ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.5, May 2010, pp.1007−1019
doi: 10.3724/SP.J.1001.2010.03531

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

演变图上的连接子图演变模式挖掘*

邹兆年+, 高 宏, 李建中, 张 硕

(哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

Mining Evolving Patterns of Connection Subgraphs over Evolving Graphs

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

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

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

Zou ZN, Gao H, Li JZ, Zhang S. Mining evolving patterns of connection subgraphs over evolving graphs. *Journal of Software*, 2010,21(5):1007−1019. http://www.jos.org.cn/1000-9825/3531.htm

Abstract: This paper investigates into the problem of mining evolving graphs, i.e. graphs changing over time. It focuses on mining evolving pattern set of connection subgraphs between given vertices in an evolving graph. A similarity function of connection subgraphs and the algorithm to compute it have been presented. Based on this similarity function, a dynamic programming algorithm with polynomial time complexity is proposed to find evolving pattern set. The experimental results on synthetic datasets show that the proposed algorithm has low error rate and high efficiency. The mining results on real datasets illustrate that the mining results have practical significance in real applications.

Key words: evolving graph; connection subgraph; evolving pattern

摘 要: 探讨演变图(即随时间变化的图)的挖掘,重点研究在演变图中挖掘连接子图的演变模式集合.提出一种连接子图的相似度函数及其快速计算算法.基于该相似度函数,提出一种发现演变模式集合的多项式时间复杂度的动态规划算法.模拟数据集上的实验结果表明,该算法具有较低的误差率和较高的效率.真实数据集上的实验结果表明,挖掘结果在真实应用中具有实际意义.

关键词: 演变图;连接子图;演变模式

中图法分类号: TP311 **文献标识码**: A

挖掘图数据(如无线传感器网络拓扑结构、化合物分子结构、社会网络等)是数据挖掘领域中的一项热点研究课题,简称图挖掘.其中,连接子图挖掘[1−3]是一个新的研究问题,近年来得到越来越多的关注.连接子图用于总结图中一组顶点之间的连接关系.它与顶点之间的距离和近邻度[3]等概念联系紧密.连接子图挖掘问题可以被形式化地描述为:给定图 $G=(V,E)$,一个非空顶点子集 $S\subseteq V$ 及一个子图质量函数 f,找出一个包含 S 的最小的 G

* Supported by the National Natural Science Foundation of China under Grant Nos.60773063, 60933001, 60903017 (国家自然科学基金); 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 National Natural Science Foundation of China and the Research Grants Council of Hong Kong of China under Grant No.60831160525 (国家自然基金委与香港资助局联合科研资助基金)

Received 2008-06-06; Accepted 2008-11-17

的连通子图 C,使得 $f(C)$最大.连接子图挖掘有很多实际应用.例如,在社会网络中,连接子图可以帮助医务人员找出易被某些疾病患者传染的个体.在学术合作网络中,连接子图可以帮助人们了解某领域内若干学者之间的合作关系.连接子图挖掘问题是 NP-完全问题[4].文献[1−3,5]提出了若干解决该问题的启发式算法.

现有的连接子图挖掘算法假定输入的图以及算法输出的连接子图均为静态图.然而,在许多实际应用中,图是随时间变化的,连接子图也是随时间变化的.我们将这种随时间变化的图称为演变图,并将演变图发生变化的时间域称为演变时间.由于现有的连接子图挖掘算法并非为演变图而设计,因此无法发现演变图中连接子图的演变模式.本文主要研究如何从演变图中发现一组顶点之间的连接子图的演变模式集合.演变模式集合是一组连接子图模式,其中每个连接子图模式代表了演变时间上的某一时间域内连接子图的共同结构特征.

连接子图演变模式挖掘有许多实际应用.近年来,无线传感器网络广泛应用于环境监测中.传感器节点感知到的数据通过传感器节点之间的无线链路以多跳的方式在传感器网络中传输.传感器网络的拓扑结构可以被建模为一个图,其中顶点表示传感器节点,边表示传感器节点之间的无线链路.由于传感器节点的移动性等原因,传感器网络的拓扑结构在传感器网络的生命期内会随时间而变化.图 1 给出一个无线传感器网络拓扑结构的例子.时间轴上方的图 G_1,G_2,G_3,G_4 表示无线传感器网络在不同时刻 t_1,t_2,t_3,t_4 的拓扑结构.

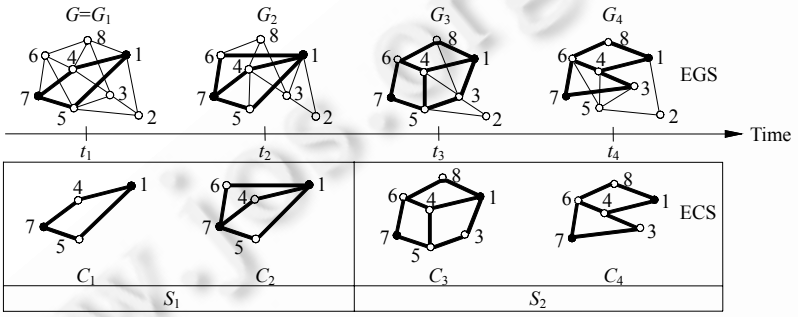


Fig.1 An example of evolving pattern set mining

图 1 演变模式集合挖掘实例

传感器网络的拓扑结构是影响路由算法、拓扑控制、网络覆盖的重要因素.在设计传感器网络时,科研人员经常对下列问题感兴趣:一组传感器节点之间的连接路径能够稳定维持多长时间?是否容易受到恶意攻击的破坏?随时间变化的趋势如何?这些问题可以根据如下有关传感器节点之间的连接子图演变模式集合的知识来解答:(1) 传感器节点之间的连接子图模式的数量;(2) 每个模式的具体结构;(3) 每个模式的持续时间.首先,通过分析每个模式的持续时间,我们能够得到一组传感器节点之间的连接路径稳定持续的时间.其次,若某种恶意攻击发生在某个演变模式的持续时间之内,即传感器节点之间的连接路径在这个时间段内保持稳定,则传感器网络的拓扑结构对于这种恶意攻击是健壮的.再次,通过对比各个模式之间的结构差异,我们能够了解传感器节点之间的连接路径随时间变化的规律.

例如,我们考察图 1 中的传感器节点 1 和节点 7.它们在 G_1,G_2,G_3,G_4 中的主要连接关系由时间轴下方的连接子图 C_1,C_2,C_3,C_4 给出.我们可以从图 1 中得到如下知识:由于 C_1 与 C_2 很相似,C_3 与 C_4 很相似,因此传感器节点 1 和节点 7 之间的连接关系可以被划分为两个阶段 S_1 和 S_2.节点 1 到节点 4 再到节点 7 的路径只在阶段 S_1 中存在,且持续时间为 t_2-t_1.若某种恶意攻击发生在 t_1 和 t_2 之间,则这种攻击对传感器网络拓扑结构的影响很小.若另一种攻击发生在 t_2 和 t_3 之间,则这种攻击很有可能破坏了传感器网络的拓扑结构,从而改变了节点 1 和节点 7 之间原有的连接子图模式.因此,设计人员可以根据这些知识来调整传感器网络的设计.

上述应用实例表明,演变图中连接子图的演变模式集合具有很重要的实际应用价值.因此,本文主要研究如何从演变图中发现一组顶点之间的连接子图的演变模式集合.演变模式集合给出如下知识:(1) 演变模式集合中模式的数量;(2) 演变模式集合中每个模式的持续时间;(3) 演变模式集合中每个模式的结构.本文将这个问题称为演变模式挖掘问题.

为解决演变模式挖掘问题,我们首先提出一种连接子图相似度函数以及计算该相似度函数的算法.该算法的时间复杂度为 $O(v\log v+e\log e)$,其中 v 和 e 分别是相似度函数的两个输入连接子图中的最大顶点数和最大边数.然后,基于连接子图相似度,我们提出一种发现连接子图的演变模式集合的动态规划方法.利用演变模式挖掘问题的重叠子结构,该动态规划算法可以在时间 $O(n^2)$内找到连接子图的演变模式集合,其中 n 为演变图在演变时间内演变得到的图的数量.我们在模拟数据集和真实数据集上进行了大量实验.实验结果表明,本文的算法具有很高的有效性、准确性和执行效率.

本文第 1 节介绍基本概念并给出演变模式挖掘问题的形式化定义.第 2 节提出一种解决演变模式挖掘问题的高效算法.第 3 节给出实验结果.第 4 节介绍相关工作.第 5 节给出本文的结论.

