TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 11/15 pp344–359
DOI: 10.26599/TST.2021.9010090
Volume 28, Number 2, April 2023

Discovering Association Rules with Graph Patterns in Temporal Networks

Chu Huang, Qianzhen Zhang, Deke Guo*, Xiang Zhao, and Xi Wang

Abstract: Discovering regularities between entities in temporal graphs is vital for many real-world applications (e.g., social recommendation, emergency event detection, and cyberattack event detection). This paper proposes temporal graph association rules (TGARs) that extend traditional graph-pattern association rules in a static graph by incorporating the unique temporal information and constraints. We introduce quality measures (e.g., support, confidence, and diversification) to characterize meaningful TGARs that are useful and diversified. In addition, the proposed support metric is an upper bound for alternative metrics, allowing us to guarantee a superset of patterns. We extend conventional confidence measures in terms of maximal occurrences of TGARs. The diversification score strikes a balance between interestingness and diversity. Although the problem is NP-hard, we develop an effective discovery algorithm for TGARs that integrates TGARs generation and TGARs selection and shows that mining TGARs is feasible over a temporal graph. We propose pruning strategies to filter TGARs that have low support or cannot make top-k as early as possible. Moreover, we design an auxiliary data structure to prune the TGARs that do not meet the constraints during the TGARs generation process to avoid conducting repeated subgraph matching for each extension in the search space. We experimentally verify the effectiveness, efficiency, and scalability of our algorithms in discovering diversified top-k TGARs from temporal graphs in real-life applications.

Key words: temporal networks; graph association rule; subgraph pattern matching; graph mining; big graphs

1 Introduction

Graph-pattern association rules have been studied to discover complex relations among entities in a static graph[1–4]. They have a traditional form $Q(u_x, u_y) \Rightarrow q(u_x, u_y)$, where antecedent $Q(u_x, u_y)$ is a graph pattern in which u_x and u_y are two designated nodes and consequent $q(u_x, u_y)$ is an edge labeled q from u_x to u_y, on which the same search conditions as in Q are imposed. Specifically, $Q(u_x, u_y) \Rightarrow q(u_x, u_y)$ states, "if an antecedent $Q(u_x, u_y)$ occurs, then the consequent $q(u_x, u_y)$ is likely to occur."

Although mining association rules has been well studied for static graphs, little has been done on considering the temporal constraints. The need for mining these temporal association rules is especially urgent since many graphs in the real world come with temporal information and are typically called temporal graphs[5–7]. To be more specific, this temporal association rule states, "if an antecedent $Q(u_x, u_y)$ occurs, then the consequent $q(u_x, u_y)$ is likely to occur in time Δt." Mining temporal graph association rules (TGARs) in a temporal graph could be very useful for many practical applications[8–10], two of which are listed as follows.

Application 1: cyberattack prevention. The leakage of personal information caused by information

• Chu Huang, Qianzhen Zhang, Deke Guo, Xiang Zhao, and Xi Wang are with the Science and Technology on Information Systems Engineering Laboratory, College of Systems Engineering, National University of Defense Technology, Changsha 410073, China. E-mail: {huangchu20, zhangqianzhen18, dekeguo, xiangzhao, wangxi19}@nudt.edu.cn.

* To whom correspondence should be addressed.
Manuscript received: 2021-10-31; revised: 2021-11-19; accepted: 2021-11-22

© The author(s) 2023. The articles published in this open access journal are distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/).

exfiltration attacks is a severe problem for cyber security. The process of information exfiltration attack can be described by a temporal association R_1 between two graph patterns Q_1 and q_1 (as illustrated in Fig. 1a), where (1) Q_1 is an information exfiltration attack, in which a victim browses a compromised website and the attacker controls a bot to create massive requests to the victim, and (2) q_1 describes the result. The rule states that "if a host is involved in an information exfiltration attack (Q_1) at some time, then this host is likely to divulge personal information within 2 min". In this way, once the occurrence of this attack pattern is discovered, the preemptive measure should be invoked in a short time to avoid possible loss.

Application 2: time-aware point of interest (POI) recommendation. Social network users tend to behave in a short period triggered by social influences[11]. For example, a temporal social influence can be represented by a temporal association R_2 between two graph patterns Q_2 and q_2 (see Fig. 1b), which capture social communities and a recommendation pattern, respectively. The rule states that "if a user u_1 has a friend u_2 and u_1 retweets u_2, and u_2 checks in at point of interests (POI) u_3 (e.g., 'British Museum') at some time t (via, e.g., Facebook Place), then it is likely u_1 will visit u_3 in 2 h".

Note that most rules in a temporal graph often pertain to the same or similar people[12, 13]. Motivated by this feature, in this paper, we focus on the problem of mining diversified top-k TGARs, to find a set of k TGARs corresponding to $q(u, u')$ that has the largest diversification score. We illustrate this problem by the following example.

Example 1 Consider the temporal graph in Fig. 2a and antecedents Q_3, Q_4, and Q_5 shown in Fig. 3, all pertaining to $q(u_1, u_3)$ in Fig. 1b. R_3, R_4, and R_5 are the temporal rules corresponding to Q_3, Q_4, and Q_5, respectively. Let $k = 2$. For diversified top-k TGARs, the result is $\{R_4, R_5\}$, since R_4 and R_5 have the greatest difference. Indeed, R_4 and R_5 find two entirely different reasons for a user to check in at POI. In particular, in R_4, the main reason for a user to check in at a POI is to retweet a friend who retweeted an exhibition in a

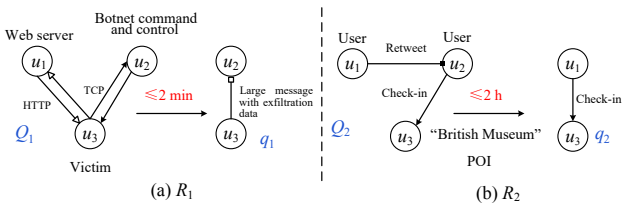


Fig. 1 Association rules in temporal graphs.

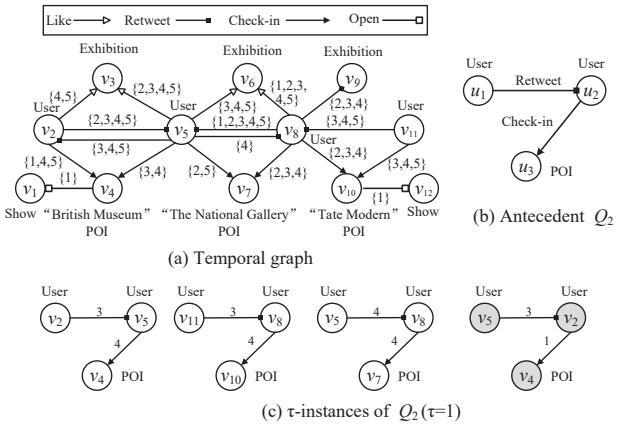


Fig. 2 Example of a temporal graph and instances.

