时序网络中短时社区搜索方法研究

顾天凯 王朝坤 楼昀恺
(清华大学软件学院 北京 100084)

摘 要 时序网络中的社区搜索问题旨在寻找符合一定时序规律的社区.短时交互特性作为时序网络的一种重要时序特征,相比于长期社区更具研究价值,可用于有效挖掘网络中核心的时序紧密结构.现有工作大多研究了时序社区的持续性、突变性、周期性等现象,尚无法建模时序社区的短时特性.针对现有工作难以满足上述需求的现状,提出 top-k 短时社区搜索这一新问题,为有效发现复杂网络中短时紧密社区提供新的解决思路.首先,针对时序网络中社区的短时特性,提出了 δ-短时社区的形式化定义用以刻画短时社区结构.同时,给出了不同时间交互下形成的时序社区的时间跨度计算方式,为衡量短时社区提供一个具体的量化指标.其次,提出了 top-k δ-短时社区搜索算法 *ShrimeCS*,分析并讨论了短时社区的判断条件分别用以判断最小 δ-短时社区和 top-k δ-短时社区,并提出了 δ-基本块结构结合判断条件用以找到最小 δ-短时社区.此外,进一步提出了强 δ-基本块结构以避免在扩展过程中出现子图冗余,从而降低搜索过程的时间开销.还分析了 top-k δ-短时社区搜索过程中基于全局时间跨度上界与渐进时间跨度上界优化的启发式计算方法,以进一步加快算法运行效率,搜索优化率相比原算法提高 64.2%以上.然后,在 5 个真实数据集和 3 个合成数据集上进行了实验,并提出了聚集因子指标用以评估时序社区中成员交互时间的接近程度.实验结果显示,*ShrimeCS* 找到社区的短时性优于基准方法,基于全局时间跨度和渐进时间跨度上界优化可以降低 17.16%以上的时间开销.在真实场景中的案例研究表明,*ShrimeCS* 找到的 top-k δ-短时社区可以捕捉到社区中时间跨度的变化,可以用来探索社区随时间的演化情况.最后,验证了所提方法的正确性,并表明 *ShrimeCS* 算法具有较好的可扩展性.

关键词 时序图;社交网络;社区搜索;短时社区;top-k 搜索
中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2022.00334

Research on Short-Time Community Search in Temporal Networks

GU Tian-Kai WANG Chao-Kun LOU Yun-Kai
(*School of Software, Tsinghua University, Beijing* 100084)

Abstract The community search problem in temporal social networks aims to find communities that can satisfy certain temporal phenomena. Short-time interactions between members are important temporal features of social networks. Compared to long-term communities, communities formed in a short time have more important research interests, as they can be used to effectively discover the potential core short-time structure with cohesiveness guaranteed in the network. However, most recent works focus on studying the phenomena of the persistent property, the bursting property and the periodic property of the temporal communities, these works fail to model the short-time phenomenon in the temporal network. In view of the current situation that it is difficult for existing works to meet the above requirements, a new problem of top-k short-time community search is proposed. At the same time, a novel and effective solution is provided for finding short-time communities in complex networks. Firstly, in order to depict the short-time phenomenon in the

收稿日期:2020-08-31;在线发布日期:2021-04-12.本课题得到国家自然科学基金(61872207)和百度资助.**顾天凯**,硕士研究生,主要研究方向为图数据、时序社区搜索与图神经网络等.E-mail:gtk18@tsinghua.org.cn.**王朝坤**(通信作者),博士,副教授,中国计算机学会(CCF)高级会员,主要研究方向为图和社交数据管理、大数据系统.E-mail:chaokun@tsinghua.edu.cn.**楼昀恺**,博士研究生,主要研究方向为图数据库、社区发现等.

temporal network, a formal definition of δ short-time community is proposed to explore the short-time structure, and a method for calculating the time interval of a temporal community which is formed under different time interactions is given to precisely measure the short-time indicator. Secondly, the top-k δ short-time community search algorithm *ShrimeCS* is proposed. The conditions to judge the minimal δ short-time community and top-k δ short-time community are carefully analyzed and discussed. After that, the δ basic block is proposed. It can be used to find the minimal δ short-time community with the help of the minimal δ short-time community judgment condition. Furthermore, the concept of δ strong block is proposed to avoid redundancy during subgraph expansion. Through utilizing the non-overlapping property of the δ strong block, the time cost can be largely reduced. In addition, we analyze and propose two heuristic approaches, which are the global time interval-based optimization approach and the progressive time interval-based optimization approach, to further improve the efficiency of *ShrimeCS*. Compared to the original algorithms, the two proposed optimization approaches can improve the optimization rate during the searching process by over 64.2%. Then, the experiments are conducted on five real-world datasets (i.e., Email, Msg, Math, SuperUser and DBLP) and three synthetic datasets. At the same time, a new metric, clustering factor (i.e., CT-factor), which is to evaluate the closeness of the interaction between two members in the temporal community, is proposed. The experimental results show that the communities found by *ShrimeCS* have better short-time features compared to other baselines. Moreover, the results demonstrate that the proposed optimization approaches (i.e., the global time interval-based optimization approach and progressive time interval-based optimization approach) can reduce the time cost by 17.16%. The case study in a real-world application scenario shows that the top-k δ short-time communities which are found by *ShrimeCS* can capture the dynamics of the time interval shifted in the community, and it is therefore beneficial to explore the community evolution through the time dimension. Finally, the correctness of *ShrimeCS* is verified, and the good scalability of *ShrimeCS* is demonstrated.

Keywords　temporal graph; social network; community search; short-time community; top-k search

1　引　言

社区是社交网络的一个重要属性. 如何挖掘社交网络图数据中的社区结构是近年来的研究热点[1]. 通常,社区具有社区内节点连接较为紧密,社区间节点连接较为松散的特点. 研究者可以借助社区结构进行大量实际应用,例如社交网络中基于条件的社区发现[2]、基于社区结构优化子图匹配[3]、层次化社区发现[4]以及利用网络上的时序属性进行社区搜索[5]等.

网络图数据的时序性近几年广受学者关注[6-9]. 在时序网络(亦称时序图)中,节点间的边具有时间属性,代表两个节点在某一或某些时刻(即时间戳)产生关联. 在时序图中存在许多与时序性相关的社区搜索问题,例如在时序动物行为网络中寻找周期性出现的动物路线以探寻周期性的迁徙现象[6],在时序车辆行驶网络中寻找某一时段内车辆数目增长率最大的路线[7],或者在时序社交网络中找出任意一个固定长度的子区间都能满足拓扑约束的时序紧密型社区[8]. 然而,在实际应用中还会遇到一类问题:如何找到一个时序社区既满足拓扑结构约束,同时社区内成员间产生关联的时刻尽量接近. 本文将这类问题称为短时社区搜索问题(Short-Time Community Search,*STCS*).

例 1.　资金交易网络中的不法洗钱行为检测. 在银行或电子金融平台对应的资金交易网络中,节点代表用户账户,时序边代表在对应时刻两个用户账户间发生资金交易. 不法分子洗钱时,通常会在较短时间内使用多个用户账户相互转账[10]. 图 1 所示

的资金交易网络中包含 9 个节点和 24 笔交易. 给定待关注账户 A,若采取基于结构的 4-*truss* 社区搜索算法[11],将得到社区$\{A,B,C,D,E,F,G,H,I\}$,这是因为该社区内任意一条边均可构成至少两个三角形. 然而因为该社区考虑了一些较远的转账记录(如最早转账记录 FI 和最晚转账记录 AE),使得社区的时间跨度长达十个月,所以无法反映真实的洗钱行为. 为了更准确发现洗钱行为,通过缩小时序社区的时间跨度我们可以得到社区 $\mathcal{C}_1=\{A,B,C,D\}$ 和 $\mathcal{C}_2=\{A,D,G,H\}$,这两个社区分别涉及 6 笔交易与 7 笔交易,且其转账记录分别集中在 5 月 19 日 14:25～15:00 间和 5 月 17 日 18:00～18:45 间. 显然,$\mathcal{C}_1$ 和 $\mathcal{C}_2$ 具有明显频繁交易模式,符合不法用户的洗钱特征,因此可将两个社区内其他账户$\{B,C,D,G,H\}$作为可能的违法账户,并开展进一步监控与分析.

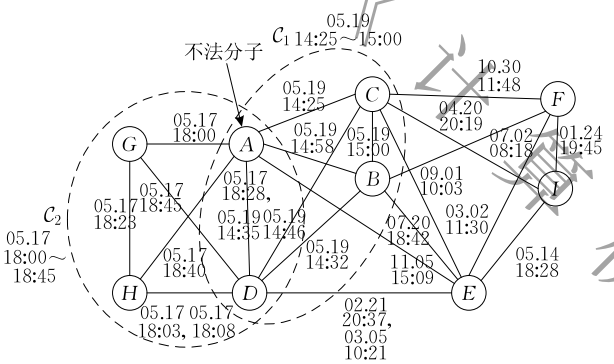

图 1 某资金交易网络(2019 年)的短时社区搜索示例

例 1 所示资金交易网络中的不法洗钱行为是短时社区搜索问题的一个具体实例,该问题在现实生活中存在较多实际应用场景. 然而现有时序社区搜索方法[6-8]均无法有效解决上述问题. 例如,持久性社区(Persistent Community,PC)搜索[8]用于在网络中找到在定长时间窗口 θ 内满足结构约束的最大社区. 但是,PC 搜索无法解决短时社区搜索问题,主要原因在于:首先 PC 搜索没有指定查询节点,因此其找到的社区可能与监控账户无关;其次,PC 搜索仅返回符合时间约束的最大社区,若监控账户在多个时间区间与不同账户进行频繁洗钱交易,则 PC 搜索无法找到其他所有账户. 最后,θ 的选取与实际应用场景密切相关,普通用户很难选取合适的 θ 值.

因此,尽管现有社区搜索方法较为成熟,但大部分工作[11-16]只考虑结构上的紧密性,未考虑时序信息,无法有效应用于时序社交网络;而考虑时序信息的已有社区搜索方法[5-8]尚无法准确刻画时序网络中的短时特征. 为此,本文提出短时社区搜索问题,旨在发现时序社交网络中同时满足拓扑结构约束以及时间跨度约束的社区.

具体地,本文的主要贡献包括:

(1) 针对时序社交网络中存在的短时社区(Short-Time Community,STC),首次提出 top-k δ-短时社区搜索问题(top-k Short-Time Community Search,top-k STCS)并给出时间跨度计算方法;

(2) 分析并证明了最小 δ-短时社区(Minimum δ-Short-Time Community,MSTC)在扩展过程中时间跨度具有单调性,并给出 MSTC 的判断条件. 进而提出 top-k *STCS* 算法 *ShrimeCS*(Short Time Community Search)以解决不同时间跨度的社区搜索问题;

(3) 提出强 δ-基本块更新算法,可以高效找到候选最小 δ-短时社区,从而降低后续搜索的空间规模. 同时,分析并证明 top-k *STCS* 具有全局时间跨度上界和渐进时间跨度上界,以此进一步优化时序社交网络规模,提升 *ShrimeCS* 的运行效率;

(4) 5 个真实数据集和 3 个合成数据集上的实验结果表明,*ShrimeCS* 可以找到符合时间跨度约束的社区,相比基准算法在时序社交网络上能计算得到更好的社区结构,并具有良好的可扩展性.

本文第 2 节介绍社区搜索与时序社区搜索的相关工作;第 3 节给出基本概念,形式化定义 δ-短时社区并提出 top-k *STCS* 问题;第 4 节提出 *MSTC* 的判定条件并给出 top-k *STCS* 算法 *ShrimeCS*;第 5 节进行算法优化,提出强 δ-基本块降低搜索开销,并利用全局上界优化和渐进上界优化提升算法运行效率;第 6 节开展实验并分析实验结果;第 7 节总结全文.

2 相关工作

2.1 社区搜索

在网络图数据中,社区这一概念尚无统一的形式化定义. 通常认为,社区是内部连接紧密而外部连接松散的子图. 因此,已有工作往往通过约束图上的拓扑结构来找到符合特定结构的社区. 常见的拓扑结构约束包括 k-*clique*[12]、k-*core*[13-17] 和 k-*truss*[11,18-19]等. 例如,Cui 等人提出重叠社区搜索算法(OCS)[12],用以发现节点所在的多个社区. Zhang 等人为区分社区内部用户参与度的问题,提出高效的 (k,p)-*core* 模型[17]. Huang 等人通过 k-*truss* 改进 k-*clique* 结构对社区搜索算法带来的计算开销,此外他们通过一种高效的索引结构

TCP-Index 用以提高算法的运行效率，并提出了基于 *TCP-Index* 的社区搜索算法用于找到包含查询节点的多个社区[11]. *EquiTruss* 进一步改进了 *k-truss* 索引结构，避免了索引的重复存储从而减少了索引的空间开销[18]. 单菁等人基于重叠社区衡量节点影响力，提出传播热点选择方法，有效解决信息传播问题[20].

另一类带有节点属性的社区搜索问题通常在找到的社区满足结构约束条件外，还要求社区中节点的属性具有一定的相似度[21-22]. 例如 Chen 等人[21]给出了 CC(Contextual Community)模型，并基于节点属性给出了社区内属性密度计算方法，该方法将返回包含查询节点且具有最大属性密度的社区. Zhang 等人[22]在 *k-core* 的基础上，设计了 KCC(Keyword-Centric)结构，该结构计算了关键词间的距离和关键词的接近程度，最后他们设计对应算法 KKCS 以找到与查询关键词最相近的社区.

2.2　时序社区搜索

时序图上的社区搜索问题是近年来的研究热点之一. 时序社区搜索旨在从时序网络中发现符合指定时序规律的社区结构. 因此在时序社区搜索方法中，不仅要关注社区的拓扑结构，还要关注社区的时序特性. 目前，常见的社区时序特性有给定时间段内的频繁性、较长时间范围内的周期性、极短时间段内的突变性等. 例如，针对时序社区在一段时间内持续出现的现象，Li 等人[8]提出 (θ,γ) *k-core* 时序社区结构，要求该结构在至少 γ 长时间段内的每个定长时间窗口 θ 中的子结构皆满足 *k-core* 约束，同时提出 TGR 算法简约时序网络规模，进而提出 PC 算法解决持久性社区搜索问题.

时序社区的周期性指的是在较长时间范围内时序社区中的每条边都以固定时间间隔出现. 为了发现时序网络中的周期性社区，Qin 等人[6]首先通过构建子图的时间支持集找到子图频繁出现的时间间隔，进而提出了基于 SPCore 的周期性社区发现算法 MPC. 因为短时社区不要求节点之间存在周期性交互，且 MPC 算法不能保证结果社区中成员的交互时间尽可能短，所以周期性社区搜索方法无法用于解决本文所提出的短时社区搜索问题.

时序社区的突变性指的是时序网络中在极短时间段内快速形成对应子图的现象. 针对时序社区的突变性，Chu 等人[7]构建了网络图的紧密度增长率模型，即通过计算 t_h 时刻下子图的紧密度函数值 $q_x(t_h)$ 从而得到时间段 Δt 内紧密度的变化率，进而提出了 TopkDBSOL 算法用于在线挖掘时序网络中的突变社区[7]. 然而，该算法不能用于解决短时社区搜索问题. 原因如下，首先该算法发现的社区不一定满足 *k-truss* 结构；其次 TopkDBSOL 算法要求返回具有最大紧密度增长率(即 $\frac{q_x(\Delta t)}{\Delta t}$)的 top-$k$ 个时序社区，而这些社区不能满足社区内成员交互时间尽可能短的要求；再次 TopkDBSOL 是一种近似方法，所得非精确解；最后该方法返回的社区不一定包含查询节点. 因此 TopkDBSOL 计算得到的社区无论在拓扑结构还是 top-k 约束条件上与短时社区均有所不同.

时序图数据上还有其他一些相关研究工作. Bansal 等人[23]在由博客构成的时序图中提出一种高效的算法用以找到由相近关键词构成的社区，Wu 等人[16]研究了时序图上的 *k-core* 分解算法. Ma 等人[24]基于边权重动态变化的时序图，提出了密集子图发现算法. Wu[25]与 Huang 等人[26]则分别研究了时序图上的最短路问题以及最小生成树问题. 然而这些工作也都不能直接用于解决本文所提出的 top-k *STCS* 问题.

