基于路径映射的相似子图匹配算法

马 静 王浩成
(辽宁大学信息学院 沈阳 110036)

摘 要 迄今为止,相关的图相似性匹配方法通常不考虑节点关系以及边权重的实际意义。提出一种基于路径映射的相似子图匹配方法,用以更精确地查找具有相似拓扑结构的加权图。其创新之处在于充分利用标签信息,综合考虑拓扑结构特征,克服了忽略节点结构关系和边权重的意义去分析图相似性的弊端。因此,该方法在很大程度上提高了图相似性匹配的应用范围和匹配精度。实验表明本方法具有较高的查询质量和效率。

关键词 子图相似,路径映射,加权图
中图法分类号 TP392 **文献标识码** A

Subgraph Similarity Matching Based on Path Mapping

MA Jing WANG Hao-cheng
(School of Information, Liaoning University, Shenyang 110036, China)

Abstract So far, the related similarity matching methods usually do not consider the relation between nodes and the practical significance of the weighted edges. This paper presented a similarity subgraph matching method based on path mapping. It can accurately locate the weighted graphs with similar topological structure. The innovations of this paper are to make full use of the attribute information and to consider the topological structure characteristics, overcoming the ignorance of the structure relations and the significance of edge weights to analysis graph similarity. Therefore, the method can greatly improve the application scope of graph similarity matching and the experiment shows that this method has higher searching quality and efficiency.

Keywords Subgraph similarity, Path mapping, Weighted graph

1 前言

近年来,图论和计算机技术相结合的数据挖掘方法日臻成熟,众多研究者纷纷选择用图来建模各种复杂的网络系统,例如,蛋白质交互网络[1]、社区关系网络[2,3]、交通网络[4]等。图的抽象描述过滤了网络系统纷繁复杂的冗余信息,只保留了网络系统的基本结构,有利于对网络系统的内在共同特性进行深入研究。

相似子图匹配通常在图同构或同态的基础上,结合某个具体应用环境找出具有相似拓扑结构的图。在相似子图匹配实际应用中,通常先将概念化的应用问题转化为合适的图,再用不同的方法进行图的相似度计算,即找到度量对应节点和边的相似度的最优匹配[5,6]。图匹配通常为NP问题,算法的复杂度随着图规模的增大呈指数增长,因而对图匹配的优化算法进行改进难度较大。近年来研究者从不同的角度提出了一些关于相似子图匹配的方法。

Grafil[7]算法通过把查询图的边松弛比转化成最大允许的特征丢失,然后在执行结构相似性计算之前快速过滤掉数据库中不满足条件的图,来提高子图相似性搜索的整体性能。

SAPPER[8]算法对查询图设定边编辑距离,将其转换成多个查询子图,并找出其中互相包含的查询图,对大的查询图进行查找之前,先对较小的查询图进行查找,可以过滤掉重复查找的部分。

以上两种方法通过指定与查询图之间差异的容忍度来得到相似的匹配结果,这样不可避免地造成图数据中原始信息的部分丢失。

LeRP[5]算法是一个求近似子图同构的方法。该方法基于计算与某节点相关联的长为 r 的路径和回路个数,得到两个图的最大近似同构子图。

SF[6]算法是一个通用图匹配算法,应用于Schema匹配,提取出节点间的邻接关系作为结构特征信息,利用节点的相似性传播,进行迭代计算,直到达到一个收敛点。

这两种图近似匹配方法都利用了基于结构的相似性计算,相比于仅考虑单个节点的特征,具有更高的匹配效率。

2 主要研究工作

归纳总结现有研究成果主要思想的优缺点,基于文献[9]提出的图同态的匹配算法,给出一种基于路径映射的相似子图匹配方法,以研究如何在大型网络社区中发现相似的群体关系。主要包括以下内容:(1)提出一种新的节点相似性计算方法。将节点的邻接关系作为结构特征信息,利用相似的传递性,即当两个节点的邻接点之间也存在相似映射,可以认为

到稿日期:2011-12-18 返修日期:2012-04-21
马 静(1986-),女,硕士,主要研究方向为数据挖掘,E-mail:majing.868893@yahoo.com.cn;王浩成(1983-),男,硕士,主要研究方向为数据挖掘。