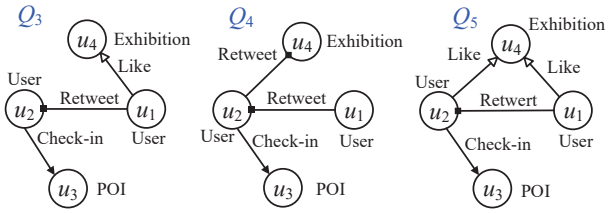


Fig. 3 Antecedents pertaining to $q_2(u_1, u_3)$ in R_2.

corresponding POI and has checked in at the POI; in R_5, the reason is that the user likes the exhibition in the corresponding POI and he (or she) retweets a friend who also likes the exhibition and has checked in at the POI.

Efforts to support TGARs seem to enjoy some success. In Ref. [14], Namaki et al. first extended graph-pattern association rules with temporal constraints over a temporal graph. They specified a time window to describe the minimal gap between the occurrences of two patterns Q_1 and Q_2. Here Q_1 and Q_2 have only one common vertex. These rules cannot model more meaningful associations among entities since: (1) their antecedents, as pattern graphs, only consider their matching instances at each timestamp and do not consider the duration feature; (2) they do not consider the diversity feature, resulting in most of the rules being highly overlapping; and (3) they use the number of minimal occurrences supported by the matches of the common vertex of Q_1 and Q_2 as a support metric. However, this setting will underestimate the actual support since these rules only once consider each match of the common vertex in Q_1 and Q_2 at each timestamp. Moreover, since Q_1 and Q_2 have only one common vertex, their method cannot be used to straightforwardly mine the graph association rule with the form $Q(u_x, u_y) \Rightarrow q(u_x, u_y)$.

To compute diversified top-k TGARs corresponding to a particular event $q(u, u')$, a straightforward solution is to enumerate all TGARs first and then select the top-

k TGARs via result diversification[15]. However, this approach is costly: (1) An excessive number of TGARs are generated; and (2) for all TGARs generated, it must compute the diversification scores for them and pick the top-k set, which is NP-hard. To this end, we propose an algorithm findTopkTGARs for diversified top-k TGARs mining that can avoid enumerating all TGARs and solve all the limitations existing in Ref. [14]. The contributions of this paper are as follows.

• We formulate support, confidence, and diversity for TGARs. We define the support of a TGAR in terms of the maximal occurrences of the antecedent and the consequent. We prove that our proposed support metric is an upper bound for alternative metrics, allowing us to guarantee a superset of patterns. We extend conventional confidence measures in terms of maximal occurrences of TGARs. We show that the support of TGARs is anti-monotonic. In addition, we define a diversification score for a set of TGARs, aiming to strike a balance between interestingness and diversity.

• We integrate pattern mining and association rule discovery in a single process to obtain high-quality rules. To avoid enumerating all TGARs to compute diversified top-k TGARs corresponding to a particular event $q(u, u')$, we propose to maintain a list L_k of diversified top-k TGARs in each round, which can help to filter TGARs that have low support or cannot make top-k as early as possible. Furthermore, in the pattern mining process, we propose an auxiliary data structure, namely, EIndex, and dynamically maintain EIndex during pattern growth. On the basis of EIndex, we can avoid conducting repeated subgraph matching for each extension in the search space.

• We conduct comprehensive experiments using three real-world datasets. The experimental results demonstrate that our algorithm achieves high effectiveness and efficiency. Remarkably, our algorithm is orders of magnitude faster than the competing baseline on all settings. We also examine one case study on the Offshore dataset. The results show that our algorithm is indeed able to identify many meaningful and diversified TGARs.

2 Problem Formulation

A temporal graph $\mathcal{G}$ is defined as a quadruple $(\mathcal{V}, \mathcal{E}, \mathbb{L}, T)$, where $\mathcal{V}$ is a set of vertices, $\mathcal{E} \in \mathcal{V} \times \mathcal{V} \times T$ is a set of (directed) edges connecting two vertices in $\mathcal{V}$, each associated with a timestamp from a universe T, and $\mathbb{L}$ is a label function that assigns labels to vertices and edges. An edge $e = \langle v, v', t \rangle$ ($t \in T$) encodes a link with label $\mathbb{L}(e)$ between v and v' that exists at timestamp t. We assume that all the timestamps in T are sorted in chronological order, joined as an arithmetic time sequence, i.e., $t_1 < t_2 < \cdots < t_{|T|}$ †. Figure 2a illustrates a temporal graph $\mathcal{G}$ with 51 temporal edges.

Given a temporal graph $\mathcal{G}$, the de-temporal graph of $\mathcal{G}$, denoted by $G = (V, E, \mathbb{L})$, is a graph that ignores all the timestamps associated with the temporal edges. More formally, for the de-temporal graph G of $\mathcal{G}$, we have $V = \mathcal{V}$ and $E = \{\langle v, v' \rangle | \langle v, v', t \rangle \in \mathcal{E}\}$.

Definition 1 (Subgraph pattern) A subgraph pattern $Q = (V_P, E_P, \mathbb{L}_P)$ is a de-temporal graph in which V_P and E_P are the sets of vertices and edges, respectively. Each vertex $u_p \in V_P$ (respectively $e_p \in E_P$) has a label $\mathbb{L}_P(u_p)$ (respectively $\mathbb{L}_P(e_p)$) specifying a predicate.

Definition 2 (Temporal subgraph homomorphism) Given a temporal graph $\mathcal{G}$ and a subgraph pattern Q, Q is temporally homomorphic to a subgraph of $\mathcal{G}$ if there is a mapping $f: V_P \to \mathcal{V}$ such that: (1) $\forall u \in V_P$, node u has the same label as node $f(u)$; and (2) if edge $\langle u, u' \rangle \in E_P$, then temporal edge $\langle f(u), f(u'), t \rangle \in \mathcal{E}$, and edge $\langle u, u' \rangle$ has the same label as emporal edge $\langle f(u), f(u'), t \rangle$.

We call a subgraph g of $\mathcal{G}$ as an instance of Q if Q is temporally homomorphic to g. For example, consider the subgraph pattern Q in Fig. 2b and temporal graph $\mathcal{G}$ in Fig. 2a. All the subgraphs of $\mathcal{G}$ in Fig. 2c are instances of Q since Q is temporally homomorphic to all of them. Furthermore, each instance g of Q has a duration, denoted by $\Delta(g)$, which is the difference between the largest and smallest timestamps in g. We only consider the instances formed within short duration since the longer an instance lasts, the less interesting it becomes[16]. We call this instance a τ-instance of Q.

Definition 3 (τ-instance) Given a temporal graph $\mathcal{G}$, a subgraph pattern Q, and a duration threshold τ, a subgraph g of $\mathcal{G}$ is a τ-instance of Q if: (1) g is an instance of Q, and (2) $\Delta(g) \leqslant \tau$.

Example 2 Figure 2c lists three 1-instances ($\tau = 1$) of the antecedent Q_2 (Fig. 2b) in TGAR R_2 (Fig. 1b) of a temporal graph (Fig. 2a). The subgraphs in gray are invalid instances of Q_2 because of the violation of the duration constraint.