3　基本概念

定义 $\mathcal{G}=(\mathcal{V},\mathcal{E})$ 为一个时序网络，其中 $\mathcal{V}$ 和 $\mathcal{E}$ 分别是 $\mathcal{G}$ 的节点集和边集. $\mathcal{E}=\{(u,v,t_{(u,v)})\mid(u,v)\in\mathcal{V}\times\mathcal{V},t_{(u,v)}\in T_{(u,v)}\}$，$u,v$ 是 $\mathcal{G}$ 中的节点，$T_{(u,v)}$ 表示节点 u,v 在 $\mathcal{G}$ 中发生交互的时间戳集合，$|\mathcal{V}|$ 和 $|\mathcal{E}|$ 分别表示 $\mathcal{G}$ 中节点数量及边数量. 我们用 $G=(V,E)$ 表示不考虑 $\mathcal{G}$ 中边的时间戳属性而形成的基础图，其中 $V=\mathcal{V}$，$E=\{(u,v)\mid(u,v,t_{(u,v)})\in\mathcal{E}\}$. $N(v)$ 为节点 v 在 G 中的邻居节点集. 给定 $e\in E$，边 e 在 G 上的支持度定义为 G 中包括 e 的三角形的数量，记作 $sup(e,G)$. 文中一些重要符号及其说明见表 1.

表 1　重要符号表

符号	说明
$\mathcal{G}=(\mathcal{V},\mathcal{E})$	一个无向、时序网络
$G=(V,E)$	一个不考虑 $\mathcal{G}$ 中边的时间戳属性形成的基础图
$N(v)$	节点 v 在图 G 上的邻居节点集合
$T_{(u,v)}$	节点 u,v 在 $\mathcal{G}$ 中产生交互的时间戳集合
$sup(e,G)$	边 e 在 G 中形成三角形的数量
TS_g	g 的一个关联时刻集
$f(TS_g^*)$	g 的最小时间跨度
$scale(TS_g)$	g 中关联时刻集的容量
$\Delta_{f(TS_g)}(u,v)$	在 g 中向外扩展一条边 (u,v)，子图时间跨度增量的集合

定义 1. 关联时刻集. 给定时序图 $\mathcal{G}$ 的子图 $g=(\mathcal{V}_g,\mathcal{E}_g)$ 及 g 对应的基础图 g，$\forall(u,v)\in E_g$，定义 g 的一个关联时刻集 $TS_g=\bigcup_{(u,v)\in E_g}\{T'_{(u,v)}\}$，其中 $T'_{(u,v)}\subseteq T_{(u,v)}$ 且 $T'_{(u,v)}\neq\varnothing$.

例 2. 如图 2 所示，令图 g_1 为节点集合 $\{v_1,v_3,v_4\}$ 在时序图 $\mathcal{G}$ 上的导出子图，g_1 中共包含 3 个节点和 4 条时序边，则 g_1 共有三个关联时刻集，分别是 $\{\{2\},\{4\},\{5\}\}$、$\{\{2\},\{4\},\{9\}\}$ 和 $\{\{2\},\{4\},\{5,9\}\}$.

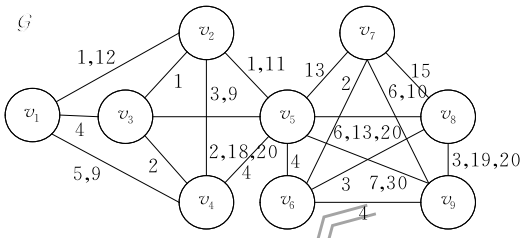

图 2 短时社区搜索的运行示例

定义 2. 时间跨度. 给定时序图 $\mathcal{G}$ 的子图 $g=(\mathcal{V}_g,\mathcal{E}_g)$，$|E_g|\geq 2$ 和 g 的一个关联时刻集 TS_g，g 在 TS_g 下的时间跨度定义为 $f(TS_g)=\max(\{t_{(u,v)}-t_{(u',v')}\mid\forall t_{(u,v)}\in T'_{(u,v)},\forall t_{(u',v')}\in T'_{(u',v')},(u,v)\neq(u',v')\wedge T'_{(u,v)}\in TS_g\wedge T'_{(u',v')}\in TS_g\})$.

值得注意的是，对于子图 g，若 $|E_g|<2$，我们约定其时间跨度为 $f(TS_g)=\infty$.

$f(TS_g)$ 刻画了 g 中边的时间戳跨度的最大值. 为便于表述，记 TS_g 中最大时间戳为 $\max(TS_g)$，最小时间戳为 $\min(TS_g)$，同时记 TS_g 的容量为 $scale(TS_g)=\sum_i|T'_{e_i}|$，$\forall T'_{e_i}\in TS_g$，即关联时刻集中各时间戳集合的规模的总和.

例 3. 如图 2 所示，令图 g_1 为节点集合 $\{v_1,v_3,v_4\}$ 在 $\mathcal{G}$ 上的导出子图，g_1 的一个关联时刻集 $TS_{g_1}=\{T'_{(v_1,v_3)},T'_{(v_3,v_4)},T'_{(v_1,v_4)}\}=\{\{4\},\{2\},\{5,9\}\}$. 则 g_1 在 TS_{g_1} 下的时间跨度 $f(TS_{g_1})$ 为 7，最大时间戳 $\max(TS_{g_1})=9$，最小时间戳 $\min(TS_{g_1})=2$，$scale(TS_{g_1})=\sum_i|T'_{e_i}|=1+1+2=4$.

定义 3. 子图最小时间跨度. 给定时序图 $\mathcal{G}$ 的子图 $g=(\mathcal{V}_g,\mathcal{E}_g)$，$|E_g|\geq 2$ 和 g 中所有关联时刻集构成的集合 $\mathcal{T}_g$，g 的最小时间跨度定义为 $f(TS_g^*)=\min(\{f(TS_g)\mid\forall TS_g\in\mathcal{T}_g\})$.

例 4. 在例 2 中，g_1 共有三个关联时刻集，时间跨度分别是 3、7、7，因此 g_1 的最小时间跨度 $f(TS_{g_1}^*)=3$.

重叠社区搜索问题是在网络中找到包含查询节点的所有社区[12,27]. 在结构条件约束中，由于 δ-*truss* 相比于其他约束条件，例如 δ-*core* 和 δ-*clique*，能较好均衡子图结构的紧密程度和计算复杂度[11]，因此，我们主要针对重叠社区结构基于 δ-*truss* 的社区搜索算法开展短时社区的研究.

定义 4. δ-短时社区. 给定时序图 $\mathcal{G}=(\mathcal{V},\mathcal{E})$ 和参数 $\delta\geq 2$，子图 $\mathcal{C}\subset\mathcal{G}$，$\mathcal{C}=(\mathcal{V}_c,\mathcal{E}_c)$，称 $\mathcal{C}$ 是一个 δ-短时社区当且仅当满足以下条件：

(1) δ-*truss*. $\forall e\in E_c$，$sup(e,C)\geq\delta-2$.

(2) 三角传递性. $\forall e_1,e_2\in E_c$，在 $\mathcal{C}$ 的基础图中 $\exists$ 三角形 $\triangle_s(e_1\in\triangle_s)$ 和三角形 $\triangle_t(e_2\in\triangle_t)$，满足 $\triangle_s=\triangle_t$ 或 $\exists\triangle_i,\triangle_{i+1},\cdots,\triangle_{i+k}$，使 $\triangle_s\cap\triangle_i\neq\varnothing$，$\triangle_t\cap\triangle_{i+k}\neq\varnothing$，$\triangle_{i+j}\cap\triangle_{i+j+1}\neq\varnothing$ 成立，其中交集不为空代表两个三角形之间存在公共边，$k\geq 0$，$j\in[0,1,\cdots,k-1]$.

(3) 最大时序子图. $\nexists\mathcal{C}'\subseteq\mathcal{G}$，$\mathcal{C}'$ 满足条件(1)和条件(2)，使得 $\mathcal{C}\subseteq\mathcal{C}'$，且 $f(TS_{\mathcal{C}}^*)=f(TS_{\mathcal{C}'}^*)$ 成立.

(4) 最大时序边容量. $\nexists TS_{\mathcal{C}}^-$，$TS_{\mathcal{C}}^-$ 满足 $f(TS_{\mathcal{C}}^-)=f(TS_{\mathcal{C}}^*)$，使得 $scale(TS_{\mathcal{C}}^-)>scale(TS_{\mathcal{C}}^*)$ 成立.

δ-短时社区在结构和时间跨度上都对社区 $\mathcal{C}$ 做出要求. 条件(1)和条件(2)使得 δ-短时社区满足 δ-*truss* 和三角传递性，条件(3)使得 $\mathcal{C}$ 是在时间跨度 $f(TS_{\mathcal{C}}^*)$ 下的最大子图，最后条件(4)要求关联时刻集 $TS_{\mathcal{C}}^*$ 在 $\mathcal{C}$ 的所有相同时间跨度中有最大时序边容量，这些约束可以有效避免冗余社区. 以条件(4)为例，在图 1 中，因为节点 H 与 D 之间分别在 5 月 17 日的 18:03 与 18:08 都存在转账记录，所以从两个时间戳中任选其一形成的关联时刻集 $TS'_{\mathcal{C}}$ 依旧满足 $f(TS'_{\mathcal{C}})=f(TS_{\mathcal{C}}^*)$，但是由于 $scale(TS'_{\mathcal{C}})<scale(TS_{\mathcal{C}}^*)$，导致 $\mathcal{C}'$ 无法满足 δ-短时社区的定义. 条件(4)从最大时序边容量的角度避免了此类社区的产生. 满足条件(1)和条件(2)的最大社区称为 δ-*truss* 社区[11].

基于上述定义，top-k δ-短时社区搜索问题定义为：给定图 $\mathcal{G}=(\mathcal{V},\mathcal{E})$，参数 δ、k 和查询节点 v_q，top-k δ-短时社区搜索问题(top-k *STCS*)是找到 v_q 所在的 k 个最小时间跨度的 δ-短时社区.

例 5. 以图 2 为例，查询节点为 v_5. top-3 4-短时社区分别为 $\{v_2,v_3,v_4,v_5\}$，$\{v_5,v_6,v_8,v_9\}$，$\{v_1,v_2,v_3,v_4,v_5\}$，其最小时间跨度分别是 3、4、4.

4　δ-短时社区搜索算法

top-k $STCS$ 的一个基本问题是找到最小 δ-短时社区($MSTC$). 本节先通过分析 $MSTC$ 的一些性质给出 $MSTC$ 的判定条件，最后通过将 $MSTC$ 扩展至 top-k 短时社区的判定条件，引出直接基于判定条件的短时社区搜索方法 $BasicSearch$.

4.1　最小 δ-短时社区的判断条件

本小节首先给出 δ-候选块以及 δ-基本块的定义并结合图示加以说明，然后提出 δ-基本块扩展过程中最小时间跨度的单调递增性.

定义 5.　δ-候选块. 给定 δ-$truss$ 社区 $\mathcal{D}=(\mathcal{V}_{\mathcal{D}},\mathcal{E}_{\mathcal{D}})$ 和查询节点 v_q，称 $\mathcal{B}$ 是 $\mathcal{D}$ 中针对 v_q 的一个 δ-候选块，如果(1) $\mathcal{B}\subseteq\mathcal{D}$，且(2) $\mathcal{B}=(\mathcal{V}_{\mathcal{B}},\mathcal{E}_{\mathcal{B}})$，$v_q\in\mathcal{V}_{\mathcal{B}}$，$\mathcal{B}$ 满足 δ-$truss$.

定义 6.　δ-基本块. 给定 δ-$truss$ 社区 $\mathcal{D}=(\mathcal{V}_{\mathcal{D}},\mathcal{E}_{\mathcal{D}})$，查询节点 v_q 和 $\mathcal{D}$ 中针对 v_q 的一个 δ-候选块 $\mathcal{B}$，称 $\mathcal{B}$ 为一个 δ-基本块，如果 $\mathcal{B}$ 中没有其他针对 v_q 的 δ-候选块.

例 6.　以图 3 中社区 $\mathcal{C}_1$ 为例，$v_q=v_5$. 在 4-$truss$ 社区 $\mathcal{C}_1=\{v_1,v_2,v_3,v_4,v_5\}$ 中，4-基本块为 $\mathcal{B}=\{v_2,v_3,v_4,v_5\}$，因为 $\mathcal{B}$ 满足 4-$truss$ 结构且 $\mathcal{B}$ 不具有其他 4-候选块.

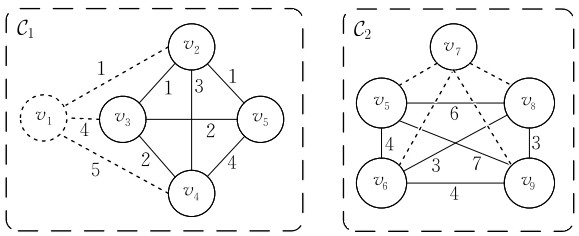

图 3　寻找最小 δ-短时社区的示意

我们首先给出 $MSTC$ 的判定条件. 给定图 $\mathcal{G}=(\mathcal{V},\mathcal{E})$，$MSTC=(\mathcal{V}_{\mathcal{M}},\mathcal{E}_{\mathcal{M}})$ 与 $TS^*_{MSTC}(\nexists TS'_{MSTC}, scale(TS'_{MSTC})>scale(TS^*_{MSTC}))$. $MSTC$ 应满足：(1) $\nexists\,\mathcal{B}\in\mathcal{G}$，$\mathcal{B}$ 是 δ-基本块，使得 $f(TS^*_{\mathcal{B}})<f(TS^*_{MSTC})$，且(2) $\nexists MSTC'\in\mathcal{G}$，$MSTC\subset MSTC'$，有 $f(TS^*_{MSTC'})=f(TS^*_{MSTC})$ 成立.

$MSTC$ 的判定条件给出了判断一个社区 $\mathcal{C}$ 是 $MSTC$ 的两个要求，即 $\mathcal{C}$ 相比 $\mathcal{G}$ 的其他社区有最小的时间跨度，并且 $\mathcal{C}$ 在该时间跨度下是容量最大的社区. 给定 $k=1$，$v_q=v_5$，$\delta=4$，图 3 展示了 v_5 所在的两个 4-$truss$ 社区 $\mathcal{C}_1=\{v_1,v_2,v_3,v_4,v_5\}$ 和 $\mathcal{C}_2=\{v_5,v_6,v_7,v_8,v_9\}$. 在图 3 的社区 $\mathcal{C}_1$ 中，4-基本块 $\mathcal{B}_1^{\min}$ 节点集合为 $\{v_2,v_3,v_4,v_5\}$，$f(TS^*_{\mathcal{B}_1^{\min}})=3$. 若考虑节点 v_1，$\mathcal{B}_1^{\min}$ 将扩展至 $\mathcal{C}_1$. 因为 $T'_{(v_1,v_4)}=\{5\}$，所以 $\mathcal{C}_1$ 中最小时间跨度 $f(TS^*_{\mathcal{C}_1})=4$. 在图 3 的社区 $\mathcal{C}_2$ 中，4-基本块共有 4 个，其中 $\mathcal{B}_2^{\min}=\{v_5,v_6,v_8,v_9\}$ 时间跨度最小，$f(TS^*_{\mathcal{B}_2^{\min}})=4$. 上述说明 $\mathcal{B}_1^{\min}$ 是时间跨度 $f(TS^*_{\mathcal{B}_1^{\min}})$ 下的最大子图，同时 $f(TS^*_{\mathcal{B}_1^{\min}})$ 在 v_5 所处的其他社区中有最小的时间跨度，因此 $MSTC$ 是 $\mathcal{B}_1^{\min}$，时间跨度是 $f(TS^*_{\mathcal{B}_1^{\min}})$.

定理 1.　δ-基本块 $\mathcal{B}(\mathcal{B}\subseteq\mathcal{G})$ 在 $\mathcal{B}$ 不断扩展的过程中最小时间跨度单调递增.

证明.　因篇幅限制，该部分证明请参看附录. 后续定理的证明同样在附录中给出.

4.2　top-k δ-短时社区搜索算法

基于定理 1，本小节将 $MSTC$ 的判定条件扩展至 top-k δ-短时社区的判定条件，并给出基于该判定条件的短时社区搜索方法.

