计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.201100005

时序图中 Top-k 稠密子图查询算法研究

穆聪聪　王一舒　袁　野　乔百友　马玉亮
东北大学计算机科学与工程学院　沈阳 110000
(mucong@stumail.neu.edu.cn)

摘　要　稠密子图的查询是图分析领域的重要研究问题之一,在社交用户相关性分析、Web 中社群分析等方面都有着广泛的应用。目前,关于稠密子图查询的研究工作主要基于静态图。而在实际应用中,时序信息会对稠密子图查询产生重要的影响,使得图拓扑结构随时间序列不断发生变化,包含的信息量也不断增加,使得已有的针对静态图的查找方法不再适用于时序图。因此,如何高效地在时序图上查找稠密子图仍然是一个挑战。为了解决上述挑战,首先规范化地定义了基于时序图的稠密子图查找问题;然后,根据图的拓扑结构和包含时间标签的边之间的相似度,提出一种基于阈值的近似查找算法 DTS-base。为了加快算法的收敛速度,提出了一个基于快速计算最大相似度时间片的优化算法 DTS-opt。最后,通过在真实数据集上的实验,证明了所提算法的高效性和可扩展性。

关键词:稠密子图;时序图;Top-k 查询
中图法分类号　TP399

Top-k Densest Subgraphs Search in Temporal Graphs

MU Cong-cong, WANG Yi-shu, YUAN Ye, QIAO Bai-you and MA Yu-liang
School of Computer Science and Engineering, Northeastern University, Shenyang 110000, China

Abstract　The dense subgraph search problem is one of the most important graph analysis problems. It is widely used in many fields, such as the social user correlation analysis in social networks, the community discovery in the Web, etc. However, the current research focuses on searching dense subgraphs on static graphs. In practical application, temporal information has an important impact on the query of dense subgraphs, which makes the topology structure of graphs change constantly with time sequences, and the amount of information contained also increases dramatically. Therefore, the existing searching methods for static graphs are no longer applicable to temporal graphs. Hence, it is still a challenge to search dense subgraphs efficiently on a temporal graph. In order to solve the above challenge, this paper first defines the Top-k densest temporal subgraphs searching problems in a standardized way. Then, to address the above challenge, this paper proposes an approximate searching algorithm DTS-base based on threshold according to the topology of the graph and the similarity between edges containing time tags. Furthermore, an optimization algorithm DTS-opt based on the fast calculation of maximum similarity time slice is proposed in order to accelerate the convergence speed. Finally, experiments on real data sets demonstrate the efficiency and scalability of the proposed algorithms.

Keywords　Densest subgraph, Temporal graph, Top-k query

1　引言

近年来,随着图查询研究的不断深入,稠密子图查询问题得到了越来越多研究者的关注[1-3]。稠密子图是指一个图中相对集中的子区域,一般被视作图结构的核心部分。稠密子图作为具有特殊结构和性质的子图,可表示现实网络中具有某种特定意义的对象群体。例如:1)在商业及金融分析[1]中,稠密子图往往代表着内部高度相关的实体结构,对了解市场动态及预测金融产品走势有着极高的价值;2)在社交网络[2-3]中,顶点代表用户,边代表用户之间的朋友关系,网络中的稠密区域可以被认为是一个社区团体。因此,对稠密子图查询的研究在实际应用中具有非常重要的现实意义。

目前,大量的研究工作给出了不同的稠密子图的定义和相应的查询算法,例如,k-core[4]、k-truss[5]、极大团[6-7]和准团[8-9]等。然而,现有算法主要是针对静态图。但是在实际应用中,很多随时间变化的网络不适合被简单地抽象为静态图,如社交网络、通信网络、交通网络以及蛋白质相互作用网络等。在这些网络中,节点之间只在某些特定的时间存在联系,

到稿日期:2020-11-02　返修日期:2021-03-16　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家重点研发计划项目(2016YFCI401900)
This work was supported by the National Key Research and Development Project(2016YFCI401900).
通信作者:袁野(yuanye@mail.neu.edu.cn)

即节点之间的联系是具有时序性的。广义上，时序图是在静态图的基础上加入了时间维度信息。狭义上，时序图包含了图的演化历史，能够为应用提供图在不同历史时刻下的信息，为通信网络[10]、社交网络[5]等领域的挖掘提供了更加丰富的数据支持。因此，本文主要研究基于时序图的稠密子图查询问题。

一个典型的基于时序图的稠密子图示例如图 1 所示。该图表示一个合作社群的网络，其中，节点表示合著者网络中的作者，节点之间的边表示作者之间存在合作，边上的时间序列表示合作的时间点。合作社群的演化主要有以下几种情形：1）如图 1 中蓝色节点构成的合作群体内部一直紧密合作，构成相对稳定。2）如图 1 中黄色节点构成的合作群体仅在{1，2}时间内紧密合作，然后各自转向其他的群体。3）如图 1 中绿色节点构成的合作群体的合作时间不稳定，但构成相对稳定；而其他内部结构松散或者构成不稳定的群体不能反映演化的过程。如图 1 中〈2，5，9，17，20〉节点集构成的子图虽然在拓扑结构上是紧密的，但是从合作时间层面上看，该子图结构并不稳定，该子图也不能反映群体的演化过程。因此，本文基于时序图提出稠密子图的查询问题，查询的结果包含以上 3 种情形，使结果中的稠密时序子图既能满足合作群体内部的紧密性，同时又能从时间维度更好地体现合作社群演化的过程。本文提出的查询算法还可以应用于海洋监测领域，查询的结果可以表示海洋污染或温度等演化的过程。

