高技术通讯 2010 年 第 20 卷 第 3 期:270 ~ 278

doi:10.3772/j.issn.1002-0470.2010.03.009

节点与决策模式两段式映射的子图查询算法①

李先通② 李建中③

(哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001)

摘 要 针对大规模图集的子图查询问题,给出了一种基于节点与决策模式映射(NDFM)的索引结构——NDFM-Index,并在此索引结构的基础上提出了一种图集的子图查询算法。NDFM-Index 利用图中关键节点所携带的结构信息以及邻居的标号分布,与决策模式形成映射,从而不通过枚举直接得到查询图所包含的索引模式,得到更小的候选集。理论与实验的分析结果表明,该算法不但能避免索引筛选过程中对查询图子图的枚举过程,而且能显著地减小候选集尺寸,进而大大降低查询图与候选集之间的子图同构测试次数,提高查询效率。

关键词 图数据集,图查询,频繁模式,决策模式,节点向量

0 引 言

图是一种通用的数据结构,它能有效地描述结构化和半结构化数据,可应用于如社会或信息网络、生物信息学、蛋白质网络、模式识别中 2D/3D 物体、有线或无线网络等许多领域。图中的节点常用于描述现实世界中的实体,边则用于描述实体之间的特征关系。例如,在计算机视觉领域,图用于描述复杂关系,如图像中的实体组织结构;在化学信息学与生物信息学中,图被用于表示化合物和蛋白质。为准确描述数据信息,常采用一种特殊的图——标号图对它们进行描述。此时,图中的节点与边分别与某属性值相关联。然而,由于近来结构化与半结构化数据的迅速累积,有效支持大规模图数据集的图查询算法已逐渐成为数据库领域中的一个具有挑战性的研究方向。本文提出了一种基于节点-决策模式映射(nodeto decision feature mapping,NDFM)的索引结构——NDFM-Index 的子图查询算法,并对其进行了详尽的论述。

1 研究背景

给定一个图数据集 GD 和一个查询图 q,图查询的目的是找到图集中 q 的所有超图,即 $Q=\{g \mid g\in GD \wedge q\subseteq g\}$。例如,在化合物中,某些子结构将直接决定化合物的特殊性质,图查询可迅速定位所有具有相似属性的分子。而且,图查询问题在生物识别技术上也有广泛应用。

目前,在图查询问题上研究人员已提出了一些高效的查询算法,其中,文献[1-4]的算法,以及基于文献[1]建立的近似查询算法[5,6],均采用图挖掘算法生成的频繁图模式(如路径、树或子图等)构建索引。而文献[7-11]则采用与此不同的构建模块,如图闭包、有向无环图分解、图集特殊构件等。同时,在一些专用领域内,还存在一些效率更高的专用算法,如应用于 XML 查询中的方法[12,13]等。

包括精确匹配和近似匹配等已经发表的算法,在查询过程中均采用"过滤与验证"机制。该机制的处理过程是通过不同的过滤手段尽可能多地将与查询图 q 无关的图从候选集中剔除,从而使验证阶段需要计算的子图同构次数尽可能多地降低,达到提高算法效率的目的。过滤阶段采用的规则为"包含逻辑(inclusion logic)",即给定标号图 g_1 与 g_2 以及 g_1 的一个子图 g'($g'\subseteq g_1$),若 g_1 是 g_2 的子图,则 g'必为 g_2 的子图($(g_1\subseteq g_2)\Rightarrow(g'\subseteq g_2)$)。反之,若 g'不是 g_2 的子图,则 g_1 也不可能是 g_2 的子图($(g'\not\subseteq g_2)\Rightarrow(g_1\not\subseteq g_2)$),$g_2$ 可直接从候选集中删除。目前已经发表的图查询算法,其过滤手段多建

① 国家自然科学基金(60773063,60473075,60533110,60803036)和 973 计划(2006CB303000)资助项目。
② 男,1973 年生,博士;研究方向:数据库,数据挖掘,图挖掘,图查询;E-mail: lxt@hit.edu.cn
③ 通讯作者,E-mail: lijzh@hit.edu.cn
(收稿日期:2009-04-28)

立于该包含逻辑的基础上。

在图查询算法中,时间复杂性最高的两个部分是对查询图进行分解以得到其包含的索引模式和验证阶段的子图同构测试。前者通常是通过枚举加子图同构的方式进行,即通过枚举得到 q 包含的所有子图,并通过子图同构的方式计算最终结果,效率低下;后者是 NP-完全问题[14]。因此,算法的效率直接被这两个子问题的效率制约,如何降低验证阶段的子图同构次数,或如何通过更高效的手段得到查询图包含的频繁模式,是图查询研究工作中的重要内容。

本文提出了一种基于节点-决策模式映射的索引结构 NDFM-Index,用于解决上述问题。决策模式为频繁模式挖掘过程中发生支持度阶跃的频繁模式。NDFM-Index 利用图中关键节点所携带的结构信息,以及邻居的标号分布,与决策模式形成映射,从而不通过枚举直接得到 q 所包含的索引模式,得到更小的候选集。

2 基本概念

定义 2-1(标号图) 一个标号图是一个五元组 $G=\{V,E,\sum E,\sum V,L\}$。其中,$V$ 代表图中节点的集合,$E\subseteq V\times V$ 代表图中边的集合。$\sum V$, $\sum E$ 分别代表节点标号的集合与边标号的集合。L 是标号函数,用于完成标号向节点和边的映射:$V\rightarrow\sum V$ 与 $E\rightarrow\sum E$。