具体地，依据如下方式将 $MSTC$ 的判定条件扩展至 top-k δ-短时社区：对于子图 $\mathcal{G}_k$ 和关联时刻集 $TS^*_{\mathcal{G}_k}(\nexists TS'_{\mathcal{G}_k}, scale(TS'_{\mathcal{G}_k})>scale(TS^*_{\mathcal{G}_k}))$，$\mathcal{G}_k$ 为第 k 最小 δ-短时社区当且仅当满足以下条件：(1) $\exists\mathcal{G}_1$，$\mathcal{G}_2,\cdots,\mathcal{G}_{k-1}$ 和关联时刻集 $TS^*_{\mathcal{G}_1},TS^*_{\mathcal{G}_2},\cdots,TS^*_{\mathcal{G}_{k-1}}$ 是 top-$(k-1)$ 最小 δ-短时社区，且有 $f(TS^*_{\mathcal{G}_1})\le f(TS^*_{\mathcal{G}_2})\le\cdots\le f(TS^*_{\mathcal{G}_{k-1}})$ 成立；(2) $\nexists\,\mathcal{G}'\subseteq\mathcal{G}$ 和 $TS^*_{\mathcal{G}'}$，$f(TS^*_{\mathcal{G}_{k-1}})<f(TS^*_{\mathcal{G}'})<f(TS^*_{\mathcal{G}_k})$ 成立；(3) $\nexists\,\mathcal{G}_{k'}\supset\mathcal{G}_k$ 和 $TS^*_{\mathcal{G}_{k'}}$，$f(TS^*_{\mathcal{G}_{k'}})=f(TS^*_{\mathcal{G}_k})$ 成立.

算法 1 给出了一种基于 top-k δ-短时社区判定条件下短时社区搜索问题的直接解决方案. 在初始化 top-k 短时社区集合，候选社区队列和当前社区计数后，算法首先在 $\mathcal{G}$ 上执行社区搜索算法找到查询节点所在的最大社区(第 1～3 行)，这里我们采用基于 δ-$truss$ 的社区搜索算法 TCP-$Index$[11] 或 $EquiTruss$[18]，在 δ-$truss$ 搜索过程中，我们构建了考虑时间因素的索引结构，以便降低在计算子图时间跨度时的时间开销. 接着遍历每个社区，找到社区中所有的 δ-基本块并分别以每个 δ-基本块为基础扩展为 k 个候选短时社区，将其加入队列 BQ 中(第 4～10 行). 最后，算法通过判定条件依次判断候选短时社区是否满足要求并保持队列的升序性质，返回所有满足条件的 top-k 个短时社区(第 11～32 行). 该阶段主要由两部分组成，分别是子图扩展(第 19～25 行)和短时社区判定(26～30 行). 在子图扩展过程中，算法根据当前子图(初始为 δ-基本块)不断扩展，寻找新的子图 $\mathcal{B}'$ 直至 $\mathcal{B}'$ 重新满足 δ-$truss$，

注意第 24 行的计算时序边 *trussness* 方法可参考文献[11],本文不再赘述.短时社区判定阶段则根据短时社区判定条件,由 $\mathcal{B}'$ 判定 $\mathcal{B}$ 是否满足短时社区条件,并在队列中加入 $\mathcal{B}'$ 直至 $\mathcal{B}'$ 达到最大社区规模 C.

算法 1. BasicSearch 方法.

输入:时序社交网络 $\mathcal{G}=(\mathcal{V},\mathcal{E})$,整数 $\delta(\delta\geq 2)$,查询节点 v_q 和参数 k

输出:$\mathcal{G}$ 中 top-k 短时社区集合 *MSTCs*

1. *MSTCs*←∅
2. *BQ*←∅,*cur_topk*=0//初始候选社区队列和计数变量
3. *Cs*←*CommunitySearch*(v_q,$\mathcal{G}$) //构建考虑时间因素的索引 *TCP-Index* 或 *EquiTruss*,根据查询节点 v_q 和图 $\mathcal{G}$ 返回 v_q 所在的所有 δ-*truss* 社区 *Cs*
4. FOR EACH *C* IN *Cs* DO //遍历每一个社区
5. 计算社区 *C* 的 *truss*,记为 *CTruss*
6. $\mathcal{B}Set$←*enumBlocks*(v_q,$N(v_q)$,δ) //详见算法 2
7. FOR EACH $\mathcal{B}$ IN $\mathcal{B}Set$ DO //遍历每一个δ-基本块
8. (*sps*,*ifs*)←*enumTopkTCCPeriod*($\mathcal{B}$,*info*,*k*) //详见算法 3
9. FOR EACH (*span*,*info*)∈(*sps*,*ifs*) DO
10. *BQ*←*BQ*∪{(*C*,$\mathcal{B}$,*CTruss*,*span*,*info*)}
11. 按 *BQ* 中 *info.max*−*info.min* 数值升序排序
12. WHILE *BQ*≠∅ DO //若队列中候选短时社区不为空
13. (*C*,$\mathcal{B}$,*Truss*,*span*,*info*)←*BQ.pop*()
14. IF $\mathcal{B}$是短时社区 THEN //top-k 短时社区判定条件
15. IF *cur_topk*>*k* THEN BREAK
16. *MSTCs*←{(*span*,*info*)}∪*MSTCs*
17. *cur_topk*++
18. ELSE //未构成短时社区
19. $\mathcal{B}'=\mathcal{B}$
20. WHILE $\mathcal{B}'\subseteq C$
21. *outer*←{v_j | $\forall v_j\in N(v_i)$,$v_j\notin V_{\mathcal{B}'}$,$v_i\in V_{\mathcal{B}'}$}
22. FOR EACH *outerNode* IN *outer* DO
23. 把与 *outerNode* 相连的边加入 $\mathcal{B}'$
24. $\mathcal{B}truss$←计算 $\mathcal{B}'$ 中每条边的 *trussness*
25. IF $\forall e\in E(\mathcal{B}')$,$\mathcal{B}truss(e)\geq\delta$ BREAK
26. (*n_span*,*n_info*)←*updateTopkTCCPeriod*($\mathcal{B}'$,*span*,*info*,*k*)//详见算法 4
27. FOR EACH (*span*′,*info*′) IN (*n_span*,*n_info*) DO
28. *BQ*←*BQ*∪{(*C*,$\mathcal{B}'$,*span*′,*info*′)}
29. IF $\mathcal{B}$是短时社区 THEN //top-k 短时社区判定条件
30. *BQ*←*BQ*∪{(*C*,$\mathcal{B}$,*span*,*info*)}
31. 按 *BQ* 中 *info.max*−*info.min* 数值升序排序
32. RETURN *MSTCs*

算法 2 首先初始化 $\delta Cands$ 和 δ-基本块的节点集合 *pans*(第 1 行),然后进入子过程(第 2 行).子过程中基线情况为:若 *pans* 中元素恰好为 δ,则将当前集合并入 $\delta Nodes$ 并回退(第 10～13 行);否则根据当前节点位置递归遍历后续节点,寻找可能节点组合(第 14～18 行).将 v_q 加入子过程返回的节点候选集合 $\delta Cands$ 中,构成 $|\delta|$ 个节点,并检查当前候选组合是否满足 δ-基本块的定义(第 4～7 行),最后把满足条件的候选集合 $\delta Cands$ 加入到 $\delta Nodes$ 中(第 8～9 行).

算法 2. enumBlocks 方法.

输入:查询节点 v_q,邻居节点集合 $N(v_q)$,参数 $\delta(\delta\geq 2)$

输出:各 δ-基本块的节点集合 $\delta Nodes$

1. $\delta Cands$←∅,$\delta Nodes$←∅,*pans*←∅
2. *enumsubBlocks*($N(v_q)$,$\delta-1$,*pans*,0,$\delta Cands$)//详见子过程
3. FOR EACH $\delta Cand$ IN $\delta Cands$ DO //遍历候选节点集合
4. 将节点 v_q 加入 $\delta Cand$
5. *verEg*←$|V_{\delta Cand}|*(|V_{\delta Cand}|-1)/2$
6. *relEg*←$|\{(u,v)\,|\,(u,v)\in E_{\delta Cand},\forall u,v\in V_{\delta Cand}\}|$ //计算 $\delta Cand$ 的边数
7. IF *verEg*==*relEg* THEN
8. $\delta Nodes$←$\delta Nodes$∪{$\delta Cand$}
9. RETURN $\delta Nodes$

子过程*enumsubBlocks*($N(v_q)$,δ,*pans*,*counter*,$\delta Cand$)

10. IF *size*(*pans*)==δ THEN
11. $\delta Cands$←$\delta Cands$∪*pans*
12. 删除 *pans* 中的最后一个元素
13. RETURN
14. WHILE *counter*<|$N(v_q)$| DO
15. *pans*←*pans*∪$N(v_q)$[*counter*]
16. *enumsubBlock*($N(v_q)$,δ,*pans*,*counter*+1,$\delta Cands$)
17. 删除 *pans* 中的最后一个元素
18. RETURN

算法 3 初始化短时社区 $span_{\mathcal{B}}$ 和跨度极值 $info_{\mathcal{B}}$,其中 $info_{\mathcal{B}}$ 是存储时间戳最小值 *info.min* 和时间戳最大值 *info.max* 的二元组集合(第 1 行).遍历子图 $\mathcal{B}$ 的时序边,将所有处于跨度极值之间的时序边组成一个集合 *il*(第 2～3 行),算法分别将 *il* 及 *info* 存入 $span_{\mathcal{B}}$ 和 $info_{\mathcal{B}}$ 中(第 4～5 行).若 k 大于 1,则计算前 k 短时社区的最小时间跨度(第 6～10 行),并将其加入 $info_{\mathcal{B}}$(第 11～12 行).最后根据 $info_{\mathcal{B}}$ 的时间跨度找到对应的短时社区并存入 $span_{\mathcal{B}}$(第 13 行),最后返回结果($span_{\mathcal{B}}$,$info_{\mathcal{B}}$)(第 14 行).

算法 3.　enumTopkTCCPeriod 方法.

输入：当前子图 $\mathcal{B}$，$\mathcal{B}$ 的时间跨度极值 $info$，参数 k

输出：$\mathcal{B}$ 的前 k 个最小短时社区与时间跨度极值（$span_{\mathcal{B}}$，$info_{\mathcal{B}}$）

1. $span_{\mathcal{B}} \leftarrow \varnothing, info_{\mathcal{B}} \leftarrow \varnothing, il \leftarrow \varnothing$
2. FOR EACH $edge$ IN $\mathcal{B}.edges$ DO //遍历时序边
3. $il \leftarrow il \cup \{edge \mid edge.t \in [info.min, info.max]\}$
4. $span_{\mathcal{B}} \leftarrow span_{\mathcal{B}} \cup \{il\}$
5. $info_{\mathcal{B}} \leftarrow info_{\mathcal{B}} \cup \{info\}$
6. IF $k>1$ THEN
7. $topkInfo \leftarrow \varnothing$
8. FOR EACH $edge$ IN $\mathcal{B}.edges$ DO
9. 更新 $info_{\mathcal{B}}$ 中各个元素，并按时间跨度升序
10. $topkInfo$ 更新为前 k 个时间跨度极值
11. FOR EACH tf IN $topkInfo$ DO
12. $info_{\mathcal{B}} \leftarrow info_{\mathcal{B}} \cup \{tf\}$
13. 根据 $info_{\mathcal{B}}$ 补全对应的短时社区 $span_{\mathcal{B}}$
14. RETURN ($span_{\mathcal{B}}, info_{\mathcal{B}}$) //返回更新后的结果

算法 4 首先找到当前子图 $\mathcal{B}$ 的新增边并初始化（第 1 行），其中 $topkInfo$ 用以更新时维护最小时间跨度（第 2 行）. 其次，通过遍历新增边，不断更新当前的跨度极值，并按时间跨度升序维护当前子图 $\mathcal{B}$ 的前 k 小的时间跨度（第 3～7 行），然后将结果存入 $infos_{\mathcal{B}}$（第 8 行）. 最后，对于 $infos_{\mathcal{B}}$ 中每一个跨度极值 $info$，将出现在 $info.min$ 和 $info.max$ 之间的时序边存入 tl 中，然后把 tl 加入 $spans_{\mathcal{B}}$，得到当前跨度极值 $info$ 下的短时社区（第 8～12 行），最后返回（$spans_{\mathcal{B}}, info_{\mathcal{B}}$）（第 13 行）.

算法 4.　updateTopkTCCPeriod 方法.

输入：当前子图 $\mathcal{B}$，短时社区 $span$，短时社区 $span$ 的时间跨度极值 $info$，参数 k

输出：当前子图 $\mathcal{B}$ 的前 k 个短时社区 $spans_{\mathcal{B}}$ 和对应的 k 个时间跨度极值 $infos_{\mathcal{B}}$

1. $edges \leftarrow \{e \mid \forall e \in \mathcal{E}_{\mathcal{B}} \cap e \notin span\}$ //得到所有新增边
2. $topkInfo \leftarrow \varnothing, infos_{\mathcal{B}} \leftarrow \varnothing, spans_{\mathcal{B}} \leftarrow \varnothing$
3. FOR EACH $edge$ IN $edges$ DO //遍历时序边
4. 根据 $edge$ 更新 $info$，将结果存入 $topkInfo$
5. FOR EACH tf IN $topkInfo$ DO
6. IF $tf.max - tf.min \geq f(TS^*_{\mathcal{B}})$ THEN
7. $infos_{\mathcal{B}} \leftarrow infos_{\mathcal{B}} \cup \{tf\}$
8. $infos_{\mathcal{B}} \leftarrow$ 将 $infos_{\mathcal{B}}$ 按时间跨度升序并取出前 k 个元素
9. FOR EACH $info$ IN $infos_{\mathcal{B}}$ DO
10. $tl \leftarrow \varnothing$
11. FOR EACH $edge$ IN $edges$ DO //遍历时序边
12. $tl \leftarrow tl \cup \{edge \mid edge.t \geq info.min \wedge edge.t \leq info.max\}$ //加入符合条件的时序边
13. $spans_{\mathcal{B}} \leftarrow spans_{\mathcal{B}} \cup \{tl\}$ //当前 $info$ 下的短时社区
14. RETURN ($spans_{\mathcal{B}}, infos_{\mathcal{B}}$)

5　短时社区搜索算法优化

因为基准算法 *BasicSearch* 中需要遍历 $C(\delta, |N(v_q)|)$ 个 δ-基本块，所以直接根据定义计算开销较大. 观察到 top-k 问题中扩展短时社区 δ-基本块间可能存在重叠，本节据此提出强 δ-基本块与上界优化，进而提出相关算法以提高 *ShrimeCS* 性能，最后分析所提算法的时间复杂度.

5.1　强 δ-基本块的更新

本小节首先定义强 δ-基本块的概念及基本性质，并给出从 δ-基本块找到强 δ-基本块的算法，然后通过引入全局上界优化和渐进上界优化提升算法效率，提出短时社区搜索方法 *ShrimeCS*.

定义 7.　强 δ-基本块. 给定 δ-基本块 $\mathcal{B}$，查询节点 v_q 和除 $\mathcal{B}$ 以外的 δ-基本块集合 $\mathcal{S}$，其中 $\mathcal{S}$ 满足 $\forall \mathcal{B}' \in \mathcal{S}, (\mathcal{V}_{\mathcal{B}} - \{v_q\}) \cap (\mathcal{V}_{\mathcal{B}'} - \{v_q\}) \neq \varnothing \wedge \mathcal{B} \neq \mathcal{B}'$，我们称 $\mathcal{B}$ 为强 δ-基本块，若 $\mathcal{S}=\varnothing$ 或对于 $\forall \mathcal{B}' \in \mathcal{S}$，有 $f(TS^*_{\mathcal{B}}) < f(TS^*_{\mathcal{B}'})$ 成立.

性质.　非重叠性. 给定两个 δ-基本块 $\mathcal{B}_1, \mathcal{B}_2$，若 $\mathcal{B}_1, \mathcal{B}_2$ 除查询节点 v_q 外至少存在一个公共节点，则 $\mathcal{B}_1, \mathcal{B}_2$ 中有且仅有一个是强 δ-基本块.

