在当前状态 t 的 $PS(t)$中的顶点对(v,μ),其中 $v\in V_m,\mu\in V_{IN}$,有如下的剪枝条件.这里,t'是由 t 通过添加顶点对(v,μ)转换而来的状态.注意,显然在 $PS(t)$中的任一顶点对(v,μ)都不满足剪枝条件 1.

剪枝条件2. $\neg(\forall x\in V_m(t)\cap N_m(v),\xi\in N_{IN}(\mu)$且 $l_m(x,v)=l_{IN}(\xi,\mu))$,其中$(x,\xi)\in M(t)$.

剪枝条件3. $|VS_m|>|VS_{IN}|$,其中 $VS_m=A_m(t)\cap N_m(v)-V_m(t),VS_{IN}=A_{IN}(t)\cap N_{IN}(v)-V_{IN}(t)$.

剪枝条件4. $|VS_m|>|VS_{IN}|$,其中 $VS_m=A_m(t)\cup N_m(v)-V_m(t),VS_{IN}=A_{IN}(t)\cup N_{IN}(v)-V_{IN}(t)$.

剪枝条件5. $|VS_m|>|VS_{IN}|$,其中 $VS_m=N_m(v)-V_m(t),VS_{IN}=N_{IN}(v)-V_{IN}(t)$.

剪枝条件 2 保证了部分映射 $M(t')$本身是一个从 g_m 的关于顶点 $V_m(t')$的诱导子图到 g_{IN} 的一个子图同构.其他剪枝条件通过分别适当地划分 v 和μ的邻居或其他顶点集,以较细的粒度向前考察顶点的连接情况,从而有较大可能地提前剪去不能产生完全映射的搜索分枝.总体来说,剪枝条件(pruning condition)可划分为 3 个级别:(1) $PCond_{vertex}$:单个顶点级条件(剪枝条件 1);(2) $PCond_{self}$:部分映射自身验证级条件(剪枝条件 2);(3) $PCond_{ahead}$:前向验证级条件(剪枝条件 3~剪枝条件 5).它们分别对应于只考察顶点对中的两个顶点,考察部分映射本身是一个从 g_m 的诱导子图到 g_{IN} 的子图同构,以及考察未来顶点以得出当前部分映射不能扩展出完全映射的情况.显然,通过哈希法,这些剪枝条件均可在$O(\max\{|V_m|,|V_{IN}|\})=O(|V_{IN}|)$时间内得以判定.

由 DFS 编码的最右扩展性质可知,在检测过程中构造匹配顶点集合时,令当前状态为 t,g_m 的关于 $V_m(t)$的诱导子图中的不在当前最右路径上的任一顶点(记为 $V_{nrp}(t)$)都不与 $V_m-V_m(t)$的任一顶点邻接.以图 1 中 g_1 和图 1(c)确定的 $dfsc_c(g_1)$以及图 3 中 g_{IN} 为例,令当前状态 t 对应的部分映射 $M(t)=\{(u_0,\mu_1),(u_1,\mu_2),(u_2,\mu_3),(u_3,\mu_0)\}$,则此时 g_1(即 g_m)的关于 $V_m(t)$的诱导子图为 g_1 中除去边(u_3,u_4)以外的子图,其不在当前最右路径(为$\langle u_0,u_3\rangle$)的顶点 u_1 或 u_2 都不与 $V_m-V_m(t)$中顶点即 u_4 邻接.因此,我们可以精炼紧缩 $A_{IN}(t)$,通过去掉只邻接于 $V_{nrp\text{-}IN}(t)$的顶点,其中 $V_{nrp\text{-}IN}(t)$是指所有按照当前部分映射由顶点 $V_{nrp}(t)$映射到 g_{IN} 的那些顶点.这样,剪枝条件得到增强而且几乎无须额外的计算消耗.具体地说,在算法中,当待添加 $PS(t)$中的顶点对(v,μ)来转换状态 t 时,如果指向 v 的前向边的源顶点 u 是当前搜索下的非最右顶点,$A_{IN}(t)$的 DFS 精炼过程如下:令最近一次添加的不属于当前搜索下最右路径的顶点为 w(w 已被考察过),通过部分映射 $M(t)$得到 w 对应在 g_{IN} 的顶点为ζ.从 $A_{IN}(t)$中去掉所有这样的顶点,其在 $V_{IN}(t)$的范围内只邻接ζ而不邻接其他顶点.对于不连通图,仅要考虑当前的连通分支.此操作可在$O(|V_{IN}|)$时间内完成.

定理 2.1(正确性). 通过搜索所有经上述剪枝条件剪枝后的状态,所有完全匹配(即子图同构)都能得到.

