Mining Diversified Top-r Lasting Cohesive Subgraphs on Temporal Networks

Longlong Lin, Pingpeng Yuan, Rong-Hua Li, and Hai Jin, *Fellow, IEEE*

Abstract—Temporal subgraph mining is recently ubiquitous. Identifying diversified and lasting ingredients is a fundamental problem in analyzing temporal networks. In this paper, we investigate the problem of finding diversified lasting cohesive subgraphs from temporal networks. Specifically, we first introduce a new model, called maximal lasting (k, σ)-core, for characterizing lasting cohesive subgraphs on temporal networks so as to the nodes in the subgraph are connected densely and also the subgraph's structure remains unchanged for a period of time. To enhance the diversity of results, we then formulate a diversified lasting cohesive subgraphs problem, which finds r maximal lasting (k, σ)-cores with maximum coverage regarding the number of vertices and timestamps. Unfortunately, we show that the optimization problem is NP-hard. To tackle this issue, we first devise a greedy algorithm named *GreLC* with (1-1/e) approximation ratio. However, *GreLC* has prohibitively high time and space complexity, resulting in poor scalability. Then, an improved DFS-based search algorithm called *TopLC* with 1/4 approximation ratio is proposed to lower the computational cost. Finally, empirical studies on six real-world temporal networks demonstrate that the proposed solutions perform efficiently and accurately, and our model is better than temporal cohesive subgraphs detected by existing approaches.

Index Terms—Cohesive subgraphs, temporal networks, lasting pattern mining, diversified top-r search

1 Introduction

Temporal networks have received attention in a wide spectrum of scenarios ranging from brain networks and communication networks to bibliographic networks and social networks [1], [2], [3]. In these networks, each edge is a triple (v_1, v_2, t) that indicates the two parties v_1 and v_2 have an interaction at time t. Analyzing the temporal nature of these networks can provide high-level insights about their structure and evolution. For example, it can reveal the time-respecting path or reachability [4], [5], periodicity of a cohesive subgraph [6], the timeline of events [7] and time constrained motifs [8].

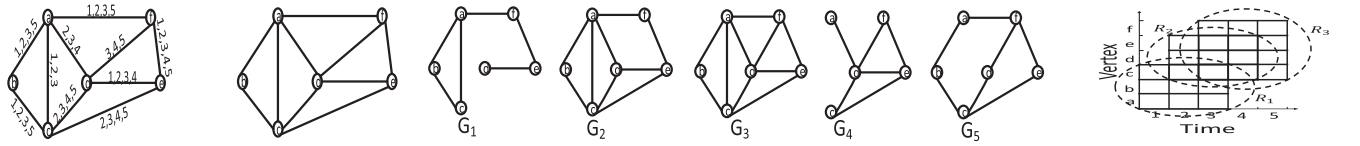
Although temporal networks are constantly evolving, some cohesive structures may be unchanged for a period of time, that is lasting cohesive subgraphs, suggesting the microscopic invariable structural feature. For instance, in dynamic protein-protein interaction networks [9], each vertex represents a protein and each edge reflects the time when the two proteins interact. A lasting cohesive subgraph is a group of densely connected proteins and the connection lasts for a period time. Such a subgraph may be able to predict which protein complexes are more prone to mutations. Another concrete example is scientific collaboration networks [10], in which an edge indicates the time when the two authors co-author a paper. A lasting cohesive subgraph can be regarded as a stable research team that lastly worked together for a period of time, which can provide evidence why the team has changed, for example, due to some members graduated from the team.

Surprisingly, detecting lasting cohesive subgraphs is meaningful and enjoys many applications, but this issue has not been adequately studied in literature. As stated in Section 6, the existing approaches on detecting static cohesive subgraphs [11] only considered the structural cohesiveness but the temporal feature of a subgraph. Until very recently, some studies tried to identify temporal cohesive subgraphs [6], [7], [10], [12], [13], [14]. For example, Wu *et al.* [13] investigated core-decomposition on temporal networks, which can be used to visualize temporal networks. Ma *et al.* [12] investigated heavy temporal subgraphs problem to analyze road traffic conditions. Rozenshtein *et al.* [7] studied the timeline of events on temporal networks. Li *et al.* [10] developed an effective branch and bound algorithm to detect maximum persistent communities in temporal networks. Qin *et al.* [6] introduced the concept of periodic clique to capture and predict periodicity of cohesive subgraphs. Chu *et al.* [14] modeled the density bursting subgraph as a temporal subgraph that aggregates its cohesiveness at the fastest speed during the corresponding time interval. Unfortunately, since the interlink structures constituted by the returned node set may be different from one timestamp to another, the existing approaches fail to consider the lasting of a subgraph. Moreover, they proposed techniques also cannot be straightforwardly applied to identify lasting cohesive subgraphs.

Here, we study an important problem of identifying diversified lasting cohesive subgraphs from a temporal network. Specifically, we propose a novel lasting cohesive

- *Longlong Lin, Pingpeng Yuan, and Hai Jin are with the National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Laboratory, Cluster and Grid Computing Laboratory, School of Computer Science & Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China. E-mail: {longlonglin, ppyuan, haijin}@hust.edu.cn.*
- *Rong-Hua Li is with the Department of Computer Science, Beijing Institute of Technology, Beijing 100081, China. E-mail: lironghuascut@gmail.com.*

Manuscript received 12 June 2020; revised 31 Dec. 2020; accepted 10 Jan. 2021. Date of publication 9 Feb. 2021; date of current version 11 Nov. 2022. (Corresponding author: Pingpeng Yuan.)
Recommended for acceptance by Yanfang (Fanny) Ye.
Digital Object Identifier no. 10.1109/TBDATA.2021.3058294



(a) Temporal graph $\mathcal{G}$ (b) Its de-temporal graph G (c) All snapshots of $\mathcal{G}$ (d) Diversified top-r lasting (k,σ)-cores

Fig. 1. De-temporal graph, snapshots, and diversified top-r lasting (k,σ)-cores of an example temporal graph.

subgraph model, named maximal lasting (k,σ)-core, based on the well-known k-core [15] (k-core is a subgraph such that each node has at least k neighbors within the subgraph). For a temporal network $\mathcal{G}$, a maximal lasting (k,σ)-core consists of a static graph G_S and a time interval I whose length is no less than σ, in which G_S is a k-core and keeps unchange during I in $\mathcal{G}$. Clearly, the maximal lasting (k,σ)-core is a natural fusion of the cohesiveness and the time duration (lasting). Unfortunately, the number of maximal lasting (k,σ)-cores may be exponentially large in the worst case (see Section 2.2 for details). Thus, enumerating all maximal lasting (k,σ)-cores is expensive and meaningless. So, we resort to research diversified top-r searching due to the importance and usefulness of diversification (see Section 6). In particular, we aim to find r maximal lasting (k,σ)-cores, saying diversified top-r lasting (k,σ)-cores, with maximum coverage regarding the number of vertices and timestamps. These maximal lasting (k,σ)-cores are distinctive and informative. In a nutshell, our main contributions are summarized as follows:

An Elegant Cohesive Subgraph Model in Temporal Graphs. We propose a novel temporal model, called diversified top-r lasting (k,σ)-cores, to capture both the diversification and the time duration of cohesive subgraphs on temporal networks. We demonstrate that the problem of identifying diversified top-r lasting (k,σ)-cores can essentially be reduced to the maximum r-coverage. Thereby, our problem is NP-hard due to the maximum r-coverage is NP-hard [16].

Two Efficient Algorithms With Approximate Guarantees. To detect diversified top-r lasting (k,σ)-cores on a temporal network, we devise a greedy algorithm named $GreLC$ with $(1-1/e)$ approximation ratio, which first enumerates all maximal lasting (k,σ)-cores as candidate results and then adopts a greedy manner to obtain the final results. However, its time and space complexity are prohibitively high, resulting in poor scalability. To reduce the complexity of $GreLC$, we further devise an improved DFS-based search algorithm called *TopLC* with $1/4$ approximation ratio, which interweaves simultaneously candidate results and pruning techniques. Namely, *TopLC* maintains dynamically at most r candidate lasting (k,σ)-cores and the candidates are updated when a new lasting (k,σ)-core is generated. Meanwhile, the candidates can be applied to facilitate the pruning process, resulting in that some pruning techniques are guided by the candidates.

Several Experimental Evaluations on Real-World Datasets. We conduct extensive experiments on six real-world temporal networks for evaluating the proposed solutions. These results illustrate that our best algorithm is 1-3 orders of magnitude faster than the baseline with most parameter settings on all datasets. For instance, on a temporal network contains over million vertices and edges, the best algorithm takes about 6 seconds to find diversified top-r lasting (k,σ)-cores with most parameter settings. The baseline, however, cannot get the results within one hour. Meanwhile, the experiments also demonstrate that the practical diversified quality of $TopLC$ is comparable to $GreLC$. In addition, by comparing with four state-of-the-art baseline models, we find that our model can indeed detect more fascinating lasting patterns that cannot be found by the baselines. For reproducibility purpose, the source code of our paper can be accessed at https://github.com/Lin021/DLCP.

2 Diversified Lasting Cohesive Subgraphs

2.1 Preliminaries

Here, we focus on an undirected and unweighted temporal graph $\mathcal{G}(V,\mathcal{E},\mathcal{T})$ without self-loops, in which V and $\mathcal{E}$ represent the node (vertex) set and the temporal edge set respectively. Each temporal edge is a triple $(u,v,t)\in\mathcal{E}$ indicating that u and v have an interaction at timestamp t. More than an interaction may occur between u and v. Namely, two temporal edges (u,v,t_1) and (u,v,t_2) are different if $t_1\neq t_2$. We let $n=|V|$ (resp. $m=|\mathcal{E}|$) be the number of vertices (resp. temporal edges). $\mathcal{T}=\{t|(u,v,t)\in\mathcal{E}\}$ is the time domain of $\mathcal{G}$. For convenience, we assume that t is an integer, because in real-world scenarios the timestamp is normally an integer. A time interval $I=[t_b,t_e]\subseteq\mathcal{T}$ is a set of continuous timestamps and let $|I|=t_e-t_b+1$ (resp. $|\mathcal{T}|$) be the number of timestamps of I (resp. $\mathcal{T}$).

Let $G(V,E)$ be the de-temporal graph of $\mathcal{G}$, in which $E=\{(u,v)|\exists(u,v,t)\in\mathcal{E}\}$ and $\bar{m}=|E|$. That is G is a static graph that removes the time attribute of $\mathcal{G}$. Let $N_G(v)=\{u|(u,v)\in E\}$ be the neighbors of v in G and $d_G(v)=|N_G(v)|$ be the degree of v in G. A graph $G_S=(S,E_S)$ is a subgraph of G, denoted by $G_S\subseteq G$, if $S\subseteq V$ and $E_S\subseteq E$. Let $\mathcal{G}_S(S,\mathcal{E}_S,\mathcal{T}_S)$ be a temporal subgraph of $\mathcal{G}$ induced by S, in which $S\subseteq V$, $\mathcal{E}_S=\{(u,v,t)\in\mathcal{E}|u,v\in S\}$ and $\mathcal{T}_S=\{t|(u,v,t)\in\mathcal{E}_S\}$. For any timestamp $t\in\mathcal{T}$, we let $G_t=(V_t,E_t)$ be the snapshot of $\mathcal{G}$ at timestamp t such that $E_t=\{(u,v)|(u,v,t)\in\mathcal{E}\}$ is an edge set existing at timestamp t and V_t is the end vertices of E_t. Fig. 1a shows a temporal graph $\mathcal{G}$ with 6 vertices, 38 temporal edges and $\mathcal{T}=\{1,2,3,4,5\}$. Figs. 1b and 1c indicate the de-temporal graph G and all snapshots of $\mathcal{G}$ respectively. We give the following two definitions for the convenience of modeling lasting cohesive subgraph later.