例 7.　如图 4 所示，设 $v_q = v_5$，在 v_5 所在的社

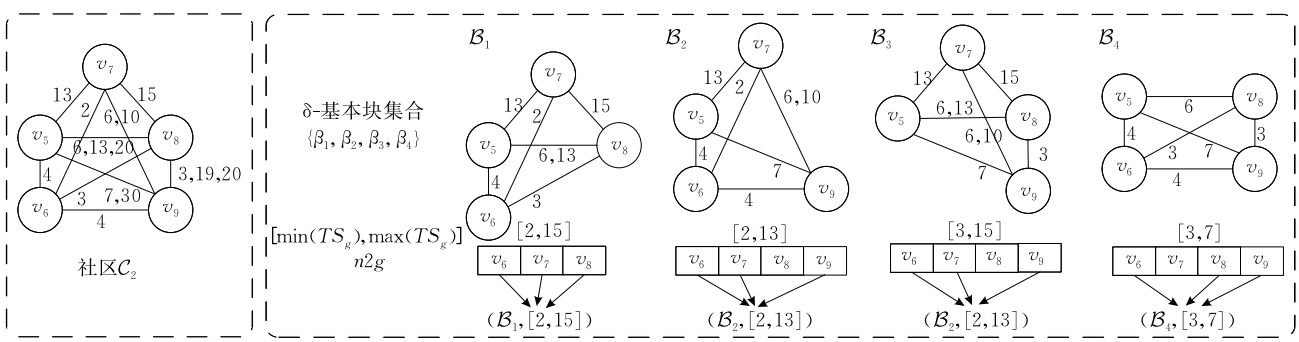

图 4　在所有 δ-基本块中找到强 δ-基本块的运行示例

区 C_2 中任意选取两个 δ-基本块，例如选取 $\mathcal{B}_1$ 和 $\mathcal{B}_4$，节点集合 $U_{\mathcal{B}_1}=\{v_5,v_6,v_7,v_8\}$，$U_{\mathcal{B}_4}=\{v_5,v_6,v_8,v_9\}$. $(U_{\mathcal{B}_1}-\{v_5\})\bigcap(U_{\mathcal{B}_4}-\{v_5\})=\{v_6,v_8\}$，$\mathcal{B}_1$，$\mathcal{B}_4$ 中存在公共节点，从而 $\mathcal{B}_1$，$\mathcal{B}_4$ 中存在强 δ-基本块. $\mathcal{B}_1$，$\mathcal{B}_4$ 的最小时间跨度分别是 13、4. 而 $\mathcal{B}_4$ 可扩展至 $\mathcal{B}'_4$，其中 $U'_{\mathcal{B}_4}=\{v_5,v_6,v_7,v_8,v_9\}$，$\mathcal{B}_1\subset\mathcal{B}'_4$，$f(TS^*_{\mathcal{B}'_4})=f(TS^*_{\mathcal{B}_1})=13$，即 $\mathcal{B}_4$ 在扩展中形成的子图 $\mathcal{B}'_4$ 与 $\mathcal{B}_1$ 有相同的时间跨度，因此 $\mathcal{B}_4$ 是强 δ-基本块.

基于强基本块的非重叠性，现在我们讨论重合节点属于不同 δ-基本块的情形. 假设 $\delta=4$，重合节点属于两个不同的 δ-基本块，其他参数情况类似.

如图 5 所示，给定 δ-基本块 $\mathcal{B}$，节点集合 $U_{\mathcal{B}}=\{v_q,v_b,v_c,v_d\}$. 假设目前的强 δ-基本块分别是 $U_{\mathcal{B}_1}=\{v_q,v_b,v_e,v_f\}$，最小时间跨度为 $|t_2-t_1|$ 和 $U_{\mathcal{B}_2}=\{v_q,v_c,v_g,v_h\}$，最小时间跨度为 $|t_4-t_3|$. 当考虑 $\mathcal{B}$ 时，强 δ-基本块可能会发生改变，存在以下三种不同情况.

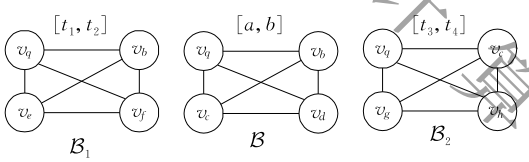

图 5 强 δ-基本块的三类更新策略，根据 $[a,b]$、$[t_1,t_2]$、$[t_3,t_4]$ 长度的不同情况选择更新策略

(1) $|b-a|\geq|t_2-t_1|$，$|b-a|\geq|t_4-t_3|$. 此时 $f(TS^*_{\mathcal{B}})$ 均大于 $\mathcal{B}_1$，$\mathcal{B}_2$ 的最小时间跨度，因此 $\mathcal{B}_1$ 和 $\mathcal{B}_2$ 可以在子图扩展过程中形成 $\mathcal{B}'\supset\mathcal{B}$，此时 $\mathcal{B}$ 不是强 δ-基本块，无需更新.

(2) $|b-a|<|t_2-t_1|$，$|b-a|<|t_4-t_3|$. 在这种情况下，$\forall\mathcal{B}_i\in\{\mathcal{B}_1,\mathcal{B}_2\}$，均有 $f(TS^*_{\mathcal{B}})<f(TS^*_{\mathcal{B}_i})$. 说明 $\mathcal{B}_1$，$\mathcal{B}_2$ 均可在 $\mathcal{B}$ 扩展过程中被包含，因此 $\mathcal{B}_1$ 和 $\mathcal{B}_2$ 不再是强 δ-基本块，删除 $\mathcal{B}_1$，$\mathcal{B}_2$，得到更新后强 δ-基本块为 $\mathcal{B}$.

(3) $|b-a|\geq|t_2-t_1|$，$|b-a|<|t_4-t_3|$（$|b-a|<|t_2-t_1|$，$|b-a|\geq|t_4-t_3|$ 情况类似）：此时，原先强 δ-基本块中，有 $f(TS^*_{\mathcal{B}_1})\leq f(TS^*_{\mathcal{B}})$，又有 $f(TS^*_{\mathcal{B}_2})>f(TS^*_{\mathcal{B}})$. 存在如下关系，$\mathcal{B}_1$ 在子图扩展过程中包含 $\mathcal{B}$，而 $\mathcal{B}$ 在子图扩展过程中包含 $\mathcal{B}_2$. 因此不加入 $\mathcal{B}$，同时删除 $\mathcal{B}_2$，即更新后强 δ-基本块是 $\mathcal{B}_1$.

算法 5 *CandidateBlock* 根据上述更新规则从 δ-基本块集合中找到强 δ-基本块集合. 首先初始化变量 $n2g$，其中 $n2g$ 存储重合节点 v，v 对应的子图 $\mathcal{B}$ 和最小时间跨度 $info$，$info$ 是时间戳最小值 $info.min$ 和时间戳最大值 $info.max$ 构成的二元组（第 1 行）. 其次算法遍历每个 δ-基本块，找到 $\mathcal{B}$ 与已存储强 δ-基本块之间重合的节点集合 cn（第 3 行）. 若 cn 为空，说明 $\mathcal{B}$ 与已存储强 δ-基本块间非重叠，加入到 $n2g$（第 4～5 行）；否则说明 $\mathcal{B}$ 与已存储强 δ-基本块之间存在节点重合，根据 $\mathcal{B}$ 的时间跨度与历史时间跨度间的关系，更新强 δ-基本块（第 6～14 行）.

算法 5. CandidateBlock 方法.

输入：δ-基本块集合 $\mathcal{BS}et$，各 δ-基本块对应的时间跨度极值 $info$ 以及查询节点 v_q

输出：强 δ-基本块集合 $SG\mathcal{BS}et$

```
1.  n2g ← ∅ //初始化变量用以存储重合节点
2.  FOR EACH B IN BSet DO
3.    cn ← {v | ∀v∈V_B, v∈n2g}
4.    IF cn == ∅ THEN
5.      n2g ← {(v, B, info) | v∈(V_B − v_q)} //更新 n2g
6.    ELSE
7.      sf ← false, lf ← false
8.      FOR EACH node IN cn DO
9.        找到当前重合节点对应的子图
10.       若B的时间跨度比子图对应的时间跨度大，将
          sf 置为 true，否则将 lf 置为 true
11.     IF sf 与 lf 皆为 true THEN
12.       根据第三种情况更新 n2g
13.     ELSE IF sf 为 true，lf 为 false THEN
14.       根据第二类情况更新 n2g
15. RETURN {(B, info) | ∀(v, B, info)∈n2g}
```

例 8. 如图 4 所示，设 $v_q=v_5$. 在 v_5 所在的社区 C_2 中共有 4 个 δ-基本块，分别是 $\mathcal{B}_1$，$\mathcal{B}_2$，$\mathcal{B}_3$，$\mathcal{B}_4$. 在算法 5 遍历 δ-基本块的过程中，$n2g$ 初始为空，因此 $n2g$ 首先存储 $\mathcal{B}_1$ 的节点与最小时间跨度，即 $(v_6,\mathcal{B}_1,[2,15])$，$(v_7,\mathcal{B}_1,[2,15])$ 和 $(v_8,\mathcal{B}_1,[2,15])$（第 4～5 行）. 当遍历至 $\mathcal{B}_2$ 时，由于 $\mathcal{B}_2$ 和 $n2g$ 之间存在相同的节点（第 3 行），即重合节点集合为 $\{v_6,v_7\}$，算法通过 $n2g$ 找到历史记录的强 δ-基本块（第 8～9 行）$(v_6,\mathcal{B}_1,[2,15])$，$(v_7,\mathcal{B}_1,[2,15])$，$f(TS_{\mathcal{B}_2^*})<f(TS_{\mathcal{B}_1^*})$，按照第二类情况更新 $n2g$（第 8～14 行），即 $(v_6,\mathcal{B}_2,[2,13])$，$(v_7,\mathcal{B}_2,[2,13])$ 和 $(v_9,\mathcal{B}_2,[2,13])$. 遍历至 $\mathcal{B}_3$ 时，由于 $\mathcal{B}_3$ 与 $n2g$ 间的重合节点 $\{v_8\}$ 的时间跨度小于 $f(TS_{\mathcal{B}_3^*})$，按照第一类情况更新 $n2g$. 最后，由于 $\mathcal{B}_4$ 和 $n2g$ 存在重合节点 $\{v_6,v_9\}$ 且 $f(TS_{\mathcal{B}_4^*})=4$，根据第二类情况更新 $n2g$. 最后得到的强 δ-基本块为 $\mathcal{B}_4$，即 $(v_6,\mathcal{B}_4,[3,7])$，$(v_8,\mathcal{B}_4,[3,7])$ 和 $(v_9,\mathcal{B}_4,[3,7])$.

5.2 全局上界优化与渐进上界优化

我们接下来进一步提出全局上界优化与渐进上界优化作为两种搜索过程中的剪枝策略以加快 *ShrimeCS* 的运行效率.

全局上界优化. 在 top-k *STCS* 中存在与参数 k 相关的时间跨度上界，通过该跨度上界可以避免大规模时序社交网络中的无效时序边，从而加速算法 *ShrimeCS* 运行效率.

定理 2. 给定查询节点 v_q 所在的社区 C 和 C 中所有关联时刻集构成的集合 $\mathcal{T}_C$，top-k *STCS* 中存在时间跨度上界 σ_C，满足 $\max(\{f(TS_C)\mid \forall TS_C\in \mathcal{T}_C\})\leq\sigma_C$. 记 $\tau(\mathcal{G})$ 为 $\mathcal{G}$ 中所有关联时刻集，同时记 $Topmax^k(\mathcal{G})$ 为 $\mathcal{G}$ 中第 k 小的时间跨度. 令 Cs 为查询节点 v_q 所在社区的集合，那么对于 v_q 的某一个社区 $\mathcal{C}$，其中 $\mathcal{C}\in Cs$ 且 $\tau(\mathcal{C})\geq k$，$\mathcal{C}$ 的第 k 小时间跨度为 $Topmax^k(\mathcal{C})$. 那么此时 top-k *STCS* 的一个可能上界为 $\delta_{\mathcal{G}^*}=\min(\{Topmax^k(\mathcal{C})\mid \forall \mathcal{C}\in Cs,\tau(\mathcal{C})\geq k\})$.

通过上述分析，计算得到的时间跨度上界 $\delta_{\mathcal{G}^*}$ 可以降低复杂网络中的搜索规模，即优化了网络的遍历开销. 具体而言，若在 $\mathcal{G}$ 中得到上界 $\delta_{\mathcal{G}^*}$ 后继续扩展当前子图，则根据定理 1，$\nexists\delta'_{\mathcal{G}^*}<\delta_{\mathcal{G}^*}$，也就是说在扩展过程中一旦得到一个新的大于 $\delta_{\mathcal{G}^*}$ 的时间跨度，由定理 1 可知，后续时序边都不需要再被考虑.

渐进上界优化. *SG*$\mathcal{B}$在扩展过程中可以得到 top-k_i δ-短时社区，根据 top-k δ-短时社区判定条件，后续子图 $g'\supset SG\mathcal{B}$ 中包含 top-(k_i+1)最小 δ-短时社区，因此需要遍历所有新增时序边上的时间戳集合. 我们指出，这些时序边中只有一部分时间戳可以得到 top-(k_i+1)最小 δ-短时社区. 通过定理 3 进一步优化 *ShrimeCS* 的搜索空间，进而提高算法运行效率.

定理 3. 子图 $g\subset\mathcal{G}$ 扩展形成子图 g' 时（$\mathcal{G}\supset g'\supset g$），关联时刻集满足 $\max(TS_g)\leq\max(TS_{g'})\wedge\min(TS_g)\geq\min(TS_{g'})$. 若 $\exists T_{(u,v)}=\{t_1,t_2,\cdots,t_m\}$，$m>k$，其中 m 是节点 u,v 产生交互的时间戳集合的大小. 最小时间跨度在 (u,v) 边上的增量分别是 $\Delta_{f(TS_{g_1^*})}(u,v)=\{\max(\{|t_1-\max(TS^*_{g_1})|,|t_1-\min(TS^*_{g_1})|\})-f(TS^*_{g_1}),\max(\{|t_2-\max(TS^*_{g_1})|,|t_2-\min(TS^*_{g_1})|\})-f(TS^*_{g_1}),\max(\{|t_3-\max(TS^*_{g_1})|,|t_3-\min(TS^*_{g_1})|\})-f(TS^*_{g_1}),\cdots,\max(\{|t_m-\max(TS^*_{g_1})|,|t_m-\min(TS^*_{g_1})|\})-f(TS^*_{g_1})\}$. 记 $t_{(i)}$ 是将 $\Delta_{f(TS_{g_1^*})}(u,v)$ 从小到大排列的第 i 个时间戳，根据定理 3，子图在扩展的过程中搜索范围最大限制在 $t_{(k)}$ 以内，从而不用进一步考虑 $t_{(k+1)},\cdots,t_{(m)}$ 对短时社区最小时间跨度的影响.