Definition 4 (Temporal graph association rule) A TGAR $R(u_x, u_y, \Delta t)$ is defined as $Q(u_x, u_y) \overset{\Delta t}{\Rightarrow} q(u_x, u_y)$, where $Q(u_x, u_y)$ is a graph pattern in which

† Without loss of generality, we interpret each timestamp as an integer, e.g., the UNIX timestamps are nonnegative integers.

u_x and u_y are two designated nodes, and $q(u_x, u_y)$ is an edge labeled q from u_x to u_y on which the same search conditions as in Q are imposed; and Δt is a constant that specifies a time interval.

We refer to Q and q as the antecedent and consequent of R, respectively. The TGAR states that for all vertices v_x and v_y in a temporal graph, if there exists a τ-instance $g(v_x, v_y)$ of $Q(u_x, u_y)$ with the largest timestamp t, then the consequent $q(u_x, u_y)$ will likely hold, within a time window $[t, t + \Delta t]$.

Example 3 Recall the first association rule for cyberattack prevention described in Application 1. This rule can be described as TGAR $R_1(u_3, u_2, \Delta t = 2\text{ min})$: $Q_1(u_3, u_2) \overset{2\text{ min}}{\Rightarrow} q_1(u_3, u_2)$, where its antecedent is the pattern Q_1, and its consequent is $q_1(u_3, u_2)$. Similarly, the temporal association for POI recommendation in Application 2 can be expressed as TGAR $R_2(u_1, u_3, \Delta t = 2\,\text{h})$: $Q_2(u_1, u_3) \overset{2\,\text{h}}{\Rightarrow} q_2(u_1, u_3)$ with Q_2 and edge $\langle u_1, u_3\rangle$ in Fig. 1.

Frequently used notations are summarized in Table 1.

3 Diversified TGARs Discovery Problem

We want to detect temporal event associations by explicitly using TGARs. To this end, we introduce support and correlation measures for TGARs (Section 3.1), followed by the formulation of the diversified TGARs discovery problem (Section 3.2).

3.1 Interestingness measures

Having an anti-monotonic support measure can allow the development of methods that effectively prune the search space; without an anti-monotonic measure, an exhaustive search is unavoidable[17]. To this end, we propose a new support metric based on the maximal occurrence supported by the designated nodes. We first introduce some important notions and then define the support, confidence, and weighted-diversity-score of TGARs.

Table 1 Notation and description.

Notation	Description
$\mathcal{G}$ / G	Temporal graph / De-temporal graph
T	Timestamps set in $\mathcal{G}$
$\mathcal{V}$ / V	Vertices set in $\mathcal{G}$ / Vertices set in G
$\mathcal{E}$ / E	Temporal edges set in $\mathcal{G}$ / Edges set in G
Q / g	Subgraph pattern / An instance of Q
τ	Duration threshold of g
Δt	Time interval threshold between Q and q
$(v_x, v_y)_t$	(v_x, v_y) exists in a $g(v_x, v_y)$ with largest timestamp t
k	Number of returned TGARs
b	Size bound of Q
δ, σ	Support and confidence thresholds of TGARs
$Adj(e_i)$	Adjacency list corresponding to $e_i \in Q$
time($\langle v, v'\rangle$)	Timestamps set of data edge $\langle v, v'\rangle$ during τ
cand(u)	Vertices set of $\mathcal{G}$ to which u can be mapped

Rule occurrence. Given a TGAR $R(u_x, u_y, \Delta t)$: $Q(u_x, u_y) \overset{\Delta t}{\Rightarrow} q(u_x, u_y)$ and a vertex pair (v_x, v_y) in $\mathcal{G}$, if: (1) $g_{t'}(v_x, v_y)$ is a τ-instance of $Q(u_x, u_y)$ with the largest timestamp t', (2) an edge $\langle v_x, v_y\rangle$ in $\mathcal{G}$ matches $q(u_x, u_y)$ at timestamp t, and (3) $0 \leqslant t - t' \leqslant \Delta t$, then the time window $[t', t]$ is an occurrence of $R(u_x, u_y, \Delta t)$ supported by $\langle v_x, v_y\rangle$ that matches $q(u_x, u_y)$ at timestamp t. For ease of exposition, we use the notation $(v_x, v_y)_t$ to describe that the vertex pair (v_x, v_y) exists in a τ-instance $g(v_x, v_y)$ of $Q(u_x, u_y)$ with the largest timestamp t.

Note that $(v_x, v_y)_t$ may exist in multiple τ-instances of $Q(u_x, u_y)$. We use a counter $TN^t_{(v_x,v_y)}$ to count the number of τ-instances of $Q(u_x, u_y)$ that contains $(v_x, v_y)_t$. Moreover, a time window may contain multiple occurrences of a TGAR. We want to further find "maximal" ones that suffice to support TGARs.

Maximal occurrence. Let $\mathcal{T}_q(v_x, v_y, t)$ be the set of the time window that includes all occurrences of a TGAR in $\mathcal{G}$ supported by the edge $\langle v_x, v_y\rangle$ at timestamp t. A maximal occurrence of TGAR $R(u_x, u_y, \Delta t)$ supported by an edge $\langle v_x, v_y\rangle$ at timestamp t is a time window $[t', t] \in \mathcal{T}_q(v_x, v_y, t)$ such that: (1) $TN^{t'}_{(v_x,v_y)} > 0$, and (2) there exists no other time window $[t'', t] \in \mathcal{T}_q(v_x, v_y, t)$ such that $TN^{t''}_{(v_x,v_y)} > 0$ and $t'' < t'$. Note that we set $TN^{t'}_{(v_x,v_y)} \leftarrow TN^{t'}_{(v_x,v_y)} - 1$ if $(v_x, v_y)_t$ is used to generate the maximal occurrence.

Example 4 Consider the TGAR $R_2(u_1, u_3, \Delta t = 2\,\text{h})$ in Fig. 1b and the temporal graph in Fig. 2a. The focus occurrences of $Q_2(u_1, u_3)$ and $q_2(u_1, u_3)$ are shown in Fig. 4. Now, we look for the maximal occurrences supported by the edge $\langle v_8, v_7\rangle$. (1) We can easily obtain $TN^2_{(v_8,v_7)} = 2$. In addition, $[2, 2] \in \mathcal{T}_q(v_8, v_7, 2)$ is a maximal occurrence of R_2 supported by $\langle v_8, v_7\rangle$ by the definition of maximal occurrence. Therefore, $TN^2_{(v_8,v_7)} \leftarrow TN^2_{(v_8,v_7)} - 1$ since $(v_8, v_7)_2$ is used to generate the maximal occurrence. Time window $[2, 3] \in \mathcal{T}_q(v_8, v_7, 3)$ and $[3, 3] \in \mathcal{T}_q(v_8, v_7, 3)$ are occurrences of R_2 supported by edge $\langle v_8, v_7\rangle$ that matches $q(v_8, v_7)$ at timestamp 3. $[2, 3] \in \mathcal{T}_q(v_8, v_7, 3)$ is a maximal occurrence of R_2 supported by edge $\langle v_8, v_7\rangle$ by the definition of maximal occurrence rather