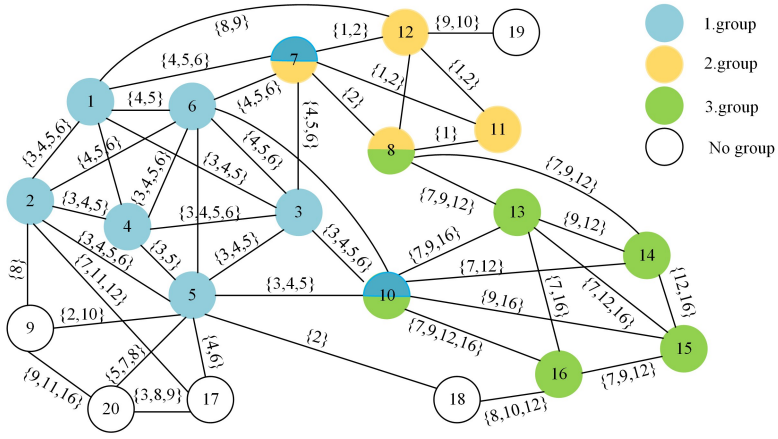

图 1　稠密时序图的示例（电子版为彩色）

Fig. 1　Example of dense temporal graph

由于在时序图中每条边上都有记录时间的标签序列，且节点之间的联系是随着时间动态变化的，而查找的稠密时序子图不仅要保证在拓扑结构上的稠密性，而且要保证子图中的边尽可能同时出现在所包含的每一个时间片中。因此，现有的静态图上的稠密子图查询算法无法直接应用于这样的时序图。目前，只有文献[11-12]研究了时序图的稠密子图查询问题。但其查找的是稠密子图可能存在的时间片区间，进而找出稠密子图。这些查找算法得到的结果子图只需要保证在某个时间区间内该子图的拓扑结构是稠密的，即这些方法也不适用于本文。因此，如何高效地在时序图上查找稠密子图仍然是一个挑战。为了解决以上挑战，本文首次提出了稠密时序子图（Dense Temporal Subgraph，DTS）模型，在静态图密度的基础上定义了稠密时序子图的密度，既考虑了边密度，又考虑了边上时间的相关性。本文的主要贡献如下：

（1）本文以时序图为对象研究稠密子图查询问题，并首次形式化地定义了时序图密度和时序图上的 Top-k 稠密子图查询问题；

（2）本文根据图的拓扑结构和包含时间标签的边之间的相似度，提出了基于阈值的稠密时序子图的基本算法 DTS-base；

（3）通过对时间片相似度矩阵的进一步优化，提出基于快速计算最大相似度时间片的 DTS-opt 算法，在很大程度上提高了 DTS-base 的查找效率；

（4）通过在 5 种真实数据集上进行测试，验证和分析了所提算法的有效性和可扩展性。

本文第 2 节介绍了相关工作；第 3 节介绍了基于时序图的稠密子图挖掘的基础知识和相关定义；第 4 节和第 5 节主要介绍了提出的稠密时序子图查询的两种近似算法；第 6 节在真实数据集上，对提出的算法进行实验评价和结果分析；最后总结全文。

2　相关工作

2.1　稠密子图模型

经典的稠密子图衡量标准主要有 3 种：度、距离和三角划分。基于度的衡量标准就是根据子图中节点拥有边的条数来度量子图的稠密性。文献[13]提出了一个挖掘最稠密准团的随机算法，而文献[4]提出了一种多项式时间间隔内连续产生所有准团的搜索算法。基于距离的衡量标准就是根据任意两个节点间的最短路径的长度来度量子图的稠密性。为了改善基于度或距离的稠密子图存在的问题，文献[5]提出了一种基于三角划分的稠密子图——k-truss 结构，即子图内每条边都包含在至少 $k-2$ 个三角形中。

另外，文献[14]定义了确定图 $G=(V,E)$ 的密度为 $\rho=|E|/|V|$，即图中顶点的平均度，并提出了一个基于最大流算法的多项式时间算法。文献[15]给出了一个线性时间的 2-近似算法。本文是在平均度的基础上加入时间维度的信息对时序子图的稠密性进行度量。因此本文提出的稠密时序子图的概念更为复杂，不能直接应用之前的方法。

2.2 动态图中的稠密子图挖掘

动态图(dynamic graph)是指会随时间发生变化的图模型或者图数据。动态图可以根据其变化的形式分为两类：1)图中拓扑结构关系发生变化；2)图中顶点和边所代表的数据对象内容，或者图中某一特定对象的评价方式发生变化[16]。动态图数据在现实生活中是广泛存在的，例如，在社交网络中建立友谊关系的过程、蛋白质交互过程中各类蛋白质分子的相互作用等。

Sarma 等[17]在分布式计算的拥挤模型的动态版本中研究了稠密子图问题。在这种模型下，他们设计了一种复杂度为$(2+\varepsilon)$的近似算法，其中需要 $O\left(D\frac{\log n}{\varepsilon}\right)$轮并行，$D$ 代表网络的直径。Angel 等[18]研究了在流边权值更新下高效维护稠密子图的问题，从而发现 Twitter 中有趣的事件。他们主要关注于查找小规模的稠密子图(最多有 10 个节点)。Valari 等[19]基于动态图研究了在每次递归移除最稠密的子图时，如何计算 Top-k 稠密子图的问题。

2.3 时序图中的稠密子图挖掘

时序图(temporal graph)[20]是一种会随着时间不断变化的图结构，时序图本质上是带有随时间变化标签的图，即图中的标签带有某种特定的时间属性。时序图强调在一个时间阈值内数据的变化。一般情况下时序图可以按照顶点间联系持续时间 $\lambda(e)$的大小分为以下两种情况。

第一种是各顶点间联系无持续性或其持续时间 $\lambda(e)$可忽略不计，即 $\lambda(e)=0$。这种情况下，图中的边$\langle u,v\rangle$ 可以用三元组(u,v,t) 进行表示，每条边$\langle u,v\rangle$都存在一组时间的序列 $T^{uv}=\{t_1,\cdots,t_n\}$ 。本文是基于 $\lambda(e)=0$ 的情况下在时序图上进行稠密子图查询问题研究的。第二种是时序图中的边在一定的时间段内被激活，即 $\lambda(e)\neq 0$。在这种时序图中，持续时间是一个重要、不可忽视且必须考虑的因素。