1 问题定义

定义 1. 设 $G=(V,E)$是一个图,其中 V 是顶点集合,E 是边集合.图 $G'=(V',E')$是一个由 G 演变得到的图(记作 $G\mapsto G'$)当且仅当:(1) $V'=V$;(2) $E'=E-E_1$ 或 $E'=E\cup E_2$,其中 $E_1\subseteq E,E_2\subseteq(V\times V)-E$.

直观上,G'是一个由 G 演变得到的图,若 G'是通过从 G 中删除一些边或者向 G 中添加一些边后得到的.

定义 2. 图 G 是一个在时间域 $T=[t_s,t_f]$上的演变图,如果 G 在 T 中任意两个时刻 t_i 和 $t_j(t_i<t_j)$的图 G_i 和 G_j 满足 $G_i\mapsto G_j$.时间域 T 称为 G 的演变时间.G 的演变图集合为 $EGS=\{\langle G_i,t_i\rangle|1\le i\le n,t_i\in T,t_s=t_1<t_2<...<t_n=t_f\}$,其中 $G_1=G,G_i\mapsto G_{i+1}(1\le i\le n-1)$.二元组$\langle G_i,t_i\rangle$表示 G 在时刻 t_i 演变为 G_i.

直观上,G 的演变图集合是 G 在时刻 $t_1,t_2,...,t_n$ 上的一系列快照.

定义 3. 设 $G=(V,E)$是一个图,$S\subseteq V$ 是 G 的一个非空顶点子集,$Q:X\to\mathrm{R}$ 是 G 的子图质量函数,其中 X 代表 G 的全部子图的集合.S 在 G 中的连接子图是一个由顶点子集 $V_C\subseteq V$ 导出的 G 的子图 $C=G[V_C]$,满足:(1) $S\subseteq V_C\subseteq V$;(2) C 是连通的;(3) $Q(C)=\max\{Q(x)|x$ 是满足条件(1)和(2)的 G 的导出子图$\}$.

连接子图质量函数是面向应用的.根据具体应用的需要,由用户给出具体的函数形式.文献[3,6]定义了几种具体的连接子图质量函数.

定义 4. 设 $G=(V,E)$是一个在演变时间 $T=[t_1,t_n]$上的演变图,$S\subseteq V$ 是 G 的一个非空顶点子集,且 $EGS=\{\langle G_i,t_i\rangle|1\le i\le n,t_i\in T\}$是 G 的演变图集合.S 在 G 中的演变连接子图集合定义为 $ECS=\{\langle C_i,t_i\rangle|1\le i\le n,t_i\in T\}$,其中,$C_i$ 是 S 在 G_i 中的连接子图.

定义 5. 设 G 是一个在演变时间 $T=[t_1,t_n]$上的演变图,$ECS=\{\langle C_i,t_i\rangle|1\le i\le n,t_i\in T\}$是顶点子集 S 在 G 中的演变连接子图集合.对于任意 $C_i,C_j\in ECS,C_i$ 与 C_j 的相似度是一个函数 $sim(C_i,C_j)$,满足性质:(1) $0\le sim(C_i,C_j)\le 1$;(2) $sim(C_i,C_j)=sim(C_j,C_i)$;(3) $sim(C_i,C_j)=1$ 当且仅当 $C_i=C_j$;(4) $sim(C_i,C_j)$越大,说明 C_i 与 C_j 越相似.

相似度函数 $sim(C_i,C_j)$的具体形式依赖于具体应用.我们将在第 2 节给出一种具体的相似度函数.

定义 6. 设 G 是一个在演变时间 $T=[t_1,t_n]$上的演变图,$ECS=\{\langle C_i,t_i\rangle|1\le i\le n,t_i\in T\}$是顶点子集 S 在 G 中的演变连接子图集合.$P=\{S_1,S_2,...,S_k\}$是 ECS 的一个划分当且仅当:(1) $S_i=\{\langle C_l,t_l\rangle,\langle C_{l+1},t_{l+1}\rangle,...,\langle C_{l+c},t_{l+c}\rangle\}$,其中,$1\le l\le n$,$l+c\le n$;(2) $S_i\cap S_j=\varnothing,\forall 1\le i<j\le k$;(3) $ECS=S_1\cup S_2\cup...\cup S_k$.$P$ 中每个元素 $S_i(1\le i\le k)$称为 ECS 的一个分段.

如果 ECS 的一个划分 P 满足如下性质:在同一分段内的连通子图非常相似,而不同分段内的连通子图非常不相似,则 P 中每个分段称为一个相似分段,P 称为相似分段集合.

定义 7. 设 G 是一个在演变时间 $T=[t_1,t_n]$上的演变图,$ECS=\{\langle C_i,t_i\rangle|1\le i\le n,t_i\in T\}$是顶点子集 S 在 G 中的演变连接子图集合,$SSS=\{S_1,S_2,...,S_k\}$是 ECS 的相似分段集合.ECS 的演变模式集合为 $EPS=\{\langle RC_i,T_i\rangle|1\le i\le k\}$,其中,$RC_i=C_{t_0}\in S_i$,满足 $\sum_{C_x\in S_i}sim(C_{t_0},C_x)=\max\left\{\sum_{C_x\in S_i}sim(C_t,C_x)\,|\,C_t\in S_i\right\}$,且 $T_i=[t_{i_s},t_{i_f}]\subseteq T$,其中,$t_{i_s}=\min\{t_k\,|\,\langle C_k,t_k\rangle\in S_i\}$,$t_{i_f}=\min\{t_k\,|\,\langle C_k,t_k\rangle\in S_i\}$.

在定义 7 中,RC_i 是 S_i 中与全部其他连接子图最相似的连接子图.因此,RC_i 可以作为 S_i 中连接子图的代表.

图 1 给出一个具体的例子来解释上述概念.图 1 中的时间轴表示演变图 G 的演变时间为 $T=[t_1,t_4]$.时间轴上方的图 G_1,G_2,G_3,G_4 分别是在时刻 t_1,t_2,t_3,t_4 由 G 演变得到的图,且 $G_1\mapsto G_2\mapsto G_3\mapsto G_4$.因此,$EGS=\{\langle G_1,t_1\rangle,\langle G_2,t_2\rangle,\langle G_3,t_3\rangle,\langle G_4,t_4\rangle\}$是 G 的一个演变图集合.时间轴下方的子图 C_1,C_2,C_3,C_4 分别是顶点子集 $S=\{1,7\}$在 G_1,G_2,G_3,G_4

上的连接子图.因此,EGS的演变连接子图集合为$ECS=\{\langle C_1,t_1\rangle,\langle C_2,t_2\rangle,\langle C_3,t_3\rangle,\langle C_4,t_4\rangle\}$.从图 1 可以看出,$C_1$与$C_2$很相似,而与 C_3和 C_4不相似;C_3与 C_4很相似,而与 C_1和 C_2不相似.因此,ECS 可以被划分为两个相似分段,即$S_1=\{\langle C_1,t_1\rangle,\langle C_2,t_2\rangle\}$和$S_2=\{\langle C_3,t_3\rangle,\langle C_4,t_4\rangle\}$,从而 ECS的相似分段集合为$SSS=\{S_1,S_2\}$.基于该相似分段集合,ECS的演变模式集合为$EPS=\{\langle C_1,[t_1,t_2]\rangle,\langle C_3,[t_3,t_4]\rangle\}$.

基于上述概念和定义,演变模式挖掘问题可以定义为下面的组合优化问题:

输入:图 G在演变时间$T=[t_1,t_n]$上的演变图集合$EGS=\{\langle G_i,t_i\rangle|1\le i\le n,t_i\in T\}$和 G的一个非空顶点子集 S.

输出:相似分段集合 $SSS=\{S_1,S_2,...,S_k\}$,使每个分段内连接子图之间的相似度最大化而不同分段内的连接子图之间的相似度最小化,并且从 SSS产生 S在 G中的连接子图的演变模式集合$EPS=\{\langle RC_i,T_i\rangle|1\le i\le k\}$.

2 演变模式挖掘算法

2.1 算法概述

给定演变图$G=(V,E)$在演变时间$T=[t_1,t_n]$上的演变图集合$EGS=\{\langle G_i,t_i\rangle|1\le i\le n,t_i\in T\}$和 G的一个非空顶点子集$S\subseteq V$.本文提出的挖掘算法通过下面 3 个阶段来发现 S在 G中的连接子图的演变模式集合.