定义 2-2(图的同构) 图的同构是一个双射 f: $V(G)\leftrightarrow V(G')$。对于图 $G=\{V,E,\sum V,\sum E,L\}$ 与图 $G'=\{V',E',\sum V',\sum E',L'\}$,若它们是同构的,则满足如下条件:

$\forall u\in V,L(u)=L'(f(u))$

$\forall u,v\in V,((u,v)\in E)\Leftrightarrow((f(u),f(v))\in E')$,且

$\forall(u,v)\in E,L(u,v)=L'(f(u),f(v))$。

若图 G 与图 G' 是同构的,记为 $G\equiv G'$。

定义 2-3(子图同构) 给定标号图 G 与 G',若 G' 中存在子图 G'' 满足 $G''\equiv G$,则称 G 与 G' 是子图同构的,记为 $G\subseteq G'$。此时,称 G 是 G' 的子图,记为 $G\subseteq G'$。称 G' 是 G 的超图,记为 $G'\supseteq G$。

子图同构问题已经被证明是 NP 完全问题[14]。然而,在图查询过程中,子图同构是无法避免的计算开销。因此,在图查询过程中,如何尽量减小子图同构测试次数,对查询算法的效率是至关重要的。

定义 2-4(图查询) 给定图集 GD,并给定查询图 q,图查询返回的结果集为 Q,其中元素为图集中 q 的超图:$Q=\{g\mid g\in GD\wedge q\subseteq g\}$。

在本文中,除特殊标记外,图的尺寸指图的边数。给定标号图 G,其尺寸表示为 $|G|$。本文提出的算法 NDFM 应用于无向标号图集合。然而,NDFM 可通过简单调整应用于有向图集合。为避免混淆,图集中的图被称为目标图。

3 图查询处理

根据包含逻辑,过滤-验证方法对图查询的处理为:

(1) 计算机索引模式中被查询图 q 包含的模式;

(2) 返回这些索引模式的支持集;

(3) 通过对这些支持集取交集得到候选集 C_q;

(4) 通过查询图 q 逐个与 C_q 中目标图进行子图同构测试进行验证以得到最终结果。

NDFM-Index 的数据结构如图 1 所示。NDFM-Index 可被分解为两部分,分别为 NDFM-Node 和 NDFM-Feature。在 NDFM-Node 中,包括两类向量。一类是 NA (node array),其中包含的信息为节点标号,节点的度,邻居的关系等。具有特定值的 NA 指向一个 CA (composed array)。CA 是另一类向量,CA 包含的信息为该节点的邻居标号分布,是一个 $|\Sigma V|$ 字节的向量。而 NDFM-Feature 由 DF (decision feature)组成,每组(NA, CA)指向一组 DF,每个 DF 包含一组决策模式的 ID,通过这些模式 ID 值与其支持集相关联。图 1 中大写字母如'VA'等表示向量集合,而小写字母如'va'等表示具体向量。

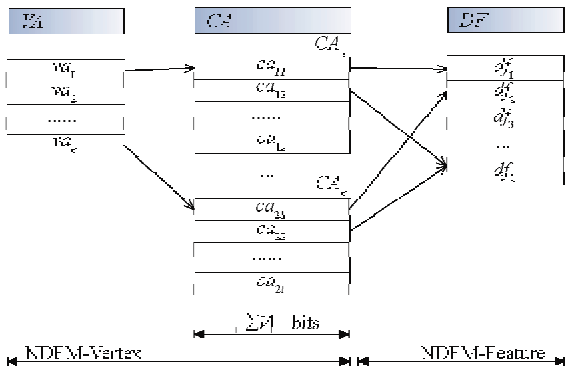


图 1 数据结构图

算法 NDFM 的整体结构如下：

(1) 离线构建索引。根据 GD 构建 NA，CA 和 DF 及互相之间的映射关系。同时记录每个频繁模式的支持集 $D_f = \{g \mid f \subseteq g \wedge g \in GD\}$。

(2) 通过查询图 q 得到其包含的关键节点，并通过对节点信息的匹配，得到一组组成向量 CA，并最终得到包含这些关键节点的决策模式。

(3) 通过包含逻辑进行过滤。对得到的决策模式，取其支持集的交集，即：$C_q = \bigcap_f D_f (f \subseteq g \wedge g \in GD)$，其中 C_q 为候选集，f 为决策模式。

(4) 验证。检验候选集中每一个标号图，以得到正确结果。

以下各小节将分别讨论节点、决策模式等构建的索引及查询算法的实现。

3.1 NDFM-Index

图查询建立索引的目的是为了尽可能多地过滤掉不正确的结果，以使候选集尺寸最小化，最终在验证阶段可通过更少的子图同构次数来得到正确结果，提高算法的整体效率。

NDFM-Index 可分为 NDFM-Node 和 NDFM-Feature 两部分，其中 NDFM-Node 由图集中关键节点组成，它们记录了这些节点的标号、邻居关系等结构信息，NDFM-Feature 则构建于决策模式之上，用于生成候选集。

3.1.1 NDFM-Node

节点是图的基本组成单元。通过节点构建索引，最简单的办法是直接使节点指向包含它的目标图。不过，这种办法下节点携带的结构信息少，对图集的过滤能力有限，因此，这种简便的方法却并不适用于过滤-验证模式。然而，利用节点构建的索引可以有效控制索引的规模，因为该索引的尺寸与 $|\sum V|$ 呈线性关系。