目前在时序图中的稠密子图的挖掘问题主要是在时序图上找出最可能出现稠密子图的区间，从而得到稠密子图。文献[11]通过计算每张快照边上的权值，为时序图构建了一个以时间戳为横坐标的粘性密度曲线，并通过该曲线得到了时序图上最可能存在稠密子图的区间。文献[12]提出了基于时序图的一种启发式稠密子图搜索策略。该方法首先通过剪枝策略移除不存在于时序子图模式中的顶点和时间阈值，然后在新的时序图中，将新的时序图划分成一系列快照，最终找到稠密子图。该方法在一定程度上保留了时序图上的时间关联性，但是完成该算法所需要的时间较长，不能很好地应用在大规模时序图上。Semertzidis 等[21]提出在时序图中所有时间快照上找到一个持久的稠密子图。文献[22]基于时序图提出了(θ,τ) 持续性 k 核子图搜索问题，也就是给定查询图中的一个顶点，通过滑动窗口模型算法找到一个在一段时间内持续存在且包含该查询点的稠密子图。

3 背景知识和问题定义

本节首先介绍了时序图的一些背景知识，接着形式化定义了稠密时序子图问题。

3.1 基本定义

定义 1(时序图) 给定时序图 $G^T=(V,E,T)$，V 表示节点的集合，E 表示边的集合，T 表示图中所有节点之间存在联系时刻的集合，$T^{uv}\in T$ 表示在节点 u 和 v 之间存在联系的时刻的集合。

例 1 如图 2 所示，节点 a 和 c 在时刻 2 和 4 时存在关系。因此，$T^{ac}=\{2,4\}$。

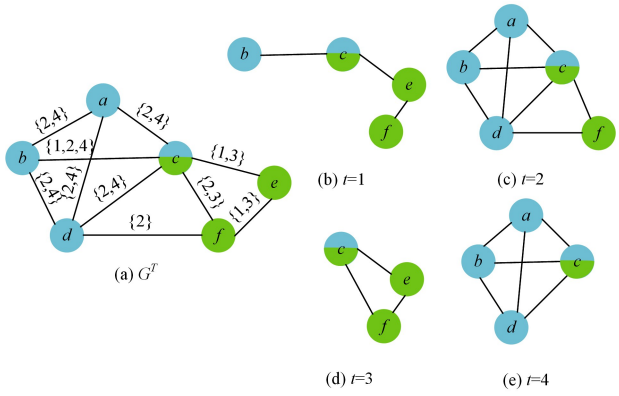

图 2 时序图及对应的时间片快照

Fig. 2 Temporal graph and corresponding snapshots

基于静态图查询稠密子图的研究通常使用平均度来度量子图的稠密性，子图内各节点之间的联系越紧密，平均度越高。同样地，子图拓扑结构的稠密性也是稠密时序子图的度量之一。时序图中平均度的概念如定义 2 所示。

定义 2(平均度) 对于时序图 $G^T=(V,E,T)$的子图 $g^T=(V',E',T')$，其中 $V'\subseteq V,E'\subseteq E,T'\subseteq T$，图 g^T 的平均度如式(1)所示：

$$\rho(g^T)=\frac{|E'|}{|T'|} \tag{1}$$

3.2 时序图的时序相似度计算

3.1 节是从子图的拓扑结构方面对时序图的稠密性进行度量，但是，从时间维度对时序图的稠密性进行度量也至关重要。因此，在得出 Top-k 稠密时序子图问题的完整定义之前，需要首先对时序图中时间片之间的相似度和时序图密度进行定义。

如图 1 所示，该稠密子图示例中所包含的 3 种情形除了对拓扑结构的稠密性进行度量以外，还需要从时间层面对结构稳定性进行度量。直观来讲，两个时间片对应的快照中包含的相同节点越多，这两个时间片就越容易出现在同一个子图中。因此，首先计算时间片之间的相似度，如定义 3 所示。

定义 3(时间片相似度) 给定时序图 $G^T=(V,E,T)$和任意两个时间片 t_1 和 t_2，时间片对应快照之间的相似度为：

$$sim(t_1,t_2)=\frac{n^2}{|E^{t_1}|\cdot|E^{t_2}|} \tag{2}$$

其中，n 表示两张快照之间相同边的数目，$|E^{t_1}|$和$|E^{t_2}|$分别表示两张快照中边的数目。

例 2 如图 2 所示，$t=2$ 和 $t=4$ 两张快照之间有 6 条相同的边，分别为$\langle a,b\rangle$，$\langle a,c\rangle$，$\langle a,d\rangle$，$\langle b,c\rangle$，$\langle b,d\rangle$和$\langle c,d\rangle$，两张快照分别有 8 条边和 6 条边，因此它们之间的相似度为 $sim(2,4)=\frac{6^2}{8\cdot 6}=0.75$。

通过定义 3 可以计算出两个时间片之间的相似度。因此，为了从时间维度更好地度量稠密时序子图的结构稳定性，

本文提出通过快照相似度矩阵计算稠密时序子图的时间片相似度。稠密时序子图的相似度计算如式(3)所示：

$$SimTemporal(G^T)=\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}sim(t_i,t_j)}{t-1} \tag{3}$$

其中，t 表示时序图 G^T 中包含的时间片的个数。$SimTemporal$ 值越高代表该时序图中快照之间的相似度越高，包含的时间片就越可能出现在最后的结果子图中。

定义 4(时序图密度) 给定一个时序图 G^T 的子图 $g^T=(V',E',T')$，其中 $V'\subseteq V,E'\subseteq E,T'\subseteq T$，子图 g^T 的密度如式(4)所示：

$$dens(g^T)=\alpha\rho(g^T)+(1-\alpha)SimTemporal(g^T) \tag{4}$$

其中，$\rho(g^T)$表示时序图 g^T 的平均度，$SimTemporal(\cdot)$表示稠密时序子图的相似度，参数 α 提供了平均度和时间片相似度之间的权衡。