这两个节点的相似值随之提高。(2) 结合边的权重和节点标签的相似值,提出一种相似路径映射方法。只有满足以下条件,才认为存在相似路径映射:①如果两个节点在查询图中是邻接的,则它们的相似节点在数据库图中也是通过路径可达的;②通过公式计算出的路径相似差度满足给定的阈值。(3) 基于以上节点相似性计算方法和路径映射方法提出的一种高效的相似子图匹配算法,在匹配过程中,兼顾标签和结构双重因素,按照节点相似度递减的顺序,沿图拓扑结构传播开来,同时,利用以上基于路径映射的方法找出最佳同态映射。

本文第 2 节介绍主要研究工作;第 3 节详细阐述节点相似性计算方法和相似路径映射方法;第 4 节为相似子图匹配算法;第 5 节给出所提方法的实验评价;最后为结束语。

3 图的相似性计算

目前大部分图的相似性算法都是基于无权网络,很大程度上影响了其应用范围。本文针对加权网络提出的图相似性计算方法,充分利用节点和边的标签信息,综合考虑了结构特征。

3.1 基于邻接点的节点相似性计算方法

这里,将网络图 G 定义为一个六元组,即 $G=(V,E,L,W,l,w)$,其中 V 是节点的集合。$E\subseteq V\times V$ 为边集合,$\langle v,u\rangle$ 表示从 v 到 u 的一条边。L 表示节点标签集合,如用户行为、用户注册信息等。W 为边权重集合,这里边权重表示用户之间的亲密度。l 为节点映射函数,即 $l_{|V\to L}$。w 为边映射函数,即 $w_{|E\to[0,1]}$。$v,u\in$ 查询图 G_P,$v',u'\in$ 数据库中图 G。

所述的相似度传播机制来自于如下观察:假设图 G 中的节点 v 与图 G' 中的节点 v' 相匹配,那么意味着节点 v 与 v' 的邻域拓扑也应该是相似的,即 v 的邻接点能很好地匹配到 v' 的邻接点上。结合加权有向图中节点的关系结构来研究节点的相似性可知,节点的邻接点之间越相似以及邻接点与节点联系越紧密,则节点的相似度越高。具体而言,任意节点对 (v,v') 的相似度需满足:

$$\mathrm{Sim}(v,v')=mat(v,v')+\alpha\cdot\frac{\sum_{i=1}^{m}\mathrm{Sim}(u,u')*(Wvu*Wv'u')}{m}+\beta\cdot\frac{\sum_{i=1}^{n}\mathrm{Sim}(u,u')*(Wuv*Wu'v')}{n}\quad(1)$$

式(1)表达的含义是任意节点对 (v,v') 的相似度 $\mathrm{Sim}(v,v')$ 来源于节点标签相似度和邻接点集相似度的平均值之和,其中 u 和 u' 分别是 v 和 v' 的邻接点中根据标签匹配的节点对,$mat(v,v')$ 表示 v 和 v' 基于节点标签相似的值,$mat(v,v')$ 可通过余弦相似度定理得到[10],且 $mat(v,v')\in[0,1]$。边的权重 W_{vu} 表示邻接点与本节点联系紧密程度的指标,因为是有向图,所以用 $(W_{vu}*W_{v'u'})$ 和 $(W_{uv}*W_{u'v'})$ 分别表示从 (v,v') 指向的邻接点与指向 (v,v') 的邻接点对节点相似性的影响程度。m 是指从 (v,v') 指向的邻接点中根据标签匹配的节点对的个数,n 是指向 (v,v') 的邻接点中根据标签匹配的节点对的个数。α,β 用来衡量两个方向的邻接点相似度对于 $\mathrm{Sim}(v,v')$ 的贡献程度(且 $\alpha+\beta=1$),下面实例中用到的 α,β 都是按照 1/2 来计算。考虑到数值计算的稳定性,需要对每次迭代计算得到的 Sim 进行归一化,使 Sim 保持在区间 $[0,1]$ 之间,即当所有的候选匹配节点对的 Sim 都计算完以后,需除以最大的 Sim 值。