阶段 1. 从 EGS中找到 S在 G中的演变连接子图集合$ECS=\{\langle C_i,t_i\rangle|1\le i\le n,t_i\in T\}$.

阶段 2. 使用具体的连接子图相似度函数计算 ECS中每对连接子图 C_i和 C_j的相似度 $sim(C_i,C_j)(1\le i<j\le n)$.

阶段 3. 利用 ECS中每对连接子图 C_i和 C_j的相似度 $sim(C_i,C_j)$,找出 ECS的最优相似分段集合 $SSS=\{S_1,S_2,...,S_k\}$,并构造演变模式集合$EPS=\{\langle RC_i,T_i\rangle|1\le i\le k\}$.

阶段 1 很简单,因为 ECS中每个连接子图 C_i可以在 G_i上应用现有的连接子图挖掘算法[1,2,5,7,8]得到.阶段 2 和阶段 3 是本文的重点.阶段 2 的困难在于如何定义具体的相似度函数以及如何快速计算两个连接子图的相似度.阶段 3 的困难在于如何快速找出最优相似分段集合.本文第 2.2 节给出了阶段 2 的解决方法;第 2.3 节提出了解决阶段 3 的算法;完整的演变模式挖掘算法在第 2.4 节给出.

2.2 连接子图的相似度

为度量图的相似度,已提出大量图的相似度度量,如图的编辑距离等,然而计算这些度量函数都是 NP-难问题[8].使用这些相似度函数会大大增加算法的复杂度.本节提出一种连接子图相似度函数及计算该相似度函数的多项式算法.该相似度函数不仅可以很好地反映连接子图的相似度,还可以大幅度提高计算效率.

定义 8. 图 $G_1=(V_1,E_1)$与 $G_2=(V_2,E_2)$的交是图$(V_i\cap V_j,E_i\cap E_j)$,记作 $G_1\odot G_2$.

定义 9. 设 $C_i=(V_i,E_i)$与 $C_j=(V_j,E_j)$是两个连接子图.C_i与 C_j的相似度为

$$sim(C_i,C_j)=\frac{|LCC|}{\max(|V_i|,|V_j|)} \tag{1}$$

其中,LCC是 $C_i\odot C_j$中的最大连通分支.

上述连接子图相似度函数满足定义 5 中指出的相似度度量的 4 条性质.

(1) 对任意连接子图 C_i和 C_j,$C_i\odot C_j$的最大连通分支 LCC中的顶点数满足$0\le|LCC|\le|V_i\cap V_j|\le\max(|V_i|,|V_j|)$.因此,$0\le sim(C_i,C_j)\le 1$.

(2) 对任意连接子图 C_i和 C_j,由于$C_i\odot C_j=C_j\odot C_i$,显然,$sim(C_i,C_j)=sim(C_j,C_i)$.

(3) 对任意连接子图 C,有$C\odot C=C$.根据定义 3,C是连通的,故 $LCC=C$,因此 $sim(C,C)=1$.

(4) 对任意连接子图 C_i和 C_j,$sim(C_i,C_j)$越大,表示 C_i与 C_j的最大公共部分在 C_i与 C_j中所占的比例越大,因此 C_i与 C_j越相似.

基于定义 9 中的相似度函数,这里提出计算两个连接子图相似度的算法(算法 1).算法 1 的输入是两个连接子图$C_i=(V_i,E_i)$和$C_j=(V_j,E_j)$.算法 1 的输出是 C_i与 C_j的相似度 $sim(C_i,C_j)$.算法第 1 行计算 C_i与 C_j的交 C.由于计算 $V_i\cap V_j$和 $E_i\cap E_j$分别需要时间 $O(v\log v)$和 $O(e\log e)$(其中 $v=\max(|V_i|,|V_j|)$,$e=\max(|E_i|,|E_j|)$),所以算法第 1 行需要时间 $O(v\log v+e\log e)$.算法第 2 行找出 $C=(V_C,E_C)$的最大连通分支 LCC.这一步可以使用 Tarjan 算法[9]在时间

$O(|V_C|+|E_C|)$内完成.算法第 3 行使用定义 9 计算 C_i 与 C_j 的相似度 $sim(C_i,C_j)$.由于$|V_C|\le|V_i\cap V_j|\le\max(|V_i|,|V_j|)=v$ 且 $|E_C|\le|E_i\cap E_j|\le\max(|E_i|,|E_j|)=e$,故算法 1 的时间复杂度为 $O(v\log v+e\log e)$.

算法 1. 计算连接子图的相似度.

输入:连接子图 $C_i=(V_i,E_i)$与 $C_j=(V_j,E_j)$.

输出:C_i 与 C_j 的相似度 $sim(C_i,C_j)$.

1. $C=C_i\odot C_j$.
2. $LCC=C$ 的最大连通分支.
3. 输出 $sim(C_i,C_j)=|LCC|/\max(|C_i|,|C_j|)$.

2.3 演变模式发现

算法的阶段 1 得到 S 在 G 中的演变连接子图集合 $ECS=\{\langle C_i,t_i\rangle|1\le i\le n\}$.算法的阶段 2 得到 ECS 中每对连接子图 C_i 与 C_j 的相似度 $sim(C_i,C_j)(1\le i<j\le n)$.利用 ECS 和这些相似度信息,我们提出一种从 ECS 中发现演变模式集合的动态规划算法.该算法包括 3 个步骤:

步骤 1. 根据每对连接子图的相似度,计算 ECS 的所有可能分段的段内相似度和段间相似度.

步骤 2. 利用 ECS 的所有可能分段的段内相似度和段间相似度,找出 ECS 的最优相似分段集合 SSS.

步骤 3. 基于 SSS,构造最终的演变模式集合 EPS.

下面给出这 3 个步骤的具体细节.

2.3.1 步骤 1(计算段内相似度与段间相似度)

我们将 $ECS=\{\langle C_i,t_i\rangle|1\le i\le n\}$的任意可能分段 $S_i=\{\langle C_l,t_l\rangle,\langle C_{l+1},t_{l+1}\rangle,...,\langle C_r,t_r\rangle\}$记作 $C_{l,r}$,其中,$1\le l\le r\le n$.S_i的段内相似度 $sim^{in}(S_i)$为

$$sim^{in}(S_i)=\frac{1}{\binom{|S_i|}{2}}\sum_{\langle C_x,t_x\rangle\in S_i,\langle C_y,t_y\rangle\in S_i,C_x\ne C_y}sim(C_x,C_y) \tag{2}$$

$sim^{in}(S_i)$表示 S_i 内所有连接子图之间的平均相似度.S_i 的段间相似度 $sim^{out}(S_i)$为

$$sim^{out}(S_i)=\frac{1}{|S_i|(n-|S_i|)}\sum_{\langle C_x,t_x\rangle\in S_i,\langle C_y,t_y\rangle\notin S_i}sim(C_x,C_y) \tag{3}$$

$sim^{out}(S_i)$表示所有在 S_i 中的连接子图与所有不在 S_i 中的连接子图之间的平均相似度.根据上述定义,$sim^{in}(C_{i,i})=0$ $(1\le i\le n)$,$sim^{out}(C_{1,n})=0$.这些特殊情况下的值与直观不相符,并且会导致错误的相似分段集合.为修正该问题,算法规定 $sim^{in}(C_{i,i})=sim^{in}(C_{1,n})(1\le i\le n)$,$sim^{out}(C_{1,n})=sim^{in}(C_{1,n})$.

直接根据公式(2)计算 $sim^{in}(C_{i,j})$需要的时间 $O\left(\binom{j-i+1}{2}\right)$.直接根据公式(3)计算 $sim^{out}(C_{i,j})$需要的时间 $O((j-i+1)(n-j+i-1))$.因此,步骤 1 总共需要时间 $O\left(\sum_{1\le i\le j\le n}\left(\binom{j-i+1}{2}+(j-i+1)(n-j+i-1)\right)\right)=O(n^4)$. 计算全部 $sim^{in}(C_{i,j})$与 $sim^{out}(C_{i,j})$的值$(1\le i\le j\le n)$.当 n 很大时,这种直接的计算方法效率很低.利用下面两个关系,我们提出一种时间复杂度为 $O(n^2)$的算法(算法 2)来计算全部 $sim^{in}(C_{i,j})$和 $sim^{out}(C_{i,j})$的值$(1\le i\le j\le n)$.