定理 2.2(搜索空间的状态数). 对于任意输入,基于上述剪枝条件的一到一子图同构检测算法所涉及的状态总数不多于 VF2[2]算法的状态总数.

结合第 2.1.3 节的分析可知,当有多个模型图时,通过一并考察多个模型图的公共 H 诱导子图,可以节省在线计算代价.从 COPE 中多个模型图到输入图的子图同构检测算法 COPESI 描述于算法 2,它对应于只寻找一个子图同构的版本;找出每个被包含模型图的在输入图中所有子图同构的版本只需去掉第 15 行、第 23 行、第 24 行.算法 COPESI 将子图同构检测过程中每个候选匹配顶点对的比较和验证划分为两部分:一部分涉及 $PCond_{vertex}$ 和 $PCond_{self}$,算法执行从当前诱导子图到输入图的检测.无论 COPE 中相应分支关联多少个模型图,这一步检测都相当于只在一个公共子图上执行;另一部分涉及 $PCond_{ahead}$,算法作用于承载一些模型图的 COPE 前缀树的子树,并按照该子树的先根遍历序检测从每个其上的模型图到输入图的子图同构.如果子树的某个节点使其关联的模型图不能子图同构到输入图,则以此节点为根的子树关联的模型图都一定不能子图同构到输入图,因而被省去检查.

算法 2. COPESI.

输入:模型图集 D 的 COPE 结构 Tr,输入图 g_{IN};

输出:结果集 $Answer_D(g_{IN})$,简记为 ANS.

1 $ANS\leftarrow\varnothing$;$TX\leftarrow Tr$ 上所有模型图 ID 列表;

2 $SIonCOPE(n_r,\varnothing,TX,ANS)$;

3 return ANS; /* n_r 是 Tr 的树根节点*/

Procedure: *SIonCOPE*(*n*,*pm*,*TX*,*ANS*) /* *pm* 为当前状态的部分映射,*TX* 为待考察模型图 ID 列表*/
4 if *pm* 中最后添加的顶点对的模型图顶点 *l* 对应 *Tr* 上节点 *n* 的图 ID 集合 *IDset*≠∅ then
5 *ANS*←*ANS*∪*IDset*; *TX*←*TX*−*IDset*;
6 if *TX*=∅ then return;
7 if *Tr* 上节点 *n* 有 *cnum*(>1)个儿子节点 then
8 将 *TX* 划分为 *cnum* 个不相交集合,即
$\mathcal{Z}$={TX_i|$\cup_i TX_i$=*TX*,$\cap_i TX_i$=∅,TX_i∈*TX*}满足 TX_i 对应于 *n* 的第 *i* 个儿子节点;
9 else $\mathcal{Z}$←{*TX*};
10 for $\mathcal{Z}$中每一个 TX_i(关于 *i* 升序) do
11 令 TX_i 中第 1 个模型图为 mg_1,基于 mg_1 根据 $PCond_{vertex}$ 和 $PCond_{self}$ 条件计算可添加到 *pm* 的候选顶点对集合 *PS*;
12 if *PS*=∅ then continue;
13 基于 *PS* 和 *pm*,对集合 A_{IN} 进行 DFS 精炼;
14 for *PS* 中每一个点对 *p* do
15 if TX_i=∅ then break;
16 *TX*′←TX_i; *ANS*′←∅;
17 for *TX*′中每一个模型图 *g* do
18 if 满足关于 *g* 的 $PCond_{ahead}$ 条件 then
19 *TX*′←*TX*′−{*g*};
20 if *TX*′≠∅ then
21 将 *p* 附加于 *pm* 中得到 *pm*′;
22 *SIonCOPE*(*n*,*pm*′,*TX*′,*ANS*′);
23 TX_i←TX_i−*ANS*′; *ANS*←*ANS*∪*ANS*′;
24 *TX*←$\cup_{S\in\mathcal{Z}}S$;