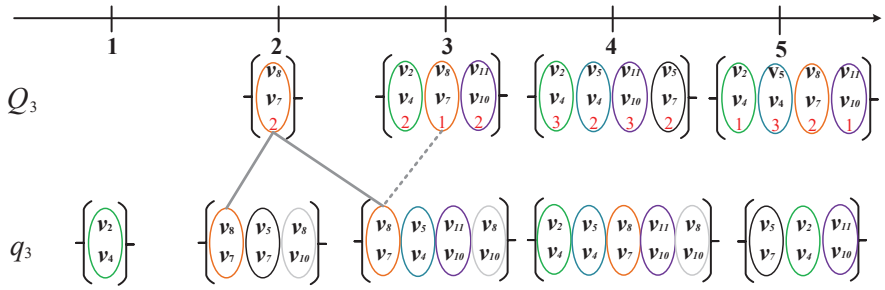


Fig. 4 Counting maximal occurrences.

than $[3, 3]$ because of $TN^2_{(v_8,v_7)} = 1$ at this moment and $2 < 3$. For $\mathcal{T}_q(v_8, v_7, 4)$, $[2, 4]$ is not the maximal occurrence of R_2 because $TN^2_{(v_8,v_7)} = 0$. Therefore, $[3, 4] \in \mathcal{T}_q(v_8, v_7, 4)$ is a maximal occurrence of R_2. In total, three maximal occurrences of R_2 are supported by the edge $\langle v_8, v_7\rangle$ at different timestamps. (2) If we consider the support metric with minimal occurrence in Ref. [14], $(v_8, v_7)_{t_1}$ is used to generate the minimal occurrence of R_2 with the matched consequent $q(v_8, v_7)$ at timestamp t_2 since there exists no time window $[t'_1, t'_2] \subset [t_1, t_2]$ (i.e., $t_1 < t'_1 \leqslant t'_2 \leqslant t_2$ or $t_1 \leqslant t'_1 \leqslant t'_2 < t_2$) such that $[t'_1, t'_2]$ is an occurrence of R_2 supported by edge $\langle v_8, v_7\rangle$. Therefore, the minimal occurrences of R_2 supported by edge $\langle v_8, v_7\rangle$ at timestamp 2 and timestamp 3 are $[2, 2]$ and $[3, 3]$, respectively.

Note that if there exist two identical occurrences of R_1 supported by different vertex pairs, we consider such cases as different occurrences rather than a single one. Indeed, it suggests that both vertices pairs are influenced by the temporal association at the same time window, which should be considered "stronger" evidence that R_1 holds.

Support. Having an anti-monotonic support measure can allow the development of methods that effectively prune the search space; without an anti-monotonic measure, an exhaustive search is unavoidable[17]. On the basis of the maximal occurrences, we define the support of TGAR R in temporal graph $\mathcal{G}$, denoted by $\text{supp}(R)$, as follows:

$$\text{supp}(R) = \frac{|m(R, u_x, u_y)|}{|\text{cand}(u_x, u_y)| \times |T|} \tag{1}$$

where $m(R, u_x, u_y)$ refers to the set of temporal edges that match $q(u_x, u_y)$ in the temporal graph $\mathcal{G}$ at each timestamp, each of which can contribute to a maximal occurrence of R, and $|\text{cand}(u_x, u_y)|$ is the number of vertex pairs $(v_x, v_y)_t$ such that v_x (respectively v_y) has the same label as u_x (respectively u_y). We normalize its size with $|\text{cand}(u_x, u_y)|$ and $|T|$, as up to $|\text{cand}(u_x, u_y)|$ vertex pairs support a maximal occurrence of R, and each vertex pair supports up to $|T|$ maximal occurrences.

We show that $\text{supp}(R)$ is anti-monotonic.

Lemma 1 For any temporal graph $\mathcal{G}$ and any TGARs R_1: $Q(u_x, u_y) \overset{\Delta t}{\Rightarrow} q(u_x, u_y)$ and R_2: $Q'(u_x, u_y) \overset{\Delta t}{\Rightarrow} q(u_x, u_y)$, $\text{supp}(R_1) \geqslant \text{supp}(R_2)$ if $R_1 \subseteq R_2$.

Here, we say $R_1 \subseteq R_2$ if $Q(u_x, u_y) \subseteq Q'(u_x, u_y)$. The set of vertex pairs $\{(v_x, v_y)_t\}$ in the temporal graph $\mathcal{G}$ is denoted by $m(Q, u_x, u_y)$. We prove Lemma 1 as follows.

Proof Consider TGARs R_1: $Q(u_x, u_y) \overset{\Delta t}{\Rightarrow} q(u_x, u_y)$ and R_2: $Q'(u_x, u_y) \overset{\Delta t}{\Rightarrow} q(u_x, u_y)$ such that $R_1 \subseteq R_2$. By definition of TGAR support, it suffices to show that $m(Q', u_x, u_y) \subseteq m(Q, u_x, u_y)$. As $Q \subseteq Q'$, for any vertex pair $(v_x, v_y) \in m(Q', u_x, u_y)$, (v_x, v_y) also exists in $m(Q, u_x, u_y)$. Otherwise, either $(v_x, v_y) \notin m(Q, u_x, u_y)$ or Q contains at least an edge that is not in Q'. Both lead to a contradiction. Hence, $m(Q', u_x, u_y) \in m(Q, u_x, u_y)$. The anti-monotonicity thus follows. ■

Example 5 Consider TGAR R_2 in Fig. 1b and the temporal graph in Fig. 2a. The focus occurrences of $Q_2(u_1, u_3)$ and $q_2(u_1, u_3)$ are shown in Fig. 4. Since a total of 10 maximal occurrences are contributed by vertex pairs $(v_8, v_7), (v_2, v_4), (v_{11}, v_{10}), (v_5, v_4), (v_5, v_7)$, respectively, $\text{supp}(R_2) = \frac{3+2+3+1+1}{6\times 5} = 0.33$. However, if we consider the support metric with minimal occurrence[14], a total of nine minimal occurrences are contributed, $\text{supp}(R_2) = \frac{9}{6\times 5} = 0.3 < 0.33$. If we adopt the concept of minimal occurrence in Ref. [14], it is a type of overkill that we will underestimate the actual support. Therefore, our proposed support metric is an upper bound for the alternative metric is proposed by Ref. [14].

Remark. The most intuitive support measure for TGAR R is to count the minimal occurrences supported

by the designated nodes of R in the time window[14]. However, this setting will underestimate the actual support since (1) for their antecedents, as pattern graphs, only their matching instances on each timestamp are considered, and the duration characteristics are neglected, (2) the setting[14] only considers each match once of the designated nodes at each timestamp, and (3) they use the number of minimal occurrences supported by the matches of the common vertex as a support metric, and some possible matching pairs will be ignored.

Confidence. The confidence of TGAR R in $\mathcal{G}$, denoted as conf(R), measures how likely a temporal edge that matches $q(u_x, u_y)$ supports a maximal occurrence of R and is defined as

$$\text{conf}(R) = \frac{|m(R, u_x, u_y)|}{|m(q, u_x, u_y)|} \tag{2}$$

where $m(q, u_x, u_y)$ is the set of temporal edges that match $q(u_x, u_y)$ in the temporal graph $\mathcal{G}$ at each timestamp.