NDFM-Node 用一个三元组 NA：(L, D, NC) 描述图集中关键节点信息。其中，L 为该节点的标号，D 表示该节点的度(或者邻居个数)，NC(neighbor condition)则表示该节点的邻居状态。此时，称 NA 为节点向量(node array)。所谓关键节点，是指该节点在标号图中的角色或位置重要于其它节点，如割点等。此处，我们采用节点的度来表示节点的重要程度，节点的度越大，其重要程度越高，为关键节点。反之，度最低或次低的节点，为非重要节点，不是关键节点。对图中节点的关键程度，有很多表示方法，通过度表示其关键程度只是表示方法之一，还可以采用如 Closeness①，Betweeness②，以及特征向量等。NC 在此处表示邻居之间的连接数，也可通过边的列表进行描述。

例如，图 2 所示为一个标号图 G，此处为方便描述，省略其边标号并仅保留部分节点标号。图中 a，b 表示节点标号，而数字表示节点编号。如果用节点向量 NA 表示这个图中的节点 a 和节点 b，则其分别为 $T_a = (a, 4, 3)$，$T_b = (b, 4, 2)$。

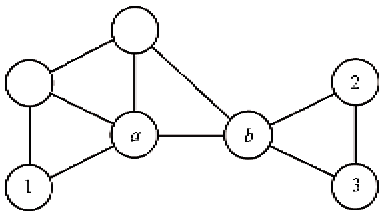


图 2 标号图 G

节点向量之间存在一种线性顺序关系。

定义 3-1(节点向量的顺序关系) 给定标号图中的节点 a，b，它们的节点向量关系如下：

(1) $NA_a = NA_b$，当且仅当 $(L_a = L_b) \wedge (D_a = D_b) \wedge (NC_a = NC_b)$。

(2) $NA_a \prec NA_b$，当且仅当

a) $(L_a = L_b) \wedge (D_a < D_b) \wedge (NC_a < NC_b)$或

b) $(L_a = L_b) \wedge (D_a = D_b) \wedge (NC_a < NC_b)$或

c) $(L_a = L_b) \wedge (D_a < D_b) \wedge (NC_a = NC_b)$。

(3) $NA_a \succ NA_b$，其它情况。

图查询过程中，在给定查询图 q 时，通常需要通过枚举加子图同构的方式得到 q 包含的频繁模式，即先通过枚举得到 q 的所有子图，并通过 q 的子图与频繁模式之间进行子图同构，最终得到 q 包含的频繁模式。在 NDFM-Node 中，对频繁模式的过滤是通过节点向量完成的，这时得到的频繁模式集合 $F_q = \{f \mid f \in F \wedge (NA_q = NA_f \vee NA_q \prec NA_f)\}$ 远小于索引中的频繁模式，从而有效地避免了枚举和大部分子图同构的计算，降低了该过程的时间复杂性。

每一个节点向量唯一确定一组属性相同的节点。如在图 2 中，设节点 v_1，v_2 和 v_3 的标号均为 c，则它们节点向量的值也相同，均为 $(c,2,1)$。然而，如果追加邻居节点的标号信息至节点向量，则可将

① 利用节点与其它节点之间的距离(最小距离)计算节点所在位置的紧密度，紧密度越高则重要程度越大。

② 用必须通过该节点的路径数来衡量该节点的重要程度，数值越大则重要程度越高。

v_1 与 v_2,v_3 区分开来,进一步增强 NDFM-Node 的过滤能力。因此,引入节点的组成向量 *CA* (composed array)来对这些节点进一步划分。

给定图 *G* 和 *G* 中的关键节点 v, v 通过节点向量 na_v 指向一组组成向量。每组 *CA* 由 $|\sum V|$ 位组成。每一位表示 v 的邻居中是否存在具有该标号的节点。然而,当 $|\sum V|$ 的值较大时,会导致 NDFM-Node 占用的空间较大。此时,可采用文献[15]中的方法,用 S_{bit} 来限定该向量的位数,而 S_{bit} 是用户可以控制的。在图 2 中,节点 v_1,v_2 和 v_3 的节点向量为(c,2,1),而它指向的组成向量则分别代表不同邻居标号分布值。

综上所述,NDFM-Node 由 *NA* 和 *CA* 两类向量组成。其中 *NA* 为节点的基本信息三元组,*CA* 则为由 *NA* 的值决定的节点邻居标号分布。

3.1.2 NDFM-Feature

定义 3-2〔**支持度**〕 图 g 关于图集 *GD* 的支持度为 *GD* 中 g 的超图所占比例值,即

$$support(g, GD) = \frac{|\{g_i \mid g_i \in GD \wedge g \subseteq g_i\}|}{|GD|}$$

在支持度定义的基础上,可直接给出频繁模式的定义。

定义 3-3〔**频繁模式**〕 给定一个图数据集 $GD = \{g_1, g_2, \cdots, g_n\}$,如果一个图模式 g 关于 *GD* 是频繁的,当且仅当它的支持度不小于一个给定的阈值 min_sup,即 $support(g, GD) \geqslant min\_sup$。