在算法 COPESI 中,*Tr* 和 *q* 是全局变量.在第 4 行,算法从节点 *n* 沿着 *TX* 中第 1 个模型图在 *Tr* 树的路径,向下找到第 1 个指向非 *l* 顶点的前向边,其上一条边对应的节点即为 *n*.第 7 行、第 8 行的实现细节讨论如下:由于每次都进行对 *Tr* 树相应子树的先根遍历效率不高,故下面给出一种模型图 ID 的表示方法,以快速划分 *TX* 列表.每个模型图的 ID 是一个二元组(*sn*,*struSeq*),其中 *sn* 是一个序号,它与传统的图 ID 意义相同;*struSeq* 是根据该图在相应 COPE 中的组织存放情况得到的一个序列,类似于 Dewey 编码[17],不同之处是一条公共路径对应于一个自然数.举例来说,图 1 中的 g_1 和 g_2 关于图 2 的 COPE 结构的 *struSeq* 分别为“1.1”和“1.2”.采用这种模型图 ID 满足第 7 行的条件,说明 *TX* 中所有模型图的 *struSeq* 的下一个自然数 *next-subseq* 对应于 *Tr* 树当前节点 *n* 的儿子节点.故只考察 *TX* 中图 ID 的 *next-subseq* 便可完成 *TX* 划分.如果第 22 行函数调用改变 A_{IN} 值,则调用后要立即恢复其值.

以图 2 的 COPE 结构和图 3 的输入图来示例 COPESI 算法.(1) 执行 *SIonCOPE*(n_r,∅,*TX*={g_1,g_2},*ANS*=∅)调用.在此调用中,第 4 行计算出 *n*=n_r;第 11 行 mg_1=g_1,得出初始 *PS*={(v_0,μ_0),(v_0,μ_1),(v_0,μ_2),(v_0,μ_3),(v_0,μ_4),(v_0,μ_5)},经 $PCond_{vertex}$ 后 *PS*={(v_0,μ_0),(v_0,μ_4)},经 $PCond_{self}$ 后 *PS* 不变;第 13 行得出 A_{IN}=∅;(2) 第 22 行调用 *SIonCOPE*(n_r,{(v_0,μ_0)},*TX*={g_1,g_2},*ANS*=∅).在此调用中,第 4 行得出 *n*=n_r;第 11 行 mg_1=g_1,得出初始 *PS*={(v_1,μ_1),(v_1,μ_5)},经 $PCond_{vertex}$ 和 $PCond_{self}$ 后 *PS* 不变;第 13 行得出 A_{IN}={μ_1,μ_5};(3) 第 22 行调用 *SIonCOPE*(n_r,{(v_0,μ_0),(v_1,μ_1)},*TX*={g_1,g_2},*ANS*=∅),在此调用中,第 4 行得出 *n*=n_1;第 11 行 mg_1=g_1,得出初始 *PS*={(v_2,μ_2),(v_2,μ_3),(v_2,μ_5)},经 $PCond_{vertex}$ 后 *PS*={(v_2,μ_2),(v_2,μ_3)},经 $PCond_{self}$ 后 *PS*={(v_2,μ_2)};第 13 行得出 A_{IN}={μ_0,μ_1,μ_2,μ_3,μ_5};(4) 第 22 行调用 *SIonCOPE*(n_r,{(v_0,μ_0),(v_1,μ_1),(v_2,μ_2)},*TX*={g_1,g_2},*ANS*=∅).在此调用中,第 4 行得出 *n*=n_2;第 7 行条件为真并执行第 8 行得出$\mathcal{Z}$={{g_1},{g_2}};然后,对于 TX_1={g_1},第 11 行 mg_1=g_1,得出初始 *PS*={(v_3,μ_3),(v_3,μ_4)},经 $PCond_{vertex}$ 后