Example 6 Consider TGAR R_2 in Fig. 1b and the temporal graph in Fig. 2a. We can verify that $m(R_2, u_1, u_3) = 10$. Hence, $\text{conf}(R_2) = \frac{10}{16} = 0.625$.

Diversification. We are interested in TGARs for a particular consequent $q(u_x, u_y)$. However, this often generates an excessive number of rules, which often pertain to the same or similar people[12, 13]. To find interesting and diverse TGARs, given a set L_k of k TGARs that pertain to the same consequent $q(u_x, u_y)$, we define the diversification of L_k, denoted by dive(L_k), as follows:

$$\text{dive}(L_k) = (1-\lambda) \sum_{R_i \in L_k} \text{conf}(R_i) + \frac{2\lambda}{k-1} \cdot \sum_{R_i, R_j \in L_k, i<j} \text{diff}(R_i, R_j) \tag{3}$$

where $\text{diff}(R_i, R_j)$ is defined as

$$\text{diff}(R_i, R_j) = 1 - \frac{|m(R_i, u_x, u_y) \cap m(R_j, u_x, u_y)|}{|m(R_i, u_x, u_y) \cup m(R_j, u_x, u_y)|} \tag{4}$$

Here, $\text{diff}(R_i, R_j)$ is used to measure the difference between two TGARs R_i and R_j in terms of the Jaccard distance of their match set. dive(L_k) is known as *max-sum diversification*. It aims to strike a balance between interestingness (measured by the revised Bayes Factor) and diversity (measured by the distance diff(,)) with a parameter λ controlled by users. We normalize the diversity metric with $\frac{2\lambda}{k-1}$ since there are $\frac{k(k-1)}{2}$ numbers for the difference sum, while there are only k numbers for the confidence sum.

Example 7 Consider the antecedents Q_3, Q_4, and Q_5 in Fig. 3, all pertaining to $q_2(u_1, u_3)$ (Fig. 1b). We can obtain R_3, R_4, and R_5 based on Q_3, Q_4, and Q_5, respectively. Let $\lambda = 0.5$, and $L_k = \{R_2, R_3, R_4, R_5\}$. We have $\text{dive}(L_k) = 2.39$.

3.2 Diversified mining problem

On the basis of the objective function dive(L_k), the diversified top-k TGARs mining problem is stated as follows.

- **Input**: A temporal graph $\mathcal{G}$, a predicate $q(u_x, u_y)$, a time interval Δt, support bound δ, confidence bound σ, size bound b, and parameters τ, λ, and k;
- **Output**: A set L_k of k nontrivial TGARs pertaining to $q(u_x, u_y)$ and Δt such that (1) dive(L_k) is maximized; and (2) for each TGAR $R \in L_k$, $\text{supp}(R) \geqslant \delta$, $|Q| \leqslant b$, and $\text{conf}(R) \geqslant \sigma$. Here, $|Q|$ is the number of edges in the antecedent of R.

The diversified mining problem is a bi-criteria optimization problem to discover TGARs for a particular consequent $q(u_x, u_y)$ with high support, bounded size, balanced interestingness, and diversity. In practice, users can freely specify the $q(u_x, u_y)$ of interests, while proper parameters can be estimated from query logs or recommended by domain experts.

Example 8 We continue with Example 7. Let $\lambda = 0.5$ and $k = 2$. The diversified top-2 TGARs in the set $\{R_2, R_3, R_4, R_5\}$ are $\{R_4, R_5\}$ with $\text{dive}(R_4, R_5) = 1.375$.

Hardness of problem. As shown in Ref. [15], deciding whether there exists a set L_k of k TGARs with $\text{dive}(L_k) \geqslant B$ for a given bound B is NP-hard. We can see that the decision problem is a special case of our problem. As a consequence, our problem is at least as hard, hence being NP-hard.

4 Diversified TGARs Discovery Algorithms

In this section, we first present a baseline algorithm for mining diversified TGARs, then we develop an advanced algorithm and propose some optimization strategies.

4.1 Baseline solution

We follow the explore-decide framework to establish a baseline for diversified TGARs discovery in a temporal graph (Algorithm 1). In particular, this framework first identifies a set of different single-edge patterns in the de-temporal graph G (Line 2). Then, it explores a candidate subgraph pattern space in a pattern generation tree, denoted by PGT, to calculate all possible TGARs (Lines 3–15). Finally, it returns a set L_k of k TGARs

Algorithm 1 findAllTGARs

Input: $\mathcal{G}$ is the temporal graph; $q(u_x, u_y)$ is the consequent; Δt is the time interval; $\sigma, b, \delta, \tau, k$ are the parameters.
Output: the set of diversified top-k TGARs in $\mathcal{G}$.
1: $r \leftarrow 0$; *candidateSet* $\leftarrow q(u_x, u_y)$; *ResultSet* $\leftarrow \varnothing$;
2: Let E be the set of all different single-edge patterns in graph G;
3: **while** $r \leqslant b$ **do**
4: $r \leftarrow r + 1$; $\mathcal{M} \leftarrow \varnothing$;
5: **for** pattern $P_R \in$ *candidateSet* **do**
6: **for** edge $e \in E$ **do**
7: **if** e can be used to extend P **then**
8: Let ext be the extension of P with e;
9: **if** ext is not already generated **then**
10: $Q \leftarrow ext/q$; Generate rule $R: Q \overset{\Delta t}{\Rightarrow} q$;
11: **if** supp$(R) \geqslant \delta$ **then**
12: $\mathcal{M} \leftarrow \mathcal{M} \cup ext$;
13: **if** conf$(R) \geqslant \sigma$ **and** Connnect$(Q) =$ **true then**
14: *ResultSet* $\leftarrow$ *ResultSet* $\cup R$;
15: *candidateSet* $\leftarrow \mathcal{M}$;
16: $L_k \leftarrow$ computeDiversity(*ResultSet*);
17: **return** L_k

such that dive(L_k) is maximized (Lines 16 and 17).

In the TGARs generation process, for each pattern P_R in the *candidateSet*, findAllTGARs tries to extend it with an edge of E (Lines 5–8). Here, P_R is the pattern that consists of $Q(u_x, u_y)$ and $q(u_x, u_y)$. All applicable extensions that have not been previously considered will be verified regarding whether they can participate in TGARs (Lines 9–14). To exclude already generated extensions (Line 9), the existing approach gSpan[18] imposes a lexicographic order among the patterns. findAllTGARs also employs this strategy for elegant enumeration of candidate patterns. Then, findAllTGARs generates rule R based on the extension ext and checks whether R can satisfy the support threshold; if so, findAllTGARs adds ext into $\mathcal{M}$ since according to the antimonotone property, the rules generated by its extensions can still be TGARs (Lines 10–12). Furthermore, findAllTGARs checks whether R satisfies the confidence threshold and Q is a connected graph; if so, findAllTGARs adds R into *ResultSet*. Finally, findAllTGARs calculates the set of diversified top-k TGARs via result diversification[15] (Lines 16 and 17).