ShrimeCS 伪代码如算法 6 所示. 该算法首先进行初始化（第 1～2 行）. 然后根据定理 2，计算给定参数 k 和时序网络 $\mathcal{G}$ 下的最大时间跨度上界 *bound*. 我们利用该上界修改社区搜索算法[11,18]，记为 *CommunitySearch*′(v_q, $\mathcal{G}$, *bound*)，使其返回社区的时间跨度不超过 *bound*（第 3～4 行）. 注意到，社区搜索算法的时间复杂度与社区规模正相关[11,18]. 通过在 δ-*truss* 搜索过程中引入最大时间跨度上界 *bound*，*CommunitySearch*′返回的社区满足 $\mathcal{G}_{r^*}\subseteq C$，其中 $\mathcal{G}_{r^*}$ 是 top-k 短时社区下经过 *bound* 约束后的最大子图，C 是原社区搜索方法返回的子图，从而使得 *CommunitySearch*′(v_q, $\mathcal{G}$, *bound*)有更优的时间复杂度. 其次，对 v_q 所处的每一个社区 $\mathcal{C}$（第 5 行），找到社区 $\mathcal{C}$ 中所有 δ-基本块记为 $\mathcal{B}Set$，并计算每个 δ-基本块 $\mathcal{B}$ 的时间戳最大值 $\max(\mathcal{B})$ 与最小值 $\min(\mathcal{B})$，用强 δ-基本块更新算法找到所有的强 δ-基本块 *SG*$\mathcal{B}Set$（算法 6～8 行），通过调用子过程 *enmuTopkTCCPeriod* 计算 *SG*$\mathcal{B}$中候选最小 δ-短时社区集合，其中 *sps* 为短时社区，*ifs* 代表（$\min(\mathcal{B})$，$\max(\mathcal{B})$），并把每一个社区 $\mathcal{C}$ 中的强 δ-基本块 $\mathcal{B}$ 及对应的最小短时社区和时间跨度极值加入到队列 *BQ* 中（第 9～11 行）. 最后，通过对队列 *BQ* 中的元素按时间跨度升序排序，找到最小短时社区（第 14～28 行）. 若当前子图 $\mathcal{B}$ 是极大子图，则算法将其加入 *MSTCs* 集合中（第 17～19 行）. 否则根据 *MSTC* 的判定条件和单调性定理，算法不断扩增当前子图 $\mathcal{B}$，直至 $\mathcal{B}$ 形成的新子图重新满足 δ 约束（第 21～28 行），算法通过前一时刻的子图 $\mathcal{B}$ 更新当前子图，并获得更新后的子图 $\mathcal{B}'$ 及对应的时间跨度极值. 若 $\mathcal{B}'$ 的时间跨度大于 $\mathcal{B}$，说明不存在其他子图满足 δ-短时社区的判定条件，此时将前一时刻的关联时刻集加入到最小短时社区 *MSTCs* 中；否则进一步扩展子图 $\mathcal{B}'$，将 $\mathcal{B}'$ 的候选短时社区 *span*′，极值 *info*′ 加入 *BQ* 直至扩展至社区 $\mathcal{C}$（第 16～35 行）.

算法 6. ShrimeCS 方法.

输入：时序社交网络 $\mathcal{G}=(\mathcal{V},\mathcal{E})$，整数 δ ($\delta\geq2$)，查询节点 v_q 和参数 k

输出：$\mathcal{G}$ 中 top-k 短时社区集合 *MSTCs*

1. *MSTCs*←∅
2. *BQ*←∅，*cur_topk*=0//初始候选社区队列和计数变量
3. 根据参数 k 和 $\mathcal{G}$ 计算全局时间跨度上界 *bound*
4. *Cs*←*CommunitySearch*′(v_q, $\mathcal{G}$, *bound*) //构建带有时间跨度上界约束索引，根据查询节点 v_q，图 $\mathcal{G}$ 和时间跨度上界 *bound* 返回 v_q 所在的所有最大时间跨度小于 *bound* 的最大 δ-*truss* 社区 *Cs*

5. FOR EACH C IN Cs DO
6. 计算社区 C 中的 $truss$，记为 $CTruss$
7. $\mathcal{BS}et \leftarrow enumBlocks(v_q, N(v_q), \delta)$ //计算 C 中所有的 δ-基本块，详见算法 2
8. $info \leftarrow$ 计算 $\mathcal{BS}et$ 中每个 δ-基本块的最小时间戳 $t_{\min}$ 和最大时间戳 $t_{\max}$，即$\{(t_{\min}, t_{\max}) \mid \forall \mathcal{B} \in \mathcal{BS}et\}$
9. $SG\mathcal{BS}et \leftarrow CandidateBlock(\mathcal{BS}et, info, v_q)$ //从 δ-基本块集合中找到所有的强 δ-基本块，详见算法 5
10. FOR EACH $SG\mathcal{B}$ IN $SG\mathcal{BS}et$ DO
11. $(sps, ifs) \leftarrow enumTopkTCCPeriod(SG\mathcal{B}, info, k)$ //详见算法 3
12. FOR EACH $(span, info) \in (sps, ifs)$ DO
13. $BQ \leftarrow BQ \cup \{(\mathcal{C}, SG\mathcal{B}, CTruss, span, info)\}$
14. 按 BQ 中 $info.max - info.min$ 数值升序排序
15. WHILE $BQ \neq \varnothing$ DO//若队列中候选短时社区不为空
16. $(\mathcal{C}, \mathcal{B}, Truss, span, info) \leftarrow BQ.pop()$
17. IF $\mathcal{B}$ 是短时社区 THEN //top-k 短时社区判定条件
18. IF $cur\_topk > k$ THEN BREAK
19. $MSTCs \leftarrow \{(span, info)\} \cup MSTCs$
20. cur_topk++
21. ELSE //未构成短时社区
22. $\mathcal{B}' = \mathcal{B}$
23. WHILE $\mathcal{B}' \subseteq C$ //不断扩展子图
24. $outer \leftarrow \{v_j \mid \forall v_j \in N(v_i), v_j \notin V_{\mathcal{B}'}, v_i \in V_{\mathcal{B}'}\}$
25. FOR $outerNode$ IN $outer$ DO
26. 把与 $outerNode$ 相连的边加入 $\mathcal{B}'$
27. $\mathcal{B}truss \leftarrow$ 计算 $\mathcal{B}'$ 中每条边的 $trussness$
28. IF $\mathcal{B}truss(e) \geqslant \delta$ BREAK
29. $(n\_span, n\_info) \leftarrow updateTopkTCCPeriod(\mathcal{B}', span, info, k)$//见算法 4
30. FOR EACH $(span', info')$ IN (n_span, n_info) DO
31. $BQ \leftarrow BQ \cup \{(\mathcal{C}, \mathcal{B}', span', info'')\}$
32. IF $\mathcal{B}$ 是短时社区 THEN
33. $BQ \leftarrow BQ \cup \{(\mathcal{C}, \mathcal{B}, span, info)\}$
34. 按 BQ 中 $info.max - info.min$ 数值升序排序
35. RETURN $MSTCs$

例 9. 如图 2 所示，假定 $\delta=4$，$v_q=v_5$，$k=3$. $ShrimeCS$ 首先初始化 $cur\_topk=0$. v_5 在 4-$truss$ 下社区 C_1，C_2 分别是$\{v_1, v_2, v_3, v_4, v_5\}$，$\{v_5, v_6, v_7, v_8, v_9\}$(第 3 行). 在 C_1 中，δ-基本块 $\mathcal{B}_1$ 为$\{v_2, v_3, v_4, v_5\}$，在 C_2 中，δ-基本块共有 4 个，分别是 $\mathcal{B}_2=\{v_5, v_6, v_7, v_8\}$，$\mathcal{B}_3=\{v_5, v_6, v_7, v_9\}$，$\mathcal{B}_4=\{v_5, v_7, v_8, v_9\}$，$\mathcal{B}_5=\{v_5, v_6, v_8, v_9\}$. C_1 中仅有一个基本块 $\mathcal{B}_1$，$f(TS^*_{\mathcal{B}_1})=3$，其中 $\min(TS_{\mathcal{B}_1})=1$，$\max(TS_{\mathcal{B}_1})=4$. $\mathcal{B}_2$ 在 C_2 的其他 δ-基本块中有最小子图跨度 $f(TS^*_{\mathcal{B}_2})=4$(第 5～13 行). 因此 BQ 中依次存储 $\mathcal{B}_1$ 和 $\mathcal{B}_2$ 的相关信息. 按照队列存取规则，首先取得 $\mathcal{B}_1$，通过扩展子图直至再次满足 δ-$truss$，可以得到子图 $\mathcal{B}'_1$ ($\mathcal{B} \subset \mathcal{B}' \subseteq C_1$)，节点集合 $U_{\mathcal{B}'_1}=\{v_1, v_2, v_3, v_4, v_5\}$，$\min(TS_{\mathcal{B}'_1})=1$，$\max(TS_{\mathcal{B}'_1})=5$. 说明 $\mathcal{B}_1$ 是时间跨度 $f(TS^*_{\mathcal{B}_1})$ 下最大的子图，算法将 $\mathcal{B}'_1$ 和 $\mathcal{B}_1$ 加入 BQ，并把 BQ 的元素按时间跨度升序排序(算法 15～33 行). 由于 $f(TS^*_{\mathcal{B}_1})$ 最小，下一次队列中又一次取得 $\mathcal{B}_1$，此时 $\mathcal{B}_1$ 为短时社区，将其加入 $MSTCs$(第 17～20 行)；然后取得 $\mathcal{B}_2$，同理将 $\mathcal{B}_2$ 及扩展 $\mathcal{B}_2$ 对应的子图加入 BQ，并在后续过程中将 $\mathcal{B}_2$ 加入 $MSTCs$. 最后取得 $\mathcal{B}'_1$，因为 $\mathcal{B}'_1$ 规模已扩展至 C_1，说明不存在满足 4-$truss$ 更大的子图，所以可直接将 $\mathcal{B}'_1$ 加入 $MSTCs$. 此时 $cur\_topk > k$，算法结束运行(第 16 行)，返回的 top-3 4-短时社区是 $(\mathcal{B}_1, f(TS^*_{\mathcal{B}_1}))$，$(\mathcal{B}_2, f(TS^*_{\mathcal{B}_2}))$ 和 $(\mathcal{B}'_1, f(TS^*_{\mathcal{B}'_1}))$.

5.3 算法复杂度分析

本小节分析了文中所提算法 $ShrimeCS$ 的时间复杂度.

引理 1. 算法 2 的时间复杂度是 $O(C^{\delta-1}_{|N(v_q)|})$.

证明. 算法 2 的输入是查询节点的邻居集合 $N(v_q)$ 和参数 δ，输出社区 $\mathcal{C}$ 下所有的 δ-基本块. 根据定义寻找 δ-基本块需要遍历除查询节点 v_q 外 $\delta-1$ 个节点，假设 v_q 在社区 $\mathcal{C}$ 中邻居节点的数量为 $|N_{\mathcal{C}}(v_q)|$，判定过程通过额外空间开销使得判定时间复杂度为常数，则算法 2 的时间复杂度是 $O(C^{\delta-1}_{|N(v_q)|})$.

引理 2. 算法 3 的时间复杂度是 $O(k \cdot |\mathcal{E}(\mathcal{G})| \cdot \max(|T_{\varepsilon_{\mathcal{G}}}|) \cdot \log(\max(|T_{\varepsilon_{\mathcal{G}}}|)))$.

证明. 算法 3 在执行中，根据定理 2，算法主过程分别与当前时序图 $\mathcal{G}$ 上的时序边两两比较并选取当前 top-k 时间跨度极值作为下一次迭代的元素. 因此在算法循环过程中，需要保证 k 个时间跨度极值且保持升序性，即 $\sum_i k(|T_{e_i}|)\log(|T_{e_i}|) = O((k|\mathcal{E}_{\mathcal{G}}| \cdot \max(|T_{\mathcal{E}(\mathcal{G})}|)) \cdot \log(\max(|T_{\mathcal{E}(\mathcal{G})}|)))$.

引理 3. 算法 4 的时间复杂度是 $O((k \cdot |\mathcal{E}_{\mathcal{G}}| \cdot \max(|T_{\varepsilon_{\mathcal{G}}}|)) \cdot \log(\max(|T_{\varepsilon_{\mathcal{G}}}|)))$.

证明. 同引理 2.

引理 4. 算法 5 的时间复杂度是 $O(|\mathcal{BS}et| \cdot |\mathcal{V}_{\mathcal{B}}|)$，其中 $\mathcal{B}$ 是 $\mathcal{BS}et$ 中的一个子图.

证明. 算法 5 需要遍历集合 $\mathcal{BS}et$. 根据更新条件，需要通过对比 $\mathcal{B}$ 中所有的节点以确定重合节点. 因此，复杂度是 $O(|\mathcal{BS}et| \cdot |\mathcal{V}_{\mathcal{B}}|)$.

引理 5. 算法 5 的时间复杂度是 $O(|C|(|\mathcal{BS}et||\mathcal{V}_{\mathcal{B}}| + \gamma k^2 \cdot |span_{\mathcal{B}}||\mathcal{E}_g| \cdot \max(|T_g|) \cdot \log(\max$

$(|T_g|)))$，其中 C 是社区数量，γ 是强 δ-基本块达到 $\mathcal{G}_{r^*}$ 时扩展子图的次数，$|span_{\mathcal{B}}|$ 代表强 δ-基本块中关联时刻集大小，$\mathcal{G}_{r^*}$（$\mathcal{G}_{r^*}\subseteq C$）是 top-$k$ *STCS* 下最大子图.

证明.　算法 6 主要分为两部分，分别是强 δ-基本块 $SG\mathcal{B}$ 发现与短时社区搜索，前者复杂度同引理 4. 在短时社区搜索中，子图规模从强 δ-基本块开始扩展直至 $\mathcal{G}_{r^*}$，即（$\mathcal{G}_{\mathcal{V}_\delta}$，$\mathcal{G}_{\mathcal{V}_{\delta'}}$，…，$\mathcal{G}_{r^*}$），其中 $\mathcal{G}_{\mathcal{V}_\delta}\leq\mathcal{G}_{\mathcal{V}_{\delta'}}\leq\cdots\leq\mathcal{G}_{r^*}$. 根据定理 1，有 $\mathcal{G}_{r^*}\subseteq\mathcal{G}_{\mathcal{V}_C}$，其中 $\mathcal{C}$ 是 δ-*truss* 和时间跨度上界约束下的最大子图，根据引理 2 和引理 3，子过程时间复杂度是 $O((k\cdot|\mathcal{E}_{\mathcal{G}}|\cdot\max(|T_{\mathcal{E}_{\mathcal{G}}}|))\cdot\log(\max(|T_{\mathcal{E}_{\mathcal{G}}}|)))$. 因此算法 6 的时间复杂度是两部分之和. 请注意，在最小短时社区搜索问题中，参数 $k=1$.

定理 4.　*ShrimeCS* 是正确的.

6　实　验

本节分别介绍了实验环境与数据集、最小短时社区对比实验、top-k 最小短时社区实验、可扩展性分析以及案例研究.

6.1　实验环境与数据集

本文实验中采用了 5 个真实数据集（Email、CollegeMsg、Math、DBLP 与 SuperUser）和 3 个合成数据集. Email 数据集是来自欧洲研究机构的邮件网络，网络中时序边代表研究人员间发送了一封邮件. CollegeMsg 数据集（简称为 Msg 数据集）是来自加州大学的在线社交网络，网络中时序边代表用户间在某时刻发送了一条私人消息. Math 数据集是来自 MathOverflow 上的用户交互网络，SuperUser 数据集来源于网页 SuperUser，时序边类型与 MathOverflow 相同. DBLP 数据集源于 2014～2016 年间的科研合作网络，网络中节点代表学者，时序边代表学者之间在某一天共同发表了一篇论文. 合成数据集中两个节点之间的时序边数量 t 变化范围为[1,10]，时序边上的时间戳在时间跨度上均匀分布，节点数量 N 变化范围为[10^3,10^5]，平均度数 *degree* 为 5，时间跨度正比于 $|E|$，时序单位为天. 数据集的相关参数如表 2 所示，注意，平均时序比 T_r 计算为 $\frac{|\mathcal{E}|}{|E|}$，即 T_r 衡量了 $\mathcal{G}$ 中时序边上时间戳的密集性. 另外，sec，min，d 分别代表秒、分、天.

表 2　数据集

数据集	$\|V\|$	$\|\mathcal{E}\|$	$\|E\|$	T_r	时序单位	时间跨度
Email	986	332 334	16 064	20.7	sec	803 d
Msg	1899	59 835	20 296	2.9	sec	193 d
Math	24 818	506 550	199 973	2.5	sec	2350 d
SuperUser	194 085	1 443 339	924 886	1.6	sec	2773 d
DBLP	496 392	1 908 097	1 693 790	1.1	d	710 d
Syn_{1000}	1000	13 849	2500	5.5	d	250 d
Syn_{10000}	10 000	137 540	25 000	5.5	d	2500 d
Syn_{100000}	100 000	749 980	250 000	3.0	d	25 000 d

为了评估社区的短时性，受到局部模块度计算方式的启发[28]，我们提出一种新的指标，聚集因子（Clustering factor，简称为 CT-factor），计算为

$$\text{CT-factor}=\begin{cases}\dfrac{f(TS_g^*)-\bar{f}(TS_g^*)}{f(TS_g^*)}, & f(TS_g^*)\neq 0\\ 0, & f(TS_g^*)=0\end{cases},$$

$$\bar{f}(TS_g^*)=\frac{\sum_{i=1}^{|E(g)|}\sum_{j=i+1}^{|E(g)|}\sum_{ki=1}^{|T_{e_i}|}\sum_{kj=1}^{|T_{e_j}|}|t_{e_i}^{ki}-t_{e_j}^{kj}|}{\sum_{i=1}^{|E(g)|}\sum_{j=i+1}^{|E(g)|}\sum_{ki=1}^{|T_{e_i}|}\sum_{kj=1}^{|T_{e_j}|}1},$$

其中 $f(TS_g^*)$ 为子图 g 最小时间跨度，$\bar{f}(TS_g^*)$ 代表子图 g 平均时间跨度，其计算为任意两条时序边 (u,v,t_1)，$(u'v',t_2)$ 时间戳差的绝对值的均值，其中 $(u,v)\neq(u',v')$. 公式中 i,j 分别用于遍历 g 上的所有静态边，ki,kj 分别用于遍历静态边 e_i,e_j 上的时间戳集合. 若 g 的最小时间跨度为零，说明 g 是由相同时间戳构成的网络，此时我们约定其聚集因子为零. 因此，CT-factor 取值范围为[0,1). 聚集因子越小，说明社区内成员的交互时间越接近.