$PS=\{(v_3,\mu_3)\}$,经 $PCond_{self}$ 后 PS 不变;第 13 行得出 $A_{IN}=\{\mu_0,\mu_1,\mu_2,\mu_3,\mu_4,\mu_5\}$;(5) 第 22 行调用 $SIonCOPE(n_r,\{(v_0,\mu_0),(v_1,\mu_1),(v_2,\mu_2),(v_3,\mu_3)\},TX=\{g_1\},ANS=\varnothing)$.在此调用中,第 4 行得出 $n=n_4$;第 11 行 $mg_1=g_1$,得出初始 $PS=\{(v_4,\mu_5)\}$,经 $PCond_{vertex}$ 和 $PCond_{self}$ 后 PS 不变;由于此时模型图中$\langle v_0,v_4\rangle$构成新的最右路径,故第 13 行紧缩一步 $A_{IN}(\mu_3)$,仍为 $\{\mu_0,\mu_1,\mu_2,\mu_3,\mu_4,\mu_5\}$;(6) 第 22 行 $SIonCOPE(n_r,\{(v_0,\mu_0),(v_1,\mu_1),(v_2,\mu_2),(v_3,\mu_3),(v_4,\mu_5)\},TX=\{g_1\},ANS=\varnothing)$调用.此时,第 4 行条件为真而执行第 5 行得到 $ANS=g_1$;第 6 行和第 23 行、第 15 行使算法回溯到(4)并从第 10 行开始考察 $TX_2=\{g_2\}$,然后第 11 行 $mg_1=g_2$,得出初始 $PS=\{(v_3,\mu_3),(v_3,\mu_4)\}$,经 $PCond_{vertex}$ 后 $PS=\{(v_3,\mu_4)\}$,经 $PCond_{self}$ 后 $PS=\varnothing$;第 12 行、第 10 行使此次递归结束并返回 $TX=\{g_2\},ANS=\{g_1\}$;然后算法回溯到(2)并从第 14 行开始考察 $p=(v_1,\mu_5)$;(7) 第 22 行调用 $SIonCOPE(n_r,\{(v_0,\mu_0),(v_1,\mu_5)\},TX=\{g_2\},ANS=\varnothing)$.在此调用中,第 4 行得出 $n=n_1$;第 11 行 $mg_1=g_2$,得出初始 $PS=\{(v_2,\mu_1),(v_2,\mu_4)\}$,经 $PCond_{vertex}$ 后 $PS=\varnothing$;第 12 行、第 10 行使此次递归结束并返回 $TX=\{g_2\}$,$ANS=\{g_1\}$;然后算法回溯到(1)并从第 14 行开始考察 $p=(v_0,\mu_4)$;(8) 第 22 行调用 $SIonCOPE(n_r,\{(v_0,\mu_4)\},TX=\{g_2\},ANS=\varnothing)$.在此调用中,第 4 行得出 $n=n_r$;第 11 行 $mg_1=g_2$,得出经 $PCond_{vertex}$ 后 $PS=\{(v_1,\mu_5)\}$,经 $PCond_{self}$ 后 PS 不变;第 13 行得出 $A_{IN}=\{\mu_2,\mu_5\}$;(9) 第 22 行调用 $SIonCOPE(n_r,\{(v_0,\mu_4),(v_1,\mu_5)\},TX=\{g_2\},ANS=\varnothing)$.在此调用中,第 4 行得出 $n=n_1$;第 11 行 $mg_1=g_2$,经 $PCond_{vertex}$ 后 $PS=\varnothing$;第 12 行、第 10 行使此次递归结束并返回 $TX=\{g_2\},ANS=\{g_1\}$;然后算法回溯至 $SIonCOPE$ 过程结束.最终算法返回 $ANS=\{g_1\}$.注意,由于此例比较简单,故未涉及 $PCond_{ahead}$ 过滤掉候选匹配顶点对的情况.

对 COPESI 的执行时间有如下分析:因为一到一的子图同构检测本身是一个 NP 完全问题,该结论可以自然推广到多个模型图的情况.同时,对于一个输入图,从一个模型图到它的子图同构检测的一个影响因素是模型图的顶点个数(因为子图同构关联着模型图的所有顶点),因此,这里对 COPESI 运行时间的分析主要是从顶点数不同的模型图的子图同构检测次数的角度进行.令 COPE 中模型图个数为 n,其中按组织结构共享相同前缀的模型图的集合分别为 $S_1,S_2,\ldots,S_k$($|S_i|$也表示为 $n(i)$),有 $\sum_{i=1,\ldots,k}n(i)=n$),所共享的前缀涉及的顶点数分别为 $cpv_1,cpv_2,\ldots,cpv_k$.后文给出 COPESI 运行时间的在线上界(on-line bound),即根据运行时情况得到的上界.对应于 $S_1=\{g_1,g_2,\ldots,g_{n(1)}\}$ 的 $n(1)$ 个模型图,令其在线执行时未由 $PCond_{ahead}$ 条件剪枝的最大深度分别为 $o\text{-}cpv_1(g_1),o\text{-}cpv_1(g_2),\ldots,o\text{-}cpv_1(g_{n(1)})$,若大于 cpv_1 则令该值为 cpv_1.对于 S_1 中的图,执行 COPESI 比逐个依次检测所节省的时间至少为 $\eta\times\Sigma_g T_{iso}(o\text{-}cpv_1(g))$,其中 $g\in\{g|g\in S_1,g\neq\mathrm{argmax}_{g'}\{o\text{-}cpv_1(g')\}\}$,$T_{iso}(v_num)$表示从顶点数为 v_num 的图到输入图的一到一检测的平均时间.η是判断 $PCond_{vertex}$ 和 $PCond_{self}$ 条件的时间总和与判断所有 3 种剪枝条件的时间总和的比值,其介于 0 到 1 之间.由于对 $PCond_{self}$ 和 $PCond_{ahead}$ 条件的判断时间相当,一个简单的估计η的方法是用一个常数近似,且其大于 0.5.因此,COPESI 运行时间的上界为

$$n\times T_{iso}-\sum_{i=1,\ldots,k}(\eta\times\Sigma_{g_i}T_{iso}(o\text{-}cpv_i(g_i)))\ ,$$