Complexity. The number of all antecedents with size bound b is denoted by $N(b)$. For each antecedent Q, findTopkTGARs takes a time of $O(|T|(|\mathcal{V}|^{|V(Q)|}))$ to compute $m(Q, u_x, u_y)$, where $V(Q)$ is the set of vertices in Q. Thus, findTopkTGARs takes $O(|T|N(b)(|\mathcal{V}|^{|V(Q)|}))$ to compute $m(Q, u_x, u_y)$ for each antecedent Q. For each generated TGAR R, a time of $Q(T)$ is taken to identify maximal occurrences of R. Moreover, algorithm findTopkTGARs takes in a total space of $O(N(b)|\text{cand}(u_x, u_y)| \times |T|)$. Indeed, (1) it maintains up to $|\text{cand}(u_x, u_y)|$ focus matches with associated timestamps up to $|T|$ for each antecedent, and (2) there exists up to $N(b)$ antecedents in the search tree.

Why costly? Algorithm findAllTGARs is not scalable enough to handle large temporal graphs because of the following drawback:

• **Drawback:** Large computational cost for diversified TGARs. FindAllTGARs will generate an excessive number of TGARs corresponding to a consequent $q(u_x, u_y)$. For all TGARs R generated, it has to compute supp(R), conf(R), and their pairwise distances and pick a top-k set based on dive(); the latter is an intractable process itself.

One can do it more efficiently, with accuracy guaranteed.

4.2 New approach

In this paper, we devise a new algorithm for diversified top-k TGARs search, denoted by findTopkTGARs, which can overcome the challenges introduced in Section 4.1. In the new algorithm, we modify the TGARs enumeration algorithm (see Lines 3–15 of Algorithm 1) to integrate diversified top-k TGARs search into the process of TGARs enumeration. Specifically, during the TGARs enumeration process, we maintain a list L_k of diversified top-k TGARs that maximize the diversification score and update the k candidates when new TGARs are reported. The new algorithm has the following three advantages.

• **High pruning power.** By integrating diversified top-k TGARs search into the process of TGARs enumeration, we can develop more pruning strategies to filter rules that have low support and cannot make top-k TGARs as early as possible in each round of the while loop in Algorithm 1.

• **High efficiency.** Our algorithm incrementally computes top-k diversified TGARs, rather than recomputing the diversification function dive() starting from scratch.

• **Guaranteed result quality.** Our algorithm can achieve a guaranteed approximation ratio of 0.5 and can be much better in practice, as verified in the experiments.

In this section, we introduce our new algorithm. In Section 4.3, we will focus on the optimization strategy

to compute the τ-instance of the antecedent in each rule.

Algorithm findTopkTGARs. Our new algorithm findTopkTGARs is shown in Algorithm 2. In each round r, it first computes the newly generated TGARs and uses a set $\mathcal{C}$ to keep track of all generated TGARs (Lines 1–7). Next, it uses a max priority queue $\mathcal{Q}$ of size $\left\lceil \frac{k}{2} \right\rceil$ to store the possible diversified top-k TGARs. Each element in $\mathcal{Q}$ is a pair of TGARs, and all TGARs in $\mathcal{Q}$ are pairwise disjoint. If $|\mathcal{Q}| < \left\lceil \frac{k}{2} \right\rceil$, findTopkTGARs iteratively selects two distinct TGARs R and R' from $ResultSet$ that maximize a revised diversification function:

$$\text{dive}^*(R, R') = \frac{1-\lambda}{k-1}(\text{conf}(R) + \text{conf}(R')) + \frac{2\lambda}{k-1}\text{diff}(R, R') \quad (5)$$

and inserts (R, R') into $\mathcal{Q}$, until $\mathcal{Q} = \left\lceil \frac{k}{2} \right\rceil$ (Lines 8 and 9). To maintain $\mathcal{Q}$, findTopkTGARs calls algorithm findDiveTGARs to incrementally compute and add a new pair $(R, R') \in ResultSet \times \mathcal{C}$ that maximizes $\text{dive}^*(R, R')$ to $\mathcal{Q}$ (Lines 10–13). This step ensures that a pair (R_1, R_2) with minimum $\text{dive}^*(R_1, R_2)$ will be replaced by (R, R') if $\text{dive}^*(R_1, R_2) < \text{dive}^*(R, R')$ (Lines 3 and 4 in Algorithm 3 findDiveTGARs).

Note that (1) not all TGARs in $\mathcal{C}$ can make diversified top-k TGARs, and (2) extending a TGAR R in $ResultSet$ may not contribute to top-k TGARs. To this end, findTopkTGARs will filter non-promising TGARs from $\mathcal{C}$ before calling algorithm findDiveTGARs (Line 10) and filter the pattern P_R generated by R from $\mathcal{M}$ if extending R cannot make top-k TGARs (Lines 14 and 15). The idea is to test whether an upper bound of marginal benefit for any TGAR pair can improve the minimum dive*-value of $\mathcal{Q}$ (denoted by dive^*_{min}), which is described as follows.

Lemma 2 Let $\text{conf}_{max}(ResultSet)$ be the maximum $\text{conf}(R')$ for $R' \in ResultSet$. A TGAR $R \in \mathcal{C}$ cannot contribute to L_k if $\frac{1-\lambda}{k-1}(\text{conf}(R) + \text{conf}_{max}(ResultSet)) + \frac{2\lambda}{k-1} \leqslant \text{dive}^*_{min}$.

Proof For each $R \in \mathcal{C}$, $\frac{1-\lambda}{k-1}(\text{conf}(R) + \text{conf}_{max}(ResultSet)) + \frac{2\lambda}{k-1}$ is an upper bound for its maximum possible increment to the dive*-value of $\mathcal{Q}$. ■

Lemma 3 Let $\text{conf}_{max}(\mathcal{C})$ be the maximum $\text{conf}(R')$ for each $R' \in \mathcal{C}$. Extending a TGAR $R \in ResultSet$ does not contribute to L_k if $\frac{1-\lambda}{k-1}(\text{conf}(R) + \text{conf}_{max}(\mathcal{C})) + \frac{2\lambda}{k-1} \leqslant \text{dive}^*_{min}$.

Proof Intuitively, $\text{conf}(R)$ is anti-monotonic with any R' expanded from R. As a result, $\text{conf}(R)$ is an upper bound of the confidence for all the possible TGARs grown from R. $\text{conf}_{max}(\mathcal{C})$ is monotonically decreasing with the increase in rounds. Hence, we can safely terminate the expansion of R if it does not satisfy the constraint of Lemma 3. ■

Finally, after the while loop, findTopkTGARs inserts R and R' into L_k for each TGAR pair $(R, R') \in$

Algorithm 2 findTopkTGARs

Input: $\mathcal{G}$ is the temporal graph; $q(u_x, u_y)$ is the consequent; Δt is the time interval; $\sigma, b, \delta, \tau, k$ are the parameters.
Output: the set of diversified top-k TGARs in $\mathcal{G}$.