频繁模式 g 的支持集是 g 在 *GD* 中所有超图的集合,记为 $D_g = \{g_i \mid g_i(GD \wedge g \subseteq g_i\}$。我们也可通过支持集来表示支持度 $support(g, GD) = |D_g| / |GD|$。

频繁模式揭示了图集内在特征。假设图集中所有频率大于 min_sup 的频繁模式组成该图集的索引,那么,给定查询图 q,如果 q 是频繁的,则包含 q 的目标图可通过索引直接从图集中得到。如果 q 不是频繁的,那么,由于频繁子图的定义,q 包含某频繁模式 f 的概率很高。由于包含 q 的目标图肯定包含 q 的所有子图,D_f 则为潜在的候选集。因此,通过频繁模式建立索引是非常必要的。

在这里,不妨进一步分析 q 本身不是频繁子图的情况。如果我们将 q 包含的所有模式按其支持度的降序排列,则得到如下模式列表: f_1, f_2, …, f_n。在这个序列中,必然存在一个阈值 i,即 $support(f_i, GD) \geqslant min\_sup$,而 $support(f_{i+1}, GD) < min\_sup$。由于所有频繁模式都加入到索引中,因此,包含 $f_k(1 \leqslant k \leqslant i)$的目标图是已知的。查询的候选集 C_q 来自这些频繁模式支持集的交集 $\bigcap_{1 \leqslant k \leqslant i} D_{f_k}$,并且,$|C_q| \leqslant support(f_i, GD)$。对大多数查询, $support(f_i, GD) \approx min\_sup$。而且,取交集的操作会生成更小的 C_q。

频繁模式来自于图挖掘算法。在挖掘过程中,通常采用效率较高的先深方法,即按先深的原则选择一条频繁边并对其逐步进行扩展,每次扩展一条边,递次生成频繁模式,直到扩展结果不再频繁为止,算法返回上一级频繁模式,继续查找可扩展的其它边。图 3 所示为频繁模式的挖掘空间,每个节点为一个频繁子图模式。从图中可以看出,尺寸越小的图距离根 root 的距离越小,图的边数为所在节点至 root 的距离。

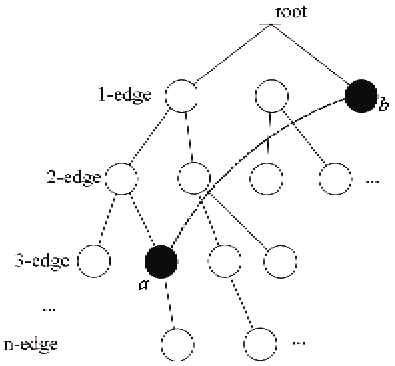


图 3 频繁模式挖掘空间

然而,频繁模式集中存在大量的冗余。例如:设图集的频繁模式集中存在如下两个路径: $p_1 = v_1 - v_2 - v_3$ 和 $p_2 = v_1 - v_2 - v_3 - v_4$。当 $D_{p1} = D_{p2}$ 时,p_1 与 p_2 是互为冗余的,因此,索引中仅需包含 p_1 或 p_2 之一即可。此时,通常保留尺寸较小的频繁模式,因为尺寸越小的频繁模式被查询图 q 包含的概率越大。显然,冗余的存在使频繁模式构成的索引庞杂低效。然而,在实际应用中,仅仅消除支持集相等的模式还不够,这时,往往给定一个选择比 δ,当某模式与其子模式之间支持度的比值不大于该选择比时,频繁模式为冗余的。然而,选择比的缺点在于,计算过程中,往往需要考查该模式所有子模式,这个代价是高昂的。如果查询提出的图集为非稳定状态,则需要反复重构索引,此时选择比的计算代价是不能容忍的。显然,索引的构建代价越低,其应用范围越广泛,可扩展性越高。因此,本文给出决策模式的概念,取代选择比,用于迅速消除冗余。

挖掘过程中,会出现频繁模式之间的支持度阶

跃。

定义 3-4（支持度阶跃） 给定图集 GD、支持度阈值 min_sup 和决策因子 σ，并通过先深方法挖掘 GD 中的频繁子图。频繁模式 f' 是通过对频繁模式 f 扩展一条频繁边得到的（在挖掘空间中是父子关系）。此时，若 $support(f, GD)/support(f', GD) \geqslant \sigma$，则称在挖掘空间节点 f 处发生支持度阶跃。

由定义可知，$support(f, GD) \geqslant support(f', GD)$，因此，$\sigma \geqslant 1$。否则，$\sigma$ 值无意义。

支持度阶跃的意义为，频繁模式 f 与 f' 仅相差一条边，但支持度却发生了明显的变化。例如，化合物集合中，苯环的出现频繁非常高。设苯环由边 $e_1, e_2, \cdots, e_6$ 组成。在挖掘过程中，由单边 e_1 向 e_6 的扩展中，支持度基本没有变化。然而，在对 $e_1, e_2, \cdots, e_6$ 进一步扩展时，支持度的变化却非常明显，即出现了支持度阶跃。

定义 3-5（决策模式） 称出现支持度阶跃的模式为决策模式（decision feature，DF）。

显然，决策模式的数量由决策因子 σ 决定。

在挖掘空间中，设 $r, f_1, f_2, \cdots, f_n$ 为一条由根至叶的路径，即 f_i 为包含 i 条边的频繁模式。设该路径上的决策模式按尺寸由小到大的顺序为 $f_{1'}, f_{2'}, \cdots, f_{k'}$。由决策模式的定义得出，$support(f_{i'}, GD) > support(f_t, GD) \geqslant support(f_{(i+1)'}, GD)$。其中，$1 \leqslant t < n$。如果给定一个合适的 σ，使 $support(f_t, GD) \approx support(f_{(i+1)'}, GD) \cdot \sigma$，则采用决策模式为索引，能最大限度地消除频繁模式间存在的冗余。