其中 $g_i\in\{g|g\in S_i,g\neq\mathrm{argmax}_{g'}\{o\text{-}cpv_i(g')\}\}$,即可以进一步表示为 $n\times T_{iso}-0.5\times\sum_{i=1,\ldots,k}(\Sigma_{g_i}T_{iso}(o\text{-}cpv_i(g_i)))$.其中,$T_{iso}$ 为从 COPE 所有模型图中的一个到输入图的一到一子图同构检测的平均时间.

2.3 索引特征生成

本文采用过滤-验证的机制来提高处理性能,这一节将给出索引特征生成方法.生成的特征子图集合采用 COPE 的形式进行组织,形成前缀树,记为 FTr.

图集中的频繁子图表征了图集的内在特征,且已被一些方法证实可作为良好的索引特征[9,10].对于从多到一的子图同构检测问题,后文将对每个频繁子图的过滤能力进行分析.给定两个频繁子图 g 和 g',满足 $g\subseteq g'$,如果 $|sup(g)|=|sup(g')|$,那么只需选择 g'作为特征而无须选择 g.因为较大的子图更不容易被输入图所包含.于是对于特征的选取来说,称 g'比 g 更显著,称 g 相对于 g'是不显著的(冗余的).对于子图 g 和一组子图 $SG=\{g_1,g_2,\ldots,g_k\}$,满足 $g\subseteq g_i,g\neq g_i$,其中 $i=1,2,\ldots,k$,如果 $\sup(g)=\cup_{i=1,2,\ldots,k}\sup(g_i)$,则 g 相对于 SG 是不显著的(冗余的).关于图 g 的显著度量δ定义为 $\delta=\dfrac{|\sup(g)|}{|\bigcup_{i=1,2,\ldots,k}\sup(f_i)|}$,其中 $f_1,f_2,\ldots,f_k$ 是所有为 g 超图的特征.为了得到显著的子图,可以设置一个最小显著水平阈值$\delta_{\min}$并保留所有显著水平不低于$\delta_{\min}$的子图.

特征生成算法包括两个步骤:(1) 频繁子图挖掘生成初始特征集;(2) 从挖掘得到的初始特征集中去除冗余子图,其以图大小(边数)降序逐层地计算每个子图的显著水平.同时,在第(1)步的频繁子图挖掘中所得到的一部分频繁子图间的包含关系被使用,这减少了在第(2)步中逐层去除冗余子图时子图同构检测数量.

在得到所有显著子图即特征集后,将其组织成第 2.1 节描述的 COPE 形式,形成前缀树 *FTr*.

因为挖掘对象均为给定模型图集,构造 COPE 的频繁 H 诱导子图挖掘和特征生成的频繁子图挖掘同时进行.在联合挖掘过程中,若一方剪枝或当前支持度小于最小阈值时,另一方按其自己方式继续进行挖掘搜索.

2.4 多到一的子图同构检测方法总结

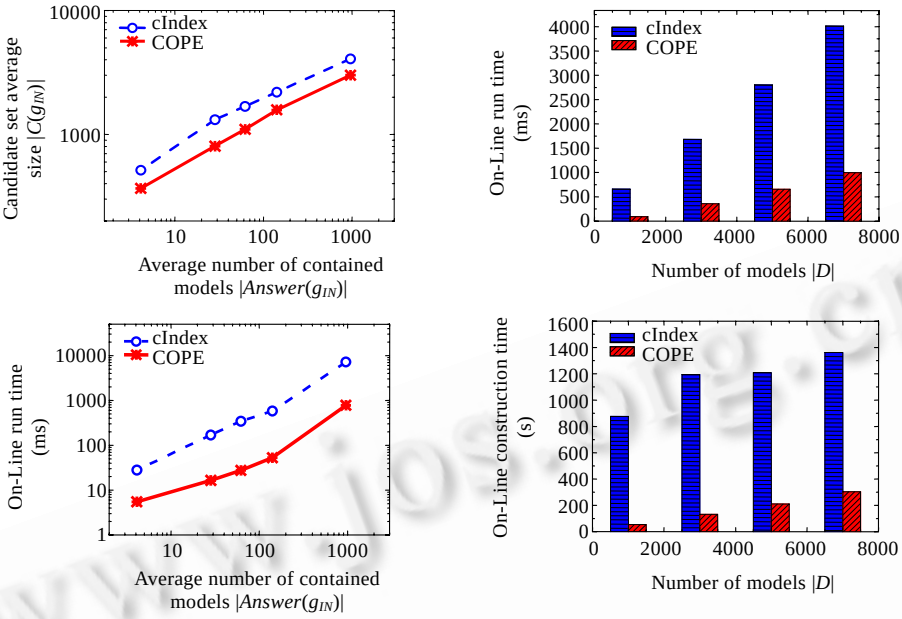
基于算法 COPESI,在 COPE 上投影出的任何模型图集合到输入图的子图包含关系均可循此检测算法得出.采用过滤-验证机制,多到一的子图同构检测包括两个步骤:(1) 以 *FTr* 和输入图 g_{IN} 为输入执行只寻找一个子图同构版本的算法 COPESI,得到所有特征子图 f 满足 $f\subseteq g_{IN}$,然后计算候选结果集 $C(g_{IN})=D-\cup_{f\subseteq g_{IN},f\in F}\sup_D(f)$,其中不记录候选图在 g_{IN} 中的子图同构;(2) 以 *DTr* 和 g_{IN} 为输入,结合候选集 $C(g_{IN})$来执行 COPESI 并得出最终结果.其中在 COPESI 执行过程中,待考察模型图列表中一旦发现非候选集中的图即从列表中删除并不予考察.