1: $r \leftarrow 0$; $candidateSet \leftarrow q(u_x, u_y)$; $\mathcal{C} \leftarrow \varnothing$; $L_k \leftarrow \varnothing$;
2: Let E be the set of all different single-edge patterns in graph G;
3: Let $\mathcal{Q}$ be a max priority queue of maximal size $\left\lceil \frac{k}{2} \right\rceil$, initially empty;
4: **while** $r \leqslant b$ **do**
5: $ResultSet \leftarrow \varnothing$;
6: Same as Lines 4–14 of Algorithm 1;
7: $\mathcal{C} \leftarrow \mathcal{C} \cup ResultSet$;
8: **if** $|\mathcal{Q}| < \left\lceil \frac{k}{2} \right\rceil$ **then**
9: Improves $\mathcal{Q}$ by incorporating pairs of TGARs from *ResultSet*
10: $\mathcal{C} \leftarrow \text{pruneTGARs}(\mathcal{Q}, \mathcal{C})$;
11: **for** Rule $R \in ResultSet$ **do**
12: **if** $R \notin \mathcal{Q}$ **then**
13: $\mathcal{Q} \leftarrow \text{findDiveTGARs}(L_k, R, \mathcal{C})$;
14: **if** $\frac{1-\lambda}{k-1}(\text{conf}(R) + \text{conf}_{max}(\mathcal{C})) + \frac{2\lambda}{k-1} \leqslant \text{dive}^*_{min}$ **then**
15: $\mathcal{M} \leftarrow \mathcal{M}/P_R$;
16: $candidateSet \leftarrow \mathcal{M}$;
17: Insert each R in $\mathcal{Q}$ into L_k;
18: **if** $k\%2 \neq 0$ **then**
19: Remove R_{min} from L_k;
20: **return** L_k;

Algorithm 3 findDiveTGARs

Input: L_k is the set of diversified top-k TGARs; Rule $R \in ResultSet$ and $R \notin \mathcal{Q}$; $\mathcal{C}$ is the set that tracks all generated TGARs.
Output: the updated set L_k of diversified top-k TGARs in $\mathcal{G}$.

1: **for** pattern $R' \in \mathcal{C}$ **do**
2: **if** $R' \notin \mathcal{Q}$ **then**
3: **if** (R_1, R_2) in $\mathcal{Q}$ has the minimum dive* value **and** $\text{dive}^*(R_1, R_2) < \text{dive}^*(R, R')$ **then**
4: Replace (R_1, R_2) by (R, R');
5: **return** L_k;

$\mathcal{Q}$ and returns L_k as the set of diversified top-k TGARs (Lines 17–20). In particular, if k is an odd number, findTopkTGARs removes R_i with the minimum $\sum_{R_j \in L_k, j \neq i} \text{dive}^*(R_i, R_j)$ from L_k (Lines 18 and 19).

The following Lemma shows the quality of the result for the diversified top-k TGARs computed using Algorithm 2.

Lemma 4 Given a temporal graph $\mathcal{G}$, a consequent pattern $q(u_x, u_y)$, and an integer k, suppose L_k^* is the optimal diversified top-k TGARs, and L_k is the diversified top-k cliques returned by Algorithm 2. We have $\dfrac{\text{dive}(L_k)}{\text{dive}(L_k^*)} \geqslant 0.5$.

Proof We use induction on k to prove Lemma 4. (1) Lemma 4 is trivially true when $k = 2$. (2) Suppose that $k > 2$. Algorithm findDiveTGARs is essentially the max-sum dispersion problem in Ref. [15], which maximizes the sum of pairwise distances for a set of data points. Since the approximation-preserving reduction in Ref. [15] has approximation ratio 2, Algorithm 2 produces a 2-approximation of the diversified top-k TGARs in $\mathcal{G}$. ■

Algorithm analysis. Obviously, Algorithm 2 can achieve a tighter bound than Algorithm 1 since $\text{dive}^*_{min} \gg \delta$. This result significantly reduces the search space since it can filter rules that cannot make top-k TGARs in the early stage in each round. Moreover, the cost of reduction takes in a total time of $O(b \times |\mathcal{C}|)$, where in each round, a linear scan of $ResultSet$ and $\mathcal{C}$ is required to identify redundant TGARs.

To realize the algorithm framework findTopkTGARs in Algorithm 2, we still need to solve the following issue:

• **Repeated subgraph matching.** In the exploring process, whenever an antecedent Q is produced, we need to re-execute temporal subgraph homomorphism calculation for Q over the overall temporal graph $\mathcal{G}$ to calculate $m(Q, u_x, u_y)$, which can be detrimental.

4.3 Optimization strategy

To avoid repeated calculations, we construct an auxiliary data structure called EIndex to represent partial solutions in a compact form. On the basis of the auxiliary data structure, whenever an extension Q' of an antecedent Q is produced, we can join the new, inserted edge with the materialized results stored in Q to update the auxiliary data structure and then return $m(Q', u_x, u_y)$ according to the updated intermediate results.

EIndex structure. Given an antecedent Q that is extended by $q(u_x, u_y)$, EIndex is defined as follows: (1) Each vertex u of EIndex has a candidate set, denoted by cand(u), that stores all vertices of $\mathcal{G}$ to which u can be mapped. (2) There is an edge between $v \in \text{cand}(u)$ and $v' \in \text{cand}(u')$ if and only if $\langle v, v' \rangle$ exists in G. (3) Each edge records its corresponding timestamps during T, each of which may participate in any τ-instances of Q. For example, the level-2 part of PGT illustrated in Fig. 5 shows the EIndex constructed for the antecedent Q_2 in Fig. 2b over the temporal graph in Fig. 2a. Note that EIndex will be updated in the pattern extension process.

Next, we propose how to initialize EIndex and maintain EIndex efficiently in the exploring process and how to verify each candidate's temporal pattern.

Algorithm calculateVertexPair. Algorithm calculateVertexPair initializes EIndex as two candidate sets cand(u_x) and cand(u_y). Then, for each Q_i in level-i ($1 \leqslant i \leqslant b$), i.e., $|Q_i| = i$, calculateVertexPair relies on an incremental maintain strategy to efficiently maintain EIndex for the extended edge $e_i = \langle u, u' \rangle$ in the current level. First, we obtain the adjacency list corresponding to e_i, i.e., $Adj(e_i)$, in EIndex as follows.

• **Case 1.** If u' is the newly introduced vertex, we check, for each vertex v in $cand(u)$, whether there is an edge $\langle v, v' \rangle$ matching $\langle u, u' \rangle$ and existing in G; if so, we add v' into cand(u') and record the corresponding timestamps of $\langle v, v' \rangle$.