图集 GD 的挖掘空间中，虽然采用先深方法的目的是尽量避免类似 Apriori 方法的重复工作，但整个过程依然存在重复挖掘的现象。如图 3 所示，节点 a 与节点 b 分别代表挖掘过程中出现的两个频繁子图。由于是先深顺序，因此，存在 $g_b \subseteq g_a$ 的情况。当此情况发生时，若 g_b 与 g_a 之间的支持度比值满足决策因子的限定条件，则定义 g_b 为决策模式，无论 g_b 与它的儿子之间的支持度比例关系。

在图 1 中，NDFM-Feature 部分（DF）与 NDFM-Node 部分的映射为由 NA 与 CA 共同制约的关键节点向包含该节点的决策模式列的映射，即每组 DF 为一个决策模式的列表，这组 DF 不但包含由 NA 决定的节点属性，同时，这些节点也满足由 CA 决定的节点邻居标号分布。

3.1.3 索引构建算法

NDFM-Index 由 NDFM-Node 和 NDFM-Feature 两部分组成。需要指出的，NDFM-Feature 由决策模式 DF 和图集中所有单边模式组成。单边模式的加入目的，是为了最大化地保证查询结果的完整性。

NDFM-Node 和 NDFM-Feature 共同组成查询算法的索引部分，其构建算法在算法 1 中给出。

算法 NDFM-Index 可分为三部分。

第一部分（1-4 行）用于生成 GD 中的频繁模式，形成先深搜索空间（DFS）树。同时，所有单边模式 e 被加入到索引模式集 F 中（第 3 行）。由于索引的构建不是在线进行的，因此，频繁模式挖掘算法的选择比较自由。

算法 1 NDFM-Index

输入：图集 GD，决策因子 σ
输出：NDFM-Index

```
1.  F ← Φ
2.  for GD 中所有单边 do
3.      F ← e
4.  挖掘频繁子图并形成 DFS 树
5.  for DFS 树中所有节点 do
6.      if support(f_k, GD) / support(f_{k-1}, GD) ⩾ σ
7.          F ← F ∪ f_k
8.      if N_i 中存在 f_k 的超图
9.          if support(f_k, GD) / support(f̂_k, GD) ⩾ σ
10.             F ← F ∪ f_k
11. for GD 中所有关键节点 v do
12.     n_i ← na_v
13.     n_i 升序排列
14.     for each n_i do
15.         c_j ← ca_i
16.         mapping(n_i, c_j)
17.         for each c_j do
18.             F_k ← {f | f ∈ F ∧ n_i c_j) ⊆ f}
19.             F_k 按尺寸降序排列
20.             mapping((n_i, c_j), F_k)
21. return n, c, F
```

第二部分（4-10 行）与第一部分是同时进行的。在查询过程中，通过决策因子 σ 计算决策模式，并将决策模式加入到集合 F 中去。在这部分算法中，N_i 表示算法已经遍历的分布式文件系统（DFS）树节点，f_k，$\hat{f}_k$ 与 $f_{(k-1)}$ 分别表示 DFS 树中的频繁模式，其中 f_k 表示当前待考察的节点，$\hat{f}_k$ 表示已考察过并保存在 N_i 中的节点，而 $f_{(k-1)}$ 表示 f_k 的父亲节点。

第三部分（11-20 行）为映射的形成部分。在第

11行找到所有 GD 中的关键节点,并通过关键节点计算其 NA,并形成 NA 与 CA 的映射。最后,算法在17-20行,形成 NA,CA 向 DF 的映射,即通过映射将模式集中选定的模式 f 与 NA,CA 关联起来。其中,na_v 表示节点 v 的 NA 向量值,ca_i 表示 NA 向量中 n_i 决定的 CA 值,F_k 表示所有被12-16行给定的 NA,CA 制约的 DF。

3.2 查询算法

查询算法在算法2中给出。

算法在第一行将整个图集 GD 看作候选集,并通过 q 包含的频繁模式支持集逐个与当前候选集取交集,最终得到真正的候选集。

算法2 NDFM-Query

输入:图集 GD,$NDFM\text{-}Index$,查询图 q
输出:候选集 C_q

1. $C_q \leftarrow GD$
2. $F_q \leftarrow \Phi$
3. $na \leftarrow q$ 中所有关键节点
4. for na 中的每组 na_i do
5. $F_i \leftarrow \text{mapping}(\text{mapping}(na_i, c_i), na_i)$
6. $F_q \leftarrow F_q \cup Fi$
7. for F_q 中每个决策模式 do
8. if $f_i \not\subseteq q$ do
9. $F_q \leftarrow F_q - f_i$
10. for F_q 中所有 f do
11. $C_q \leftarrow C_q \cap D_f$
12. return C_q

算法在开始时,先得到查询图 q 中所有关键节点的节点向量 na_i,通过映射得到与之相关的组成向量集合,并通过 NA,CA 与 DF 间的映射关系,得到与这些关键节点相关的决策模式(第5行)。在第7-9行的循环中,进一步验证得到的决策模式被查询图 q 包含。最终,算法通过第10-11行得到最终的候选集并输出。