关系 1.

$$sim^{in}(C_{i,j})=\frac{1}{|C_{i,j}|}\left[sim^{in}(C_{i,j-1})\binom{|C_{i,j-1}|}{2}+sim^{in}(C_{i+1,j})\binom{|C_{i+1,j}|}{2}+sim(C_i,C_j)-sim^{in}(C_{i+1,j-1})\binom{|C_{i+1,j-1}|}{2}\right] \tag{4}$$

关系 2.

$$\begin{aligned}sim^{out}(C_{i,j})=&\frac{1}{|C_{i,j}|(n-|C_{i,j}|)}\Big[sim^{out}(C_{j,j})(n-1)+sim^{out}(C_{i,j-1})|C_{i,j-1}|(n-|C_{i,j-1}|)+\\&2\cdot sim^{in}(C_{i,j-1})\binom{|C_{i,j-1}|}{2}-2\cdot sim^{in}(C_{i,j})\binom{|C_{i,j}|}{2}\Big]\end{aligned} \tag{5}$$

算法 2. 计算段内相似度与段间相似度.

输入:演变连接子图集合 $ECS=\{\langle C_i,t_i\rangle|1\le i\le n\}$和 ECS 中每对连接子图 C_i 和 C_j 的相似度 $sim(C_i,C_j)$.

输出:$sim^{in}(C_{i,j})$与 $sim^{out}(C_{i,j})(1\le i\le j\le n)$.

1. **for** i=1 **to** n **do**
2. $sim\text{-}in[i,i]=0$.
3. $sim\text{-}out[i,i]=\frac{1}{n-1}\sum_{1\le j\le n,j\ne i}sim(C_i,C_j)$.
4. **for** i=1 **to** n−1 **do**
5. $sim\text{-}in[i,i+1]=sim(C_i,C_{i+1})$.
6. **for** k=2 **to** n−1 **do**
7. **for** i=1 **to** n−k **do**
8. $sim\text{-}in[i,i+k]=sim^{in}(C_{i,i+k})$,其中 $sim^{in}(C_{i,i+k})$由公式(4)计算得到.
9. **for** k=1 **to** n−1 **do**
10. **for** i=1 **to** n−k **do**
11. $sim\text{-}out[i,i+k]=sim^{out}(C_{i,i+k})$,其中 $sim^{out}(C_{i,i+k})$由公式(5)计算得到.
12. **for** i=1 **to** n **do**
13. $sim\text{-}in[i,i]=sim\text{-}in[1,n]$.
14. $sim\text{-}out[1,n]=sim\text{-}in[1,n]$.
15. **return** 存放在数组 *sim-in* 和 *sim-out* 中的结果 $sim^{in}(C_{i,j})$与 $sim^{out}(C_{i,j})(1\le i\le j\le n)$.

在算法 2 中,所有 $sim^{in}(C_{i,j})$和 $sim^{out}(C_{i,j})$的值被分别存放在两个二维数组 *sim-in*[1..*n*,1..*n*]和 *sim-out*[1..*n*,1..*n*]中,其中,$sim^{in}(C_{i,j})$的值存放在数组元素 *sim-in*[*i*,*j*]中,$sim^{out}(C_{i,j})$的值存放在数组元素 *sim-out*[*i*,*j*]中.算法的基本思想是按照 $k=j-i$ 递增的顺序来计算所有 $sim^{in}(C_{i,j})$与 $sim^{out}(C_{i,j})$的值.给定演变连接子图集合 $ECS=\{\langle C_i,t_i\rangle|1\le i\le n\}$和 ECS 中每对连接子图 C_i 和 C_j 的相似度 $sim(C_i,C_j)$,算法 2 的具体过程如下:首先,算法的第 1 行~第 3 行对数组元素 *sim-in*[*i*,*i*]和 *sim-out*[*i*,*i*]进行初始化($1\le i\le n$),其中,*sim-in*[*i*,*i*]被初始化为 0,*sim-out*[*i*,*i*]被初始化为 $\sum_{1\le j\le n,j\ne i}sim(C_i,C_j)\big/(n-1)$. 然后,第 4 行和第 5 行将数组元素 *sim-in*[*i*,*i*+1]初始化为 C_i 和 C_{i+1} 的相似度 $sim(C_i,C_{i+1})(1\le i\le n-1)$.接下来,在第 6 行~第 8 行,按照 $k=j-i$ 从 2 到 n−1 的顺序使用公式(4)计算 $sim^{in}(C_{i,j})$的值并存放在 *sim-in*[*i*,*j*]($1\le i\le j\le n,j=i+k$)中.在第 9 行~第 11 行,按照 $k=j-i$ 从 1 到 n−1 的顺序使用公式(5)计算 $sim^{out}(C_{i,j})$的值并存放在 *sim-out*[*i*,*j*]($1\le i\le j\le n,j=i+k$)中.在第 12 行~第 14 行,将 $sim^{in}(C_{i,i})(1\le i\le n)$和 $sim^{out}(C_{1,n})$的值全部重新赋值为 *sim-in*[1,*n*].最后,第 15 行输出存放在数组 *sim-in* 和 *sim-out* 中的结果 $sim^{in}(C_{i,j})$与 $sim^{out}(C_{i,j})(1\le i\le j\le n)$.

算法的第 1 行~第 3 行需要时间 $O(n^2)$.第 4 行和第 5 行需要时间 $O(n)$.利用关系 1 和关系 2,每个 $sim^{in}(C_{i,j})$和 $sim^{out}(C_{i,j})$的值可以分别在第 8 行和第 11 行进行计算,其时间复杂度均为 $O(1)$.由于 ECS 总共有$(n^2+n)/2$ 个可能分段,故第 6 行~第 11 行的时间复杂度为 $O(n^2)$.最后,第 12 行~第 14 行需要时间 $O(n)$.因此,算法 2 的时间复杂度为 $O(n^2)$.

2.3.2 步骤 2(寻找最优相似分段集合)

利用步骤 1 计算得到的 ECS 的所有可能分段的段内相似度与段间相似度,步骤 2 寻找演变连接子图集合 $ECS=\{\langle C_i,t_i\rangle|1\le i\le n\}$的相似分段集合 $SSS=\{S_1,S_2,...,S_k\}$,使得下面的 Badness 函数值最小化:

$$Badness(\{S_1,S_2,\ldots,S_k\})=\sum_{i=1}^{k}\left(\frac{sim^{out}(S_i)}{sim^{in}(S_i)}\right)^{\alpha} \tag{6}$$

在公式(6)中,$\left(\frac{sim^{out}(S_i)}{sim^{in}(S_i)}\right)^{\alpha}$ 表示相似分段 S_i 的质量损失,称为 S_i 的 Badness 值,并记作 $Badness(S_i)$.当 $sim^{in}(S_i)$越大(即 S_i 内连接子图之间的相似度越大)而 $sim^{out}(S_i)$越小(即 S_i 内的连接子图与 S_i 外的连接子图之间的相似度越小)时,$Badness(S_i)$越小,表示 S_i 的分段质量越高.反之,$Badness(S_i)$越大,表示 S_i 的分段质量越低.同理,

$Badness(\{S_1,S_2,...,S_k\})$描述了相似分段集合$\{S_1,S_2,...,S_k\}$的质量损失,称为$\{S_1,S_2,...,S_k\}$的 Badness 值.相似分段集合的质量越高,其 Badness 值越小.在公式(6)中,α是一个正实数,用来控制输出的演变模式集合的解析度,即较大的α值会导致相似分段数量的增加,从而能够分辨出分段之间的细微差别.α的作用还将在第 4 节中进行说明.

一种解决该问题的简单方法是枚举 ECS 的所有可能相似分段集合,并从中找出 Badness 值最小的相似分段集合.由于这种简单的方法需要枚举 ECS 中的 2^{n-1} 个可能的相似分段集合,其时间复杂度为 $O(2^n)$.显然,这种简单的枚举方法不可行.我们给出一种时间复杂度为 $O(n^2)$的动态规划算法.

首先,分析相似分段集合发现问题的优化解的子结构.我们将相似分段集合发现问题的最优解称为最优相似分段集合,并将$\{\langle C_i,t_i\rangle|1\le i\le j\}$记作$ECS(j)$.显然,$ECS(n)=ECS$.假设$SSS=\{S_1,S_2,...,S_k\}$是$ECS$的最优相似分段集合且 SSS 的最后一个分段是 $S_k=\{C_i,C_{i+1},...,C_n\}$(如图 2 所示).$\{S_1,S_2,...,S_{k-1}\}$必然是 $ECS(i-1)$的最优相似分段集合.否则,必存在某个 $ECS(i-1)$的相似分段集合 SSS'比$\{S_1,S_2,...,S_{k-1}\}$具有更小的 Badness 值.因此,$SSS'\cup\{S_k\}$将是比 SSS 具有更小的 Badness 值的 ECS 的相似分段集合.这与 SSS 是 ECS 的最优相似分段集合矛盾.