下面给出节点相似值的计算步骤:首先初始化节点相似值 Sim,当 $mat(v,v')>0$ 时,将 v 和 v' 看作是根据标签匹配的节点对,$\mathrm{Sim}(v,v')=mat(v,v')$,否则,$\mathrm{Sim}(v,v')=0$。$mat(v,v')>0$ 的节点对 (v,v') 作为候选匹配节点对,按照 $\mathrm{Sim}(v,v')$ 值由大到小的顺序从候选匹配节点对中依次选择节点对 (v,v'),如果 $\mathrm{Sim}(v,v')$ 值相同(如初始情况下),则可随机选一个。根据式(1)迭代计算 $\mathrm{Sim}(v,v')$,直到 $\mathrm{Sim}(v,v')$ 不再变化或者达到某个稳定阈值时,停止更新。设 ξ 为相似度阈值,当且仅当 $\mathrm{Sim}(v,v')\geqslant\xi$ 时,则认为 v 和 v' 是基于邻接点相似匹配的节点对,构造节点匹配对集合 $Match$。

3.2 一种基于路径映射的方法

边到路径的映射方法允许查询的两图有类似的拓扑结构,并且可以更准确地反映顶点间关系,同时大大提高匹配的范围,本节给出一种基于路径映射的相似性计算方法。

对于节点带标签、边加权的图来说,进行相似性路径映射时,需要考虑路径两端节点的相似性信息和路径的权重。路径两端节点越相似,路径上平均权重之差越小,路径存在相似性映射的可能就越大。由此,给出一个衡量路径相似情况的指标——路径相似差度 $PaBalen(E,E')$,只有当 $PaBalen(E,E')$ 的值在给定的相似阈值 θ 内时,才认为 E 和 E' 存在相似路径映射。根据路径两端节点的相似性信息和路径的权重定义路径相似差度公式:

$$PaBalen(E,E')=\frac{1}{\mathrm{Sim}(v,v')+\mathrm{Sim}(u,u')}*\left|\frac{WP_{vu}}{m}-\frac{WP_{v'u'}}{n}\right|\quad(2)$$

式中,$v,u\in G_P$,$v',u'\in G$,v,v' 和 u,u' 分别是路径 E,E' 两端的点,m 表示 v 到 u 经过的边数目,n 表示 v' 到 u' 经过的边数目,因为是边到路径的映射,所以 $m=1,n\geqslant1$。$\frac{WP_{vu}}{m}$ 是指从 v 到 u 的路径上权重的平均值,$\frac{WP_{v'u'}}{n}$ 同理。将路径 E,E' 的平均权重之差 $\left|\frac{WP_{vu}}{m}-\frac{WP_{v'u'}}{n}\right|$ 记作 $WP_{(E,E')}$。给出一个差度阈值 θ,路径 E,E' 存在相似映射,当且仅当 $PaBalen(E,E')\leqslant\theta$ 时。

下面给出相似路径映射的步骤:如果 $(v,v')\in Match$ 是相似匹配的节点,v 和 u 是邻接点,且 $(u',u)\in Match$,首先判断 v' 和 u' 之间是否可达,如果可达,那么根据式(2)计算 $PaBalen(E,E')$。若计算出的路径相似差度 $PaBalen(E,E')\leqslant\theta$,则返回 (E,E'),否则认为不存在相似路径映射。

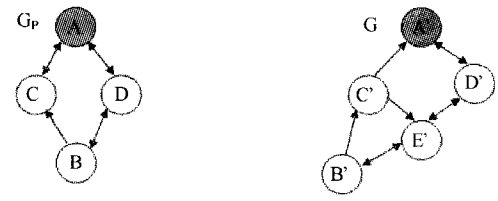


图 1 路径相似映射图

为方便描述,用一个简单图来表示路径映射过程,如图 1 所示,若 $(A,A')(B,B')\in Match$,由于两点之间的路径可能不止一条,如从 B 到 A 的路径有 2 条,从 B' 到 A' 的路径有 3 条,那么可以得到 $2*3$ 个 $WP_{(E,E')}$,找出其中最小的 $WP_{(E,E')}$。因为 $WP_{(E,E')}$ 的值越小,$PaBalen(E,E')$ 的值越小,(E,E') 存在相似性映射的可能越大,所以将 $WP_{(E,E')}$ 最小

的值代入式(2)计算。如果得到的 $PaBalen(E,E')\leqslant\theta$，那么就认为 E 和 E' 之间存在相似路径映射。否则，E 和 E' 之间不存在相似路径映射。