定义 5(Top-k 稠密时序子图(DTS)问题) 给定一个时序图 $G^T=(V^T,E^T,T)$，一个参数 λ 和一个正整数 k，找到 k 个稠密时序子图集 $G^T=\{g_1^T,g_2^T,\cdots,g_k^T\mid g_i^T\subseteq V,1\leqslant i\leqslant k\}$，使得回报函数 $r(G^T)$最大，且 $\forall g_i^T,g_j^T\subseteq V,\frac{|g_i^T\cap g_j^T|}{|g_i^T\cup g_j^T|}\leqslant\lambda$。

其中，回报函数(reward function) $r(G^T)=\sum_{i=1}^{k}dens(g_i^T)$。$dens(g_i^T)$表示稠密时序子图 g_i^T 的密度，参数 λ 作为任意两个时序子图的重叠上界，允许两个时序子图之间存在一定数量的公共节点。

4 基于阈值的 DTS-base 算法

本节提出了一种基于阈值的解决时序图中 Top-k 稠密子图查询问题的算法 DTS-base，如算法 1 所示。DTS-base 算法思路是：首先，在时序图 G^T 中找到前 k 个平均度最大的子图。然后，计算每个时序子图最大的时间片相似度和时序密度。最后，以前 k 个时序子图为基础计算出搜索的 *lowerbound*，从而保证找到所有候选时序子图，取前 k 个时序密度最大的子图作为结果集。

算法 1 DTS_base(G^T,k,λ)

输入：时序图 $G^T=(V^T,E^T,T)$，参数 $k>0,\lambda\in[0,1]$

输出：时序密度最大的 k 个子图集$\mathcal{G}$

1. G←DS(G^T,k,λ)；
2. ($\forall g_i^T\in G$)←sim(g_i^T)
3. 初始化 lowerbound
4. WHILE 找不到满足 lowerbound 的子图
5. k=k+1；
6. g_k^T=DS(G,1,λ)；
7. if($\rho(g_k^T)+lb_{temporal}>lb_{total}$)
8. 计算 dens(g_k^T)并加入到结果集 G^T；
9. RETURN G^T
10. Function DS(G^T,k,λ)：
11. 初始化结果集 G=∅
12. WHILE 找到小于 k 个子图
13. 运行贪心算法找到一个 2-近似的最稠密子图 g_i；
14. 在 G 中移除 g_i 中与外部联系不密切的⌈(1−λ)|V_i|⌉个节点及它们之间的边；
15. G=G∪g_i；
16. END WHILE
17. RETURN G
18. Function sim(g_i^T)：
19. 根据时间片快照之间的相似度构造相似度矩阵 Matrix(g_i^T)
20. SimTemporal(g_i^T)←Matrix(g_i^T)；
21. WHILE SimTemporal(g_i^T)增加
22. 找出平均相似度最小的时间片 t；
23. T=T\{t}；
24. SimTemporal(g_i^T)←Matrix(g_i^T)
25. END WHILE
26. RETURN g_i^T

4.1 计算 k 个平均度最大的子图

结果集中的每个子图在拓扑结构上应首先满足两个条件：一是稠密性，即子图中节点之间是联系紧密的；二是在允许重叠的前提下，子图之间尽可能少地包含重复的节点。针对条件一，我们的目的是从 G 中得到 k 个平均度最大的子图。直观地看，在每次查询最大平均度的过程中，我们可以使用贪心策略依次删除与度数最小的节点 $v_{\min}$ 有关的边。算法维护一个顶点集 S，初始化 $S\leftarrow V$。在每次迭代中，找到并删除 S 的诱导子图中度数最小的节点及其相关的边。然后从 S 中删除该节点并进行下一次迭代，当 S 为空时停止迭代。通过上述过程，即可找到平均度最大的顶点集 S。文献[15]证明了该贪心算法为 2-近似的。

针对条件二，可以通过引入一个重叠上限 λ 来控制每个子图的重叠程度。参数 λ 提供了稠密度和重叠程度之间的权重。一个小的 λ 更强调子图之间的重叠度更低，从而使得结果集中包含更多的节点；而一个大的 λ 更强调子图是稠密的，且结果集中子图之间的重叠度更高。要想满足两次贪心迭代得到的子图之间的重叠上限为 λ，只要在下一次迭代中保留$\lfloor\lambda|V_i|\rfloor$个节点，即删除$\lceil(1-\lambda)|V_i|\rceil$个节点。但是，不是每一个节点都可以随意删除，为了在下一次迭代中更容易地找到拓扑结构更为稠密的子图，我们需要保留与外部联系密切的节点，删除与外部联系不密切的节点，这样才能保留更多的边。因此，在每次贪心策略执行之后，从 G 中移除 G_i 中与外部联系不密切的$\lceil(1-\lambda)|V_i|\rceil$个节点及它们之间的边，从而在下一次迭代中既能保证有较大的稠密度，又能同时满足子图之间的重叠上限 λ。在我们的实验评估中也证明了参数 λ 在不同取值下的效果。具体执行过程如算法 1(第 10－17 行)所示。

4.2 计算相似度最大的时间片

4.1 节从拓扑结构的角度分析了结果集中的子图需要满足的条件，从而得到了既保证稠密性又同时满足 λ 重叠上限的 k 个稠密子图。但是，从时序的角度来看这些子图的结构并不一定是稳定的。即时间片扩展到子图中，并不能保证子图中的所有边大致包含相同的时间片集合，也就不能说明该子图在时序的角度是结构稳定的。因此，本节主要介绍如何计算稠密子图的时序密度。在计算之前，首先引入定理 1。

定理 1 通过贪心迭代得到的子图中任意一行时间片之间的平均相似度为$\frac{1}{n-1}\sum_{j=1}^{n}sim(t_i,t_j)\geqslant\frac{1}{2(n-1)}$，其中 n 为该