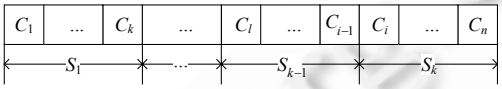


Fig.2 Structure of the optimal similar segment set

图 2 最优相似分段集合的结构

根据最优相似分段集合的优化子结构,我们得到计算 ECS 的最优相似分段集合的 Badness 值的递归方程.设 $Badness(j)$表示 $ECS(j)$的最优相似分段集合的 Badness 值.当 $j=0$ 时,$ECS(0)=\varnothing$,因此 $Badness(0)=0$.当 $j>0$ 时,若 $C_{i,j}$ 是 $ECS(j)$的最优相似分段集合中的最后一个相似分段,则 $Badness(j)=Badness(i-1)+Badness(C_{i,j})$.由于 i 总共有 j 个可能的取值$\{1,2,...,j\}$,因此 $Badness(j)$的递归方程为

$$Badness(j)=\begin{cases}0, & \text{if } j=0\\ \min\limits_{1\le i\le j}(Badness(i-1)+Badness(C_{i,j})), & \text{if } j>0\end{cases} \tag{7}$$

基于该递归方程,我们提出一种计算 ECS 的最优相似分段集合的 Badness 值的动态规划算法(算法 3)以及构造 ECS 的最优相似分段集合的算法(算法 4).算法 3 的基本思想是按照 j 从 1 到 n 的顺序使用方程(7)计算 $Badness(j)$.为构造ECS的最优相似分段集合,算法3在计算$Badness(j)$的同时使用一个数组$s[1..n]$存储相关信息.当 i 满足 $Badness(j)=Badness(i-1)+Badness(C_{i,j})$时,即当 i 是 $ECS(j)$的最优相似分段集合中最后一个分段中的第一个连接子图的下标时,我们将 i 存放在数组元素 $s[j]$中.给定演变连接子图集合 $ECS=\{\langle C_i,t_i\rangle|1\le i\le n\}$,段内相似度 $sim^{in}(C_{i,j})(1\le i\le j\le n)$,段间相似度 $sim^{out}(C_{i,j})(1\le i\le j\le n)$和正实数$\alpha$,算法 3 的工作过程如下:首先,算法的第 1 行初始化 $Badness(0)$为 0.然后,第 2 行~第 6 行按照 j 从 1 到 n 的顺序计算 $Badness(j)$并维护数组 $s[1..n]$.最后,第 7 行返回 $Badness(n)$和数组 s.易证算法 3 的时间复杂度为 $O(n^2)$.

算法 3. 计算最优相似分段集合的 Badness 值.

输入:演变连接子图集合 $ECS=\{\langle C_i,t_i\rangle|1\le i\le n\}$,段内相似度 $sim^{in}(C_{i,j})(1\le i\le j\le n)$,段间相似度 $sim^{out}(C_{i,j})(1\le i\le j\le n)$,正实数 α.

输出:$adness(n)$和数组 $s[1..n]$.

```
1.  Badness(0)=0.
2.  for j=1 to n do
3.    for i=1 to j do
4.      if Badness(j)>Badness(i−1)+Badness(C_{i,j}) then
5.        Badness(j)=Badness(i−1)+Badness(C_{i,j}).
6.        s[j]=i.
7.  return Badness(n) and s.
```

算法 4. 构造最优相似分段集合.

输入:数组 $s[1..n]$.

输出:*ECS* 的最优相似分段集合.

1. 初始化栈 *Stack* 为空.
2. $t=n$.
3. **while** $t>0$ **do**
4. 将分段$\{\langle C_{s[t]},t_{s[t]}\rangle,\langle C_{s[t]+1},t_{s[t]+1}\rangle,...,\langle C_t,t_t\rangle\}$压入栈 *Stack*.
5. $t=s[t]-1$.
6. **while** *Stack* 不为空 **do**
7. 输出 *Stack* 栈顶的最优相似分段.
8. *Pop*(*Stack*).

由于 $s[j]$存放了 *ECS*(*j*)的最优相似分段集合中最后一个分段的第 1 个连接子图的下标,*ECS*(*j*)的最优相似分段集合中最后一个分段是$\{\langle C_{s[j]},t_{s[j]}\rangle,\langle C_{s[j]+1},t_{s[j]+1}\rangle,...,\langle C_j,t_j\rangle\}$.算法 4 通过从 $s[n]$到 $s[1]$反向迭代访问数组 *s* 得到 *ECS* 的最优相似分段集合.给定数组 *s*,算法 4 的过程如下:首先,算法的第 1 行初始化一个空的栈 *Stack*.然后,第 2 行~第 5 行从 $s[n]$开始反向访问数组 *s*,找到 *ECS*(*t*)的最优相似分段集合中的最后一个分段$\{\langle C_{s[t]},t_{s[t]}\rangle,\langle C_{s[t]+1},t_{s[t]+1}\rangle,...,\langle C_t,t_t\rangle\}$并将其压入栈中.最后,第 6 行~第 8 行依次弹出栈中的分段.若 *ECS* 的最优相似分段集合的分段数为 *k*,则算法 4 总共需要对 *s* 进行 *k* 次迭代访问.由于 $k\le n$,算法 4 的时间复杂度为 $O(n)$.

综合算法 3 和算法 4,步骤 2 可以在 $O(n^2)$时间内输出 *ECS* 的最优相似分段集合.

2.3.3 步骤 3(构造演变模式集合)

在步骤 2 得到最优相似分段集合 $SSS=\{S_1,S_2,...,S_k\}$的基础之上,步骤 3 从 *SSS* 中提取演变模式集合 $EPS=\{\langle RC_i,T_i\rangle|1\le i\le k\}$,其中,$RC_i$ 是 S_i 中满足$\sum_{C_x\in S_i} sim(C_{t_0},C_x)=\max\left\{\sum_{C_x\in S_i} sim(C_t,C_x)\,|\,C_t\in S_i\right\}$的连接子图 C_{t_0},并且 $T_i=[t_{i_s},t_{i_f}]$ 是分段 S_i 的时间区间,其中,$t_{i_s}=\min\{t_k\,|\,\langle C_k,t_k\rangle\in S_i\}$,$t_{i_f}=\max\{t_k\,|\,\langle C_k,t_k\rangle\in S_i\}$.

综上所述,算法的阶段 3 可以在时间 $O(n^2)$内找到 *ECS* 的最优相似分段集合和演变模式集合.

2.4 完整算法

综合第 2.1 节~第 2.3 节的算法,本节给出从演变图中挖掘连接子图演变模式集合的完整算法(算法 5).给定演变图 *G* 在演变时间 $T=[t_1,t_n]$上的演变图集合 $EGS=\{\langle G_i,t_i\rangle|1\le i\le n,t_i\in T\}$和 *G* 的一个非空顶点子集 *S*,算法过程如下:首先,对 *EGS* 中的每个图 G_i $(1\le i\le n)$,算法第 2 行使用现有的连接子图挖掘算法 CePS[5]从 G_i 中挖掘 *S* 在 G_i 上的连接子图 C_i,从而找到 *S* 在 *G* 中的演变连接子图集合 $ECS=\{\langle C_i,t_i\rangle|1\le i\le n,t_i\in T\}$.然后,对 *ECS* 中的每对连接子图 C_i 和 C_j,第 4 行使用算法 1 计算 C_i 与 C_j 的相似度.接下来,第 5 行使用算法 2 计算 *ECS* 的所有可能分段的段内相似度 $sim^{in}(C_{i,j})$与段间相似度 $sim^{out}(C_{i,j})(1\le i\le j\le n)$.第 6 行和第 7 行使用算法 3 和算法 4 计算 *ECS* 的最优相似分段集合 $SSS=\{S_1,S_2,...,S_k\}$.最后,第 8 行~第 10 行构造最终的演变模式集合 *EPS*.

算法 5. 演变模式挖掘算法.

输入:演变图集合 $EGS=\{\langle G_i,t_i\rangle|1\le i\le n\}$及顶点子集 *S*.

输出:最优演变模式集合.