4 相似子图匹配算法

基于文献[9]提出的图同态的匹配算法，利用以上节点相似度计算方法和加权路径映射方法对其进行改进，提出了一种基于路径映射的相似子图匹配算法，该算法的基本思想如下：

给定查询图 Gp、数据库中图 G、mat、ξ、θ，计算从 Gp 到 G 的相似子图映射 σm。首先调用基于邻接点的节点相似性计算方法，结合标签信息和结构特征来为图 Gp 和 G 中的节点建立节点匹配对集 $Match$。(1)从 $Match$ 中每次都挑选相似值 $\mathrm{Sim}(v,v')$ 最大的节点对 (v,v') 进行匹配，然后利用路径相似差度作为路径映射条件来快速地剪枝，将 $Match$ 分成两部分，一部分是能继续进行匹配的节点对集合，另一部分是不能继续进行匹配的节点对集合。(2)两部分再分别重复进行以上步骤，最后递归返回包含匹配对 (v,v') 的映射 σ_1 和不包含匹配对 (v,v') 的映射 σ_2 中的较大者，作为本次最佳映射 σ。(3)循环调用步骤(1)直到找出最多的节点映射，返回最佳映射 σ_m。

算法开始前需要完成以下工作，以确保匹配的质量：

a) 根据节点匹配对集合 $Match$ 为 Gp 中的节点创建匹配列表 H。H 中的每个节点 v 都有 $H[v].good$ 和 $H[v].minus$。其中 $H[v].good$ 是通过映射 σ 可能和 v 匹配的候选节点的集合，$H[v].minus$ 是通过映射 σ 不能和 v 匹配的节点集合。

b) 为 Gp 中的每个节点建立邻接表 L。

c) 为 G 建立可达矩阵 $L1$。若任意两点 (v',u') 之间最多经过 3 条边可达，则 $L1[v',u']=1$，否则 $L1[v',u']=0$。

d) I 是相互矛盾的匹配对 (v,u) 的集合。对于任何两个匹配对 (v,v')、(v,v'')，若 v 和 v' 相似匹配，则 v 将不可能和 v'' 相似匹配，此时认为这两个匹配对是互相矛盾的。

算法 1 Sim_Match

输入：Gp(V,E,L,W,l,w),G(V',E',L',W',l',w'),mat(v,u),ξ

输出：从 Gp 到 G 的相似子图映射 σ_m

1. 计算节点相似值 Sim;
2. 对 Gp 中每个节点 v do;
3. L[v]:= {v' | v' ∈V',(v,v')∈E,v'是 v 的邻接点};
4. H[v].good:= {v' | v' ∈V',mat(v,v') ≥ ξ};
5. H[v].minus:= ∅;
6. 为 G 建立可达矩阵 L1;
7. σ_m:=∅;
8. 当 sizeof (H)>sizeof (σ_m) 时,do
9. (σ,I):= greedyMatch(L,L1,H);
10. H:=H\I;
11. 如果 sizeof (σ) > sizeof (σ_m),那么 σ_m:=σ;
12. 返回 σ_m;

算法 2 greedyMatch

输入：Gp 的匹配列表 H

输出：Gp[H]到 G 的节点映射 σ,冲突匹配对集合

1. 如果 H 为空,那么返回 (∅,∅);
2. 从 H 中选择使 Sim 的值最大的节点对(v,v');
3. H[v].minus:= H[v].good \ {v'};
4. H[v].good:= ∅;
5. H:= trimMatching(v,v',L,L1,H);/* 将 H 分成 H^+和 H^-两部分 */
6. 如果 H[v].good 非空,那么
7. H^+[v].good:= H[v].good;
8. H^+[v].minus:= ∅;
9. 如果 H[v].minus 非空,那么
10. H^+[v].good:= H[v].minus;
11. H^-[v].minus:= ∅;
12. (σ_1,I_1):= greedyMatch(L,L1,H^+);
13. (σ_2,I_2):= greedyMatch(L,L1,H^-);
14. σ:= max($\sigma_1\cup\{(v,v')\},\sigma_2$);
15. I:= max($I_1,I_2\cup\{(v,v')\}$);
16. 返回(σ,I);