Definition 1 (Time Support Set). *The time support set of a subgraph G_S in a temporal graph $\mathcal{G}$ is defined by $Sup(G_S)=\{t|G_S\subseteq G_t\}$.*

Definition 2 (σ-Lasting Support Set). *Given a positive integer σ, a σ-lasting support set of G_S, denoted by $l_\sigma(G_S)\subseteq Sup(G_S)$, is a time interval such that $|l_\sigma(G_S)|\geq\sigma$.*

By Definition 2, σ-lasting support set describe the time duration of a subgraph in a temporal graph.

2.2 Problem Definition

One representative cohesive subgraph model is k-core [15], which has been widely applied in community search, user engagement and influence evaluation [17], [18], [19]. Based on this, we define an elegant temporal cohesive subgraph model by integrating the time duration into the k-core.

Definition 3 (k-Core [15]). *For static graph G and positive integer k, the k-core $G_C = (C, E_C)$ is a subgraph of G such that $d_{G_C}(u) \geq k$ for any $u \in C$, and G_C is maximal k-core if there is no other k-core containing G_C.*

Definition 4 (Lasting (k,σ)-Core). *For temporal graph $\mathcal{G}$ and positive integer k and σ, a lasting (k,σ)-core $R = (G_S, I)$, satisfying*

i) Structural cohesiveness: *G_S is a k-core;*
ii) Time duration: *I is a σ-lasting support set of G_S;*

Definition 5 (Maximal Lasting (k,σ)-Core). *A lasting (k,σ)-core (G_S, I) is maximal if there is no other lasting (k,σ)-core $(G_{S'}, I')$ satisfying $G_S \subseteq G_{S'}$ and $I \subseteq I'$.*

Parameter k depicts the structural cohesiveness of subgraph G_S and σ controls the time duration of G_S. Conceptually, the maximal lasting (k,σ)-core is different from k-core, since it enables better analysis of the lasting pattern by incorporating the time duration into k-core. Namely, it can capture lasting cohesive subgraphs from the evolving process of temporal graphs very well. Meanwhile, the maximal lasting (k,σ)-core has the following superiorities: (1) the maximal lasting (k,σ)-core has several elegant properties to facilitate the algorithm design of the following problem discussed in Section 2.3 ; (2) It or they can be obtained in linear time when the time interval is fixed; (3) the maximal lasting (k,σ)-core help effectively identify some fascinating temporal patterns as illustrated in our experiments.

Example 1. Consider temporal graph $\mathcal{G}$ in Fig. 1a. Let $k = 2$ and $\sigma = 3$, subgraph $G_S = \{(a,b),(b,c),(c,a)\}$ is a 2-core and its time support set is $\{1,2,3\}$. Thus, time interval [1,3] is a 3-lasting support set of G_S. Moreover, there does not exist another lasting (k,σ)-core $(G_{S'}, I')$ such that $G_S \subseteq G_{S'}$ and $[1,3] \subseteq I'$. Consequently, pair $(G_S, [1,3])$ is a maximal lasting (2,3)-core of $\mathcal{G}$. In a similar way, we can derive that $(\{(d,c),(d,e),(c,e)\},[2,4])$ and $(\{(c,e),(e,f),(f,d),(d,c)\},\ [3,5])$ are also maximal lasting (2,3)-cores of $\mathcal{G}$.

Let $LC^k_\sigma(\mathcal{G})$ be the set of maximal lasting (k,σ)-cores of $\mathcal{G}$. Lasting (k,σ)-cores in $LC^k_\sigma(\mathcal{G})$ may overlap (e.g., in Example 1, $(\{(d,c),(d,e),(c,e)\},[2,4])$ and $(\{(c,e),(e,f),(f,d),(d,c)\},[3,5])$). As a result, the number of maximal lasting (k,σ)-cores may be exponentially large in the worse case. Thus, enumerating all maximal lasting (k,σ)-cores is expensive and meaningless. An interesting and meaningful problem is to output diversified top-r lasting (k,σ)-cores due to its practicality in real-world applications [7], [20]. Intuitively, the number of vertices and timestamps are two pivotal factors in characterizing the diversity of the resulting lasting (k,σ)-cores. So, we borrow diversified measurement in [20] and define the coverage of maximal lasting (k,σ)-cores as follows:

Definition 6 (Coverage). *For a set of maximal lasting (k,σ)-cores $\mathcal{R} = \{(G_{S_1}, I_1), (G_{S_2}, I_2), \ldots,\} \subseteq LC^k_\sigma(\mathcal{G})$ in temporal graph $\mathcal{G}$, the coverage of $\mathcal{R}$ is denoted by $cov(\mathcal{R}) = \cup_{(G_{S_i}, I_i) \in \mathcal{R}} \{(v,t) | v \in S_i, t \in I_i\}$.*

Based on Definition 5 and 6, we formally define diversified top-r lasting (k,σ)-cores as follows:

Definition 7 (Diversified Top-r Lasting (k,σ)-Cores). *For temporal graph $\mathcal{G}$ and parameter k, σ and r, the diversified top-r lasting (k,σ)-cores of $\mathcal{G}$ is denoted by $\mathcal{R} = \{(G_{S_1}, I_1), (G_{S_2}, I_2), \ldots,\} \subseteq LC^k_\sigma(\mathcal{G})$ such that (i) $|\mathcal{R}| \leq r$; (ii) $|cov(\mathcal{R})|$ is maximum.*

Example 2. Reconsider temporal graph $\mathcal{G}$ in Fig. 1a. Let $r = 2$, $k = 2$ and $\sigma = 3$, there are three maximal lasting (2,3)-cores by Example 1: $R_1 = (\{(a,b),(b,c),(c,a)\},[1,3])$, $R_2 = (\{(d,c),\ (d,e),(c,e)\},[2,4])$ and $R_3 = (\{(c,e),(e,f),(f,d),(d,c)\},[3,5])$. The coverage of R_1, R_2 and R_3 are illustrated in Fig. 1d. By Definition 7, the diversified top-2 lasting (2,3)-cores is $\{R_1, R_3\}$. Since R_1 and R_3 only overlap on vertex c at time $t = 3$, we have $|cov(\{R_1, R_3\})|$=3*3+3*4-1*1=20.

Problem Statement (DLCP). For temporal graph $\mathcal{G}$ and parameter k, σ and r, the **D**iversified **L**asting **C**ohesive subgraph **P**roblem (abbr. DLCP) aims to identify the diversified top-r lasting (k,σ)-cores from $\mathcal{G}$.

2.3 Problem Analysis

In this subsection, we first show $DLCP$ is NP-hard and then present some elegant properties for designing effectively algorithms in next sections.

Theorem 1 (Hardness). *$DLCP$ is NP-hard.*

Proof. For a family of sets $\mathcal{F} = \{S_1, S_2, ...\}$ and parameter r, the maximum r-coverage problem is to return subset $\mathcal{R} \subseteq \mathcal{F}$ such that $|\mathcal{R}| \leq r$ and $\cup_{S_i \in \mathcal{R}} \{s | s \in S_i\}$ is maximum. Therefore, we can prove this theorem by reducing a special case of $DLCP$ to the maximum r-coverage problem in polynomial time. Specifically, we build a collection of sets $\mathcal{F} = \{S_1, S_2, ...\}$ based on $\mathcal{G} = (V, \mathcal{E}, \mathcal{T})$, such that $S_{i_j} = \{(v_{i1}, j), (v_{i2}, j), ...\}$ and $G_{\{v_{i1}, v_{i2}, ...\}}$ is a maximal k-core of G_j for $j \in \mathcal{T}$. Thus, identifying $DLCP$ from the temporal graph $\mathcal{G}$ is equivalent to return the maximum r-coverage from $\mathcal{F}$ when $\sigma = 1$. Since the reduction can be done in polynomial time and the maximum r-coverage problem is NP-hard [16], the $DLCP$ is also NP-hard. □

Given vertex set S and interval I with $|I| \geq \sigma$, let $G_S(I) = (V_S(I), E_S(I))$ be a subgraph, in which $E_S(I) = \cap_{t \in I} \{(u,v) | (u,v,t) \in \mathcal{E}_S\}$ and $V(I)$ is the end vertices of $E(I)$. We use $G(I)$ to denote $G_S(I)$ when $S = V$.

Property 1. *Finding the maximal lasting (k,σ)-core in I is equivalent to calculating the maximal k-core in $G(I)$.*

Property 2. *For any lasting (k,σ)-core (G_S, I) of temporal graph $\mathcal{G}$, G_S is a k-core.*

By Definition 5, Property 1, 2 are true. For convenience, we let $C_S^k(I)$ be the maximal k-core of $G_S(I)$. Note that we need $\sum_{t\in I,(u,v,t)\in\mathcal{E}_S} O(1)$ time to compute $C_S^k(I)$.

Property 3. *Given vertex set S, positive integer k and two intervals $I, I^{'}$, $C_S^k(I^{'}) \subseteq C_S^k(I)$ if $I \subseteq I^{'}$.*

Proof. It can be obtained that $G_S(I') \subseteq G_S(I)$ if $I \subseteq I^{'}$, which implies that $d_{G_S(I')}(v) \leq d_{G_S(I)}(v)$ for any $v \in V_S(I')$. Consequently, $C_S^k(I^{'}) \subseteq C_S^k(I)$ by Definition 3. □

Corollary 1. *Given vertex set S, positive integer k and two intervals $I = [t_s, t_e]$, $I^{'} = [t_s^{'}, t_e^{'}]$, $Kcore(C_S^k(I) \cap C_S^k(I')) = C_S^k([t_s, t_e^{'}])$ if $t_s < t_s^{'}$, $t_e < t_e^{'}$ and $t_e \geq t_s^{'}$, where $Kcore(G)$ is the maximal k core of G.*

Proof. $I \subseteq [t_s, t_e^{'}]$ and $I' \subseteq [t_s, t_e^{'}]$ since $t_s < t_s^{'}$, $t_e < t_e^{'}$ and $t_e \geq t_s^{'}$. According to Property 3, we have $C_S^k([t_s, t_e^{'}]) \subseteq C_S^k(I)$ and $C_S^k([t_s, t_e^{'}]) \subseteq C_S^k(I')$. Thereby, $C_S^k([t_s, t_e^{'}]) \subseteq C_S^k(I) \cap C_S^k(I')$. Furthermore, $C_S^k(I) \cap C_S^k(I')$ is also the subgraph of $G_S([t_s, t_e^{'}])$. So, $Kcore(C_S^k(I) \cap C_S^k(I')) = C_S^k([t_s, t_e^{'}])$. □

3 The GreLC Algorithm

Since $DLCP$ is NP-hard as demonstrated in Theorem 1, it is almost impossible to tackle $DLCP$ in polynomial time unless P=NP. Thereby, in this section, we devise a simple greedy algorithm called $GreLC$ with a guaranteed approximation ratio (Algorithm 1). Specifically, $GreLC$ includes two stages. The algorithm first calls Algorithm 2 to enumerate all maximal lasting (k, σ)-cores (Line 1). Then, it adopts a greedy manner to obtain an approximate result (Line 2-5).