1. **for** $i=1$ **to** n **do**
2. 使用 CePS 算法寻找 *S* 在 G_i 中的连接子图 C_i.
3. **for** $1\le i\le j\le n$ **do**
4. 使用算法 1 计算 C_i 与 C_j 的相似度 $sim(C_i,C_j)$.
5. 使用算法 2 计算所有 $sim^{in}(C_{i,j})$与 $sim^{out}(C_{i,j})(1\le i\le j\le n)$.
6. 使用算法 3 计算 *Badness*(*j*)与数组 $s[1..n]$.
7. 使用算法 4 构造 *ECS* 的最优相似分段集合 $SSS=\{S_1,S_2,...,S_k\}$.

8. **for** i=1 **to** k **do**

9. T_i=[t_l,t_r],其中 S_i={C_l,C_{l+1},...,C_r}.

10. $RC_i = C_{t_0}$ 使得 $C_{t_0} \in S_i$且$\sum_{C_x \in S_i} sim(C_{t_0}, C_x) = \max\left\{\sum_{C_x \in S_i} sim(C_t, C_x) \mid C_t \in S_i\right\}$.

11. **return** EPS={⟨RC_i,T_i⟩|1≤i≤k}.

下面分析算法 5 的时间复杂度.由于 CePS 算法的时间复杂度为 $O(|E|)$[3],因此算法的第 1 行和第 2 行需要时间 $O\left(\sum_{i=1}^{n}|E_i|\right)$.第 3 行和第 4 行需要时间 $O(n^2(v\log v+e\log e))$,其中,v=max{$|V_{C_i}||1\le i\le n$}且 e=max{$|E_{C_i}||1\le i\le n$}.如上所述,第 5 行和第 6 行的时间复杂度均为 $O(n^2)$,第 7 行的时间复杂度为 $O(n)$.由于$|S_i|$>0,第 8 行~第 10 行需要时间 $O\left(\sum_{i=1}^{k}|S_i|^2\right)\le O\left(\left(\sum_{i=1}^{k}|S_i|\right)^2\right)=O(n^2)$.因此,算法 5 的时间复杂度为 $O\left(\sum_{i=1}^{n}|E_i|+n^2(v\log v+e\log e)\right)$.

3 实验结果

3.1 模拟数据集上的实验结果

本节给出本文算法在模拟数据集上的误差率和效率的实验结果.根据上述讨论,我们知道算法的阶段 2 和阶段 3 是本文算法的重点.它们影响了算法的误差率和效率.因此,在模拟数据集上的实验中,我们假定已得到算法阶段 1 找到的演变连接子图集合 ECS,并仅对算法阶段 2 和阶段 3 在 ECS 上的误差率和效率进行考察.

首先,我们考察算法的误差率.给定演变连接子图集合 ECS,设 $\{S_1^*,S_2^*,...,S_m^*\}$ 是没有任何误差的、准确的 ECS 的相似分段集合,设$\{S_1,S_2,...,S_k\}$是由本文算法得到的 ECS 的相似分段集合.算法的误差率为

$$ErrorRate=\frac{\sum_{1\le i\le j\le n}\left|gap(C_i,C_j)-gap'(C_i,C_j)\right|}{\max\left[\sum_{1\le i\le j\le n}\left|\min(k-1,j-i)-gap(C_i,C_j)\right|,\sum_{1\le i\le j\le n}\left|gap(C_i,C_j)\right|\right]} \tag{8}$$

其中,$gap(C_i,C_j)$=|q−p|,若 $C_i\in S_p^*,C_j\in S_q^*,S_p^*,S_q^*\in\{S_1^*,S_2^*,...,S_m^*\}$, 而 $gap'(C_i,C_j)$=|q'−p'|,若 $C_i\in S_{p'},C_j\in S_{q'},S_{p'},S_{q'}\in\{S_1,S_2,...,S_k\}$.公式(8)的分子表示 $\{S_1^*,S_2^*,...,S_m^*\}$ 与 $\{S_1,S_2,...,S_k\}$的绝对误差,分母表示 $\{S_1^*,S_2^*,...,S_m^*\}$ 与任意可能相似分段集合的最大绝对误差.因此,误差率表示 $\{S_1^*,S_2^*,...,S_m^*\}$ 与$\{S_1,S_2,...,S_k\}$的归一化误差.影响算法误差率的因素是 Badness 函数中的参数α.后面将给出算法误差率关于α的实验结果.

其次,我们考察算法的效率,即算法的执行时间.影响算法效率的因素有 3 个:演变连接子图集合中连接子图的数量、连接子图中顶点的数量和连接子图中边的数量.后面将给出算法效率关于这 3 个因素的实验结果.

我们使用 C 语言实现了本文的算法,并进行了大量的实验.用于实验的 PC 机有 Intel Pentium IV 3.0GHz 处理器,512MB 内存,运行 Windows XP 操作系统.

3.1.1 模拟数据集

实验中使用的模拟数据集包含 n 个连接子图{$C_1,C_2,...,C_n$},并具有 k 个相似分段{$S_1,S_2,...,S_k$},其中,每个相似分段内的连接子图非常相似,而不同分段内的连接子图很不相似.模拟数据集有 6 个参数(见表 1).给定一组参数,按照如下方式产生数据集中各个分段 $S_1,S_2,...,S_k$:对于每个分段 $S_i(1\le i\le k)$,首先确定 S_i 内连接子图的数量 $|S_i|$.$|S_i|$服从均值为 n/k 的泊松分布.同时,使用 Erdos-Ranyi 随机图模型生成一个图 $Seed_i=(V_i,E_i)$作为生成 S_i 中连接子图拓扑结构的种子,其中,$|V_i|$服从均值为 μ_v 的泊松分布,$|E_i|$服从均值为 μ_e 的泊松分布,V_i 包含 S,其余顶点 V_i−S 从 V_C−S 中随机选取.接下来,分段 S_i 中每个连接子图 C_j 通过如下过程产生:首先初始化 C_j=$Seed_i$;然后对任意 $u,v\in V_{C_j}$,如果边$(u,v)\in E_{C_j}$,则以概率 P 删除(u,v),否则,以概率 P 添加边(u,v).在得到模拟数据集之后,可将生成的分段集合$\{S_1,S_2,...,S_k\}$作为没有任何误差的真实的相似分段集合来考察演变模式挖掘算法的误差率.

Table 1 Parameters of synthetic dataset

表 1 模拟数据集的参数

Parameter	Description
n	Number of connection subgraphs
k	Number of segments
μ_v	Average number of vertices in connection subgraphs
μ_e	Average number of edges in connection subgraphs
S	Input vertex subset
V_C	Set of candidate vertices in connection subgraphs
P	Probability to add or delete edges

3.1.2 算法误差率的实验结果

实验 1 考察算法在不同数据集上的误差率.实验中使用 5 组模拟数据集:$D_1,D_2,...,D_5$.每组数据集 D_i 包含 1 000 个随机生成的演变连接子图集合,其中每个演变连接子图集合有相同的参数.$D_1,D_2,...,D_5$ 中演变连接子图集合的参数为 n=100,μ_v=10,μ_e=20,S={s_1,s_2},V_C={$v_1,v_2,...,v_{40}$},P=5%,k 分别为 4,6,8,12 和 16.表 2 给出算法误差率实验结果.实验结果显示,对于每组模拟数据集,算法平均误差率的最小值(表 2 中用下划线标注)在 0.039 5~0.151 8 之间,远低于随机相似分段集合的平均误差率 0.5.这说明本文算法的误差率很低.

实验 2 考察 Badness 函数的参数α对算法误差率的影响.实验使用的数据集与实验 1 中的数据集相同.图 3 给出当α从 1 变化到 1 000 时算法分别在数据集 $D_1,D_2,...,D_5$ 上的平均误差率.随着α的增加,平均误差率首先快速下降.当α到达一定值(约为 10)时,平均误差率变得很低.当α继续增加并超过某值(约为 500)时,平均误差率开始增加.其原因在于α可以控制算法结果的解析度.当α很小时,算法结果的解析度很低,分段数少于真实的分段数,造成结果与真实分段集合的差异较大,故误差率很高.当α很大时,算法结果的解析度很高,分段数大于真实的分段数,同样导致误差率较高.实验结果说明α能够控制算法得到的演变模式集合的解析度.