算法 3 trimMatching

输入：v,v',H,L,L1,Sim(v,v'),路径相似差度阈值 θ

输出：更新的列表 H

1. 对 L[v]∩ H 中任意节点 u 和 H[u].good 中节点 u' do
/* 对 v 的邻节点进行剪枝 */
2. 如果 L1[v',u']=0,那么
3. H[u].good:= H[u].good\ {u'};
4. H[u].minus:= H[u].minus ∪ {u'};
5. 否则
6. E:={the edge from v to u};
7. E':={the path from v' to u'};
8. 计算 PaBalen(E,E');
9. 如果 PaBalen(E,E')>θ,那么
10. H[u].good:= H[u].good\ {u'};
11. H[u].minus:= H[u].minus∪{u'};
12. 返回 H;

相似子图匹配算法整体执行过程如下，首先为 Gp 中每个节点建立邻接表 L，匹配列表 H，为 G 建立可达矩阵 $L1$(1－6 行)，将映射 σ_m 初始化为空(7 行)并且调用算法 greedyMatch(8－9 行)。算法 greedyMatch 首先从 H 中挑选 Sim 值最大的节点对 (v,v')，然后递归计算包含匹配对 (v,v') 的映射 σ_1 和不包含匹配对 (v,v') 的映射 σ_2，返回 $\sigma_1\cup\{(v,v')\}$ 和 σ_2 中的较大者。同时，greedyMatch 还计算出相互矛盾的匹配对的集合 I_1 和 I_2，并且返回 I_1 和 $I_2\cup\{(v,v')\}$ 中的较大者作为 I。得到了 greedyMatch 的返回结果 σ、I 后，从 H 中去除 I 并且选择 σ_m 和 σ 中的较大者(10－11 行)。循环调用 greedyMatch 直到 σ_m 比 H 大为止，最后返回映射 σ_m(12 行)。

贪婪匹配算法运行过程如下，首先选择一个匹配对 (v,v')，将 $H[v].good$ 的其他节点移到 $H[v].minus$ 中，并将 $H[v].good$ 设为空(2－4 行)。假设 (v,v') 已经匹配，就调用程序 trimMatching 对 v 的邻接点 u 进行剪枝(5 行)。更新后的 H 被分成 H^+ 和 H^- 两部分，H^+ 中的节点 v 满足 $H[v].good$ 不为空，否则 v 属于 H^-(6－11 行)，再对 $G[H^+]$ 和 $G[H^-]$ 循环调用 greedyMatch 来计算 σ_1 和 σ_2(12－13 行)。返回 σ_1(包含匹配对 (v,v') 的映射)和 σ_2(不包含匹配对 (v,v') 的映射)中的较大者，来判断 (v,v') 不是一个好的选择。如果 (v,v') 不是一个好的匹配，那么将它置于 I_2 中，同时，greedyMatch 还计算出相互矛盾的匹配对的集合 I(14－15 行)。

剪枝算法运行过程如下，输入一个候选匹配对 (v,v') 和当前的匹配列表 H，假设 (v,v') 已经匹配，对于 v 的任意的邻

接点 u 来说，u' 是 $H[u].good$ 中的候选节点。如果 v' 到 u' 不可达，那么将 u' 从 $H[u].good$ 中移到 $H[u].minus$ 中(2－4行)；如果 v' 到 u' 可达，则计算路径相似差度 $PaBalen(E,E')$ (6－8行)。如果 $PaBalen(E,E')$ 大于给定的阈值 θ，也就说明从 E 到 E' 不存在路径映射，那么将 u' 从 $H[u].good$ 中移到 $H[u].minus$ 中(9－11行)，返回更新后的 H(12行)。

5 实验评价

由于本文算法 Sim_Match 是由图同态匹配算法 compMaxCard[9] 改进而来，因此设计几组相关实验，将 Sim_Match 算法与 compMaxCard 算法和查找相似节点的 SF (Similarity Flooding)[6] 算法进行比较，验证结果表明，本算法在相似匹配方面具有较高的精确度和效率。本文所有实验均在配置为：AMD Athlon(tm)64 * 2 Dual core Processor 5000 + 2.6GHz，1G 内存的 WinXP SP3 PC 上完成。每次实验都重复执行超过 5 次且将平均值记录下来。