Algorithm 1. $GreLC(\mathcal{G}, k, \sigma, r)$

Input: Temporal graph $\mathcal{G}(V, \mathcal{E}, \mathcal{T})$ and three parameters k, σ and r
Output: The diversified top-r lasting (k, σ)-cores of $\mathcal{G}$
1 $LC_\sigma^k(\mathcal{G}) \leftarrow Enum(\mathcal{G}, k, \sigma)$ and $\mathcal{R} \leftarrow \emptyset$
2 **for** *i=1 to r* **do**
3 $R^* \leftarrow argmax_{R\in LC_\sigma^k(\mathcal{G})}(|cov(\mathcal{R} \cup \{R\})| - |cov(\mathcal{R})|)$
4 $\mathcal{R} \leftarrow \mathcal{R} \cup \{R^*\}$ and $LC_\sigma^k(\mathcal{G}) \leftarrow LC_\sigma^k(\mathcal{G}) \setminus \{R^*\}$
5 return $\mathcal{R}$

3.1 Enumeration all Maximal Lasting (k, σ)-Cores

A Naive Approach. As described in Property 1, for an interval I whose length is no less than σ, we can attain the maximal k-core of $G(I)$ as the maximal lasting (k, σ)-core in I. Thus, in order to enumerate all maximal lasting (k, σ)-cores from a temporal graph $\mathcal{G}(V, \mathcal{E}, \mathcal{T})$, a naive approach is to identify the maximal k-core in each possible subgraph $G(I)$ with linear time by the peeling algorithm [21] and then eliminate those non-maximal lasting (k, σ)-cores by checking all these lasting (k, σ)-cores. Unfortunately, the total number of possible time intervals is quadratic in $\mathcal{T}$, resulting in that the total number of subgraph $G(I)$ may also be quadratic in $\mathcal{T}$. Furthermore, the approach needs to visit the whole vertex set V and takes $O(m)$ time to compute $G(I)$. Consequently, the time complexity of the naive approach is $O(|\mathcal{T}|^2 \cdot m + T_{check})$ (T_{check} is the time of maximal check), which is prohibitively high when $|\mathcal{T}|$ is very large as illustrated in Section 5.

An Efficient Algorithm. According to properties discussed in Section 2.3, we propose an efficient bottom up algorithm, which is sketched in Algorithm 2. The core insight of the algorithm is to generate lasting (k, σ)-cores by a bottom up tree. Fig. 2 presents this insight, in which all leaf nodes of the tree are all lasting (k, σ)-cores with an interval length of σ and all non-leaf nodes are computed from two adjacent tree nodes in the previous level.

Algorithm 2. $Enum(\mathcal{G}, k, \sigma)$

Input: Temporal graph $\mathcal{G}(V, \mathcal{E}, \mathcal{T})$ and two parameters k and σ
Output: The all maximal lasting (k, σ)-cores of $\mathcal{G}$
1 Let G be the de-temporal graph of $\mathcal{G}$
2 Let $G_S(S, E_S)$ be the maximal k-core of G
3 $LC_\sigma^k(\mathcal{G}) \leftarrow \emptyset$ and $Q \leftarrow \emptyset$
4 **for** *i=1 to* $|\mathcal{T}_S| - \sigma + 1$ **do**
5 **if** $C_S^k([i, i+\sigma-1]) \neq \emptyset$ **then**
6 $Q.push((C_S^k([i, i+\sigma-1]), [i, i+\sigma-1]))$
7 **while** $Q \neq \emptyset$ **do**
8 $l = |Q|$
9 **if** *l=1* **then**
10 $R \leftarrow Q.pop()$, $LC_\sigma^k(\mathcal{G}) \leftarrow LC_\sigma^k(\mathcal{G}) \cup \{R\}$
else
11 $R_1 = (G_{S_1}, [t_s, t_e]) \leftarrow Q.pop()$, $LC_\sigma^k(\mathcal{G}) \leftarrow LC_\sigma^k(\mathcal{G}) \cup \{R_1\}$
12 **for** *i=1 to* $l-1$ **do**
13 $R_2 = (G_{S_2}, [t_s^{'}, t_e^{'}]) \leftarrow Q.pop()$
$LC_\sigma^k(\mathcal{G}) \leftarrow LC_\sigma^k(\mathcal{G}) \cup \{R_2\}$
14 **if** $t_e \geq t_s^{'}$ *and* $Kcore(G_{S_1} \cap G_{S_2}) \neq \emptyset$ **then**
15 $Q.push((Kcore(G_{S_1} \cap G_{S_2}), [t_s, t_e^{'}]))$
16 $R_1 \leftarrow R_2$
17 **for** $(G_S, I) \in LC_\sigma^k(\mathcal{G})$ **do**
18 **if** *there is a lasting* (k, σ)*-core* $(G_{S'}, I') \in LC_\sigma^k(\mathcal{G})$ *s.t.* $G_S \subseteq G_{S'}$ *and* $I \subseteq I'$ **then**
19 $LC_\sigma^k(\mathcal{G}) = LC_\sigma^k(\mathcal{G}) \setminus \{(G_S, I)\}$
20 return $LC_\sigma^k(\mathcal{G})$

Specifically, Algorithm 2 first applies Property 2 to prune some unqualified vertices, which can essentially lower the cost of computing $G(I)$ (Line 1-2), and then initializes set $LC_\sigma^k(\mathcal{G})$ to collect all maximal lasting (k, σ)-cores, queue Q to store temporary lasting (k, σ)-cores (G_S, I) so that the new lasting (k, σ)-core can be generated from Q by Corollary 1 (Line 3). In Line 4-16, the algorithm executes the bottom up tree framework as illustrated in Fig. 2. Specifically, in Line 4-6, the algorithm computes all non-empty lasting (k, σ)-cores (G_S, I) in Q with $|I| = \sigma$. In Line 7-16, the algorithm applies the Corollary 1 to compute the rest lasting (k, σ)-cores (G_S, I) with $|I| > \sigma$, where $Kcore(G)$ is used to compute the maximal k-core of G. In Line 17-19, the algorithm eliminate those non-maximal lasting (k, σ)-cores by checking all these lasting

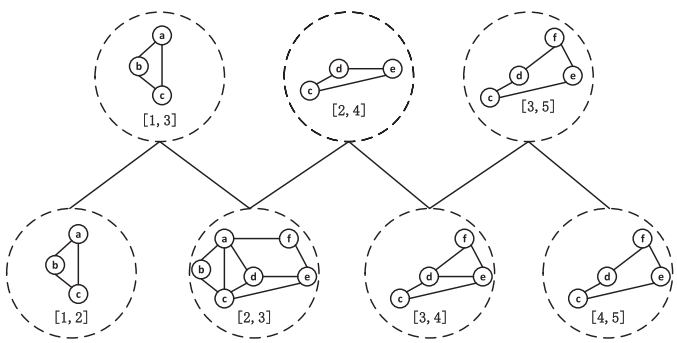


Fig. 2. The bottom up tree example.

(k,σ)-cores. Clearly, Algorithm 2 can identify all maximal lasting (k,σ)-cores of $\mathcal{G}$.

Example 3. Reconsider temporal graph $\mathcal{G}$ in Fig. 1a. We assume that $k=2$ and $\sigma=2$. It is easy to know that G is a 2-core, since each vertex v in G has at least two neighbors. Thus, by Property 2, we do not prune any unqualified nodes. Then, Fig. 2 illustrates Line 4-16 of Algorithm 2. Concretely, all leaf nodes of the tree in Fig. 2 show all lasting (2,2)-cores with $|I|=2$. The rest lasting (2,2)-cores with $|I|>2$ can be iteratively generated from the previous level, as illustrates in Fig. 2.

3.2 Analysis

Our greedy algorithm *GreLC*, which is outlined in Algorithm 1, can generate a diversified top-r lasting (k,σ)-cores with $(1-1/e)$ approximation ratio. The approximation ratio is guaranteed based on the correctness of Algorithm 2 and the greedy framework [16]. Since *GreLC* calls Algorithm 2, we next give the complexity analysis of Algorithm 2.

Theorem 2. *The time complexity of Algorithm 2 is* $O(|\mathcal{T}|\cdot m + |\mathcal{T}|^2\cdot\bar{m}+T_{check})$.

Proof. The algorithm first takes $O(\bar{m})$ to get maximal k-core of de-temporal G (Line 2). Then, Line 4-6 run in $O(|\mathcal{T}|\cdot m)$ time to compute the shortest interval results. Subsequently, the algorithm takes $O(|\mathcal{T}|^2\cdot\bar{m})$ to get the longer interval results (Line 7-16). Finally, the algorithm takes T_{check} time to eliminate the non-maximal lasting (k,σ)-cores in Line 17-19. Putting these together, Algorithm 2 takes $O(|\mathcal{T}|\cdot m+|\mathcal{T}|^2\cdot\bar{m}+T_{check})$ to get all maximal lasting (k,σ)-cores in total. □

Remark 1. *Even though the worse-case time complexity of Algorithm 2 is comparable to the naive approach, it is actually faster as illustrated in Section 5.*

Note that for any maximal lasting (k,σ)-core (G_S,I) in $LC_\sigma^k(\mathcal{G})$, we only store the vertices of G_S and the two endpoints of the interval I in real storage. By doing so, we just need to traverse vertices and timestamps in $\mathcal{R}$ and R to get $cov(\mathcal{R}\cup\{R\})$ and $cov(\mathcal{R})$.

Theorem 3. *The time complexity and space complexity of Algorithm 1 are* $O(|\mathcal{T}|\cdot m+|\mathcal{T}|^2\cdot\bar{m}+T_{check}+|LC_\sigma^k(\mathcal{G})|\cdot n\cdot r^2\cdot|\mathcal{T}|)$ *and* $O(|LC_\sigma^k(\mathcal{G})|\cdot n+m)$ *respectively.*

Proof. For the time complexity, the algorithm first takes $O(|\mathcal{T}|\cdot m+|\mathcal{T}|^2\cdot\bar{m}+T_{check})$ to run the *Enum* procedure (Line 1). Then, Line 3 runs in $O(|LC_\sigma^k(\mathcal{G})|\cdot nr|\mathcal{T}|)$ time since computing $|cov(\mathcal{R}\cup\{R\})|-|cov(\mathcal{R})|$ takes $O(nr|\mathcal{T}|)$ time for any $R\in LC_\sigma^k$. Putting these together, Algorithm 1 takes $O(|\mathcal{T}|\cdot m+|\mathcal{T}|^2\cdot\bar{m}+T_{check}+|LC_\sigma^k(\mathcal{G})|\cdot n\cdot r^2\cdot|\mathcal{T}|)$ in total.

For the space complexity, Algorithm 1 needs $O(n)$ extra space to store each maximal lasting (k,σ)-core. In addition, we also need to store the temporal graph. Therefore, the space complexity of Algorithm 1 is $O(|LC_\sigma^k(\mathcal{G})|\cdot n+m)$. □

Drawbacks of GreLC*Algorithm.* Although *GreLC* has a bounded approximation ratio, it cannot handle large temporal graphs as shown in the experiments. The reasons are as follows: (1) the number of maximal lasting (k,σ)-cores increases squarely as $|\mathcal{T}|$ grows. Thus, *GreLC* requires a lot of memory to store $LC_\sigma^k(\mathcal{G})$ for selecting greedily; (2) the time complexity of selecting greedily diversified top-r lasting (k,σ)-cores significantly increases with increasing $LC_\sigma^k(\mathcal{G})$; (3) enumerating the maximal lasting (k,σ)-core in each $G(I)$, checking maximal (Line 17-19 of Algorithm 2) and maximum r-coverage are isolated. Therefore, the following four challenges need to be addressed. The first challenge is how to avoid storing all maximal lasting (k,σ)-cores in memory. The second challenge is how to avoid enumerating all maximal lasting (k,σ)-cores for computing the final result efficiently. The third is how to interleave enumeration procedure, checking maximal and maximum r-coverage. The final challenge is how to guarantee the quality of results when not all maximal lasting (k,σ)-cores are enumerated and stored. In following section, we will propose some more efficient algorithms to solve $DLCP$ with a comparable and guaranteed approximation ratio.

4 The TopLC Algorithm

We devise a DFS-based search algorithm, named *TopLC*, to dynamically store at most r candidate maximal lasting (k,σ)-cores that facilitates the pruning process. Specifically, we first propose some rules to effectively update the candidate results when a new maximal lasting (k,σ)-core is generated in Section 4.1. Then, we introduce the DFS-based search algorithm in Section 4.2. Finally, a powerful temporal graph reduction algorithm is developed in Section 4.3, which essentially reduces the search cost.