实验选取基于 *truss* 约束的社区搜索算法 (k,Δ,θ)-*truss*[5]，*TCP-Index*[11] 和 *EquiTruss*[18]，(k,Δ,θ)-*truss* 通过计算时序三角形持续时间找到 k-*truss* 持续社区结构，*TCP-Index* 分别为图中每个节点的邻居集合构建了基于 *trussness* 的最小生成树 Tx，根据 Tx 高效查找节点所在的社区，*EquiTruss* 改进 *TCP-Index* 中相同节点在不同最小生成树中重复出现的问题，通过构建超图的方式提出空间复杂度更低的索引结构. 为了将 *ShrimeCS* 与基准算法在短时社区搜索问题的运行时间开销上比较，

我们对(k,Δ,θ)-*truss*，*TCP-Index* 和 *EquiTruss* 进行修改，通过滑动时间窗口的方式进行短时社区搜索，并将改进后的算法记为(k,Δ,θ)-*truss-Slide*，*TCP-Index-Slide* 和 *EquiTruss-Slide*. 最后，我们比较基于 δ-基本块的 *BasicSearch* 与 *ShrimeCS* 的时间开销，以展示强 δ-基本块的优化效果.

本实验中代码均为 C++，运行环境为 Windows Server 2008 R2，内存大小为 256 GB，CPU 为 Intel Xeon E5-2650，硬盘为 2 TB.

6.2 最小短时社区对比实验

为了比较 *ShrimeCS* 和 *EquiTruss* 的短时社区搜索的性能，我们在 Email 和 Msg 数据集上进行了对比实验. 为了验证 *ShrimeCS* 在不同 δ 参数下均具有更好的效果，我们设置 δ 取值为 3～7 分别进行实验.

我们首先使用 CT-factor 评价不同社区搜索方法的效果. 结果如图 6 所示，图 6 展示 *ShrimeCS* 在 $k=1$ 时与基准算法随不同 δ 下的变化情况. 可以看到，*ShrimeCS* 的聚集因子始终小于基准算法，其中在 Email 数据集上短时性最为明显，随着 δ 的提升，*ShrimeCS* 找到的短时社区聚集因子保持在 0.30～0.65 之间，而 *EquiTruss* 和 *TCP-Index* 找到的社区始终处于 0.90 以上. 说明 *ShrimeCS* 找到的社区短时性更优. 此外，随着 δ 的增大，*ShrimeCS* 的聚集因子呈增大趋势，这是由于 δ 的增大使得社区最小时间跨度提升明显而新增时序边的交互时间相对分散，导致社区内平均时间跨度与社区内最小时间跨度差距变大，从而使得聚集因子逐渐增大. 注意到图 6(b)中 $\delta=7$ 时，*ShrimeCS* 找到的社区略优于基准算法，这是因为 Msg 数据集中 $\delta=7$ 时社区较小，*ShrimeCS* 与基准算法找到的社区相近. 图 7 中 Msg 的曲线也间接印证该原因. 另外，即使在社区相近时，*ShrimeCS* 发现的短时社区仍优于基准算法发现的社区，这是因为 *ShrimeCS* 引入了时间跨度约束，有效过滤了在满足拓扑结构约束下导致社区最小时间跨度变大的时序边，从而避免聚集因子的增大.

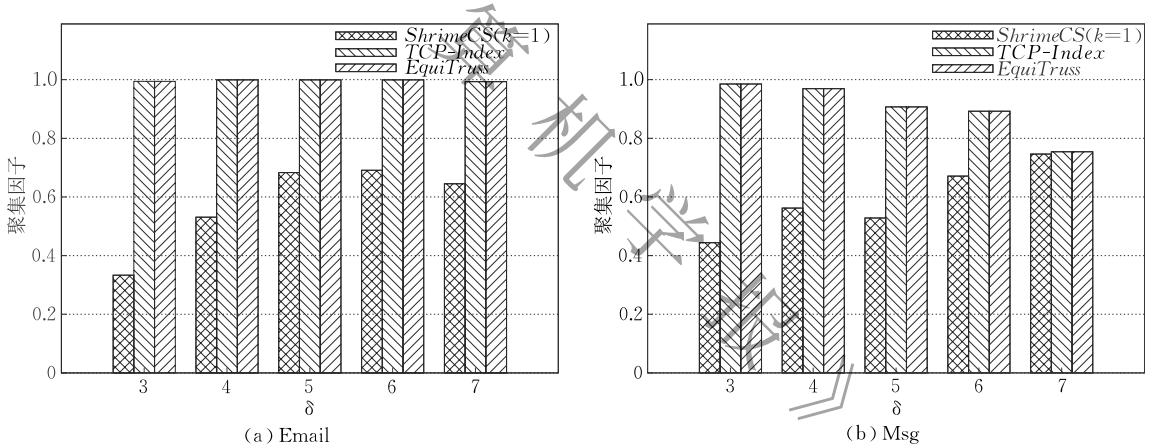

图 6 *ShrimeCS* 与基准算法在聚集因子指标上的对比实验

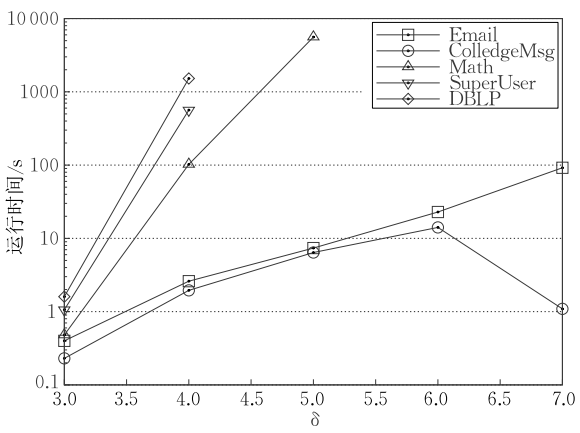

图 7 *ShrimeCS* 在不同数据集上的性能实验($k=1$)

我们接着以社区搜索用时为评价指标进行实验. 图 7 展示了 *ShrimeCS* 算法在不同 δ 下的用时. 请注意，当查询时间超过 3 小时，我们提前终止查询. 从图 7 可以看到，搜索最小短时社区与 δ-*truss* 和网络规模相关. δ 越高，*ShrimeCS* 计算复杂度越大，计算开销越高.

同时 *ShrimeCS* 的复杂度与原社区规模有关. 例如在 CollegeMsg 数据集中 $\delta=7$ 时，满足条件的是一个由 8 个节点组成的社区，*ShrimeCS* 运行时间反而减少. 这是因为社区内节点数量少，时序边也对应减少，所以节省了寻找 δ-基本块用时.

表 3 展示了 $k=1$ 时，*ShrimeCS*、*TCP-Index-Slide*、*EquiTruss-Slide* 以及(k,Δ,θ)-*truss-Slide* 分别在 4 个数据集(Email、Msg、Math、SuperUser)下不同 δ-*truss* 的运行时间对比($\delta=3$，$\delta=4$)，“—”表示运行时间超过 3 小时. 从表 3 中可以看到，

表 3　*ShrimeCS*、*TCP-Index-Slide*、*EquiTruss-Slide*、(k,Δ,θ)-*truss-Slide* 运行时间对比实验（单位：s）

δ	Dataset	*ShrimeCS*	*TCP-Index-Slide*	*EquiTruss-Slide*	(k,Δ,θ)-*truss-Slide*
$\delta=3$	Email	0.40	1.42	0.97	30.62
	Msg	0.23	1.34	0.82	58.94
	Math	0.48	756.74	514.78	—
	SuperUser	1.06	5070.62	3442.98	—
$\delta=4$	Email	2.60	4.42	3.01	114.80
	Msg	1.95	3.93	2.65	98.09
	Math	102.26	2490.34	1698.78	—
	SuperUser	563.56	16386.90	11017.54	—

ShrimeCS 在不同的 δ-*truss* 下均有最快的运行时间. 其中，随着数据集中时间跨度的不断增大，*ShrimeCS* 与其他两个算法在时间效率上的差距不断拉大. 其原因在于，*ShrimeCS* 算法通过全局上界优化限制了子图扩展过程中的最大规模，从而避免发生最坏情况下在全图中搜索的情况，即 *ShrimeCS* 中通过基于强 δ-基本块进行扩展的方式是一种局部搜索方式，受到数据集时间跨度影响较小，具有较好的稳定性.

图 8 展示了 *BasicSearch* 方法中直接基于 δ-基

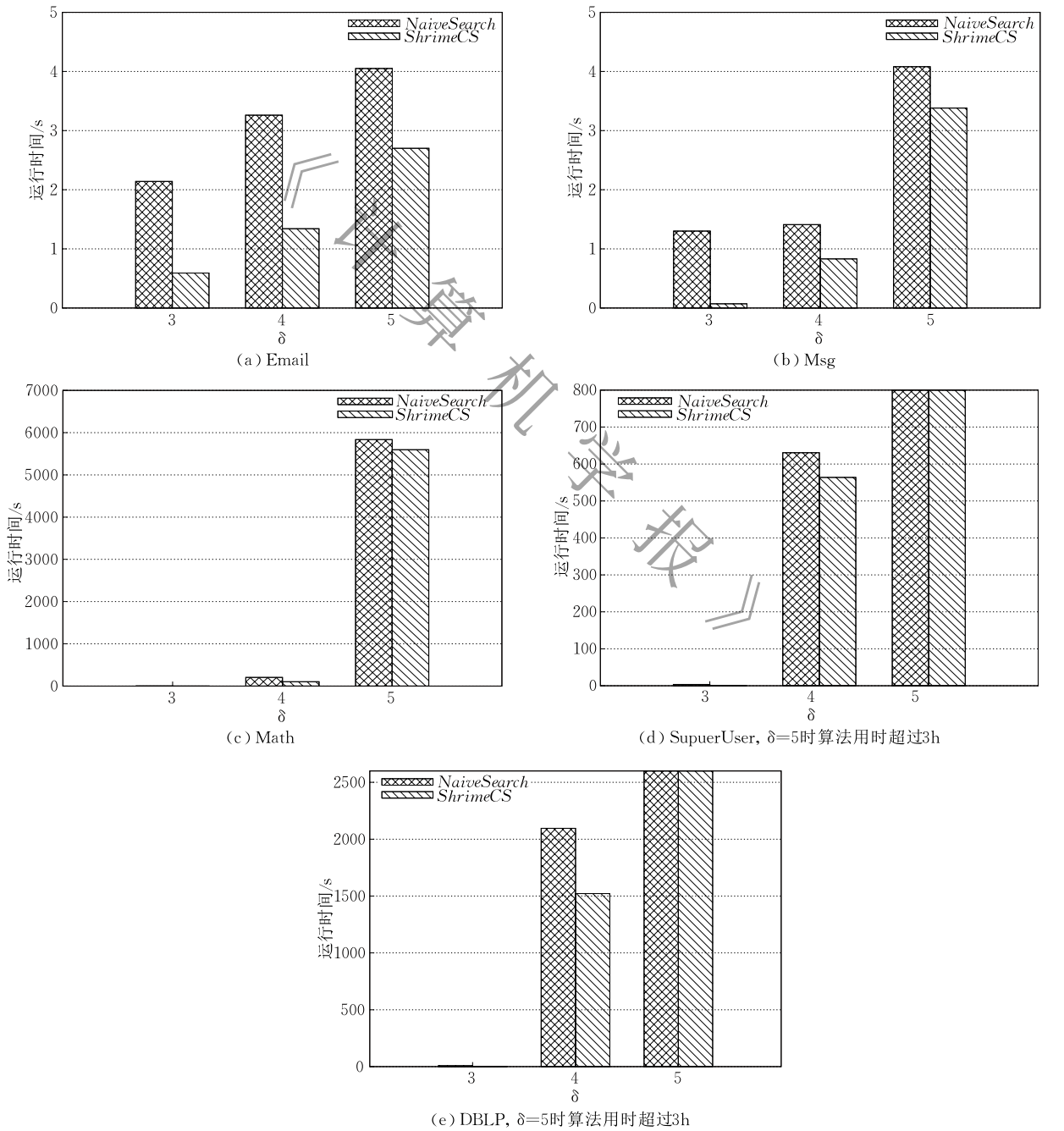

图 8　*BasicSearch* 中基于 δ-基本块与 *ShrimeCS* 中基于强 δ-基本块的运行时间对比实验

本块和 *ShrimeCS* 中基于强 δ-基本块随不同 δ-*truss* 下的运行时间对比. 图 8 结果表明,通过第 5 节所提的强 δ-基本块算法,*ShrimeCS* 的用时普遍比 *BasicSearch* 减少 17.16% 以上. 请注意,在该实验中,*BasicSearch* 与 *ShrimeCS* 除 δ-基本块外参数设置均保持一致. 图 8 展示结果说明,基于强 δ-基本块的搜索方式相比基于 δ-基本块的搜索方式,前者能在后续搜索过程中避免找到重复的短时社区,从而降低时间开销. 同时,δ-*truss* 越小,强 δ-基本块的优化效果越明显,这是因为 δ-*truss* 越小,δ-基本块数量越多,从而 *ShrimeCS* 利用非重叠性在搜索短时社区过程中避免了较大的时间开销.

6.3 top-k 最小短时社区实验

本节通过实验验证两类时间跨度上界带来的优化效果,取 $\delta=3$,$k=100$,分别以 *ShrimeCS*(简记 *Shrime*),*ShrimeCS* w. o. *GLOBAL*(简记 *Shrime-G*)以及 *ShrimeCS* w. o. *LOCAL*(简记 *Shrime-L*)代表原算法,去除全局上界优化的算法以及去除渐进上界优化的算法.