Table 2 Average error rate on five groups of datasets

表 2 5 组数据集上的平均误差率

Dataset	k	α					
		1	10	50	100	500	1 000
D_1	4	0.695 4	0.157 1	0.151 8	0.151 8	0.185 9	0.639
D_2	6	0.855 6	0.170 5	0.128 3	0.133 6	0.159 3	0.738 1
D_3	8	0.901 4	0.099 9	0.081 8	0.082	0.098 7	0.676 9
D_4	12	0.962 6	0.067 9	0.048 8	0.049 3	0.068 7	0.647 8
D_5	16	0.980 9	0.054 4	0.039 5	0.039 8	0.065	0.656 3

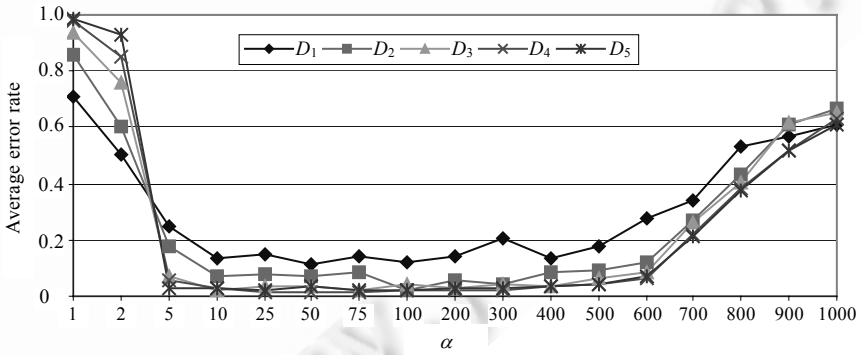


Fig.3 Average error rate of the algorithm with respect to parameter α in Badness function

图 3 Badness 函数中的参数α发生变化的情况下算法的平均误差率

3.1.3 算法效率的实验结果

实验 3 考察连接子图数量对算法执行时间的影响.实验使用 100 个随机生成的演变连接子图集合,其参数均为 k=10,μ_v=10,μ_e=20,S={s_1,s_2},V_C={$v_1,v_2,...,v_{40}$},P=5%,n 在实验中变化.实验中,α=100.图 3(a)给出当 n 从 100 变化到 1 000 时,算法在 100 个演变连接子图集合上的平均执行时间.实验结果表明,算法的平均执行时间随着 n

的增加而增大.这是因为算法阶段 2 和阶段 3 的时间复杂度为 $O(n^2(v\log v+e\log e))$.

实验 4 考察连接子图平均顶点数对算法执行时间的影响.实验使用 100 个随机生成的演变连接子图集合,其参数均为 n=100,k=10,P=5%,S={s_1,s_2},|V_C|=4μ_v,μ_e=2μ_v,μ_v 在实验中变化.实验中,α=100.图 4(b)给出当 μ_v 从 10 变化到 20 时,算法在 100 个演变连接子图集合上的平均执行时间.实验结果表明,算法的平均执行时间随着 μ_v 的增加而增加.这是因为算法阶段 2 和阶段 3 的时间复杂度为 $O(n^2(v\log v+e\log e))$,其中 v 是连接子图的最大顶点数,与 μ_v 的大小直接相关.

实验 5 考察连接子图平均边数对算法执行时间的影响.实验使用 100 个随机生成的演变连接子图集合,其参数均为 n=50,k=10,μ_v=10,P=5%,S={s_1,s_2},|V_C|=40,μ_e 在实验中变化.实验中,α=100.图 4(c)给出当 μ_e 从 15 变化到 40 时,算法在 100 个演变连接子图集合上的平均执行时间.实验结果表明,算法的平均执行时间随着 μ_e 的增加而增加.这是因为算法阶段 2 和阶段 3 的时间复杂度为 $O(n^2(v\log v+e\log e))$,其中 e 是连接子图的最大顶点数,与 μ_e 的大小直接相关.

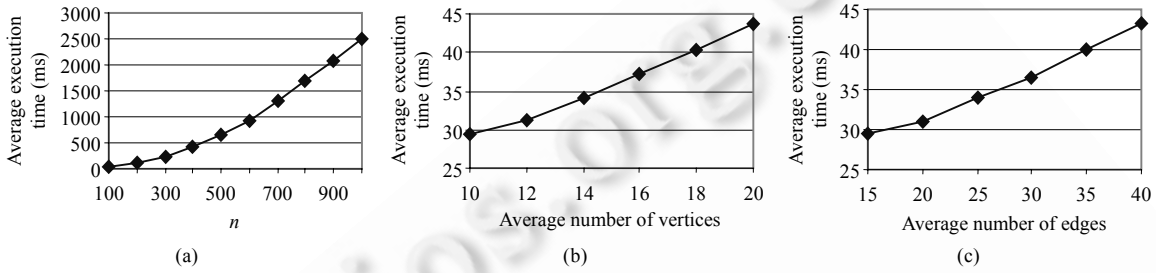


Fig.4 Average execution time of the proposed algorithm

图 4 本文算法的平均执行时间

3.2 真实数据上的实验结果

为了考察本文算法在实际应用中的有效性,我们在 Enron 电子邮件数据集上进行了实验.该数据集包含 1998 年 10 月 30 日~2002 年 9 月 22 日期间,151 个用户的 252 759 封电子邮件.我们从该数据集中提取了 205 个电子邮件通信图.每个图分别取自一个星期内的电子邮件数据.在星期 $week_i$ 提取的图 G_i=(V_i,E_i)中,V_i 表示电子邮件数据集中出现的全部 30 123 个电子邮件地址,E_i={(v_1,v_2)|在 $week_i$ 中,v_1 发信给 v_2,或者 v_2 发信给 v_1}.

实验考察演变时间 T=[$week_{58}$,$week_{179}$]上的演变图集合 EGS={⟨G_i,$week_i$⟩|58≤i≤179}(因为这段时间内的电子邮件通信非常活跃)、顶点子集 S={andrew.fastow@enron.com(Andrew Fastow,Enron 公司前首席财政官),jeff.skilling@enron.com(Jeffrey Skilling,Enron 公司前总裁兼 CEO)}、Badness 函数的参数α=3、连接子图中最大顶点数=10(算法 CePS 所需的输入参数[5]).演变模式挖掘算法返回的结果如下:

$$SSS=\{\{\langle C_{58},t_{58}\rangle,\langle C_{59},t_{59}\rangle,...,\langle C_{95},t_{95}\rangle\},\{\langle C_{96},t_{96}\rangle,\langle C_{97},t_{97}\rangle,...,\langle C_{148},t_{148}\rangle\},$$
$$\{\langle C_{149},t_{149}\rangle,\langle C_{150},t_{150}\rangle,...,\langle C_{157},t_{157}\rangle\},\{\langle C_{158},t_{158}\rangle,\langle C_{159},t_{159}\rangle,...,\langle C_{179},t_{179}\rangle\}\},$$

EPS={⟨C_{58},[$week_{58}$,$week_{95}$]⟩,⟨C_{119},[$week_{96}$,$week_{148}$]⟩,⟨C_{155},[$week_{149}$,$week_{157}$]⟩,⟨C_{158},[$week_{158}$,$week_{179}$]⟩}.

下面结合 Enron 公司案件调查过程中的一些重要事件来说明上述挖掘结果具有实际意义.

(1) 在时间段 T_1=[$week_{58}$,$week_{95}$]内,Fastow 与 Skilling 的大量旧的电子邮件被删除,导致数据不完整,从而在 T_1 内仅发现很少量连接子图.因此,将 T_1 划分为一个分段.

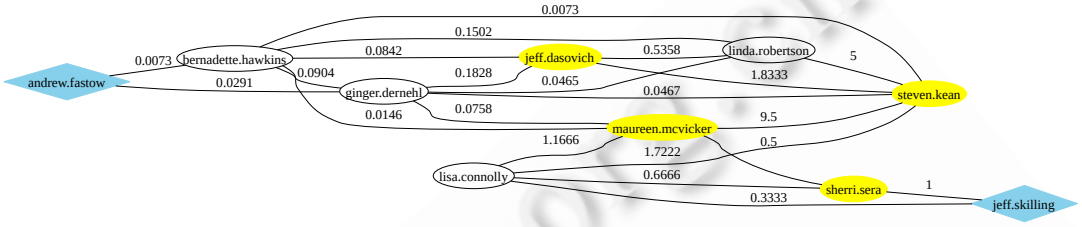
(2) 与 T_1 相比,在 T_2=[$week_{96}$,$week_{148}$]内的每个图中都可以发现 Fastow 与 Skilling 之间的连接子图,并且连接子图中的节点主要是 Enron 公司的管理者,如 Steven Kean(Enron 公司前执行副总裁),Kenneth Lay(Enron 公司前董事会主席,CEO)等.事实上,这段时间正值 Enron 公司财务危机爆发,高层管理者私人通信频繁.另外,在 2000 年 8 月 23 日(即 T_1 和 T_2 的交界处),Enron 公司股价达到每股 90.75 美元的最高点.这在案件调查过程中被视作财务危机爆发的重要标志.因此,将 T_1 和 T_2 划分为两个分段是符合事实的.