4.1 Updating the Candidate Results

Assume that our search algorithm keeps at most r candidate maximal lasting (k,σ)-cores in a set $\mathcal{R}$, an important problem is how to update $\mathcal{R}$ when a new maximal lasting (k,σ)-core H is generated. For any $R\in\mathcal{R}$, we let $\Delta(R,\mathcal{R})=cov(R)\backslash cov(\mathcal{R}\setminus\{R\})$ and $R_{min}=\arg\min_{R\in\mathcal{R}}|\Delta(R,\mathcal{R})|$. Intuitively, $\Delta(R,\mathcal{R})$ represents the private coverage of R in $\mathcal{R}$ and R_{min} has minimal private coverage among all maximal lasing (k,σ)-cores in $\mathcal{R}$. The following two rules are used to update set $\mathcal{R}$:

Rule 1. When $|\mathcal{R}|<r$, $\mathcal{R}=\mathcal{R}\cup\{H\}$;

Rule 2. When $|\mathcal{R}|=r$ and $|cov((\mathcal{R}\setminus\{R_{min}\})\cup\{H\})|\geq(1+\frac{1}{r})|cov(\mathcal{R})|$, $\mathcal{R}=(\mathcal{R}\setminus\{R_{min}\})\cup\{H\}$;

Obviously, Rule 1 is self-explanatory. For Rule 2, we only update set $\mathcal{R}$ when the coverage increases is not less than $\frac{1}{r}$ times before the update. Consequently, the two rules can guarantee that $\mathcal{R}$ has an approximate ratio of $\frac{1}{4}$ according to [16].

Basic Update Method. For updating set $\mathcal{R}$, a basic method is to traverse vertices and timestamps in H and $\mathcal{R}$. The basic method needs $O(nr^2\cdot|\mathcal{T}|)$ time to get R_{min} and $O(nr\cdot|\mathcal{T}|)$ time to get $cov(R)$ or $cov((\mathcal{R}\setminus\{R_{min}\})\cup\{H\})$. Thereby, the time complexity of the basic update method is $O(nr^2\cdot|\mathcal{T}|)$. Unfortunately, our search algorithm may need to update $\mathcal{R}$ frequently (see Section 4.2), thus using this basic method is time consuming. To reduce the complexity of the update procedure and to boost algorithm design in Section 4.2, we convert Rule 2 into the following rule:

Rule 2′. When $|\mathcal{R}|=r$ and $|\Delta(H,(\mathcal{R}\setminus\{R_{min}\})\cup\{H\})|\geq|\Delta(R_{min},\mathcal{R})|+\frac{|cov(\mathcal{R})|}{r}$, $\mathcal{R}=(\mathcal{R}\setminus\{R_{min}\})\cup\{H\}$;

Effective Update Method. With this new rule, we propose an effective algorithm with two indexes to execute the update,

which is outlined in Algorithm 3. Algorithm 3 maintains dictionary A and B. For each entry of A, the key is a pair (v,t) and the value is $A(v,t)=\{(G_S,I)\in\mathcal{R}|v\in S,t\in I\}$. For each entry of B, the key is an integer i and the value is $B(i)=\{R\in\mathcal{R}||\Delta(R,\mathcal{R})|=i\}$. Clearly, R_{min} can be got from B in constant time by indexing the smallest key of B. In Line 1-2, the algorithm executes the Rule 1. Rule 2′ is executed in Line 3-6. In Algorithm 3, $Pcov(H,\mathcal{R})$ is used to compute the private coverage of H in $(\mathcal{R}\setminus\{R_{min}\})\cup\{H\}$, which can be divided into two parts. One part is not covered by $\mathcal{R}$ (Line 9-11), and the other is only covered by H (Line 12-13). For operation $Add(\mathcal{R},H)$, if (v,t) is not a key in A, the algorithm initializes $A(v,t)=\{H\}$ and $|\Delta(H,\mathcal{R})|$ is increased by 1 due to (v,t) is only covered by H (Line 17-19). If (v,t) is a key in A and only covered by a single maximal lasting (k,σ)-core R, the algorithm updates B in Line 21-22 since (v,t) will not be covered only by R after inserting H. In the same way, $Delete(\mathcal{R})$ removes R_{min} from R and B (Line 25). Meanwhile, the procedure updates dictionary A and B accordingly (Line 26-32).

Algorithm 3. Update($\mathcal{R}$, H, r)

Input: The candidate result $\mathcal{R}$, a mew maximal lasting (k,σ)-core H and a parameter r
Output: The updated result $\mathcal{R}$

1 **if** $|\mathcal{R}|<r$ **then**
2 $Add(\mathcal{R},H)$
  else
3 $|cov(\mathcal{R})|=|A|$
4 **if** $Pcov(H,\mathcal{R})\geq|\Delta(R_{min},\mathcal{R})|+\frac{|cov(\mathcal{R})|}{r}$ **then**
5 $Delete(\mathcal{R})$
6 $Add(\mathcal{R},H)$
  Procedure $Pcov(H,\mathcal{R})$
7 $count\leftarrow 0$ and obtain R_{min} from B
8 **for** $v,t\in H$ **do**
9 **if** $(v,t)\notin A$ **then**
10 $count\leftarrow count+1$
11 continue
12 **if** $v,t\in R_{min}$ *and* $|A(v,t)|=1$ **then**
13 $count\leftarrow count+1$
14 return count
  Procedure $Add(\mathcal{R},H)$
15 add H into $\mathcal{R}$ and $|\Delta(H,\mathcal{R})|\leftarrow 0$
16 **for** $v,t\in H$ **do**
17 **if** $(v,t)\notin A$ **then**
18 $A(v,t)=\{H\}$ and $|\Delta(H,\mathcal{R})|\leftarrow|\Delta(H,\mathcal{R})|+1$
19 continue
20 **if** $|A(v,t)|=1$ *and* $A(v,t)=\{R\}$ **then**
21 move R in B from $B(|\Delta(R,\mathcal{R})|)$ to $B(|\Delta(R,\mathcal{R})|-1)$
22 $|\Delta(R,\mathcal{R})|\leftarrow|\Delta(R,\mathcal{R})|-1$
23 insert H into $A(v,t)$
24 insert H into B based $|\Delta(H,\mathcal{R})|$
  Procedure $Delete(\mathcal{R})$
25 obtain R_{min} from B and remove R_{min} from $\mathcal{R}$ and B
26 **for** $v,t\in R_{min}$ **do**
27 remove R_{min} from $A(v,t)$
  if $|A(v,t)|=1$ *and* $A(v,t)=\{R\}$ **then**
28 move R in B from $B(|\Delta(R,\mathcal{R})|)$ to $B(|\Delta(R,\mathcal{R})|+1)$
29 $|\Delta(R,\mathcal{R})|\leftarrow|\Delta(R,\mathcal{R})|+1$
30 continue
31 **if** $|A(v,t)|=0$ **then**
32 remove (v,t) from A

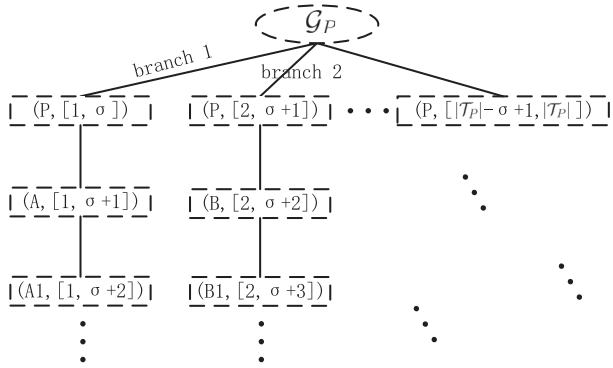


Fig. 3. Illustration for DFS-based search algorithm.

Theorem 4. *The time complexity of Algorithm 3 is* $O(\max\{|V(H)|\cdot|I(H)|,|V(R_{min})|\cdot|I(R_{min})|\})$*, in which* $V(H)$ *and* $I(H)$ *(resp.* $V(R_{min})$ *and* $I(R_{min})$*) are the vertices and interval of* H *(resp.* R_{min}*).*

Proof. Since an entry can be deleted from or inserted to a dictionary in constant time. Thereby, the time complexity of $Pcov(H,\mathcal{R})$, $Add(\mathcal{R},H)$ and $Delete(\mathcal{R})$ are $O(|V(H)|\cdot|I(H)|)$, $O(|V(H)|\cdot|I(H)|)$ and $O(|V(R_{min})|\cdot|I(R_{min})|)$ respectively. So, the time complexity of Algorithm 3 is $O(\max\{|V(H)|\cdot|I(H)|,|V(R_{min})|\cdot|I(R_{min})|\})$. □

Remark 2. *Clearly, Algorithm 3 is theoretically less time complexity than the previous basic update method, and we have also verified it experimentally in Section 5.*

4.2 The DFS-Based Search Algorithm

The high-level insight of the search algorithm is to generate new maximal lasting (k,σ) -cores in depth-first manner, that is, to explore the timestamp as far as possible along each search branch before backtracking. Fig. 3 illustrates the insight, root node represents the reduced temporal graph by Algorithm 5 in Section 4.3 and each of the remaining tree nodes is a search space (S,I), in which S represents the candidate vertices and I refers to the search interval. The search order of the algorithm is to execute search branch 1 first, then execute search branch 2 after completion, and so on. Before describing the DFS-based search algorithm in detail, we design three powerful pruning techniques: candidate vertex pruning, early termination and checking maximal.

Definition 8. (Lasting (k,σ)-Interval of Node) *For temporal graph* $\mathcal{G}$*, parameter* k *and* σ *and node* u*, interval* I *is called a lasting* $((k,\sigma)$*-interval of* u *if* $|I|\geq\sigma$ *and* $d_{G_t}(u)\geq k$ *for any* $t\in I$*. Furthermore, we let* $LI(u)$ *be all lasting* $((k,\sigma)$*-intervals of* u*.*

Based on Definition 8, we propose the following lemma for pruning some unqualified candidate vertices.

Lemma 1. (Candidate Vertex Pruning) *For any search space* (S,I)*, we can safely prune vertex* u *from* S *without losing any lasting* (k,σ)*-core in* I *if* I *is not included in any interval of* $LI(u)$*.*

Proof. By Definition 8, there is time $t\in I$ such that $d_{G_t}(u)<k$ if I is not included in any interval of $LI(u)$. Thus, u cannot be included in any lasting (k,σ)-core in I according to

Definition 4. Consequently, u is removed from S without loss of accuracy. □

The DFS-based search also has the following powerful advantage: if the current search space cannot generate a lasting (k,σ)-core or is unlikely to improve the coverage quality according to Rule 2′, we can safely prune the whole search space. We state the advantage as follows.