子图所包含时间片的个数。

证明：利用数学归纳法，已知只包含一个时间片的子图的相似度为1，所以通过贪心策略迭代得到的子图的稠密时序子图的相似度 $SimTemporal(G^T)\geqslant 1$。当结果集中子图只包含两个时间片，则该稠密时序子图的相似度一定大于或等于1，因此 $sim(t_1,t_2)=sim(t_2,t_1)\geqslant\frac{1}{2}$。假设结果集中子图只包含 $n-1$ 个时间片，并且该子图中任意一行时间片的平均相似度 $\frac{1}{n-2}\sum_{j=1}^{n-1}sim(t_i,t_j)\geqslant\frac{1}{2(n-2)}$，即 $\sum_{j=1}^{n}sim(t_i,t_j)\geqslant\frac{1}{2}$，那么对于包含 n 个时间片的子图，相当于增加了一行和一列，该行和列中的每一个数值表示新增加的这个时间片和前 $n-1$ 个时间片的相似度，已知 $\sum_{i=1}^{n-1}\sum_{j=1}^{n-1}sim(t_i,t_j)\geqslant\frac{n-1}{2}$，同时为了保证该稠密时序子图的相似度 $SimTemporal(G^T)\geqslant 1$，即 $\frac{1}{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n}sim(t_i,t_j)\geqslant 1$，又 $\sum_{i=1}^{n}\sum_{j=1}^{n}sim(t_i,t_j)=\sum_{i=1}^{n-1}\sum_{j=1}^{n-1}sim(t_i,t_j)+2\sum_{j=1}^{n}sim(t_i,t_j)=n$。因此对于新增加的第 n 行，$\frac{1}{n-1}\sum_{j=1}^{n-1}sim(t_i,t_j)\geqslant\frac{1}{2(n-1)}$，证毕。

首先，将时间片扩展到子图，目的是要找到在当前时序子图中使得相似度最大的时间片的集合。在这个过程中可以构造一个相似度矩阵 $\boldsymbol{Matrix}(\boldsymbol{G}^T)$，矩阵中的每一个元素 m_{ij} 代表第 i 个时间片和第 j 个时间片快照之间的相似度，矩阵中的第 i 行代表第 i 个时间片快照与其他时间片快照的相似度向量，每一行的平均值越高代表该时间片快照和其他时间片快照之间的相似度越高，反之亦然。因此，为了使最后得到的时间片相似度 $SimTemporal(G^T)$ 尽可能大，可以迭代地删除当前矩阵中相似度最小的时间片对应的行和列。根据定理1可知，如果本次迭代中存在一行时间片之间的平均相似度小于 $\frac{1}{2(n-1)}$，则还可以继续迭代，直到 $SimTemporal(G^T)$ 不再增加为止。具体执行过程如算法1(第18－26行)所示。

例3 以图2为例，假设图2(a)为算法1得到的稠密子图，可以构造出时序图2(a)各个快照之间的相似度矩阵，计算出该时序图的时间片相似度：

$$\boldsymbol{Matrix}(\boldsymbol{G}^T)=\begin{bmatrix}0 & \frac{1^2}{3\times 8} & \frac{2^2}{3\times 3} & \frac{1^2}{3\times 6}\\ \frac{1^2}{8\times 3} & 0 & \frac{1^2}{8\times 3} & \frac{6^2}{8\times 6}\\ \frac{2^2}{3\times 3} & \frac{1^2}{3\times 8} & 0 & 0\\ \frac{1^2}{6\times 3} & \frac{6^2}{6\times 8} & 0 & 0\end{bmatrix}$$

$$\Rightarrow\begin{bmatrix}0.1806\\ 0.2778\\ 0.1620\\ 0.2685\end{bmatrix}\Rightarrow SimTemporal(G^T)=0.89$$

按照时间片相似度的大小依次删除时间片3，1，4并重新计算时序子图的相似度，可分别得到相似度为0.847 222，1.5和1，因此删除时间片{3，1}可以得到相似度最高的时间片。

4.3 计算前 k 个稠密时序子图

通过上文我们能够找到 k 个平均度最大的子图并计算出它们对应的时序子图的最大时间片相似度，然后，根据式(4)可以计算出当前 k 个稠密时序子图的密度。根据式(4)可以看出，稠密时序子图的密度为平均度和时间片相似度的加权之和。在这种情况下，不能保证找到的前 k 个平均度最大的子图即为最后的结果。因此，为了保证在剩余子图中不存在比当前 k 个中任意一个稠密时序子图密度更大的子图，本节提出一个 *lowerbound* 以便于找出所有的候选子图，从而得出前 k 个密度最大的稠密时序子图。

直观来说，如果下一次迭代找到的子图的平均度加上已找到的 k 个稠密时序子图中最小的时间片相似度的总和大于当前 k 个稠密时序子图中最小的密度，则需要判断当前子图的实际相似度，从而决定是否将其加入到候选结果集中；反之，迭代终止。因此，前 k 个稠密时序子图中最小的时间片相似度和最小时序密度就是 *lowerbound*。根据 *lowerbound* 查找前 k 个时序密度最大的子图的具体执行过程如算法1(第1－9行)所示。

5 优化的 DTS-opt 算法

第4节介绍了找到稠密时序子图的3个步骤。从 DTS-base 算法中可以看出，在得到每一个时序子图后，需要构造该子图中涉及到的所有时间片之间的相似度矩阵，通过迭代地删除平均相似度最小的一行和一列，可以得到相似度最大的时间片集合，从而计算出该稠密时序子图的相似度。显然，迭代删除的部分在3个步骤中的时间复杂度是最大的。但是，在许多时序图中时间片的规模往往和相似度最大的时间片集合的规模差距较大。所以可以通过对部分时间片进行约减，缩减相似性矩阵的规模，从而减少计算的次数，达到提升计算速度的效果。因此，本文提出了一个基于快速计算最大相似度时间片的优化算法 DTS-opt。