图 9 展示 *Shrime*、*Shrime-L* 与 *Shrime-G* 在四

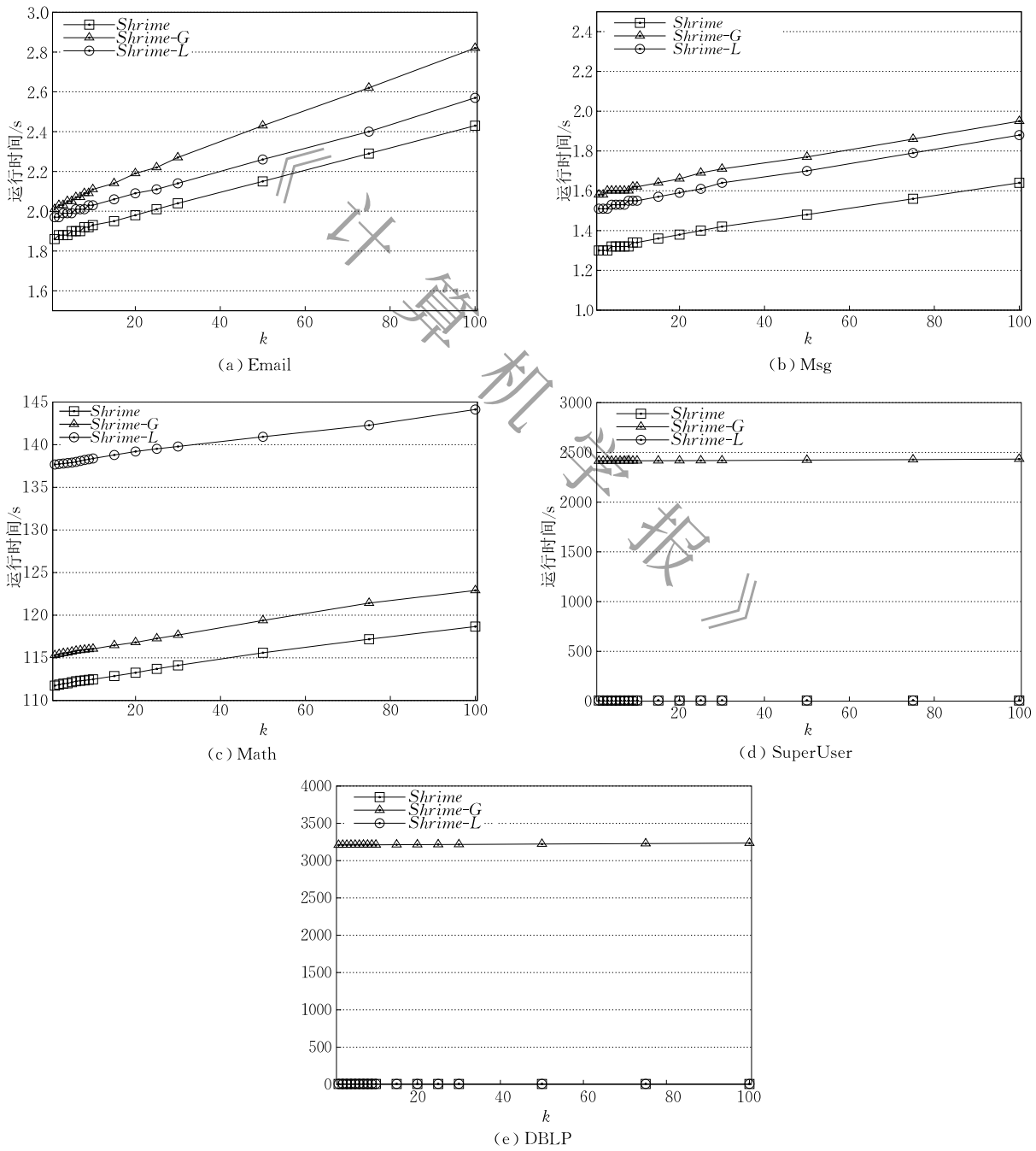

图 9 *Shrime*、*Shrime-L*、*Shrime-G* 的运行时间对比实验

个数据集(Email,Msg,Math,SuperUser)上用时随 k 的变化情况,其中参数 k 变化范围为 $k=1$ 至 $k=100$. 如图 9(a)～图 9(e)所示,随参数 k 的不断增大,算法用时增长稳定,*Shrime* 比 *Shrime-G* 运行更快,说明全局上界优化可以通过删减大量无效边加快运行速率,并且在大规模时序网络中效果尤为明显,同样论证了利用定理 2 确定时间跨度上界的可行性. 如图 9(d)所示,*Shrime-G* 用时在 2400 s 左右,*Shrime* 只需 5 s,说明全局上界优化计算得到的时间跨度上界使得算法在搜索阶段通过预剪枝最多扩展至原图规模下的一个局部子图,该局部子图的规模远远小于原图,从而大幅加速了算法用时. 同时,渐进上界优化对时序边较多的数据集优化效果明显,这是因为在子图扩展过程中,渐进上界优化可以将遍历范围限制在 $k\cdot\max(|T_{e_i}|)$ 范围内,而去除优化后遍历范围扩大至 $\prod_i |T_{e_i}|$,加大了遍历开销.

因此,全局上界优化可以加速算法初期的运行效率,渐进上界优化可以加速算法在计算过程中的运行效率,两者对 *ShrimeCS* 算法有显著提升.

表 4 展现 *ShrimeCS*($k=1$)与 *ShrimeCS*($k=5$)全局上界优化效果($\delta=3$). 其中,搜索优化率计算为经全局上界优化后过滤的边数量与图中所有时序边数量之间的比值. 全局上界优化效果与 k 值相关,k 值越小,优化效果越好. 从表 4 中可以看到,引入全局上界优化可以大幅删减无效的时序边,$k=1$ 时搜索优化率达到 75%～94%;$k=5$ 时,搜索优化率为 60%～75%.

表 4　全局上界优化的性能实验　(单位:s)

ShrimeCS($k=1$)\用时	优化前用时	优化后用时	搜索优化率/%
Email	0.47	0.05	94.4
Msg	0.25	0.02	88.5
Math	25.13	0.03	76.2
SuperUser	167.47	0.11	80.6
ShrimeCS($k=5$)\用时	优化前用时	优化后用时	搜索优化率/%
Email	0.47	0.36	64.2
Msg	0.25	0.06	68.4
Math	25.13	0.05	75.7
SuperUser	167.47	0.27	74.7

表 5 对比了在 Email 数据集上基于 *ShrimeCS* 的最小短时社区和第 top-100 短时社区与 *EquiTruss* 找到的社区之间的时间跨度. 可以看到,*EquiTruss* 找到的社区最小时间跨度较大,社区内成员之间的交互间隔最长可达 803 d. 而 *ShrimeCS* 给出的最小短时社区,在 $\delta=3$ 时可以找到最小时间跨度为 18 min 的短时社区. 直至第 top-100 短时社区,其社区内成员的交互时间跨度均较短,最长时间跨度不超过 157 d,体现短时社区特点. 例如 $\delta=5$ 的最小短时社区长度是 13 d,说明社区内成员在时间跨度为 13 d 的时段内邮件交往频繁.

表 5　社区时间跨度的对比实验

δ 约束	top-1 *STC*	top-100 *STC*	*EquiTruss*
$\delta=3$	18 min	68 d	803 d
$\delta=4$	8 d	72 d	803 d
$\delta=5$	13 d	124 d	803 d
$\delta=6$	38 d	49 d	803 d
$\delta=7$	143 d	157 d	803 d

6.4　可扩展性分析

本节对 *ShrimeCS* 的可扩展性进行分析. 可扩展性具体表现为随数据集规模的扩大和查询参数 k 的增大,*ShrimeCS* 算法运行时间的变化情况. 此处以 top-k 3-短时社区为例,在不同规模的数据集及不同查询参数 k 下的用时如表 6 所示.

表 6　*ShrimeCS* 的可扩展性实验　(单位:s)

数据集	k				
	1	10	20	50	100
Syn_{1000}	1.7	1.7	1.9	2.4	2.6
Syn_{10000}	220.4	221.1	223.8	230.1	240.7
Syn_{100000}	8164.6	8167.6	8172.3	8180.9	8194.4

表 6 在合成数据集上对 *ShrimeCS* 进行了可扩展性实验. 从表 6 中可以看到,在不同规模的数据集下,随着参数 k 取值的不断提高,*ShrimeCS* 用时随 k 增长稳定,这表明通过渐进上界优化可以有效减小在扩展子图过程中计算 top-k 最小短时社区的搜索空间. 另外,当 k 相同时,可以看到随着数据集规模的增加,算法运行时间的增长速度逐渐减缓,这是因为全局上界优化通过计算时间跨度上界,可以更好的预先删减网络中无效的时序边,降低无效时序边对算法运行效率的影响.

6.5　案例研究

本小节在 DBLP 数据集上进行案例研究. 图 10 展示了在 DBLP 数据集上的 top-k δ-短时社区与 δ-*truss* 社区,其中 $k=3$,$\delta=3$. 我们以查询节点 Madhav V. Marathe 为例,分别展示采用社区搜索算法(*EquiTruss*)与短时社区搜索算法(*ShrimeCS*)之后得到的社区结构.

EquiTruss 返回的社区如图 10 上半部分所示,共有 15 个节点. 显然,我们通过上述信息仅能知道 Madhav V. Marathe 曾与这些学者有过合作关系. 因为此时社区不带有时间信息,所以我们无法得到 Madhav V. Marathe 在 2014～2016 年间与其他学

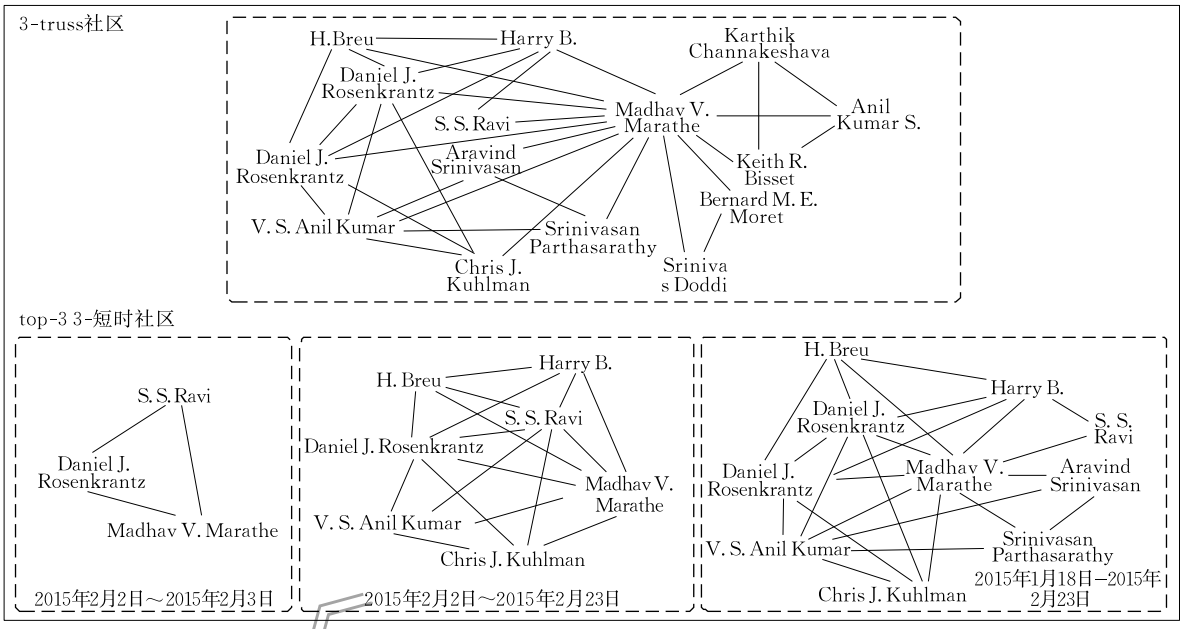

图 10 在 DBLP 数据集上不同社区搜索算法的比较(上半部分是 *EquiTruss* 得到的社区结构,下半部分是 *ShrimeCS* 得到的 top-3 短时社区结构($\delta=3$,$k=3$).可以看到,*ShrimeCS* 更好地展现了查询节点所在社区的活跃时间与社区变化)

者合作的活跃时段,也无法了解与其合作的群体随时间的变化情况.这说明基准算法对时间跨度不敏感,无法很好地表达短时特性.

ShrimeCS 返回的社区如图 10 下半部分所示,其计算得到的 top-3 短时社区进一步展现了 Madhav V. Marathe 在 2015 年中与其他学者的合作关系.其中,top-1 短时社区聚焦在 2015 年 2 月 2 日,节点数为 3;top-2 短时社区聚焦在 2015 年 2 月 2 日至 2015 年 2 月 23 日,节点数增长为 7,时间跨度也随之增长到 21 天;top-3 短时社区聚焦在 2015 年 1 月 18 日至 2015 年 2 月 23 日,节点数为 10,时间跨度为 36 天.于是我们可以得到 Madhav V. Marathe 与其他学者合作的变化情况,同时也可发现 Madhav V. Marathe 在 2015 年间较为活跃的时间区间主要集中在 1～2 月之间.

7 结束语

短时社区搜索是一种在时序社交网络中针对时序特点而提出的满足时间跨度要求的搜索问题.它既包含网络拓扑结构约束,也包含时间跨度约束.

本文给出了短时社区的形式化定义,结合分析时序社交网络上短时社区的特点给出并证明了社区扩展过程中的时间跨度单调性与短时社区判断条件,并基于此给出了 top-k 短时搜索算法 *ShrimeCS*.同时提出了强 δ-基本块的更新算法,用以删减扩展过程中可被其他子图包含的 δ-基本块,另外根据全局上界优化和渐进上界优化的启发式算法,进一步提高算法运行效率.实验结果表明短时社区搜索算法可以有效发现时序社交网络中的短时社区,并在合成数据集和真实数据集上取得较好的效果.

对于短时社区搜索问题,由于相同社区具有不同时间跨度,因此可以希望在短时社区中进行社区演化分析,衡量短时社区不同用户的参与度变化.

参 考 文 献

[1] Wang M, Wang C K, Yu J X, et al. Community detection in social networks: An in-depth benchmarking study with a procedure-oriented framework. Proceedings of the VLDB Endowment, 2015, 8(10): 998-1009

[2] Zhu Jun-Chao, Wang Chao-Kun. Approaches to community search under complex conditions. Journal of Software, 2019, 30(3): 552-572(in Chinese)
(竺俊超, 王朝坤. 复杂条件下的社区搜索方法. 软件学报, 2019, 30(3): 552-572)

[3] Lou Yun-Kai, Wang Chao-Kun. Optimization approach to subgraph matching algorithms using community structure information. Journal of Frontiers of Computer Science and Technology, 2019, 13(1): 1-22(in Chinese)
(楼昀恺, 王朝坤. 使用社区结构信息的子图匹配算法优化方法. 计算机科学与探索, 2019, 13(1): 1-22)

[4] Kang Ying, Gu Xiao-Yan, Yu Bo, et al. A multilevel community detection algorithm for large-scale social information networks. Chinese Journal of Computers, 2016, 39(1): 169-182(in Chinese)
(康颖，古晓艳，于博等. 一种面向大规模社会信息网络的多层社区发现算法. 计算机学报，2016，39(1)：169-182)

[5] Xu Lan-Tian, Li Rong-Hua, Wang Guo-Ren, et al. Research on K-truss community search algorithm for temporal networks. Journal of Frontiers of Computer Science and Technology, 2019, 14(9): 1482-1489(in Chinese)
(徐兰天，李荣华，王国仁等. 面向时序图的 K-truss 社区搜索算法研究. 计算机科学与探索，2019，14(9)：1482-1489)

[6] Qin H, Li R H, Wang G, et al. Mining periodic cliques in temporal networks//Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE). Macao, China, 2019: 1130-1141

[7] Chu L, Zhang Y, Yang Y, et al. Online density bursting subgraph detection from temporal graphs. Proceedings of the VLDB Endowment, 2019, 12(13): 2353-2365

[8] Li R H, Su J, Qin L, et al. Persistent community search in temporal networks//Proceedings of the 2018 IEEE 34th International Conference on Data Engineering(ICDE). Paris, France, 2018: 797-808

[9] Wu An-Biao, Yuan Ye, Qiao Bai-You, et al. The influence maximization problem based on large-scale temporal graph. Chinese Journal of Computers, 2019, 42(12): 2647-2664(in Chinese)
(吴安彪，袁野，乔百友等. 大规模时序图影响力最大化的算法研究. 计算机学报，2019，42(12)：2647-2664)

[10] Wen D, Huang Y, Zhang Y, et al. Efficiently answering span-reachability queries in large temporal graphs//Proceedings of the 2020 IEEE 36th International Conference on Data Engineering(ICDE). Dallas, USA, 2020: 1153-1164

[11] Huang X, Cheng H, Qin L, et al. Querying k-truss community in large and dynamic graphs//Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data. Snowbird, USA, 2014: 1311-1322

[12] Cui W, Xiao Y, Wang H, et al. Online search of overlapping communities//Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. New York, USA, 2013: 277-288

[13] Sozio M, Gionis A. The community-search problem and how to plan a successful cocktail party//Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington, USA, 2010: 939-948

[14] Wen D, Qin L, Zhang Y, et al. I/O efficient core graph decomposition at web scale//Proceedings of the 2016 IEEE 32nd International Conference on Data Engineering(ICDE). Helsinki, Finland, 2016: 133-144

[15] Bi F, Chang L, Lin X, et al. An optimal and progressive approach to online search of top-k influential communities. Proceedings of the VLDB Endowment, 2018, 11(9): 1056-1068

[16] Wu H, Cheng J, Lu Y, et al. Core decomposition in large temporal graphs//Proceedings of the 2015 IEEE International Conference on Big Data (Big Data). Santa Clara, USA, 2015: 649-658

[17] Zhang C, Zhang F, Zhang W, et al. Exploring finer granularity within the cores: Efficient (k, p)-core computation//Proceedings of the 2020 IEEE 36th International Conference on Data Engineering(ICDE). Dallas, USA, 2020: 181-192

[18] Akbas E, Zhao P. Truss-based community search: A truss-equivalence based indexing approach. Proceedings of the VLDB Endowment, 2017, 10(11): 1298-1309

[19] Huang X, Lu W, Lakshmanan L V S. Truss decomposition of probabilistic graphs: Semantics and algorithms//Proceedings of the 2016 International Conference on Management of Data. San Francisco, USA, 2016: 77-90

[20] Shan Jing, Shen De-Rong, Kou Yue, et al. Approach for hot spread node selection based on overlapping community search. Journal of Software, 2017, 28(2): 326-340(in Chinese)
(单菁，申德荣，寇月等. 基于重叠社区搜索的传播热点选择方法. 软件学报，2017，28(2)：326-340)

[21] Chen L, Liu C, Liao K, et al. Contextual community search over large social networks//Proceedings of the 2019 IEEE 35th International Conference on Data Engineering(ICDE). Macau, China, 2019: 88-99

[22] Zhang Z, Huang X, Xu J, et al. Keyword-centric community search//Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE). Macau, China, 2019: 422-433

[23] Bansal N, Chiang F, Koudas N, et al. Seeking stable clusters in the blogosphere//Proceedings of the 33rd International Conference on Very Large Data Bases. Vienna, Austria, 2007: 806-817

[24] Ma S, Hu R, Wang L, et al. Fast computation of dense temporal subgraphs//Proceedings of the 2017 IEEE 33rd International Conference on Data Engineering(ICDE). San Diego, USA, 2017: 361-372

[25] Wu H, Cheng J, Huang S, et al. Path problems in temporal graphs. Proceedings of the VLDB Endowment, 2014, 7(9): 721-732

[26] Huang S, Fu A W C, Liu R. Minimum spanning trees in temporal graphs//Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. Melbourne, Australia, 2015: 419-430

[27] Palla G, et al. Uncovering the overlapping community structure of complex networks in nature and society. Nature, 2005, 435(7043): 814-818

[28] Wang X, Chen G, Lu H. A very fast algorithm for detecting community structures in complex networks. Physica A: Statistical Mechanics and Its Applications, 2007, 384(2): 667-674

附录 A.