Algorithm 4. *TopLC($\mathcal{G}$, k, σ, r)*

Input: Temporal graph $\mathcal{G}(V,\mathcal{E},\mathcal{T})$ and three parameters k, σ and r
Output: The diversified top-r lasting (k,σ)-cores of $\mathcal{G}$
1 $(\mathcal{G}_P, LI) \leftarrow Reduction(\mathcal{G}, k, \sigma)$
2 $\mathcal{R} \leftarrow \emptyset$, *flag* $\leftarrow \emptyset$
3 **for** $i = 1$ *to* $|\mathcal{T}_P| - \sigma + 1$ **then**
4 $Search(P, [i, i+\sigma-1]$, *flag*$)$
5 return $\mathcal{R}$
Procedure $Search(S, [t_s, t_e]$, *flag*$)$
6 $D \leftarrow \emptyset$
7 **for** $u \in S$ **do**
8 **if** $[t_s, t_e]$ *is not included in any interval of* $LI(u)$ **then**
9 $D.push(u)$
10 $S \leftarrow S \setminus D$
11 **if** $|\mathcal{R}| = r$ *and* $|S|(|\mathcal{T}_P| - t_s + 1) < |\Delta(R_{min}, \mathcal{R})| + \frac{|cov(\mathcal{R})|}{r}$ **then**
12 return
13 **if** $C_S^k([t_s, t_e]) = \emptyset$ **then**
14 return
15 **if** *flag* $= \emptyset$ **then**
16 *flag*$.push(C_S^k([t_s, t_e]))$
else
17 H=*flag*$.pop()$
18 **if** H *is maximal by Lemma 3* **then**
19 $Update(\mathcal{R}, H, r)$
20 *flag*$.push(C_S^k([t_s, t_e]))$
21 **if** $t_e = |\mathcal{T}_P|$ **then**
22 return
23 **else**
Search(S, $[t_s, t_e + 1]$, *flag*)

Lemma 2. (Early Termination) *For any search space $(S,[t_s,t_e])$, we can safely prune the whole search space if it satisfies one of the following conditions: (1) $C_S^k([t_s,t_e]) = \emptyset$; (2) $|\mathcal{R}| = r$ and $|S|(|\mathcal{T}_P| - t_s + 1) < |\Delta(R_{min},\mathcal{R})| + \frac{|cov(\mathcal{R})|}{r}$.*

Proof. The first condition is obvious. For any lasting (k,σ)-core H of descendants of $(S,[t_s,t_e])$, $|\Delta(H,(\mathcal{R}\setminus\{R_{min}\})\cup\{H\})| \le |S|(|\mathcal{T}_P| - t_s + 1)$. Thereby, $|\Delta(H,(\mathcal{R}\setminus\{R_{min}\})\cup\{H\})| \le |\Delta(R_{min},\mathcal{R})| + \frac{|cov(\mathcal{R})|}{r}$. Consequently, by Rule 2′, we can safely prune the whole search space due to none of the descendants of $(S,[t_s,t_e])$ can be included in our $DLCP$. □

Recall that in Algorithm 2, we check the maximality of Definition 5 based on all lasting (k,σ)-cores. The time complexity increases with the number of all lasting (k,σ)-cores. However, our search algorithm doesn't enumerate all lasting (k,σ)-cores in advance. Thus, one significant challenge in implement DFS-based search algorithm is how to check maximality without enumerating all lasting (k,σ)-cores. Fortunately, we can effectively check maximality based on the following observation.

Algorithm 5. *Reduction($\mathcal{G}$, k, σ)*

Input: Temporal graph $\mathcal{G}(V,\mathcal{E},\mathcal{T})$ and two parameters k, σ
Output: The reduced temporal graph
1 Let $G_C = (C, E_C)$ be the maximal k-core of de-temporal graph G of $\mathcal{G}$
2 $Q \leftarrow \emptyset$, $D \leftarrow \emptyset$ and $LI \leftarrow [\emptyset]$
3 **for** $v \in C$ **do**
4 $count \leftarrow 0$
5 **for** $t \leftarrow 1 : |\mathcal{T}_C|$ **do**
6 $d_t(v) \leftarrow |N_{G_t}(v) \cap C|$
7 **if** $d_t(v) \ge k$ **then**
8 $count = count + 1$ and continue
9 $Q.push((v,t))$ and $d_t(v) = 0$
10 **if** $count < \sigma$ **then**
11 $Q.push((v,t_1))$ and $d_{t_1}(v) = 0$, $t_1 \in [t - count : t-1]$
else
12 $LI(v) = LI(v) \cup \{[t - count, t-1]\}$
13 $count \leftarrow 0$
14 **while** $Q \neq \emptyset$ **do**
15 $(v,t) \leftarrow Q.pop()$ and $D = D \cup \{(v,t)\}$
16 **for** $u \in N_{G_t}(v) \cap C$ *and* $d_t(u) \neq 0$ **do**
17 $d_t(u) = d_t(u) - 1$
18 **if** $d_t(u) < k$ **then**
19 UpdateLasting($LI(u), t$)
20 return $(\mathcal{G}_C \setminus D, LI)$
Procedure UpdateLasting($LI(u), t$)
21 $Q.push((u,t))$, $d_t(u) = 0$
22 Let $(s,e) \in LI(u)$ and $s \le t \le e$
23 $LI(u) = LI(u) \setminus \{[s,e]\}$
24 **if** $t - s \ge \sigma$ **then**
25 $LI(u) = LI(u) \cup \{[s, t-1]\}$
else
26 $Q.push((u,t_1))$ and $d_{t_1}(u) = 0$, $t_1 \in [s : t-1]$
27 **if** $e - t \ge \sigma$ **then**
28 $LI(u) = LI(u) \cup \{[t+1, e]\}$
else
29 $Q.push((u,t_2))$ and $d_{t_2}(u) = 0$, $t_2 \in [t+1 : e]$

Observation 1. *A search space cannot be contained by another search space in the subsequent search branch, but it can be contained by its descendants.*

Lemma 3 (Checking Maximal). *Given a lasting (k,σ)-core $(G_S,[t_s,t_e])$, $(G_S,[t_s,t_e])$ is maximal if (i) $S \neq V(R)$, in which R is the lasting (k,σ)-core generated by the search space $(S,[t_s,t_e+1])$; (ii) $(G_S,[t_s,t_e])$ is maximal in $\mathcal{R}$.*

Proof. According to Observation 1, it is only necessary to pay attention to the descendants of $(S,[t_s,t_e])$ and the elements in $\mathcal{R}$ when doing the maximal check on a lasting (k,σ)-core $(G_S,[t_s,t_e])$. Thus, we can safely claim that $(G_S,[t_s,t_e])$ is maximal if conditions (1) and (2) hold. □

With these powerful pruning techniques, we introduce an efficient algorithm $TopLC$, which is sketched in Algorithm 4, to implement the DFS-based search. Concretely, the algorithm first applies Algorithm 5 to reduce some unqualified nodes and obtain all lasting ((k,σ)-intervals of each node in Line 1 (more detail in Section 4.3). And then the algorithm initializes the candidate result set $\mathcal{R}$ and a flag (Line 2). Note that the flag is used to test whether the element in flag is maximal by Lemma 3. Subsequently, the algorithm performs depth-first search by starting each

search branch with the shortest interval (Line 3-4). Finally, the algorithm outputs $\mathcal{R}$ as the diversified top-r lasting (k,σ)-cores of $\mathcal{G}$.

For the Procedure $Search$, it first checks whether each node in S should be preserved by Lemma 1 (Line 6-9). And then the algorithm invokes Lemma 2 to determine whether prune the current search space (Line 11-20). If it is, the algorithm jumps to next search branch (Line 11-14). Otherwise, in Line 15-20, the algorithm invokes Lemma 3 to execute the maximal check and calls Algorithm 3 to update $\mathcal{R}$. In Line 21-23, the algorithm extends the current search interval into next search space. Clearly, Algorithm 4 performs a DFS-based enumeration search with some update rules in Section 4.1. Thereby, it can derive correctly $DLCP$ based on the correctness of above three pruning techniques and Algorithm 5, which is stated in Section 4.3.

4.3 Temporal Graph Reduction

In this subsection, we design a temporal graph reduction algorithm to implement Line 1 of Algorithm 4.

Lemma 4 (Temporal Graph Reduction). *For a vertex u, let $\mathcal{T}_u^- = \{t | d_{G_t}(u) < k\}$ and $\mathcal{T}_u^+ = \{I | d_{G_t}(u) \geq k, t \in I\}$. We can safely prune u at any timestamp $t \in \mathcal{T}_u^-$ and within $I \in \mathcal{T}_u^+$ such that $|I| < \sigma$.*

By Definition 4, we know the lemma is clearly true. Intuitively, $\mathcal{T}_u^-$ and $\mathcal{T}_u^+$ are used to prune some unqualified vertices with low degrees and short durations respectively. Meanwhile, we also know $LI(u) = \{I \in \mathcal{T}_u^+ || I| \geq \sigma\}$ according to Definition 8. Thus, we design Algorithm 5 to implement Lemma 4 and get all lasting (k,σ)-intervals of each node.

Concretely, Algorithm 5 first computes the maximal k-core of G (Line 1) and initializes queue Q to maintain all (v,t) that means v need to be deleted at time t by Lemma 4 and set D to collect deleted pair (v,t) (Line 2). Subsequently, in Line 4-13, the algorithm checks whether each (v,t) should be deleted and obtains temporary $LI(v)$. And then, the algorithm processes iteratively pair (v,t) in Q to reduce more unqualified nodes and update $LI(v)$ (Line 14-19 and Line 21-29). Finally, the algorithm returns the reduced temporal graph and all lasting (k,σ)-intervals of each node (Line 20).

The following example illustrates the whole procedure of DFS-based search algorithm.

Example 4. Reconsider temporal graph $\mathcal{G}$ in Fig. 1a. We assume that $k=2$, $\sigma=2$ and r=3. Algorithm 4 first recalls Algorithm 5 to obtain $\mathcal{G}_P$ and LI. In particular, by Line 3-13 of Algorithm 5, a is pruned at time 4 and 5 due to its low degrees and short duration according to Lemma 4. Similarly, b and d are pruned at time 5 and 1, respectively. Algorithm 5 further processes iteratively the previous deletion vertices and updates LI. Consequently, f and e are both pruned at time 1. Meanwhile, we obtain $LI(a) = \{[1,3]\}$, $LI(b) = \{[1,3]\}$, $LI(c) = \{[1,5]\}$, $LI(d) = \{[2,5]\}$, $LI(e) = \{[2,5]\}$, $LI(f) = \{[2,5]\}$. Then, Algorithm 4 executes depth-first search as illustrated in Fig. 3. The search space $(\{a,b,c,d,e,f\},[1,3],\emptyset)$ is processed first. Since [1,3] is not included in any interval of $LI(d)$, $LI(e)$ or $LI(f)$, d, e and f are pruned by Lemma 1. And then the search space passes the early termination as stated in Lemma 2 and is pushed in flag for later maximal checking. Subsequently, it extends the current search space $(\{a,b,c\},[1,3],C^k_{\{a,b,c\}}[1,3])$ into next search space $(\{a,b,c\},[1,4],C^k_{\{a,b,c\}}[1,3])$. For search space $(\{a,b,c\},[1,4],C^k_{\{a,b,c\}}[1,3])$, since [1,4] is not included in any interval of $LI(a)$ or $LI(b)$, a and b are pruned by Lemma 1 and the search space cannot pass the early termination, resulting in that the search branch is cut off and jumps to next search branch. Subsequently, the search space $(\{a,b,c,\ d,e,f\},[2,4],C^k_{\{a,b,c\}}[1,3])$ is processed. Since [2,4] is not included in any interval of $LI(a)$ or $LI(b)$, a and b are pruned by Lemma 1. And then the search space passes the early termination and the element $C^k_{\{a,b,c\}}[1,3]$ in flag also passes maximal checking by Lemma 3, resulting in that $C^k_{\{a,b,c\}}[1,3]$ can be used to update $\mathcal{R}$ by executing Algorithm 3. For the remaining search space, we use the same procedure, and we will obtain the diversified top-3 lasting (2,2)-cores: $\{C^k_{\{a,b,c\}}[1,3], C^k_{\{c,d,e\}}[2,4], C^k_{\{a,b,c,d,e,f\}}[2,3]\}$.

5 Experimental Evaluation

Here, we conduce extensive experiments for evaluating the efficiency and effectiveness of our proposed solutions. All experiments are conducted on a server with an Intel Xeon 2.50GHZ CPU and 32GB memory running Ubuntu 18.04.