3 实验结果与分析

实验的运行环境为 PIV 3.0GHz CPU,2GB 内存,运行 RedHat Linux 8.0 操作系统的 PC 机.实验数据集采用 AIDS(AIDS antiviral screen dataset)的真实数据.实验主要考察提出方法的效率和可扩展性等,即与最新的方法 cIndex[7]比较检测的执行效率、索引特征的过滤能力、可扩展性及离线构造时间.其中,两种方法均采用对于每个被包含模型图只寻找一个子图同构的版本.cIndex 和本文方法(以下统称为 COPE)均用 C 语言实现,gcc 编译.

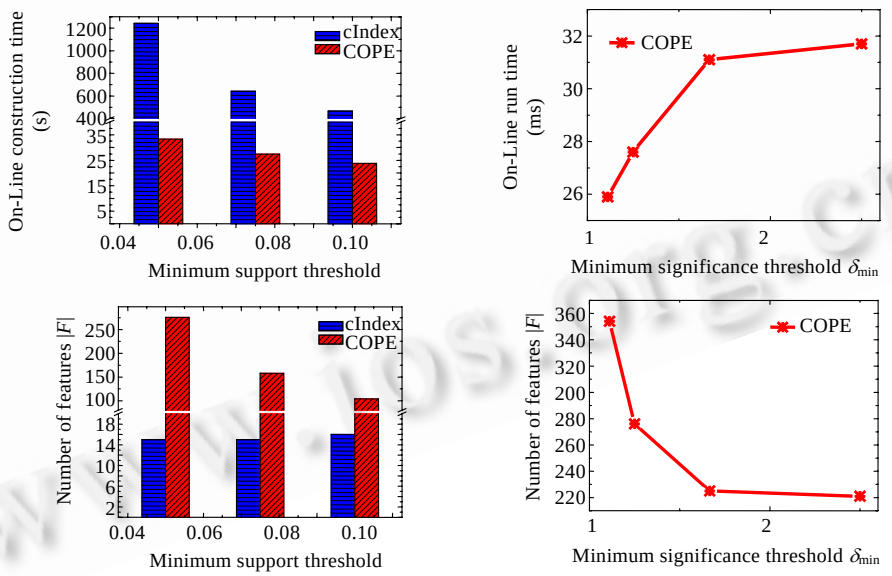
若无特别说明,实验中 cIndex 和 COPE 的各个参数设置如下.本文在选取模型图集和输入图集上采用与 cIndex 相同的方法.在多于 40 000 个化合物的 AIDS 数据集上进行频繁子图挖掘,将支持度介于 0.5%~10%之间的频繁子图集合记为 D_0.实验中,模型图集是从 D_0 中随机选取 10 000 个图构成,记为 D_{10000}.从 AIDS 中随机选取 10 000 个图形成数据集 *W*,在 *W* 中随机选取 2 000 个图作为输入图集 G_{IN}.对于 cIndex,其余 8 000 个图作为历史输入图集 *L*.获得对比子图(contrast subgraph)所用的最小支持度阈值 σ_c=0.05(此为与 0.1 和 0.01 比较而具有最小平均候选集大小的值).对于 cIndex-TopDown,叶节点内历史输入图最少数目 min_*size* 设置为 100.对于 COPE,最小显著水平阈值 $\delta_{\min}$=1.25;COPE 构造(*DTr*,*FTr*)及挖掘初始特征的最小支持度阈值 σ_D^I 和 σ_F 均为 0.05.在 cIndex 的 3 种方法即 cIndex-Basic,cIndex-BottomUp 和 cIndex-TopDown 中,选取 cIndex-Basic 用于离线构造的比较,因它产生最少数量的索引特征;选取 cIndex-TopDown 比较在线处理,因其在 3 种方法中执行效率最高.