本文实验使用的数据来自于腾讯微博。腾讯微博是一个公共的大型网络社区，有超过 1 亿的用户，内容覆盖领域广。从中选取 3 个不同种类的用户群，分别记为用户群 1、用户群 2、用户群 3，从每个用户群提炼出每个月用户浏览、下载的内容、用户间的交互信息等，作为用户的标签信息(节点标签)和用户交互的频度(边权重)。由于网络图非常庞大，可以仅考虑关键的“骨骼网络”，“骨骼网络”中只包含出入度超过一定阈值的节点。对于图 G 来说，生成 G 的“骨骼图”Gs，Gs 是 G 的子图并且 Gs 中的每个节点 v 的度满足：v 的度 $\geqslant G$ 中节点度的平均值 $+\alpha \times G$ 中节点度的最大值，α 是介于[0,1]之间的常数。

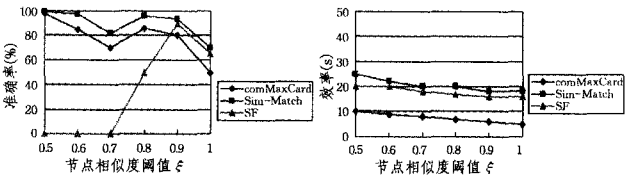
鉴于同一个网络可以有不同版本的图表示，这些不同版本的图应该是相似匹配的，基于这一点来评价算法的精确度。为各用户群生成一个包含 11 个子图的集合 TA，集合中的子图代表了每个用户群中 11 个不同版本的存档图。生成了 TA 以后，根据时间戳得到网络图的序列，将这 11 个图根据得到的序列进行排序，将原始的图看作 $G1$，然后将其他 10 个不同版本的子图 $G2$ 运用不同的方法与 $G1$ 进行相似匹配，根据发现相似图匹配的比例表示算法的精确度。

先为每个用户群生成一个包含 11 个子图的集合 TA，然后基于 TA，设定 $\alpha=0.2$，生成“骨骼图”集合，表 1 为 3 个用户群的网络骨骼表。

表 1 用户群骨骼表

网络图	网络图 G			骨骼网络(α=0.2)	
	节点数	边数	节点度的平均值	节点度的最大值	节点数	边数
用户群 1	20,000	66,000	6.60	260	250	17,544
用户群 2	6,600	40,678	12.32	388	55	938
用户群 3	7,000	12,218	3.49	200	120	864

由于 Sim_Match 算法需要用到节点相似度阈值 ξ 和路径相似差度阈值 θ，为了保证实验的可靠性，将不同 ξ 下的 Sim_Match 算法的准确率和效率进行了实验，得到的结果如图 2(a)所示。从图中可以看出，算法的准确率对于 ξ 的变化不是很敏感，大多数情况下都在 70%以上。当 ξ 低于 0.7 时，算法准确率最低，因为此时 $G1$ 的节点很容易在 $G2$ 中找到它的匹配节点。当 $\xi>0.7$ 时，说明 $G1$ 中的节点在 $G2$ 中找到与它匹配节点的机率越大，但 $\xi>0.9$ 时，从数据库图中发现相似子图的要求越高，越近似于精确匹配，所以找到的相似子图会减少。图 2(b)表示不同 ξ 下算法的执行效率。

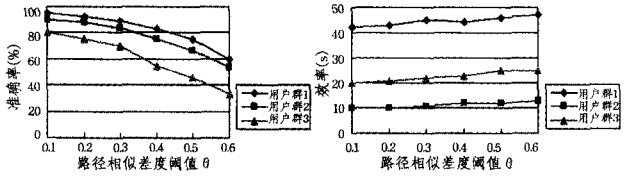


(a)准确率-节点相似度阈值 ξ　　(b) 效率-节点相似度阈值 ξ

图 2 Sim_Match 算法的准确率和效率-节点相似度阈值 ξ

令节点相似度阈值 $\xi=0.8$，将路径相似差度阈值 θ 对于算法的影响进行实验。从图 3(a)中 3 个用户群的实验结果可知，算法的精确度和效率都随 θ 值的增加而递减，而且图 3(a)－(b)两图中都在 $\theta=0.3$ 时变化幅度最大，因此当 θ 保持在[0－ 0.3]之间时，算法的性能最佳。图 3(b)显示了 θ 对算法效率的影响。