5.1 Experimental Setup

Datasets. Six real-world datasets[1] with different types are used in our experiments. *Lkml* and *Enron* are communication networks that appear in Linux kernel mailing list and *Enron* company respectively, in which vertices represent users while edge (u,v,t) signifies that u sent v a message at time t. *Last* is a co-listening network that appears in the Last.fm streaming platform. *DBLP* is a scientific collaboration network, in which vertices represent authors and each temporal edge (u,v,t) refers to the authors u and v coauthored a publication at time t. *Wiki* is a network showing that users edit the same page at the same timestamp. *Epin* is a co-rating network, where the timestamps of each edge represent when the two users co-rated one common item together. In our experiments, the self-loops are deleted and the directed temporal graphs are converted into undirected temporal graphs. In a nutshell, the statistical information of datasets are exhibited in Table 2.

Parameters. Our model involves three parameters: σ (the parameter of time duration), k (the parameter of k-core) and r (the parameter of diversification). Let p and q be the percentage of nodes and temporal edges of original graph respectively for testing the scalability of our algorithms. The default values and ranges of parameters are exhibited in Table 3. Unless otherwise stated, we set the default values of other parameters when changing a parameter.

5.2 Efficiency Evaluation

We implement five different algorithms to test the efficiency of the proposed algorithms: $Naive$, $Enum$, $GreLC$, $TopLC$ and $TopLC$-B. $Naive$ and $Enum$ are used to enumerate all maximal lasting (k,σ)-cores by applying the naive approach discussed in Section 3.1 and Algorithm 2 respectively. $GreLC$ and $TopLC$ are used to identify diversified top-r lasting (k,σ)-cores by applying Algorithms 1 and 4 respectively.

1. All datasets can be accessed from http://konect.unikoblenz.de/

TABLE 1
Summary of Notations

Symbol	Definition
$\mathcal{G}, \mathcal{G}_S$	a temporal graph and its temporal subgraph
$V, \mathcal{E}, \mathcal{T}$	the node set, temporal edge set and time domain of $\mathcal{G}$
G, G_S	the de-temporal graph of $\mathcal{G}$, a subgraph of G
$Sup(G_S)$	time support set of G_S in Definition 1
$l_\sigma(G_S)$	σ-lasting support set of G_S in Definition 2
k, σ, r	the parameters used in our model
$LC_\sigma^k(\mathcal{G})$	the set of maximal lasting (k,σ)-cores of $\mathcal{G}$
$cov(\mathcal{R})$	the cover of $\mathcal{R}$ in Definition 6
$LI(u)$	all lasting ((k,σ)-intervals of u in Definition 8

$TopLC$-B is the $TopLC$ algorithm with the basic update method stated in Section 4.1. Since no existing work mines diversified top-r lasting (k,σ)-cores from temporal networks, we use $Naive$ and $GreLC$ as baselines for efficiency testing. The experimental results are reported as follows.

Exp-1: Running Time of Naive and Enum. In this experiment, we report the running time of enumerating all maximal lasting (k,σ)-cores. Specifically, we report the running time of $Naive$ and $Enum$ with default parameters on all datasets. Other parameters can obtain similar trends. By Fig. 5, the running time of the $Enum$ is 1-3 orders of magnitude faster than the $Naive$ on all datasets. For instance, $Enum$ takes 89864 milliseconds to enumerate all maximal lasting (k,σ)-cores from DBLP dataset, while $Naive$ takes 5231769 milliseconds. These results demonstrate that Corollary 1 applied in $Enum$ can indeed reduce the computational cost.

Exp-2: Running Time of Various DLCP Mining Algorithms. Fig. 6 illustrates the efficiency of various $DLCP$ mining algorithms with default parameters on all datasets. Clearly, $TopLC-B$ is consistently faster than $GreLC$ on all datasets. For instance, $TopLC-B$ takes 22521 milliseconds to mine $DLCP$ from DBLP dataset, while $GreLC$ takes 109421 milliseconds. These results indicate our proposed pruning techniques in Section 4.3 are very effective in practice. Moreover, $TopLC$ is more efficient than $TopLC-B$ on all datasets. For instance, $TopLC$ only takes 6042 milliseconds to identify $DLCP$ from DBLP dataset. These results are consistent with the theoretical analysis in Section 4.1. However, we also observe that $TopLC$, $TopLC-B$ and $GreLC$ have similar running time on Last, Wiki and Epin. The reason for this phenomenon can be explained as follows. These three datasets are interactive frequently, that is, the topology of each dataset changes significantly from one timestamp to another. Thereby, they have fewer maximal lasting (k,σ)-cores, resulting in that the optimization strategies in $TopLC$ are greatly discounted.

TABLE 2
Datasets

Dataset	$n=\lvert V\rvert$	$m=\lvert\mathcal{E}\rvert$	$\lvert\mathcal{T}\rvert$	$\bar{m}=\lvert E\rvert$	$\mathcal{TU}$
Lkml	26,885	328,092	98	159,996	1 Month
Enron	86,803	498,994	49	296,831	1 Month
Last	992	4,432951	77	369,973	27 Day
DBLP	1,824,701	11,865,584	80	8,344,615	1 Year
Wiki	298,386	18,086,734	101	10,519,921	56 Day
Epin	109,757	33,412,111	25	24,994,363	21 Day

$\mathcal{TU}$ is the time unit for each snapshot.

TABLE 3
Parameter Setting

Parameter	Range	Default Value
σ	2,3,4,5,6	3
k	3,4,5,6,7	4
r	5, 10, 15, 20, 25	10
p	20%, 40%, 60%, 80%, 100%	100%
q	20%, 40%, 60%, 80%, 100%	100%

Exp-:3 Running Time of TopLC With Varying Parameters. Because $TopLC$ is faster than the other algorithms, we only report the effect of different parameters on the efficiency of $TopLC$ in this experiment. Figs. 4a and 4b illustrate the results with varying k and σ on all datasets. As we have seen, the running time decreases with increasing k or σ. Because the performance of pruning candidate vertices and early termination is improved with increasing k or σ, more vertices or search spaces are pruned. Fig. 4c reports the results with varying r. Differently, the running time of $TopLC$ remains quite stable with varying r on all datasets. Because the number of maximal lasting (k,σ)-cores is unchange even for a large r, resulting in that the number of search spaces does not change with r. Thereby, the $TopLC$ is insensitive to parameter r. It further proves that i$TopLC$ is more efficient than the greedy algorithm $GreLC$.

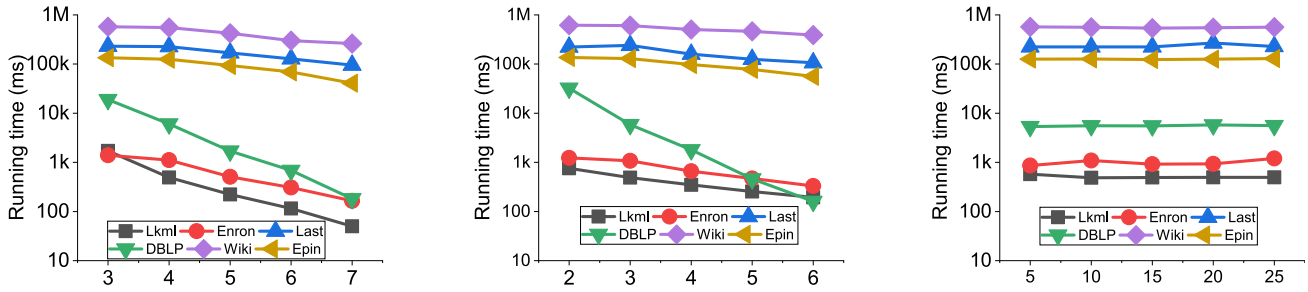
Exp-4: Coverage Quality of GreLC and TopLC With Varying Parameters. We report the coverage quality of $GreLC$ and $TopLC$ with varying parameters on Enron and DBLP in Fig. 7. Note that we do not consider the effect of r on the coverage quality, because the bigger r is, the better the quality is according to Definition 7. The other datasets can also observe similar trends. As can be seen, the coverage quality obtained by $GreLC$ and $TopLC$ are almost equal. It further demonstrates the approximation ratios of *TopLC* and *GreLC* are also almost equal in practice. As expected, the coverage decreases with an increasing k or σ. Because with a larger k or σ, the restriction of maximal lasting (k,σ)-core will be stronger, resulting in that the coverage of $DLCP$ decreases with increasing k or σ.

Exp-5: Scalability Testing. The larger dataset Epin is chosen to evaluate the scalability of $GreLC$ and $TopLC$ under default parameter. Concretely, we generate five temporal subgraphs by varying parameter p and q as stated in Table 3 (i.e., randomly choosing 20-100 percent nodes or temporal edges from the Epin). Subsequently, we test the running time of $GreLC$ and $TopLC$ on these temporal subgraphs. As Fig. 8 illustrates, $GreLC$ and $TopLC$ scales near liner with respect to the size of the temporal subgraphs. Consequently, our algorithms can handle large real-world temporal networks.

5.3 Effectiveness Evaluation

For testing the effectiveness, we compare our model against four baseline models: $Kcore$, $DivClique$ [20], $TDense$ [7] and $Pcore$ [10]. The $Kcore$ is a conventional cohesive subgraph model that applies the peeling algorithm [21] to compute the maximal k-core from the de-temporal graph G. The $DivClique$, $TDense$ and $Pcore$ are state-of-the-art temporal cohesive subgraph models (see Section 6 for details).

Effectiveness Metrics. Most well-known effectiveness metrics (e.g., conductance or density) only consider structural



(a) Vary k (b) Vary σ (c) Vary r

Fig. 4. Running time of $TopLC$ with varying parameters.

information but temporal attributes. Fortunately, there are two goodness metrics, burstiness [14] and temporal conductance [22], which can measure the average density and the temporal separability of a single temporal subgraph respectively. Let $\mathcal{R} = \{(S_1, I_1), (S_2, I_2), \ldots, (S_r, I_r)\}$ be a set of temporal subgraphs, we generalize above two metrics to accommodate our top-k problem as follows.

Average burstiness (AB) calculates the density of the internal structure of the temporal subgraphs. That is, good temporal cohesive subgraphs should be densely connected by internal structure. Formally, $AB(\mathcal{R}) = [\sum_{(S_i,I_i)\in\mathcal{R}} \frac{|\{(u,v,t)\in\mathcal{E}|u,v\in S_i,t\in I_i\}|}{|S_i|(|S_i|-1)|I_i|}]/r$.

Average temporal conductance (ATC) measures the separability of temporal subgraphs, that is good temporal cohesive subgraphs well-separated from the rest of the graph. Formally, $ATC(\mathcal{R}) = [\sum_{(S_i,I_i)\in\mathcal{R}} \frac{cut(S_i,I_i)}{\min\{vol(S_i,I_i),vol(V\setminus S_i,I_i)\}}]/r$, in which $cut(S,I) = |\{(u,v,t) \in \mathcal{E} | u \in S, v \in V \setminus S, t \in I\}|$ and $vol(S,I) = \sum_{u\in S}\sum_{t\in I} d_{G_t}(u)$.

Intuitively, the larger $AB(\mathcal{R})$ is, the denser $\mathcal{R}$ is in the whole temporal extent. In a similar way, the smaller $ATC(\mathcal{R})$ is, the farther away $\mathcal{R}$ is from the rest of the graph.

Exp-6: Effectiveness of Kcore, DivClique, TDense, Pcore and DLCP. In this experiment, we report the effectiveness of the temporal subgraphs detected using different models under with their default parameters on all datasets. The other parameters can also observe similar results. According to Table 4, we see that the best scores are achieved by our model on all datasets except for Lkml in terms of ATC metric. The result indicates our model is more capable of preserving temporal separability than other baseline models. On the other

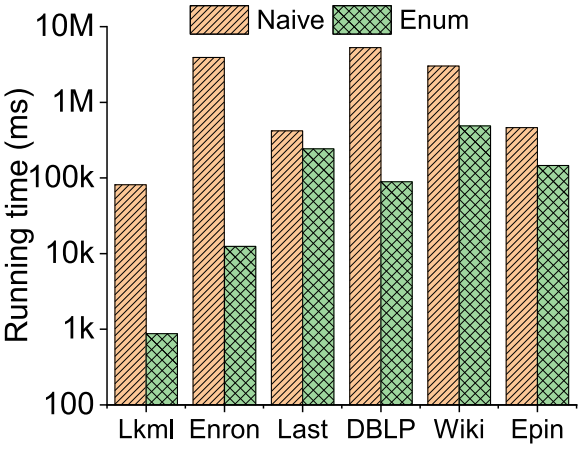


Fig. 5. Running time of $Naive$ and $Enum$.