图 4(a)给出了子图同构检测执行效率的实验结果.为了观察所提出的方法对具有不同结果集大小的输入图相应的执行效率,2 000 个测试输入图集按其结果集大小被划分为 5 部分:[0,20],[20,40],[40,100],[100,200],[200,∞].横坐标表示结果集的平均大小(被包含的模型图的平均数量).图 4(a)上图的纵坐标为平均候选集大小,我们看到,COPE 的候选集大小比 cIndex-TopDown 要小,表明 COPE 的索引特征的过滤能力更好(第 2.3 节).图 4(a)下图的纵坐标表示每个输入图作用在相应候选集上的平均响应时间(毫秒),相比之下,COPE 的响应时间小得多,效率提升约 1 个数量级.提升的原因是不仅候选图更少,而且模型图集的压缩组织能够减小在线计算代价(第 2.1 节、第 2.2 节).



(a) Candidate set size and on-line run time
(a) 候选集大小和执行时间

(b) Scalability
(b) 可扩展性



(c) Off-Line construction performance
(c) 离线构造性能

(d) Sensibility of $\delta_{\min}$
(d) $\delta_{\min}$ 敏感性

Fig.4 Processing performance, scalability, off-line construction performance, and sensibility of $\delta_{\min}$

图 4 处理性能、可扩展性、离线构造性能和最小显著水平阈值$\delta_{\min}$敏感性

图 4(b)给出了可扩展性实验结果.所选用的 4 个模型图集分别是从 D_0 中随机选取 10 000,30 000,50 000 和 70 000 个图所得.横坐标表示模型图集的大小,图 4(b)上图的纵坐标表示 G_{IN} 中每个输入图的平均执行时间(ms),图 4(b)下图的纵坐标为离线构造时间(s).通过这两个图我们可以看到,COPE 具有良好的可扩展性.

图 4(c)展示了离线构造性能的实验结果.比较 cIndex-Basic 和 COPE 方法在 0.05,0.075 及 0.10 的最小支持

度阈值下的 D_{10000} 的离线构造时间和特征数目.需要注意的是,cIndex-Basic 关联的最小支持度阈值用于挖掘初始对比子图;在本文方法中,构建 COPE 的诱导子图挖掘和特征生成的频繁子图挖掘的最小支持度阈值相等,等于该实验中指定的最小支持度阈值.通过实验结果我们可以看到,不但检测执行效率更高,提出方法的预处理阶段的离线构造时间也短得多,尽管特征数目多于 cIndex 的方法.原因是因为 cIndex 在构造对比子图时,在频繁模式挖掘得到初始子图模式集合后,需要得到从初始子图集合的每个子图到历史输入图集的每个输入图之间的子图包含关系.这导致了大量的子图同构检测.同时,*FTr* 的引入大大减少了在线处理中的过滤时间.

图 4(d)考察最小显著水平阈值 $\delta_{\min}$ 的变化对离线构造和在线检测的影响.其中,测试输入图集选用[40,100].在这两个图中,横坐标表示 $\delta_{\min}$ 取值,为 1.1111, 1.2500, 1.6667 和 2.5000.图 4(d)上图的纵坐标表示每个输入图平均执行时间(ms),图 4(d)下图的纵坐标为索引特征数量.通过该实验可以看出,随着 $\delta_{\min}$ 的增大,特征数量逐渐减小,执行时间逐渐变长.在实际应用中,我们需要在检测执行时间和索引大小上做出折衷.

4 结 论

为了高效地执行从预先给定的多个模型图到一个在线提交的输入图的子图同构检测,本文提出了图集合压缩组织方法、多到一的子图同构检测算法和索引特征生成方法.在所提出的方法中,多个图的互相同构的特定诱导子图被合并在一起,使得在执行子图同构检测时,合并的子图以相当于在一个图上执行检测算法的方式被考察,从而避免了考察大量重复状态的时间消耗.分析结果说明,所提出方法的计算代价比现有方法小得多.实验结果显示,所提出方法的在线执行时间比现有方法快约一个数量级,索引构造时间也缩短很多.