从算法 DTS-base 中可以看出，在得到稠密子图之后，需要把时间信息扩展到该子图中，通过比较各个快照之间的相似度，从而构造一个精确相似度矩阵。但是，这种精确的相似度矩阵在算法刚开始的时候是没有必要的，因为时间片的规模往往很大，也就意味着会存在很多快照之间的相似度很低，算法需要执行比较长的一段时间才会慢慢趋向收敛。因此，我们的目的是想让算法一开始就从趋向收敛的一个相似度矩阵开始计算，通过少量的迭代过程，找到对应的稠密时序子图的最大相似度。

直观来看，某个时间片对应的子图的边数越多，则有着更大的概率存在与之相似度大的子图，并且该子图大概率会出现在最后的结果集中。另外，两个时间片快照相似度存在一个上界，如式(5)所示：

$$sim(t_1,t_2)\leqslant\frac{\min(|E^{t_1}|,|E^{t_2}|)^2}{|E^{t_1}|\cdot|E^{t_2}|}\tag{5}$$

定理2 如果某个子图的边数和边数比它大的最小子图的边数之比大于2，则子图的时间片相似度不再增加。

证明：假设将所有子图按照边数排序为 $\{y_1,y_2,y_3,\cdots,y_n,\cdots\}$，根据式(5)可以得到前 $n-1$ 个子图的时间片相似度

矩阵，对应的相似度为 $\frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n-1}\frac{y_j}{y_i}}{n-2}$，那么对于新加入的 y_n，如果使得相似度不再增加，则有 $\frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n-1}\frac{y_j}{y_i}}{n-2}-\frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n-1}\frac{y_j}{y_i}}{n-1}\geqslant\frac{2\sum_{i=1}^{n-1}\frac{y_n}{y_i}}{n-1}$，继续通分可得 $\sum_{i=1}^{n-1}\sum_{j=i+1}^{n-1}\frac{y_j}{y_i}\geqslant(n-2)\sum_{i=1}^{n-1}\frac{y_n}{y_i}$，即 $\frac{2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n-1}\frac{y_j}{y_i}}{n-2}\geqslant 2\sum_{i=1}^{n-1}\frac{y_n}{y_i}$。不等式左边的部分可以代表前 $n-1$ 行的平均值之和，右边的部分代表新增加的 y_n 与前 $n-1$ 个子图之间的相似度之和的 2 倍。因此，当 $\frac{y_n}{y_{n-1}}\geqslant 2$ 时，子图的时间片相似度不再增加。证毕。

根据式(5)和定理 2，可以首先按照每张快照中的边数进行排序，然后以从大到小的顺序按照式(5)中的相似度的上界进行扩展，从而在不计算实际相似度的情况下可以将相似度矩阵扩展到一个较小的范围，最后再计算实际的相似度从而保证更快的收敛。优化后的 DTS-opt 的具体执行过程如算法 2 所示。后续的实验也证明了该算法的有效性。

算法 2 DTS_opt(g_i^T)

输入：时序图 $G^T=(V^T,E^T,T)$，参数 k>0，λ∈[0,1]

输出：时序密度最大的 k 个稠密子图集 $\mathcal{G}^T$

1. 找到前 k 个平均度最大的候选时序子图；
2. 对于每一个候选时序子图 g_i^T
3. 根据子图中时间片出现的次数排序；
4. WHILE SimTemporal(g_i^T)增加
5. 按照时间片出现的次数依次构造相似度矩阵 Matrix(g_i^T)；
6. 扩展得到一个较小的 Matrix(g_i^T)；
7. Matrix(g_i^T)计算实际的 SimTemporal(g_i^T)；
8. 根据 lowerbound 继续搜索候选时序子图
9. 返回前 k 个时序密度最大的子图作为结果集 G^T
10. RETURN G^T.

例 4 以图 2 为例，按照各时间片包含的边数排序{2,4,1,3}使用式(5)依次构建近似的相似性矩阵。以{2,4}为例，$sim(2,4)\leqslant\frac{\min(8,6)^2}{8\times 6}=0.75$，构建的近似的相似度矩阵为 $\begin{vmatrix}0 & 0.75\\ 0.75 & 0\end{vmatrix}$，因此{2,4}构建的子图相似度为 1.5。同理可得，{2,4,1}构建的子图相似度为 1.406 25。可以看出，由于时间片{1}的加入使得相似度下降，故提前找到构建相似度矩阵的集合{2,4}，缩小了计算的量级。

6 实验和评估

本节主要对本文提出的 DTS-base 和 DTS-opt 算法的有效性和扩展性在真实数据集上进行验证和分析。

6.1 实验数据和参数

6.1.1 实验数据

本文实验选用数据集规模从小到大的 5 种真实数据集。其中，数据集 mit，mathoverflow 和 hep-ph 来自 KONECT-NETWORK 数据库。数据集 contact 和 email-eu 来自 Stanford Large Network Database Collection。实验数据的相关信息如表 1 所列。

表 1 数据集信息

Table 1 Information of datasets

Dataset	$\|E\|$	$\|V\|$	Temporal edge	Time slice
mit	96	2 539	1 086 404	3 452
contact	274	2 124	28 244	15 662
email-eu	986	24 929	332 334	20 788
mathoverflow	21 688	239 978	506 550	2 350
hep-ph	28 093	3 148 447	4 596 803	5 253

表 1 中，数据集 mit 是来自 MIT 的 Reality Mining 朋友关系数据集，数据集 contact 是堆栈交换网站 Super User 上进行交互所生成的，数据集 email-eu 由加利福尼亚大学欧文分校的在线社交网络上发送的私人消息组成，数据集 mathoverflow 是堆栈交换网站 Math Overflow 上的时间交互网络所生成的，数据集 hep-ph 是学术网站 arXiv 上的合著者网络所生成的。

6.1.2 实验参数

实验中，我们首先对比了本文算法 DST-base 和 DST-opt 的运行效率。然后，测试了两个算法在不同输入参数下的性能表现及其扩展性。参数设置如表 2 所列。本文所有算法全部采用 C++编程实现，实验的硬件环境为 Intel Core I5 处理器和 8GB 内存，运行 Windows10 操作系统。