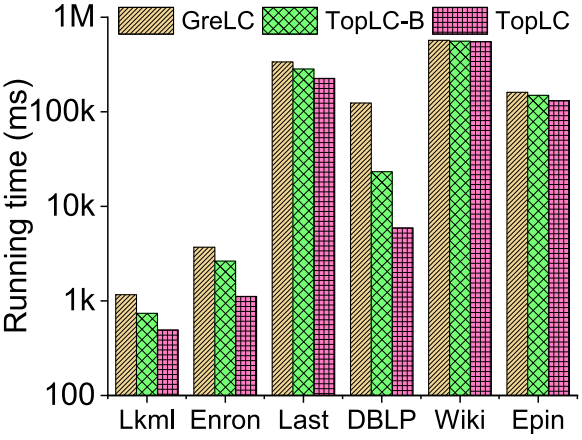
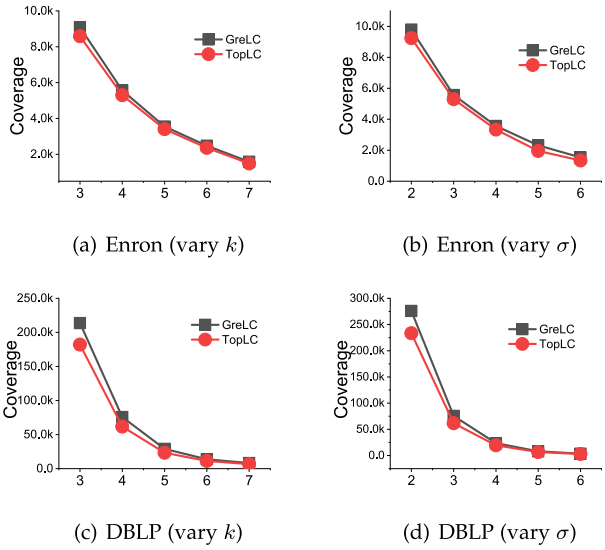


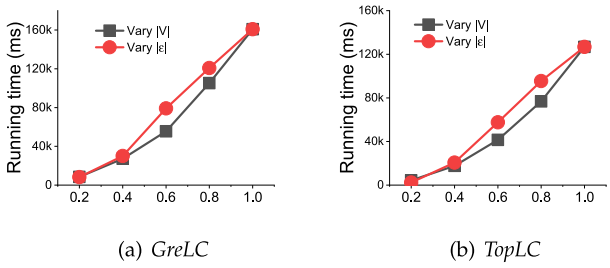
Fig. 6. Running time of various $DLCP$ mining algorithms.



(a) Enron (vary k) (b) Enron (vary σ)

(c) DBLP (vary k) (d) DBLP (vary σ)

Fig. 7. Coverage quality of $GreLC$ and $TopLC$ with varying parameters.



(a) $GreLC$ (b) $TopLC$

Fig. 8. Scalability on Epin ($k = 4$, $\sigma = 3$ and $r = 10$).

TABLE 4
Effectiveness of $Kcore$, $DivClique$, $TDense$, $Pcore$, and $DLCP$

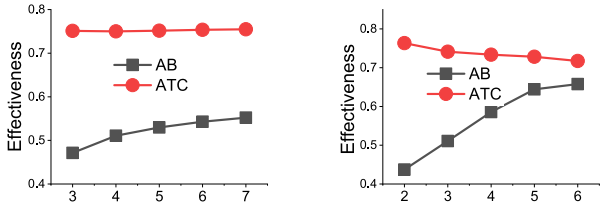
		Lkml	Enron	Last	DBLP	Wiki	Epin
AB	$Kcore$	10^{-4}	10^{-5}	0.11	10^{-7}	10^{-5}	10^{-4}
	$DivClique$	0.01	0.03	0.74	10^{-3}	0.48	0.35
	$TDense$	**0.68**	**0.23**	**0.85**	**0.15**	**0.59**	**0.63**
	$Pcore$	0.02	0.01	0.34	10^{-2}	0.23	0.19
	$DLCP$	0.48	0.07	0.77	0.01	0.34	0.51
ATC	$Kcore$	**0.40**	0.61	1	0.63	0.96	0.86
	$DivClique$	0.99	0.98	0.97	0.85	0.99	0.94
	$TDense$	0.93	0.85	0.96	0.72	0.87	0.89
	$Pcore$	0.82	0.69	0.61	0.68	0.98	0.89
	$DLCP$	0.76	**0.60**	**0.56**	**0.56**	**0.67**	**0.74**

The best result in each model is highlighted in bold.

hand, the $TDense$ is better than other models in terms of AB metric (but the $TDense$ has poor ATC metric), and our model is runner-up and slightly weaker than $TDense$. This is because that $TDense$ obtains the subgraphs with maximum total density, which is proportional to the AB metric. More generally, $DivClique$ and $Pcore$ outperform $Kcore$ w.r.t AB metric, while they are worse than $Kcore$ in terms of the ATC metric. The reason is that the $DivClique$ and $Pcore$ consider the temporal dimension of the graphs, so these subgraphs are more dense in the whole temporal extent. In a nutshell, this experiment indicates our model can find much denser and more separable sub-graphs in terms of temporal feature than the baselines.

Exp-7: Effectiveness of DLCP With Varying Parameters on Epin. Fig. 9 illustrates the effectiveness of our $DLCP$ model on Epin dataset by varying k or σ with $r = 10$. The other datasets can also obtain similar results. By Fig. 9a, AB increases with an increasing k. Because the structural cohesiveness of lasting (k, σ)-core is improved with a larger k, these subgraphs are more dense. Since k does not affect the temporal separability, ATC is stable with increasing k. As seen in Fig. 9b, AB increases while ATC decreases with an increasing σ. Because with a larger σ, the time duration of lasting (k, σ)-core will be stronger, these subgraphs are more dense and far away from the rest of the graph.

Exp-8: Case Study on DBLP. Fig. 10 visualizes the top-2 temporal cohesive subgraphs containing node Prof. Jiawei Han detected by our model on DBLP datasets with $k = 3$ and σ=3. Note that the cohesive subgraph identified by $Kcore$[2] contains 1225183 authors that come from diverse research domains and their cooperation was intermittent rather than lasting. This results indicate that $Kcore$ is ineffective to identify lasting cohesive subgraphs. However, as shown in Fig. 10, our model can identify lasting cohesive subgraphs from a temporal network. For example, in Fig. 10a, the cohesive subgraph obtained by our model is a stable research team, because all researchers in this cohesive subgraph collaborate closely and lastly with Prof. Jiawei Han from 2009 to 2011. Additionally, we also look at the homepage of Prof. Jiawei Han (https://hanj.cs.illinois.edu/), and find that other authors in Fig. 10a are indeed Han's academic partners during 2009-2011. Similar results can also be seen in Fig. 10b, in which other authors are Han's students during 2014-2016. These results indicate



(a) Vary k (b) Vary σ

Fig. 9. Effectiveness of $DLCP$ with varying parameters on Epin.

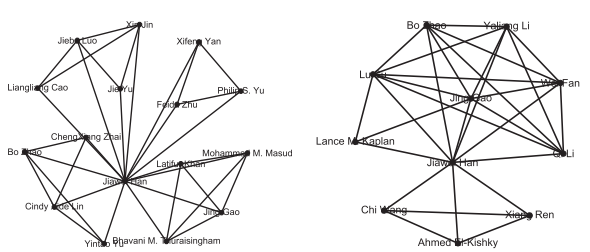
that our model can indeed identify lasting cohesive subgraphs from a temporal network.

6 Related Work

Identifying diversified lasting cohesive subgraphs from temporal networks is a novel problem that has not been adequately investigated in literature.

Cohesive Subgraph Mining. Retrieving cohesive subgraphs on static networks is to seek subgraphs such that the vertices in the subgraph are connected densely, which has been well investigated in past decades [11]. Notable models include densest subgraph [23], [24], clique [25], [26], quasi-clique [27], [28], k-truss [29], [30], k-core [15], [31], and more. For example, Epasto *et al.* [23] studied densest subgraph on large and highly dynamic networks, where the edges are deleted uniformly and added adversely. Yuan *et al.* [25] investigated diversified cliques problem, which is to identify k maximal cliques so as to maximize vertex coverage. Boden *et al.* [27] investigated γ-quasi-clique on multi-layer attributed graphs. Zhang *et al.* [29] adopted the k-truss to model the engagement and tie strength of social networks, and investigated the anchored k-truss problem that is to preventing network unraveling by fixed some critical users. The model of k-core [15] is the closest to our work. The core decomposition has been widely investigated [31] with the applications in social networks, visualization, software engineering, and so forth. Furthermore, cohesive subgraphs are also central to many high-impact applications, including user engagement [18], influence evaluation [19] and community search [17].

Temporal Graph Analyzing. Temporal graph analysis has recently emerged as an important research filed ranging from computer science and bioscience to physics and mathematics [1], [2], [3]. In the literature, some classical graph analysis problems have been extended to temporal graphs, such as reachability and shortest path queries [4], [5], motifs mining [8] and minimum spanning tree [32]. Our work is also related to temporal cohesive subgraph mining, which has been discussed with different disciplines [6], [7], [10], [12], [14], [20],



(a) 2009-2011 (b) 2014-2016

Fig. 10. Case study on DBLP.

2. We do not visualize this cohesive subgraph due to it is too large to show in a figure

[33]. For example, [12] modeled a dense temporal subgraph so as to maximize the sum of edge weights from a special temporal graph with unchanged topology but changing edge weights over time. Li *et al.* [10] explored persistent communities, a k-core structure in any θ-length subinterval during a given interval. Chu *et al.* [14] studied burst communities, where the community aggregates its average degree at the fastest speed. Galimberti *et al.* [33] proposed the span-core model for core decomposition on temporal graphs. Qin *et al.* [6] proposed an interesting temporal model, called periodic cliques, for characterizing the periodic behavior of cohesive subgraphs. Yang *et al.* [20] investigated the diversified γ-denses on temporal graphs and devised a divide-and-conquer algorithm framework with some powerful pruning strategies. γ-dense maintains a γ-quasi-clique structure in any timestamp in a given time interval, but their interlink structures may be different from one timestamp to another. Thus, the γ-dense cannot model the lasting of a cohesive subgraph. Rozenshtein *et al.* [7] extended the traditional densest subgraph [23], [24] to temporal networks for modeling the timelines of events. Specifically, they split the whole time domain into k non-overlapping intervals, such that the intervals span subgraphs with maximum total density. Again, their method also cannot capture the lasting of a cohesive subgraph. Unlike these literatures, our work is to study the diversification and lasting of cohesive subgraphs on temporal networks.

Lasting Pattern Discovering. To our knowledge, there are only few studies on lasting patterns [34], [35], [36], [37]. In spatio-temporal networks, convoy pattern (e.g., a group of people moving together for at least σ consecutive timestamps) can be regraded as a lasting pattern [37]. In dynamic sensor networks, the lasting connected component can serve as a backbone for convenient message transmission [35]. Ahmed *et al.* [34] proposed a stable subgraph model such that the detected subgraph is unchanged during a certain time period. However, their work did not consider the cohesiveness of the stable subgraph. So, Liu *et al.* [36] devised a stochastic algorithm framework to extract a single cohesive subgraph on dynamic networks such that the subgraph has the highest accumulated density and long-lasting. Although the model is similar to our lasting (k, σ)-core, it is a probabilistic subgraph model that characterizes the edge appearance with probability in a dynamic network. It extracted only a cohesive subgraph (i.e., it did not consider the diversification of results). Consequently, these approaches cannot directly detect diversified top-r lasting (k, σ)-cores.