References:

[1] Conte D, Foggia P, Sansone C, Vento M. Thirty years of graph matching in pattern recognition. Int'l Journal of Pattern Recognition and Artificial Intelligence, 2004,18(3):265−298.

[2] Cordella LP, Foggia P, Sansone C, Vento M. A (sub)graph isomorphism algorithm for matching large graphs. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2004,26(10):1367−1372.

[3] Ullmann JR. An algorithm for subgraph isomorphism. Journal of the ACM, 1976,23(1):31−42.

[4] Sengupta K, Boyer KL. Organizing large structural modelbases. IEEE Trans. on Pattern Analysis and Machine Intelligence, 1995, 17(4):321−332.

[5] Messmer BT, Bunke H. A decision tree approach to graph and subgraph isomorphism detection. Pattern Recognition, 1999,32(12): 1979−1998.

[6] Messmer BT, Bunke H. Efficient subgraph isomorphism detection: A decomposition approach. IEEE Trans. on Knowledge and Data Engineering, 2000,12(2):307−323.

[7] Chen C, Yan X, Yu PS, Han J, Zhang DQ, Gu X. Towards graph containment search and indexing. In: Koch C, Gehrke J, Garofalakis MN, Srivastava D, Aberer K, Deshpande A, Florescu D, Chan CY, Ganti V, Kanne CC, Klas W, Neuhold EJ, eds. Proc. of the Int'l Conf. on Very Large Data Bases. Vienna: ACM, 2007. 926−937.

[8] Zhou SG, Yu ZC, Jiang HL. Concepts, issues, and advances of searching in graph-structured data. Communications of the China Computer Federation, 2007,3(8):59−65 (in Chinese with English abstract).

[9] Yan X, Yu PS, Han J. Graph indexing: A frequent structure-based approach. In: Weikum G, Konig AC, Dessloch S, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Paris: ACM, 2004. 335−346.

[10] Cheng J, Ke Y, Ng W, Lu A. FG-Index: Towards verification-free query processing on graph databases. In: Chan CY, Ooi BC, Zhou A, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Beijing: ACM, 2007. 857−872.

[11] Washio T, Motoda H. State of the art of graph-based data mining. SIGKDD Explorations, 2003,5(1):59−68.

[12] Yan X, Han J. gSpan: Graph-Based substructure pattern mining. In: Agrawal R, Dittrich K, Ngu AHH, eds. Proc. of the IEEE Int'l Conf. on Data Mining. Maebashi City: IEEE, 2002. 721−724.

[13] Wang W, Zhou HF, Yuan QQ, Lou YB, Shi BL. Mining frequent patterns based on graph theory. Journal of Computer Research and Development, 2005,42(2):230−235 (in Chinese with English abstract).

[14] Li XT, Li JZ, Gao H. An efficient frequent subgraph mining algorithm. Journal of Software, 2007,18(10):2469−2480 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/2469.htm

[15] Liu Y, Li J, Gao H. Summarizing graph patterns. In: Bernstein P, Bertino E, Dayal U, Lockemann P, Weikum G, Whang KY, eds. Proc. of the IEEE Int'l Conf. on Data Mining. Cancun: IEEE, 2008. 903−912.

[16] Fürer M, Raghavachari B. Approximating the minimum degree spanning tree to within one from the optimal degree. In: Smithies M, ed. Proc. of the ACM/SIGACT-SIAM Symp. on Discrete Algorithms. Orlando: SIAM, 1992. 317−324.

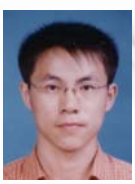
[17] Tatarinov I, Viglas S, Beyer KS, Shanmugasundaram J, Shekita EJ, Zhang C. Storing and querying ordered XML using a relational database system. In: Franklin MJ, Moon B, Ailamaki A, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Madison: ACM, 2002. 204−215.

附中文参考文献:

[8] 周水庚,蔚赵春,蒋豪良.图结构数据搜索的概念、问题与进展.中国计算机学会通讯,2007,3(8):59−65.

[13] 汪卫,周皓峰,袁晴晴,楼宇波,施伯乐.基于图论的频繁模式挖掘.计算机研究与发展,2005,42(2):230−235.

[14] 李先通,李建中,高宏.一种高效频繁子图挖掘算法.软件学报,2007,18(10):2469−2480. http://www.jos.org.cn/1000-9825/18/2469.htm



张硕(1982－),男,黑龙江肇东人,博士生,主要研究领域为图数据管理.

高宏(1966－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为并行数据库,数据仓库.

李建中(1950－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为并行数据库,传感器网络.

邹兆年(1979－),男,博士生,主要研究领域为图数据挖掘.