参考上面实验结果，令节点相似度阈值 $\xi=0.8$，相似路径差度阈值 $\theta=0.2$，利用网络骨骼图集合 TA 来比较 3 种方法的精确度和效率，得到表 2 所列结果。



(a) 准确率-路径相似差度阈值 θ　　(b) 效率-路径相似差度阈值 θ

图 3 Sim_Match 算法的准确率和效率-路径相似差度阈值 θ

表 2 3 种算法的精确度和效率

算法	精确度(%)			效率(秒)		
	用户群 1	用户群 2	用户群 3	用户群 1	用户群 2	用户群 3
compMaxCard	77	84	53	3.78	0.59	1.87
SF	45	33	28	35.26	6.43	14.83
Sim_Match	92	89	70	42.18	9.52	19.65

精确度

由 3 种算法精确度实验可知，SF 的准确率大部分都在 50%以下，本文提出的 Sim_Match 比 compMaxCard 的精确度高 5%～17%。特别是在用户群 1 和用户群 2 中说明，越是大的稠密图，邻接点的相似性对本节点相似性的影响越大，因此相似性计算越精确。由此可见，Sim_Match 在图相似性匹配方面有较好的匹配质量。

效率

Sim_Match 的高精确度是以牺牲高运行时间为代价的。由于 compMaxCard 仅仅根据节点属性来判断节点相似性，而 SF 和 Sim_Match 要多次迭代计算节点的相似性，并且 Sim_Match 还要进行带权路径的相似性计算，那么 Sim_Match 和 SF 的时间肯定要劣于 compMaxCard，但剪枝算法使得子图相似算法的性能有一定程度的提高。因此在相似性计算方面，Sim_Match 算法的运行效率还有待提高。

从实验结果可以看出，Sim_Match 在图相似性匹配方面有较好的匹配质量，大部分准确性都在 70%以上，但是由于需要多次迭代计算节点的相似性，使得其效率随着网络图的增大，其运行时间递增得比较快，对于在大图中发现相似子图方面还有待改进。

结束语 本文重点研究了图数据库中的相似子图匹配方法，提出一种基于图相似性计算的相似子图匹配算法，按照节点相似值递减的顺序，沿图拓扑结构传播开来，并在匹配过程中利用路径的相似差度进行剪枝来缩减匹配节点候选集，从而减小搜索空间。实验结果表明，本文提出的相似子图匹配算法具有较好的匹配质量和运行效率。

参 考 文 献

[1] The Gene Ontology Consortium. Gene Ontology: Tool for the Unification of Biology[J]. Nature genetics, 2000, 25(1): 25-29

[2] Watts D J, Strogatz S H. Collective dynamics of " small-world" networks[J]. Nature, 1998, 393(6684): 440-442

[3] Borgatti S P, Mehra A J, Brass D J, et al. Network Analysis in the Social Sciences[J]. Science, 2009, 323: 892-895

[4] Kalapala V, Sanwalani V, Clauset A, et al. Scale in Road Networks[EB/OL]. http://arxiv. org/abs/physics/0510198, 2007-10-27

[5] DePiero F, Krout D. An algorithm using length-r paths to Approximate subgraph isomorphism[J]. Pattern Recogn. Lett., 2003, 24(1-3): 33-46

[6] Melnik S, Garcia-Molina H, Rahm E. Similarity fooding: A versatile graph matching algorithm[C]// ICDE. 2002: 1-45

[7] Yan X, Yu P S, Han J. Substructure Similarity Search in Graph Databases[C]// Proceedings of the ACM SIGMOD International Conference on Management of Data. Baltimore, Maryland, USA: ACM Press, 2005: 766-777

[8] Zhang Shi-jie, Yang Jiong, Jin Wei. SAPPER: Subgraph Indexing and Approximate Matching in Large Graphs[C]// VLDB. 2010: 1-10

[9] Fan Wen-fei, Li Jian-zhong, Ma Shuai, et al. Graph Homomorphism Revisited for Graph Matching[C]// VLDB. 2010: 1-12

[10] 韩瑞凯，孟嗣仪，刘云. 基于兴趣相似度的社区结构发现算法研究[J]. 铁路计算机应用，2010，19(10)：10-14

[11] Shang H, Zhu K, Lin X, et al. Similarity Search on Supergraph Containment[C]// Proceedings of the 26th International Conference on Data Engineering (ICDE). IEEE Computer Society Press, Long Beach, California, USA, 2010: 637-648