Diversified Top-r Searching. The goal of this problem is to identify the top-r results that are most related to user-initiated queries in consideration of diversity [38]. In literature, some specific problems have been studied, such as skyline query [39], keyword search [40], document retrieval [41], [42], pattern matching [43], ranking [44], maximal clique [45], coherent core [46] and (k, r)-core [47] by taking diversification into consideration. However, these approaches only consider the structure or keyword but the temporal nature of a subgraph. Moreover, it is not clear how the techniques can be applied to solve our proposed diversified top-r lasting (k, σ)-cores. Thus, to the best of our knowledge, we are the first to combine lasting pattern into diversified top-r searching.

7 Conclusion and Future Work

In this paper, we are the first to systematically propose a novel diversified top-r lasting (k, σ)-cores to model both the diversification and the time duration of cohesive subgraphs on temporal graphs. And then we demonstrate that our problem is NP-hard. Subsequently, a greedy algorithm *GreLC* with $(1 - 1/e)$ approximation ratio and a DFS-based search algorithm *TopLC* with $1/4$ approximation ratio are proposed to effectively tackle our problem. Finally, our comprehensive experiments illustrate the efficiency, scalability and effectiveness of our solutions.

There are some interesting further directions. (1) Adopting others possible models (e.g., clique, k-truss or densest subgraph) to model the lasting cohesive subgraph on temporal graphs. (2) Considering online temporal cohesive subgraph search problem on big temporal networks, which is more personalized by inputting given query seeds.

Acknowledgments

This work was supported by the National Key Research & Development Program of China under Grant 2018YFB1004002 and NSFC under Grants 61672255 and 61932004.

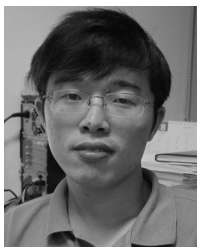
References

[1] P. Holme, "Modern temporal network theory: A colloquium," *Eur. Physical J. B*, vol. 88, no. 9, pp. 1–30, 2015.
[2] P. Rozenshtein and A. Gionis, "Mining temporal networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 3225–3226.
[3] Y. Wang, Y. Yuan, Y. Ma, and G. Wang, "Time-dependent graphs: Definitions, applications, and algorithms," *Data Sci. Eng.*, vol. 4, pp. 352–366, 2019.
[4] H. Wu, Y. Huang, J. Cheng, J. Li, and Y. Ke, "Reachability and time-based path queries in temporal graphs," in *Proc. IEEE 32nd Int. Conf. Data Eng.*, 2016, pp. 145–156.
[5] H. Wu, J. Cheng, S. Huang, Y. Ke, Y. Lu, and Y. Xu, "Path problems in temporal graphs," *Proc. VLDB Endowment*, vol. 7, no. 9, pp. 721–732, 2014.
[6] H. Qin, R. Li, G. Wang, L. Qin, Y. Cheng, and Y. Yuan, "Mining periodic cliques in temporal networks," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 1130–1141.
[7] P. Rozenshtein, F. Bonchi, A. Gionis, M. Sozio, and N. Tatti, "Finding events in temporal networks: Segmentation meets densest-subgraph discovery," in *Proc. IEEE Int. Conf. Data Mining*, 2018, pp. 397–406.
[8] R. Kumar and T. Calders, "2SCENT: An efficient algorithm to enumerate all simple temporal cycles," *Proc. VLDB Endowment*, vol. 11, no. 11, pp. 1441–1453, 2018.
[9] I. W. Taylor *et al.*, "Dynamic modularity in protein interaction networks predicts breast cancer outcome," *Nat. Biotechnol.*, vol. 27, no. 2, pp. 199–204, 2009.
[10] R. Li, J. Su, L. Qin, J. X. Yu, and Q. Dai, "Persistent community search in temporal networks," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2018, pp. 797–808.
[11] L. Chang and L. Qin, "Cohesive subgraph computation over large sparse graphs," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 2068–2071.
[12] S. Ma, R. Hu, L. Wang, X. Lin, and J. Huai, "Fast computation of dense temporal subgraphs," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2017, pp. 361–372.
[13] H. Wu *et al.*, "Core decomposition in large temporal graphs," in *Proc. Int. Conf. Big Data*, 2015, pp. 649–658.
[14] L. Chu, Y. Zhang, Y. Yang, L. Wang, and J. Pei, "Online density bursting subgraph detection from temporal graphs," *Proc. VLDB Endowment*, vol. 12, no. 13, pp. 2353–2365, 2019.
[15] S. B. Seidman, "Network structure and minimum degree," *Soc. Netw.*, vol. 5, no. 3, pp. 269–287, 1983.
[16] G. Ausiello, N. Boria, A. Giannakos, G. Lucarelli, and V. T. Paschos, "Online maximum k-coverage," *Discrete Appl. Math.*, vol. 160, no. 13–14, pp. 1901–1913, 2012.

[17] M. Sozio and A. Gionis, "The community-search problem and how to plan a successful cocktail party," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 939–948.
[18] K. Bhawalkar, J. M. Kleinberg, K. Lewi, T. Roughgarden, and A. Sharma, "Preventing unraveling in social networks: The anchored k-core problem," *SIAM J. Discrete Math.*, vol. 29, no. 3, pp. 1452–1475, 2015.
[19] M. Kitsak *et al*., "Identification of influential spreaders in complex networks," *Nat. Phys.*, vol. 6, no. 11, pp. 888–893, 2010.
[20] Y. Yang, D. Yan, H. Wu, J. Cheng, S. Zhou, and J. C. S. Lui, "Diversified temporal subgraph pattern mining," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1965–1974.
[21] V. Batagelj and M. Zaversnik, "An O(m) algorithm for cores decomposition of networks," *CoRR*, vol. cs.DS/0310049, 2003.
[22] D. J. DiTursi, G. Ghosh, and P. Bogdanov, "Local community detection in dynamic networks," in *Proc. IEEE Int. Conf. Data Mining*, 2017, pp. 847–852.
[23] A. Epasto, S. Lattanzi, and M. Sozio, "Efficient densest subgraph computation in evolving graphs," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 300–310.
[24] Y. Fang, K. Yu, R. Cheng, L. V. S. Lakshmanan, and X. Lin, "Efficient algorithms for densest subgraph discovery," *Proc. VLDB Endowment*, vol. 12, no. 11, pp. 1719–1732, 2019.
[25] L. Yuan, L. Qin, X. Lin, L. Chang, and W. Zhang, "Diversified top-k clique search," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2015, pp. 387–398.
[26] L. Chang, "Efficient maximum clique computation over large sparse graphs," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 529–538.
[27] B. Boden, S. Günnemann, H. Hoffmann, and T. Seidl, "MiMAG: Mining coherent subgraphs in multi-layer graphs with edge labels," *Knowl. Inf. Syst.*, vol. 50, no. 2, pp. 417–446, 2017.
[28] C. E. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. A. Tsiarli, "Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 104–112.
[29] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, "Efficiently reinforcing social networks over user engagement and tie strength," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2018, pp. 557–568.
[30] Y. Zhang and J. X. Yu, "Unboundedness and efficiency of truss maintenance in evolving graphs," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 1024–1041.
[31] F. Bonchi, A. Khan, and L. Severini, "Distance-generalized core decomposition," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 1006–1023.
[32] S. Huang, A. W. Fu, and R. Liu, "Minimum spanning trees in temporal graphs," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2015, pp. 419–430.
[33] E. Galimberti, A. Barrat, F. Bonchi, C. Cattuto, and F. Gullo, "Mining (maximal) span-cores from temporal networks," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 107–116.
[34] R. Ahmed and G. Karypis, "Algorithms for mining the evolution of conserved relational states in dynamic networks," in *Proc. IEEE Int. Conf. Data Mining*, 2011, pp. 1–10.
[35] F. Kuhn, N. A. Lynch, and R. Oshman, "Distributed computation in dynamic networks," in *Proc. 42nd ACM Symp. Theory Comput.*, 2010, pp. 513–522.
[36] X. Liu, T. Ge, and Y. Wu, "Finding densest lasting subgraphs in dynamic graphs: A stochastic approach," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 782–793.
[37] F. Orakzai, T. Calders, and T. B. Pedersen, "k/2hop: Fast mining of convoy patterns with effective pruning," *Proc. VLDB Endowment*, vol. 12, no. 9, pp. 948–960, 2019.
[38] M. Drosou and E. Pitoura, "Search result diversification," *SIGMOD Record*, vol. 39, no. 1, pp. 41–47, 2010.
[39] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, "Selecting stars: The k most representative skyline operator," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2007, pp. 86–95.
[40] E. Demidova, P. Fankhauser, X. Zhou, and W. Nejdl, "*DivQ*: Diversification for keyword search over structured databases," in *Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2010, pp. 331–338.
[41] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong, "Diversifying search results," in *Proc. 2nd ACM Int. Conf. Web Search Data Mining*, 2009, pp. 5–14.
[42] A. Angel and N. Koudas, "Efficient diversity-aware search," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2011, pp. 781–792.
[43] W. Fan, X. Wang, and Y. Wu, "Diversified top-k graph pattern matching," *Proc. VLDB Endowment*, vol. 6, no. 13, pp. 1510–1521, 2013.
[44] R. Li and J. X. Yu, "Scalable diversified ranking on large graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 9, pp. 2133–2146, Sep. 2013.
[45] L. Yuan, L. Qin, X. Lin, L. Chang, and W. Zhang, "Diversified top-k clique search," *VLDB J.*, vol. 25, no. 2, pp. 171–196, 2016.
[46] R. Zhu, Z. Zou, and J. Li, "Fast diversified coherent core search on multi-layer graphs," *VLDB J.*, vol. 28, no. 4, pp. 597–622, 2019.
[47] F. Zhang, X. Lin, Y. Zhang, L. Qin, and W. Zhang, "Efficient community discovery with user engagement and similarity," *VLDB J.*, vol. 28, no. 6, pp. 987–1012, 2019.

Longlong Lin is currently working toward the PhD degree from the HuaZhong University of Science and Technique, China. His current research interests include social network analysis and temporal network mining.

Pingpeng Yuan received the PhD degree in computer science from Zhejiang University. He is currently a professor with the School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include databases, knowledge representation and reasoning, and NLP, with a focus on high performance computing. During exploring his research, he implements systems and innovative applications in addition to investigating theoretical solutions and algorithmic design. He is also the principle developer in multiple system prototypes, including TripleBit, PathGraph, and SemreX.



Rong-Hua Li received the PhD degree from the Chinese University of Hong Kong, in 2013. He is currently a professor with the Beijing Institute of Technology, Beijing, China. His research interests include graph data management and mining, social network analysis, graph computation systems, and graph-based machine learning.

Hai Jin (Fellow, IEEE) received the PhD degree in computer engineering from the Huazhong University of Science and Technology, in 1994. He is currently a Cheung Kung Scholars chair professor of computer science and engineering, Huazhong University of Science and Technology. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. He worked with the University of Hong Kong between 1998 and 2000, and as a visiting scholar with the University of Southern California between 1999 and 2000. He was awarded Excellent Youth Award from the National Science Foundation of China in 2001. He is the chief scientist of ChinaGrid, the largest grid computing project in China, and the chief scientists of National 973 Basic Research Program Project of Virtualization Technology of Computing System, and Cloud Security. He is a CCF fellow, and a member of the ACM. He has coauthored 22 books and published more than 700 research papers. His research interests include computer architecture, virtualization technology, cluster computing and cloud computing, peer-to-peer computing, network storage, and network security.