4 算法效率分析

NDFM的查询过程,时间消耗分布为:$T_{\text{query}} = T_{\text{filter\_nodes}} + T_{\text{filter\_features}} + T_{\text{isoCand}} \times | C_q |$。其中,$T_{\text{query}}$ 为查询需要的总体时间,$T_{\text{filter\_nodes}}$ 为通过关键节点过滤时需要的时间消耗,$T_{\text{filter\_features}}$ 为通过频繁模式过滤需要的时间消耗,T_{isoCand} 为子图同构需要的时间消耗,而 $| C_q |$ 为候选集大小。$T_{\text{filter\_nodes}}$ 为对节点向量,组成向量的计算开销,由于节点的计算可在多项式时间内完成,因此,相比于其它通过枚举得到 q 的所有子图的算法,显然效率更高。同时,通过 NA 与 CA 得到的模式集规模更小,远小于由 q 的枚举得到的子图集,因此,在得到被 q 包含的索引模式集的过程,需要的子图同构次数少得多,即 $T_{\text{filter\_features}}$ 的值远小于其它算法。

另外,查询过程另一个时间开销大的部分为 $T_{\text{isoCand}} \times | C_q |$,而决定其开销大小的关键因素为候选集尺寸。实验结果表明,通过 NA、CA 和 DF 的合作,得到的候选集规模也小于其它算法。

5 实验分析

本节通过实验来证实NDFM-Index的查询效率与大规模图集中的查询能力。在文献[2]中,Zhang等给出的实验结果显示,TreePi的执行效率明显好于gIndex。因此,本节的实验将围绕与TreePi的比较进行。

实验数据集来自两部分。一部分是发展医疗计划(Developmental Therapeutics Program,DTP)提供的数据集,其中包括多于43000种的化合物。该数据可免费从网站上获得。另一部分是模拟数据,模拟数据来自文献[16]中提供的图数据模拟器。该模拟器允许用户定义图的个数,图的平均尺寸,种子图个数,种子图平均尺寸,以及标号的个数。

在实验结果图中,当坐标为查询图尺寸时,单位为查询图中包含的节点个数。当坐标为侯选集平均尺寸时,单位为侯选集中图的个数。当坐标为数据集尺寸时,单位为该图集中包含图的个数。

本实验运行于Pentium IV CPU,主频为3.2GHz,主存为512MB,硬盘空间为120G,编译环境为gcc/g++。

5.1 真实数据

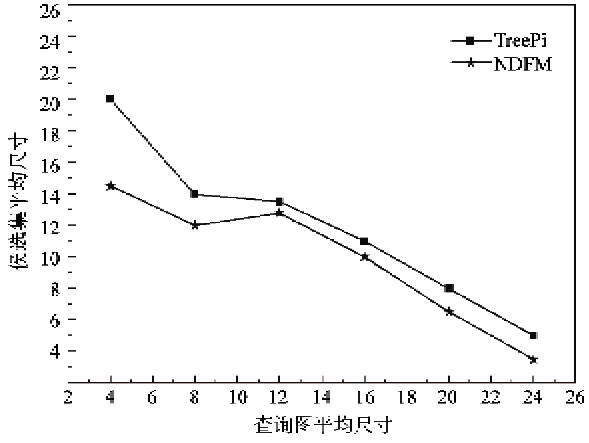
本部分给出真实数据的实验结果。在这部分实验中,从DTP中随机选取10000个图作为数据集,同时,随机选择1000个图作为查询图的生成器。生成的查询图边数由 m 控制,Q_m 为不同的查询图集合。共有6组 Q_m,每组包含边数为 m 的图1000个。m 的取值分别为4,8,12,16,20,24。同时,每组具有相同边数的查询图被划分为两组,分别为大支持集与小支持集,其临界值为50。

在与TreePi比较时,TreePi中的参数定义如下。对于支持度函数的参数,我们分别设定为 $\alpha = 5, \beta =$

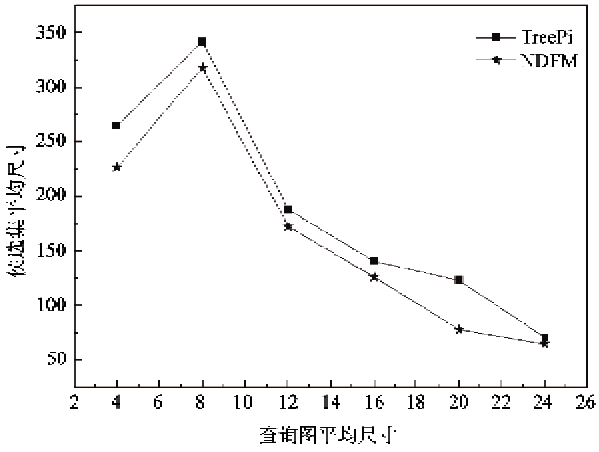
2，$\eta=10$。同时，设定 $\gamma=1.5$。这些给定的参数值，与文献[2]中给出的参数相同，这样，能得到对TreePi公平的结果。

首先检验NDFM-Index的过滤能力，得到更小的候选集是提高查询效率的基础。NDFM-Index过滤的能力来自于更准确的决策模式选择。

图4所示为平均支持集尺寸实验结果的比较。从图中可以非常清楚地看到，NDFM得到的候选集明显小于TreePi。而且，无论查询图是频繁的还是非频繁的，NDFM-Query得到的候选集尺寸均小于TreePi。