[12] 刘宝生，闫莉萍. 几种经典相似性度量的比较研究[J]. 计算机应用研究，2006，23(11)：1-3

[13] 陈琼，李辉辉，肖南峰. 基于节点动态属性相似性的社会网络社区推荐算法[J]. 计算机应用，2010，30(5)：1-5

(上接第 105 页)

行，发生死锁，需要对源程序进行修改。

```
    Full statespace search for:
  never claim+(|t|_0)
  assertion violations+(if within scope of claim)
  acceptance cycles+(faimess disabled)
  invalid end state-(disabled by never claim)
State-vector 68 byte, depth reached 24, errors: 1
  12 states, stored
  0 states, matched
  12 transitions(=stored+matched)
  5 atomic steps
hash conflicts: 0(resolved)
Stats on memory usage (in Megabytes):
  0.001 equivalent memory usage for states(stored * (State-vector
  +overhead))
  0.287 actual memory usage for states(unsuccessful compression:
        29819.05%) state-vector as stored=25032 byte+16byte
        overhead
  2.000 memory used for has table(-w19)
  0.343 memory used for DFS stack(-m10000)
  2.539 total actual memory usage
```

图 7 验证结果

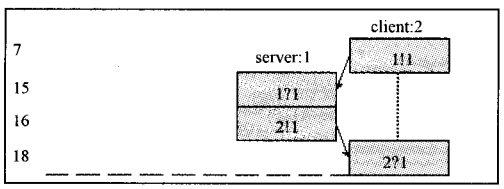


图 8 消息通道执行序列

结束语 模型检测技术是检测网络通信软件运行时问题、提高软件开发质量的有效手段。本文围绕检验 socket 通信程序中 socket 函数调用顺序的正确性进行探讨。分析顺序结构的 socket 程序，对 socket 通信程序中 socket 函数调用序列进行抽取，提出 Promela 模型抽取方案，并对模型所应满足的性质用线性时态逻辑(LTL)进行规约，再利用模型检测工具对所抽取的模型进行检测。下一步的工作主要体现在：1)能对有选择、循环、条件结构的 socket 通信程序进行分析；2)支持 socket 的 select 函数，一个服务多个客户模式，读、写、select 任意组合且阻塞和非阻塞混合，区分 UDP/TCP 通信等建模。

参 考 文 献

[1] de la Cámara P. Checking the reliability of socket based communication software[J]. Int J Softw Tools Technol Transfer, 2009, 11: 359-374

[2] Martinez J, Jimenez C. Software model checking for Internet protocols with java pathfinder[C]// 6th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems. 2008: 91-100

[3] 全嘉辉，张欢欢. 基于 FeaVer 的 MINIX 3 验证和改进[J]. 计算机工程，2010，36(22)：46-48

[4] Holzmann G J, Smith M H. Software model checking: extracting verification models from source code [J]. Software testing, verification and reliability, 2001, 11: 65-79

[5] Havelund K, Pressburger T. Model Checking Java Programs Using Java PathFinder[J]. Int J STTT, 2000, 2: 366-381

[6] Corbett J C, Dwyer M B. Bandera: extracting finite state models from java source code [C]// Software Engineering (Proceedings of the 2000 International Conference). 2000: 439-448

[7] 王大伟，张大方. 一种自动化模型检测 ANSI-C 程序的实用方法[J]. 计算机工程与科学，2010，32(4)：79-82

[8] Stevens W R. TCP/IP 详解，卷 1，协议(英文版) [M]. 北京：人民邮电出版社，2010

[9] Stevens W R. TCP/IP 详解，卷 2，实现(英文版) [M]. 北京：人民邮电出版社，2010

[10] Stevens W R. UNIX 网络编程(卷 1)：套接字联网 API(第 3 版) [M]. 北京：人民邮电出版社，2010

[11] 肖美华，薛锦云. 基于 SPIN/Promela 的并发系统验证[J]. 计算机科学，2004，31(8)：201-203

[12] Holzmann G J. The SPIN Model Checker, Primer and Reference Manual [M]. Addison-Wesley, 2003