引理 6. δ-基本块是符合 δ-*truss* 约束的最小子图.

证明. 反证法. 假设 δ-基本块不是符合 δ-*truss* 约束的最小子图,于是存在一个比 δ-基本块更小的子图 g',使得 g' 是包含查询节点 v_q 的最小 δ-*truss* 子图.

先假设 δ-基本块包含 k 个节点,因为 δ-基本块满足 δ-*truss*,即对于其中任意一条边,其支持度不小于 $\delta-2$. 所以 δ-基本块中每个节点至少会和 $\delta-1$ 个节点相连. 于是我们有以下不等式,其中 $|E|$ 代表 δ-基本块的边数:

$$\frac{k(k-1)}{2}\geq|E|\geq\frac{k(\delta-1)}{2}.$$

又因为 δ-基本块的任意一个子图不满足 δ-*truss* 条件,所以移除 δ-基本块的一个节点形成的子图也不满足 δ-*truss* 条件. 于是我们有以下不等式,其中 $|E_1|$ 代表形成子图的边数:

$$\frac{(k-1)(k-2)}{2}\leq|E_1|<\frac{(k-1)(\delta-1)}{2}.$$

综合上述两式,有 $\delta\leq k<\delta+1$,得出 $k=\delta$,所以 δ-基本块中有 δ 个节点.

因为子图 g' 是 δ-基本块的子图,所以 g' 的边数小于 δ-基本块的边数,即 $|E'|<|E|\leq\frac{\delta(\delta-1)}{2}$. 又因为 g' 同样满足 δ-*truss*,所以还需要满足 $|E'|\geq\frac{\delta(\delta-1)}{2}$. 于是,$|E'|$ 既需要满足 $|E'|\geq\frac{\delta(\delta-1)}{2}$,还需要满足 $|E'|<\frac{\delta(\delta-1)}{2}$,两者产生矛盾. 因此假设不成立,原命题成立,即 δ-基本块是符合 δ-*truss* 约束的最小子图. 原命题得证. 证毕.

引理 7. 经过 δ-*truss* 分解后得到的图 $\mathcal{G}$ 中,若至少存在一条边,其支持度 $sup>\delta-2$,那么图 $\mathcal{G}$ 可被分解为规模更小的子图 $\mathcal{G}'$,$\mathcal{G}'$ 仍然满足 δ-*truss*.

证明. 根据引理 6 可知,$\mathcal{G}'$ 至少是一个 δ-基本块. 这是因为对于 δ-基本块中的任意一条边,其支持度等于 $\delta-2$,δ-基本块依然满足 δ-*truss*. 证毕.

引理 8. 若图 $\mathcal{G}$ 的最小时间跨度为 $f(TS^*_{\mathcal{G}})$,新加入节点与边使得图 $\mathcal{G}$ 扩大为 $\mathcal{G}'$,那么图 $\mathcal{G}'$ 的最小时间跨度 $f(TS^*_{\mathcal{G}'})$ 满足 $f(TS^*_{\mathcal{G}'})\geq f(TS^*_{\mathcal{G}})$.

证明. 假设原图 $\mathcal{G}$ 中时间跨度最大值和时间跨度最小值分别落在不同的两条边 e_1,e_2 上(其他情况同理),则有 $|t^i_{e_1}-t^j_{e_2}|=f(TS^*_{\mathcal{G}})$. 对于新加入的节点集合和边集合形成的扩增图 $\mathcal{G}'$,假设存在时序边 e' 上的时间戳为 $t^k_{e'}$,如果 $t^k_{e'}$ 大于 $\max(t^i_{e_1},t^j_{e_2})$ 或者 $t^k_{e'}$ 小于 $\min(t^i_{e_1},t^j_{e_2})$,则 $\mathcal{G}'$ 的时间跨度极值发生变化,使得 $|t^k_{e'}-\max(t^i_{e_1},t^j_{e_2})|$ 或 $|t^k_{e'}-\min(t^i_{e_1},t^j_{e_2})|>|t^i_{e_1}-t^j_{e_2}|$,从而图 $\mathcal{G}'$ 的时间跨度大于原图 $\mathcal{G}$. 而假若在新增的时序边集合中,任意一个时间间隔都不大于 $\max(t^i_{e_1},t^j_{e_2})$ 且不小于 $\min(t^i_{e_1},t^j_{e_2})$,则扩增图 $\mathcal{G}'$ 的时间跨度不会发生变化,此时 $\mathcal{G}'$ 的最小时间跨度保持原图 $\mathcal{G}$ 大小. 因此,最小时间跨度满足 $f(TS^*_{\mathcal{G}'})\geq f(TS^*_{\mathcal{G}})$. 证毕.

定理 1 的证明. 由上述引理 6~8 可得. 首先,根据引理 6,$\mathcal{B}$ 是符合 δ-*truss* 约束的最小子图,此时记 $\mathcal{B}$ 的最小时间跨度为 $f(TS^*_{\mathcal{B}})$. 接下来,假设 $\mathcal{B}$ 在扩展过程中依次形成子图 $g_1g_2\cdots g_m$,引理 7 说明 $g_1g_2\cdots g_m$ 都可以至少被分解为 $\mathcal{B}$. 最后,根据引理 8,最小时间跨度满足 $f(TS^*_{g_m})\geq\cdots\geq f(TS^*_{g_2})\geq f(TS^*_{g_1})\geq f(TS^*_{\mathcal{B}})$. 因此,$\delta$-基本块 $\mathcal{B}(\mathcal{B}\subseteq\mathcal{G})$ 在 $\mathcal{B}$ 不断扩展的过程中最小时间跨度单调递增,定理 1 得证. 证毕.

定理 2 的证明. 分情况讨论. 若 $\tau(\mathcal{C})\geq k$,定理 2 显然成立. 这是因为选取 $\sigma_C=\max(\{f(TS_C)\mid\forall TS_C\in\mathcal{T}_C\})$ 即可满足条件,我们直接选取社区 C 中的最大时间跨度作为时间跨度上界. 若 $\tau(\mathcal{C})<k$,此时我们选取 $\sigma_C=\max(\{f(TS_C)\mid\forall TS_C\in\mathcal{T}_C,\forall C\in Cs\})$,其中 Cs 是查询节点 v_q 所处的所有社区. 这时 σ_C 一定是满足条件的时间跨度上界,这是因为如果 $\sum_{C\in Cs}\tau(C)\geq k$,那在 σ_C 范围内能找到 k 个短时社区. 而如果 $\sum_{C\in Cs}\tau(C)<k$,则说明在网络中不存在 k 个短时社区,由于 σ_C 是所有时间跨度下的最大值,所以 σ_C 依旧是时间跨度的上界,且此时 σ_C 是 top-k *STCS* 的一个上确界. 证毕.

定理 3 的证明. 假设当前子图 g 的最大时时间戳和最小时间戳分别为 $t_{\max}=\max(TS_g)$,$t_{\min}=\min(TS_g)$,那么子图 g 所有时间戳应落在 $[t_{\min},t_{\max}]$ 范围内. 在子图 g 扩展形成子图 g' 的过程中,会有新增的时序边,记这些新增时序边上的时间戳构成的集合为 ΔT. 则 ΔT 中的时间戳满足以下三种情况:(1) ΔT 中所有的时间戳都落在 $[t_{\min},t_{\max}]$ 范围内,此时 g' 的最大时间戳和最小时间戳不会发生变化,仍为 $t_{\max}$ 和 $t_{\min}$;(2) ΔT 中存在时间戳落在 $(t_{\max},\infty)$ 范围内,则 g' 的最大时间戳发生变化,$\max(TS_{g'})=\max(\{t\mid\forall t\in\Delta T\})>t_{\max}$,同时,$g'$ 的最小时间戳满足 $\min(TS_{g'})\leq t_{\min}$;(3) ΔT 中存在时间戳落在 $(-\infty,t_{\min})$ 范围内,则 g' 的最小时间戳发生变化,$\min(TS_{g'})=\min(\{t\mid\forall t\in\Delta T\})<t_{\min}$,同时,$g'$ 的最大时间戳满足 $\max(TS_{g'})\geq t_{\max}$. 因此,关联时刻集满足 $\max(TS_g)\leq\max(TS_{g'})\wedge\min(TS_g)\geq\min(TS_{g'})$. 证毕.

定理 4 的证明. *ShrimeCS* 方法的正确性证明主要分为如下三步:证明 δ-基本块是符合 δ-*truss* 约束的最小子图,最小 δ-短时社区的正确性以及从最小短时社区出发可以找到 top-k δ-短时社区的正确性.

证明 δ-基本块是符合 δ-*truss* 约束的最小子图,该部分可由引理 6 证得.

证明最小 δ-短时社区的正确性. 首先,*ShrimeCS* 方法从强 δ-基本块集合 *SGBSet* 出发寻找最小 δ-短时社区,*SGBSet* 是在 δ-基本块集合 *BSet* 下应用强 δ-基本块的非重叠性质得到的所有结果. 由于 δ-基本块是符合 δ-*truss* 约束的最小子图,不存在满足 δ-*truss* 的子图 $g'(g'\notin SGBSet)$ 会符合如下条件:$\forall SG\mathcal{B}\in SGBSet$,$f(TS^*_{g'})<f(TS^*_{SG\mathcal{B}})$. 这是因为若 $g'\supset SG\mathcal{B}$,则 $f(TS^*_{g'})\geq f(TS^*_{SG\mathcal{B}})$(由定理 1 保证),而若 $g'\subset SG\mathcal{B}$,则 g' 不存在(由引理 6 保证). 因此可以满足 *MSTC* 的第一个判定条件. 接下来,*ShrimeCS* 方法在 *SGBSet* 基础上

不断扩展子图，并计算扩展后子图的最小时间跨度，用以判断是否存在一个更大的子图拥有与其相同的最小时间跨度.因此可以满足 *MSTC* 的第二个判定条件.最后，综合两者可以证得 *ShrimeCS* 可以找到最小 δ-短时社区.

证明从最小短时社区出发可以找到 top-k δ-短时社区的正确性.这部分的证明可以采用数学归纳法.首先证明从最小短时社区出发可以找到第 top-1 最小 δ-短时社区.该部分可由证明最小 δ-短时社区的正确性证得.接下来证明如果从最小短时社区出发可以找到第 top-i 最小 δ-短时社区，那么从最小短时社区出发可以找到第 top-$(i+1)$最小 δ-短时社区.*ShrimeCS* 从强 δ-基本块集合 *SGBSet* 出发不断扩展子图，假设在这过程中依次形成子图 $g_1, g_2, \cdots, g_n$，第 top-i 最小 δ-短时社区为 g_p $(p<n)$，强 δ-基本块的非重叠性保证了 $g_1, g_2, \cdots, g_n$ 中不存在重叠子图，根据定理 1，最小时间跨度满足 $f(TS^*_{g_n}) \geq \cdots \geq f(TS^*_{g_2}) \geq f(TS^*_{g_1})$.因为 *ShrimeCS* 在扩展过程中遍历了所有扩展子图，所以 $g_1 g_2 \cdots g_n$ 中包含了从 *SGBSet* 出发的所有结果.因此，若存在 $f(TS^*_{g_q}) = f(TS^*_{g_p})$ 且 g_p 与 g_q 之间不存在包含关系，则 g_q 即为第 top-$(i+1)$最小 δ-短时社区.否则，若存在 $f(TS^*_{g_r}) = \cdots = f(TS^*_{g_{q+1}}) = f(TS^*_{g_q}) > f(TS^*_{g_p})$ 且 g_{q+1} 与 g_q 之间不存在包含关系，则 g_q 即为第 top-$(i+1)$最小 δ-短时社区.不然只需在 $g_1 g_2 \cdots g_n$ 中找到满足如下条件的子图 g_q：$f(TS^*_{g_{q+1}}) > f(TS^*_{g_q}) = \cdots = f(TS^*_{g_{p+1}}) > f(TS^*_{g_p})$ 且 $g_q \supset \cdots \supset g_{p+1} \supset g_p$，则 g_q 即为第 top-$(i+1)$最小 δ-短时社区.满足 top-k δ-短时社区的判断条件，因此可以证明从最小短时社区出发可以找到第 top-$(i+1)$最小 δ-短时社区.综合上述，可以证实 *ShrimeCS* 能够从最小短时社区出发找到 top-k δ-短时社区.

综合上述所有的结论可以证明 *ShrimeCS* 方法的正确性.

证毕.

GU Tian-Kai, M. S. candidate. His research interests include graph data, temporal community search and graph neural network.

WANG Chao-Kun, Ph. D., associate professor. His research interests include graph and social data management, big data system.

LOU Yun-Kai, Ph. D. candidate. His research interests include graph database, community detection.

Background

The community search problem is to find all the densely connected subgraph on the large graph G, given by the query node v. In the past literature, most of the studies put forward quite a variant of constraints on the subgraph structure, such as k-core, k-truss. Recently, there are growing interests in finding densely connected communities on the attribute and temporal graphs. Some studies aim to explore the periodic patterns appearing in the temporal network, other works also research on finding the densely bursting communities in a short time.

Short-time frequent interactions are an important temporal feature on the social network, which can effectively mine the short-time structure. In view of the current situation that it is difficult for existing work to meet the above requirements, we propose a new problem of top-k short-time community search to provide an effective solution for finding short-time communities in complex networks. First, a formal definition of δ-short-time community is given, and a method for calculating the time span of a community formed under different time interactions are provided. Second, the algorithm top-k δ-short-time community search *ShrimeCS* is proposed. In addition, it analyzes and proves the heuristic calculation method of the upper bound of global clustering length as well as the progressive clustering length in the top-k problem to further accelerate the efficiency of the method. Finally, the experimental results on synthetic data sets and real data sets confirm the correctness and scalability of the proposed method.

With previous contributions on proposed efficient structural constraints for community search, we in this paper extend the constraints considering both the temporal and the structural cohesiveness to further solving the community search problem on the temporal graph.

Therefore, it is claimed that our work provides a novel and efficient algorithm to find the short-time community. It is also claimed that our work proposes a new community search problem in the temporal social network.

This work is supported by the National Natural Science Foundation of China (No. 61872207) and Baidu Inc.