(3) 在 T_3=[$week_{149}$,$week_{157}$]中,Fastow 与 Skilling 之间的连接子图中的节点由管理者变为普通员工,这与 T_2

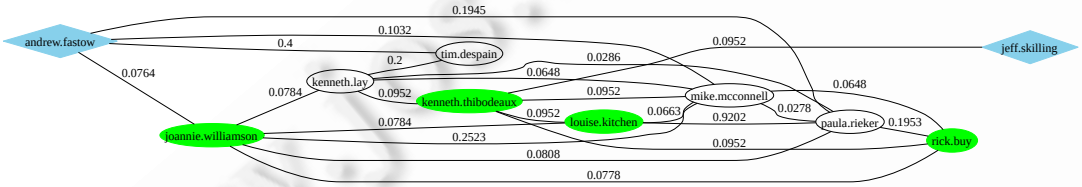
中连接子图的模式截然不同.事实上,Skillinig 于 2001 年 8 月 14 日(即 T_2 和 T_3 的交界处)辞去 Enron 公司总裁的职务,并继续担任 Enron 公司的顾问.因此,Skilling 与其他管理者之间的私人电子邮件通信减少,转为由秘书作为通信的中间媒介.因此,将 T_2 和 T_3 划分为两个分段是合理的.

(4) 在 T_4=[$week_{158}$,$week_{179}$]中没有发现任何 Fastow 与 Skilling 之间的通信路径.其原因在于,美国证券交易委员会在 2001 年 10 月 24 日(即 T_3 和 T_4 的交界处)对 Enron 公司展开调查.Enron 公司随后辞退了 Fastow.从此,不再有任何与 Fastow 的电子邮件通信.因此,将 T_4 划分为一个分段是与事实相符的.

图 5 给出了 T_2 和 T_3 上的代表性连接子图模式 RC_2=C_{119} 和 RC_3=C_{155},其中,顶点表示电子邮件地址(域名省略),菱形顶点分别表示 Fastow 和 Skilling.图 5(a)中的灰色椭圆形顶点表示 T_2 中最常出现在连接子图中的邮件地址(多为管理者).图 5(b)中的灰色椭圆形顶点表示 T_3 中最常出现在连接子图中的邮件地址(多为普通职员).



(a) RC_2



(b) RC_3

Fig.5 Representative connection subgraph patterns RC_2 and RC_3 over T_2 and T_3

图 5 T_2 和 T_3 上的代表性连接子图模式 RC_2 和 RC_3

4 相关工作

连接子图是一种新的知识类型,用于概述图中一组顶点之间的主要连接关系.它与图中顶点之间的距离、近邻度[2,8]等概念存在着密切的联系.连接子图挖掘问题是 NP-完全问题[4].文献[1−3,5]提出了若干挖掘连接子图的启发式算法.然而,这些算法仅考虑静态图上的连接子图挖掘,而没有考虑演变图上连接子图挖掘.

动态图挖掘也引起了越来越多的关注.文献[6]研究了多种真实图数据(如 Web 图、Internet 图等)的全局结构特性随时间变化的规律,发现图的密度随时间增大、图的直径随时间减小,并提出一种符合这种变化规律的图生成模型.文献[10]研究了真实社会网络中社团结构形成以及随时间变化的规律.这类研究工作关注于用实验手段发现大规模图数据的全局结构特性随时间变化的规律.动态图挖掘的另一方面的工作主要关注于动态图的社团结构.文献[11]提出一种动态规划算法来挖掘动态图中的社团结构及其随时间变化的模式.文献[12]提出一种在网络日志中发现特定时间区间内稳定的关键词聚类的算法.文献[7]提出一种在动态图中发现社团结构及其发生变化的时间点的信息论方法.图数据的全局结构随时间变化的规律可以解释现实世界中复杂网络的形成和发展的内在机理,却无法给出图数据中局部结构随时间变化的模式,因此无法解决图数据管理和分析中的具体问题.然而,本文的连接子图演变模式挖掘却可以回答图数据中一部分特定节点之间的连接关系及其随时间变化的规律.

5 结 论

本文提出了演变图上的连接子图演变模式挖掘问题,并给出一种解决该问题的高效算法.该算法利用演变

连接子图集合的最优相似分段集合的重叠子结构,使用动态规划算法在多项式时间内发现连接子图的演变模式集合.该算法的时间复杂度为 $O\left(\sum_{i=1}^{n}|E_i|+n^2(v\log v+e\log e)\right)$.其中,$n$ 是输入演变图集合中图的数量;E_i 是 G 演变得到的图 G_i 的边数;$v=\max\{|V_{C_i}||1\le i\le n\}$,$e=\max\{|E_{C_i}||1\le i\le n\}$,分别是全部连接子图$\{C_i|1\le i\le n\}$中最大的顶点数和边数.在模拟数据集上的实验结果表明,算法具有较低的误差率和较高的效率.算法在真实数据集上的实验结果表明,挖掘结果在真实应用中具有实际意义.

References:

[1] Faloutsos C, McCurley KS, Tomkins A. Fast discovery of connection subgraphs. In: Kim W, Kohavi R, Gehrke J, DuMouchel W, eds. Proc. of the 10th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM Press, 2004. 118−127.

[2] Koren Y, North SC, Volinsky C. Measuring and extracting proximity in networks. In: Eliassi-Rad T, Ungar LH, Craven M, Gunopulos D, eds. Proc. of the 12th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM Press, 2006. 245−255.

[3] Tong H, Faloutsos C. Center-Piece subgraphs: Problem definition and fast solutions. In: Eliassi-Rad T, Ungar LH, Craven M, Gunopulos D, eds. Proc. of the 12th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM Press, 2006. 404−413.

[4] Conrad J, Gomes CP, Hoeve WJ, Sabharwal A, Suter J. Connections in networks: Hardness of feasibility versus optimality. In: Hentenryck PV, Wolsey LA, eds. Proc. of the 4th Int'l Conf. on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Brussels: Springer-Verlag, 2007. 16−28.

[5] Tong H, Faloutsos C, Koren Y. Fast direction-aware proximity for graph mining. In: Berkhin P, Caruana R, Wu X, eds. Proc. of the 13th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM Press, 2007. 747−756.

[6] Leskovec J, Kleinberg JM, Faloutsos C. Graphs over time: Densification laws, shrinking diameters and possible explanations. In: Grossman R, Bayardo RJ, Bennett KP, eds. Proc. of the 11th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM Press, 2005. 177−187.

[7] Sun J, Faloutsos C, Papadimitriou S, Yu PS. GraphScope: Parameter-Free mining of large time-evolving graphs. In: Berkhin P, Caruana R, Wu X, eds. Proc. of the 13th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM Press, 2007. 687−696.

[8] Bunke H. On a relation between graph edit distance and maximum common subgraph. Pattern Recognition Letters, 1997,18(9): 689−694. [doi: 10.1016/S0167-8655(97)00060-3]

[9] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. 2nd ed., The MIT Press and McGraw-Hill Book Company, 2001. 552−557.

[10] Backstrom L, Huttenlocher D, Kleinberg J, Lan X. Group formation in large social networks: Membership, growth, and evolution. In: Eliassi-Rad T, Ungar LH, Craven M, Gunopulos D, eds. Proc. of the 12th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM Press, 2006. 44−54.

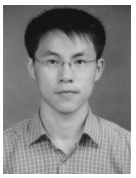
[11] Tantipathananandh C, Berger-Wolf TY, Kempe D. A framework for community identification in dynamic social networks. In: Berkhin P, Caruana R, Wu X, eds. Proc. of the 13th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM Press, 2007. 717−726.

[12] Bansal N, Chiang F, Koudas N, Tompa FW. Seeking stable clusters in the blogosphere. 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 24th Int'l Conf. on Very Large Data Bases. New York: ACM Press, 2007. 806−817.

邹兆年(1979－),男,吉林长春人,博士生,主要研究领域为图数据挖掘.

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

高宏(1966－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据仓库,传感器网络.



张硕(1982－),男,博士生,主要研究领域为图数据管理.