(a) 小支持集情况



(b) 大支持集情况

图4 平均支持集尺寸

图5所示为真实数据集上两个算法的索引构建时间比较。构建过程中，数据集的尺寸为 x 轴，且其个数分别为2000，4000，6000，8000和10000。虽然两个算法的索引构建时间均随图集规模的增加而增加，但TreePi是在图集规模小时增加快，而NDFM是在图集规模大时增加速度快。这是由于TreePi在索引构建过程中，频繁树的挖掘在达到一定规模之后，其增长速度逐渐趋于稳定，而NDFM则随着图集规模的增加，*NA*，*CA* 和 *DF* 之间的映射关系变得越来越复杂，因此，计算开销增加快。

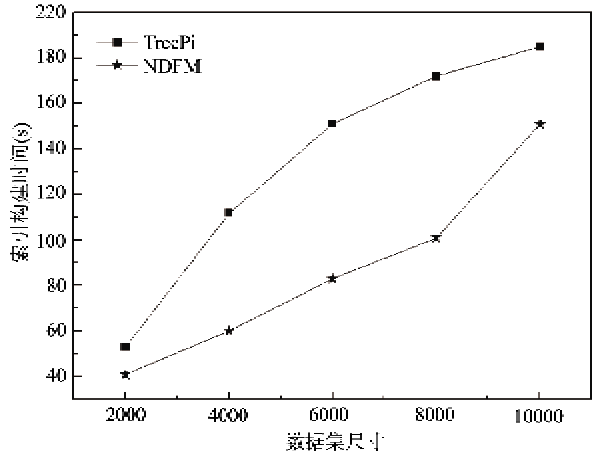


图5 索引构建时间的比较

图6所示为两个算法查询时间的比较，采用的查询图与图4所示实验类似，变边的数量分为4，8，12，16，20，24共6组。由实验结果可以看出，NDFM的查询效率相较于TreePi更平稳，这个结果是由于查询过程中，NDFM的计算直接和节点与决策模式之间的映射相关，因此，效率表现变化不大。就总体效率而言，NDFM-Query总体查询效率是TreePi的1/2到2/3左右。

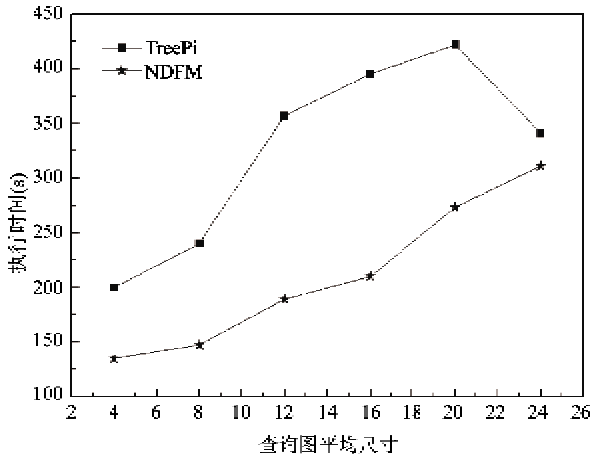


图6 查询的时间消耗

5.2 模拟数据

图7和图8分别给出了NDFM-Query与TreePi

的索引构建时间和查询时间在模拟数据集上的比较结果。模拟数据设置为利用 1000 个种子图生成 8000 个数据图,包含 40 个不同的标号值。种子平均边数为 10,而生成图的平均边数为 20。该数据记为 D8kI10T20S1kL40。