表 2 实验参数设置

Table 2 Experimental parameter settings

Parameter	Value
The size of result of researching temporal densest subgraphs, k	2 **4** 6 8 10
The parameter of subgraph overlapping upper bound, λ	0 0.2 **0.4** 0.6 0.8
The parameter of adjusting average degree and time slice similarity, α	0.3 0.4 **0.5** 0.6 0.7

注：加粗数值为默认值

6.2 实验结果和分析

6.2.1 算法性能对比实验

首先，我们对比了本文提出的 DTS-base 算法和 DTS-opt 算法的运行效率。我们设定了稠密子图大小限制 $k=4$，平均度和时间片相似度的均衡参数 α 为 0.5，子图重叠上限参数 λ 为 0.4。实验通过在不同数据集中运行两种算法，比较它们的运行时间和运行内存。图 3 和图 4 给出了在 5 种不同数据集上两种算法的运行效率。可以看出，DTS-opt 算法的运行速度和运行内存明显优于 DTS-base 算法。这是因为 DTS-opt 算法采用了扩展的 ***Matrix*** 矩阵提前找到相似性矩阵需要计算的规模的最小值，从而大大减少和缩减了矩阵的运算时间和规模。

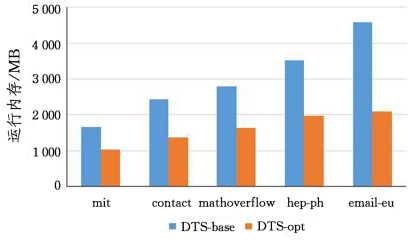

图 3 DTS-base 和 DTS-opt 算法运行内存的比较

Fig. 3 Comparison of memory between DTS-base and DTS-opt

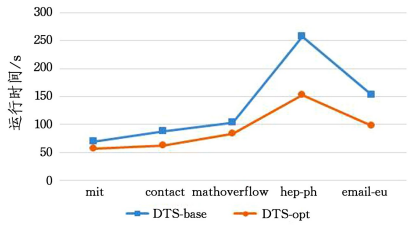

图 4 DTS-base 和 DTS-opt 算法运行时间的比较

Fig. 4 Comparison of runtime between DTS-base and DTS-opt

随后，我们对比了两种算法在时间片个数维度上的优化。我们设定了稠密子图大小限制 $k=4$，平均度和时间片相似度的均衡参数 α 为 0.5，子图重叠上限参数 λ 为 0.4。由图 5 可以看出，DTS-opt 算法在扩展相似度矩阵方面有着很明显的作用。

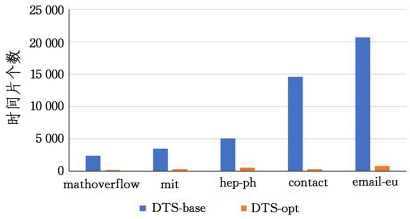

图 5 DTS-base 和 DTS-opt 算法时间片的比较

Fig. 5 Comparison of the number of time slice between DTS-base and DTS-opt

6.2.2 DTS 算法参数实验

我们通过实验检验 DTS-base 和 DTS-opt 算法的效率与其输入参数之间的关系，包括结果集合的大小 k、平均度和时间片相似度的均衡参数 α 和子图重叠上限参数 λ。在测试某一参数时，保持其他参数不变，并给出多组数据集上算法运行时间和相应测试参数之间的关系。

首先，令结果集大小 k 从 2 变化到 10 并测试算法时间。固定均衡参数 $\alpha=0.5$，子图重叠上限参数 $\lambda=0.4$。由图 6 可以看出，在同一数据集中，算法运行时间和 k 呈接近线性的增长趋势。当 k 增大时，算法运行时间的增加体现在两方面：一方面算法需要用前 k 个稠密时序子图的最小时间片相似度和最小时序密度来设置阈值，k 的增加会使得运行次数呈线性增加；另一方面，算法需要不断更新 G 中节点的状态并建立不同的相似性矩阵来约减时间片。

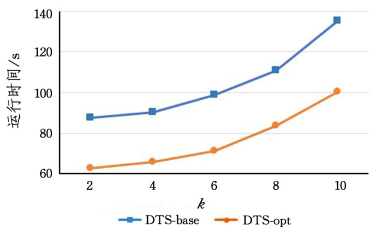

图 6 不同结果集大小 k 对算法的影响

Fig. 6 Impact of different k on algorithms

随后，我们检验了算法效率与均衡参数 α 之间的关系。固定结果集大小 $k=5$，子图重叠上限参数 $\lambda=0.4$，测试同一数据集下不同 α 的算法运行时间。由图 7 可以看出，算法运行时间和均衡参数 α 之间没有明显的线性关系。这是因为 α 越大，表示平均度在时序图密度中的比重越大，但是与时序图时间片相似度的计算不存在因果关系。同时在挖掘稠密子图的过程中使用贪心策略，得出结果子图的平均度基本上是从大到小排列的，所以改变均衡参数 α 的大小并不能使得运行时间有明显的改变。

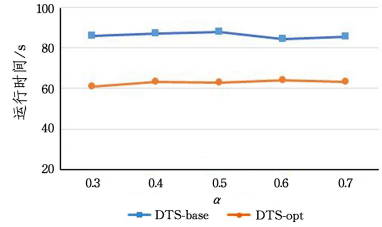

图 7 不同参数 α 对算法的影响

Fig. 7 Impact of different α on algorithms

最后，我们检验了算法效率与子图重叠上限参数 λ 之间的关系。实验中固定结果集的大小 $k=5$，平均度和时间片相似度的均衡参数 $\alpha=0.5$。由图 8 可以看出，在同一数据集中，算法的运行时间和 λ 基本呈线性的增加趋势。这是因为 λ 越大，表示在下一次迭代中可以考虑的节点数量就越多，得到的结果子图的规模就越大。随着结果子图规模的增大，涉及的时间片的数量也随之增加，从而延长运行时间。