从实验结果分析得出,模拟数据的实验结果与真实数据的结果类似。

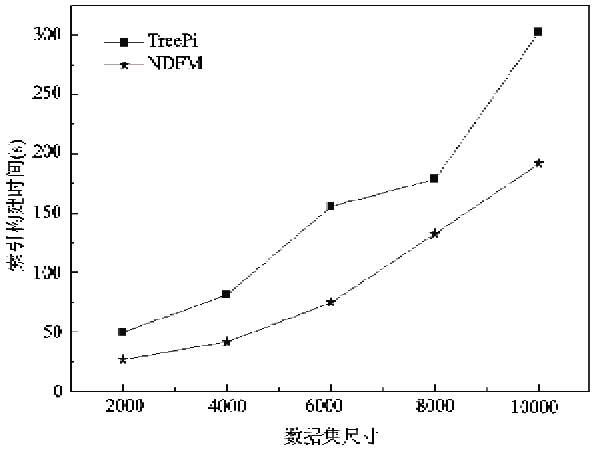


图 7　索引构建时间的比较

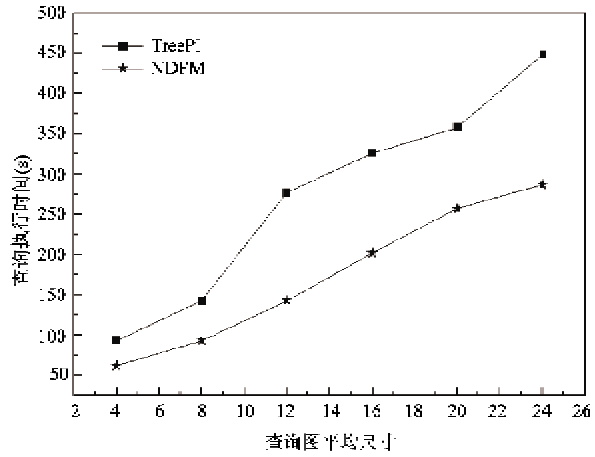


图 8　查询执行时间的比较

6　结　论

本文提出了一种基于节点与决策模式两段式映射的图查询算法。此算法采用了过滤-验证机制。与其它图查询算法不同,本文利用节点向量 *NA*,组成向量 *CA* 和决策模式 *DF* 组成索引结构。通过查询图中包含的关键节点信息,利用索引中 *NA-CA* 和 (*NA*, *CA*)-*DF* 之间的映射,可迅速得到查询图包含的频繁模式,避免了采用枚举加子图同构的方式带来的额外开销。同时,决策模式概念的引入,使得候选集的尺寸更小,从而进一步提高了查询的效率。性能分析和实验结果证明,NDFM-Index 的执行效率和可扩展性均明显优于其它图查询算法。

NDFM 的主要贡献如下:(1)在查询之初, NDFM 通过图集节点信息得到查询图包含的频繁模式索引项。采用这种方式,有效避免了在得到 q 包含的频繁模式过程中需要进行的枚举方法,降低了查询算法在得到索引项的过程中的时间复杂性。(2)NDFM 利用决策模式组织索引。决策模式的优点在于,一方面可通过低代价计算得到,另一方面,可得到更小的候选集,从而减少验证阶段子图同构次数,进一步提高了算法的整体效率。

参考文献

[1] Yan X, Yu P, Han J. Graph indexing: a frequent structure based approach. In: Proceedings of the ACM Special Interest Group on Management of Data (SIGMOD). Paris, France: ACM Press, 2004. 335-346

[2] Zhang S, Hu M, Yang J. TreePi: a novel graph indexing method. In: Proceedings of the 23nd International Conference on Data Engineering (ICDE), Istanbul, Turkey, 2007. 966-975

[3] Cheng J, Ke Y, Ng W, et al. FG-Index: towards verification free query processing on graph databases. In: Proceedings of the ACM Special Interest Group on Management of Data (SIGMOD), Beijing, China, 2007. 857-872

[4] Zhao P, Yu J, Yu P. Graph indexing: Tree + Delta >= Graph. In: Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB), University of Vienna, Austria, 2007. 938-949

[5] Yan X, Yu P, Han J. Substructure similarity search in graph databases. In: Proceedings of the ACM Special Interest Group on Management of Data (SIGMOD), Baltimore, Maryland, USA, 2005. 766-777

[6] Yan X, Zhu F, Han J, et al. Searching substructures with superimposed distance. In: Proceedings of the 22nd International Conference on Data Engineering (ICDE), Atlanta, Georgia, USA, 2006. 88

[7] Shasha D, Wang J, Giugno R. Algorithmics and applications of tree and graph searching. In: Proceedings of the Principles of Database Systems (PODS), Madison, Wisconsin, USA, 2002. 39-52

[8] He H, Singh A. Closure-tree: an index structure for graph queries. In: Proceedings of the 22nd International Conference on Data Engineering (ICDE), Atlanta, Georgia, USA, 2006. 38-47

[9] Williams D, Huan J, Wang W. Graph database indexing us-

ing structured graph decomposition. In: Proceedings of the 23nd International Conference on Data Engineering (ICDE), Istanbul, Turkey, 2007. 976-985

[10] Jiang H, Wang H, Yu P, et al. GString: a novel approach for efficient search in graph databases. In: Proceedings of the 23nd International Conference on Data Engineering (ICDE), Istanbul, Turkey, 2007. 566-575

[11] Zou L, Chen L, Yu J X, et al. A novel spectral coding in a large graph database. In: Proceedings of the 11st International Conference on Extending Database Technology (EDBT), Nantes, France, 2008. 181-192

[12] Gupta K, Suciu D. Stream processing of XPath queries with predicates. In: Proceedings of the ACM Special Interest Group on Management of Data (SIGMOD), San Diego, California, USA, 2003. 419-430

[13] Bohannon P, Fan W, Flaster M, et al. Information preserving XML schema embedding. In: Proceedings of the 31st International Conference on Very Large Data Bases (VLDB), Trondheim, Norway, 2005. 85-96

[14] Garey M, Johnson D, Bezos J. Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: W. H. Freeman and Company, 1979

[15] Bloom B. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 1970, 13(7):422-426

[16] Kuramochi M, Karypis G. Frequent subgraph discovery. In: Proceedings of the International Conference on Data Engineering (ICDE), San Jose, California, USA, 2001. 313-326

A subgraph query algorithm based on two-step mapping on vertex to decision feature

Li Xiantong, Li Jianzhong

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

Abstract

To solve the problem of subgraph query processing in large graph databases, the paper gives a two-step node to decision feature mapping (NDFM) indexed structure, named the NDFM-Index, and based on it, proposes a subgraph query processing algorithm. The NDFM-Index uses the mapping between key nodes, with the distribution of neighbors labels, and decision features to get the indexed features which are included by query graph avoiding enumeration method. The results of theoretical analysis and experimental evaluation show that the proposed method not only avoids the enumeration method of getting subgraphs of query graph, but also effectively reduces the subgraph isomorphism tests between the query graph and graphs in candidate answer set in verification stage.

Key words: graph dataset, graph query, frequent feature, decision feature, vertex array