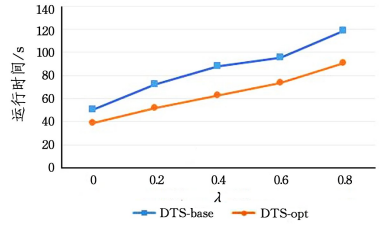

图 8 不同参数 λ 对算法的影响

Fig. 8 Impact of different λ on algorithms

结束语 本文提出了时序图上稠密子图查询问题，并提出了两种算法来解决该问题。首先，本文规范化地定义了稠密时序子图和 Top-k 稠密时序子图查询问题。为解决该问题，本文提出一种基于阈值算法的近似查找算法 DTS-base。此外，为了加快算法的收敛速度，本文还提出了优化算法 DTS-opt。最后，通过大量的实验分析，验证了本文算法的有效性和可扩展性。

在未来的工作中将会做如下的深入研究：1)将现有的算法扩展到动态的场景下，即考虑边/顶点的动态加入/删除；2)研究图上的查询处理与挖掘问题，如图聚类、图压缩等都可以以稠密子图为基础进行研究，未来可以在稠密时序子图的基础上对这些问题进行研究。

参 考 文 献

[1] FRATKIN E, NAUGHTON B T, BRUTLAG D L, et al. Motifcut: regulatory motifs finding with maximum density subgraphs [J]. Bioinformatics, 2006, 22(14): 150-157.

[2] RAVI K, PRABHAKAR R, SRIDHAR R, et al. Trawling the web for emerging cyber-communities[J]. Computer Networks, 1999, 31(11-16): 1481-1493.

[3] SOZIO M, GIONIS A. The community-search problem and how

to plan a successful cocktail party [C]// Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). Washington, DC, USA, 2010: 939-948.

[4] CHENG J, KE Y, CHU S, et al. Efficient core decomposition in massive networks[C]// Proceedings of the 27th International Conference on Data Engineering. Hannover, Germany, 2011: 51-62.

[5] UNO T. An efficient algorithm for solving pseudo clique enumeration problem[J]. Algorithmica, 2010, 56(1): 3-16.

[6] CHENG J, KE Y, FU W C, et al. Finding maximal cliques in massive networks by h*-graph [C]// Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. Indianapolis, Indiana, USA, 2010: 447-458.

[7] TSOURAKAKIS C. The k-clique densest subgraph problem [C]// Proceedings of the 24th International Conference on World Wide Web. Florence, Italy, 2015: 1122-1132.

[8] ABELLO J, RESENDE M G C, SUDARSKY S. Massive quasi-clique detection [C]// Theoretical Informatics, 5th Latin American Symposium. Cancun, Mexico, 2002: 598-612.

[9] MITZENMACHER M, PACHOCKI J, PENG R, et al. Scalable large near-clique detection in large-scale networks via sampling [C]// Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Sydney, NSW, Australia, 2015: 815-824.

[10] LI R H, QIN L, YU J X, et al. Influential community search in large networks[J]. Proceedings of the VLDB Endowment, 2015, 8(5): 509-520.

[11] MA S, HU R J, WANG L, et al. Fast Computation of Dense Temporal Subgraphs [C]// Proceeding of the 33rd IEEE International Conference on Data Engineering. San Diego, CA, USA, 2017: 361-372.

[12] YANG Y, YAN D, WU H H, et al. Diversified Temporal Subgraph Pattern Mining [C] // Proceeding of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining. New York, USA, 2016: 1965-1974.

[13] TSOURAKAKIS C, BONCHI F, GIONIS A, et al. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees [C]// Proceeding of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Chicago, IL, USA, 2013: 104-112.

[14] GOLDBERG A V. Finding a maximum density subgraph[D]. Berkeley: University of California, 1984.

[15] CHARIKAR M. Greedy approximation algorithms for finding dense components in a Graph[C]// Proceeding of the Approximation Algorithms for Combinatorial Optimization, Third International Workshop. Saarbrücken, Germany, 2000: 84-95.

[16] YANG Y J, GAO H, LI J Z. Survey on querying and mining algorithms on dynamic graphs[J]. Intelligent Computer and Application, 2013, 3(4): 24-27.

[17] SARMA A D, LALL A, NANONGKAI D, et al. Dense subgraphs on dynamic networks [C]// Proceeding of the Distributed Computing-26th International Symposium. Salvador, Brazil, 2012: 151-165.

[18] ANGEL A, KOUDAS N, SARKAS N, et al. Dense subgraph maintenance under streaming edge weight updates for real-time story identification[J]. The VLDB Journal, 2014, 23(2): 175-199.

[19] VALARI E, KONTAKI M, PAPADOPOULOS A N. Discovery of top-k dense subgraphs in dynamic graph collections [C]// Proceeding of the Scientific and Statistical Database Management-24th International Conference. Chania, Crete, Greece, 2012: 213-230.

[20] WU H, HUANG Y, CHENG J, et al. Efficient processing of reachability and time-based path queries in a temporal graph [C]// Proceedings of the 32nd International Conference on Data Engineering. Helsinki, Finland, 2016: 145-156.

[21] SEMERTZIDIS K, PITOURA E, TERZI E, et al. Finding lasting dense subgraphs[J]. Data Mining and Knowledge Discovery, 2019, 33(5): 1417-1445.

[22] LI Y, LIU J S, ZHAO H Q, et al. Research on Sustained Cohesive Subgraph Search Algorithm in Large Temporal Graphs[J]. Computer Engineering and Applications, 2021, 4: 1-11.

MU Cong-cong, born in 1995, postgraduate, is a member of China Computer Federation. His main research interests include temporal graph and graph data managment.

YUAN Ye, born in 1981, professor, is a member of China Computer Federation. His main research interests include graph databases, probabilistic databases, and social network analysis.