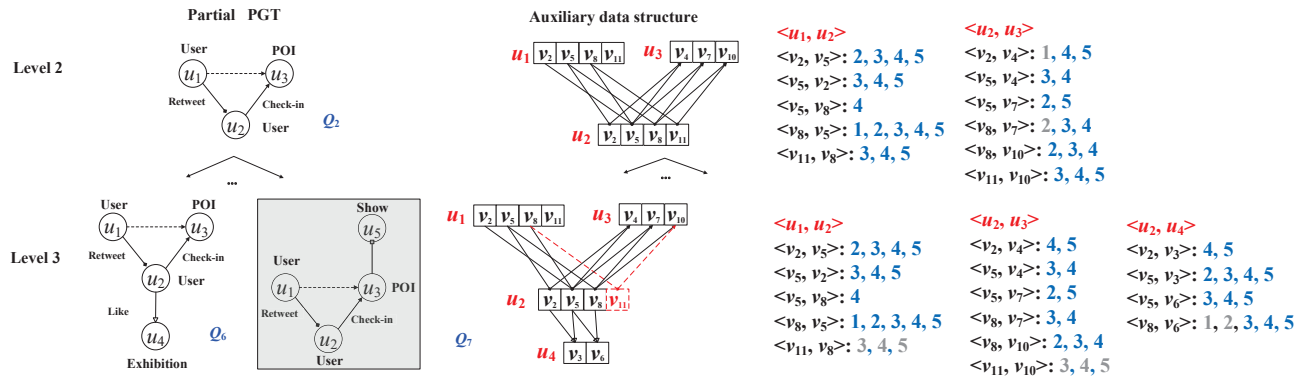


Fig. 5 Sample pattern generation tree and EIndex.

• **Case 2.** If u and u' already exist in Q_i, we use the same manner to construct the adjacency list $Adj(e_i)$ corresponding to cand(u') and cand(u) and then record the timestamps of each edge in $Adj(e_i)$.

Since the support of a rule R is calculated by the τ-instance of Q, it is safe to prune the invalid timestamps of each edge $\langle v, v'\rangle$ in $Adj(e_i)$ because of the duration constraint. Note that there is an extension order EO for all edges of Q_i according to the subgraph pattern generation tree PGT. Let $e_i.N$ denote the set of edges before e_i in EO that are adjacent to e_i, and time($\langle v, v'\rangle$) denote the timestamps of $\langle v, v'\rangle$ in $\mathcal{G}$. Intuitively, for each edge $\langle v, v'\rangle$ in $Adj(e_i)$, a timestamp t is in time($\langle v, v'\rangle$) only if for each edge $e' \in e_i.N$, there is a candidate edge in $Adj(e')$ that is joinable with $\langle v, v'\rangle$ and has a timestamp t' such that $|t' - t| \leqslant \tau$. To satisfy this requirement, we maintain a counter $\mathcal{N}^t_{\langle v,v'\rangle}$ for each timestamp t in time($\langle v, v'\rangle$) to count the number of neighbors of e_i before e_i in EO that have a candidate with timestamp t' joinable with $\langle v, v'\rangle$ such that $|t' - t| \leqslant \tau$; and NE records the number of edges in $e_i.N$. time($\langle v, v'\rangle$) is the set of timestamps satisfying $\mathcal{N}^t_{\langle v,v'\rangle} = NE$. If time($\langle v, v'\rangle$) $= \varnothing$, then edge $\langle v, v'\rangle$ will be removed from $Adj(e_i)$. Furthermore, vertex v (respectively v') will be removed from cand(u) (respectively cand(u')) if v (respectively v') has no neighbor in the candidate set of one neighbor of u (respectively u'); and the corresponding adjacency list of v (respectively v') will also be deleted from EIndex. After that, we set $\mathcal{N}^t_{\langle v,v'\rangle} = 0$ for all data edges in EIndex that have a positive count.

Lemma 5 A data edge $\langle v, v'\rangle$ in $Adj(e_i)$ with timestamp t has a joinable neighbor with timestamp t' in $Adj(e')$ s.t. $|t' - t| \leqslant \tau$ for every $e' \in e_i.N$ if and only if $\mathcal{N}^t_{\langle v,v'\rangle} = NE$.

Lemma 6 Let $v.cnt$ denote the number of query neighbors of u that have a candidate v' adjacent to v and $d(u)$ denote the degree of u. A data vertex v has a neighbor in cand(u') for every neighbor u' of u if and only if $v.cnt = d(u)$.

CalculateVertexPair then prunes the adjacency lists for other pattern edges of Q_i according to the reverse order of EO, i.e., $\overline{EO}$. In detail, for each edge $e \in \overline{EO}$, calculateVertexPair applies Lemma 5; a timestamp t of a data edge $\langle v, v'\rangle \in Adj(e)$ is pruned if $\mathcal{N}^t_{\langle v,v'\rangle} \neq NE$. Next, calculateVertexPair applies Lemma 6; a data vertex $v \in$ cand(u) and its corresponding adjacency list are pruned if $v.cnt \neq d(u)$. Note that the above pruning process only considers the edges after e in EO that are adjacent to e to prune the adjacency lists. Thus, a data edge $\langle v, v'\rangle \in Adj(e)$ might not have any joinable neighbor in $Adj(e')$, where e' is the neighbor of e before e in EO. As a result, calculateVertexPair uses the extension order EO for further refining the adjacency lists of pattern edges. The expansion process terminates if $Adj(e) = \varnothing$ ($e \in Q_i$).

Example 9 Consider the sample pattern generation tree in Fig. 5. Let $\tau = 1$. When processing the antecedent Q_6 in level 3, we first obtain the adjacency list corresponding to the extension edge $\langle u_2, u_4\rangle$. Here, we have $EO = \{\langle u_1, u_2\rangle, \langle u_2, u_3\rangle, \langle u_2, u_4\rangle\}$. When processing edge $\langle v_8, v_6\rangle$ in $Adj(\langle u_2, u_4\rangle)$, we have $N^1_{\langle v_8,v_6\rangle} = 1$. Since $NE = 2$, we remove 1 from time($\langle v_8, v_6\rangle$). Similarly, we remove 2 from time($\langle v_8, v_6\rangle$). Next, we prune the adjacent lists of the pattern edges in Q_6 according to the $\overline{EO}$ and EO, respectively. By applying Lemma 5, all timestamps in time($\langle v_{11}, v_{10}\rangle$) and time($\langle v_{11}, v_8\rangle$) will be removed. By applying Lemma 6, v_{11} is removed from cand(u_2) since v_{11} has no neighbor in cand(u_4). In level 3, Q_7 will not be extended since $Adj(\langle u_3, u_5\rangle) = \varnothing$.

After updating the EIndex for each antecedent Q_i in level-i ($1 \leqslant i \leqslant b$), calculateVertexPair calculates $m(Q_i, u_x, u_y)$ to compute the corresponding measures. That is, for each vertex pair (v_x, v_y) where $v_x \in$ cand(u_x) and $v_y \in$ cand(u_y), calculateVertexPair adopts the well-known BackTracking algorithm to calculate all the τ-instances of Q_i and then obtains the largest timestamps of each τ-instance. Additionally, we use the counter $TN^t_{(v_x,v_y)}$ to count the number of the τ-instances of Q_i that contain $(v_x, v_y)_t$. To minimize the number of recursive calls for subgraph matching, we determine a good matching order based on the cost model proposed in Ref. [19].

Complexity. Let Q_i be an antecedent in the PGT tree at level-i. CalculateVertexPair first takes a time of $O(|\mathcal{E}| \times |Q_i|)$ to update the auxiliary data structure EIndex corresponding to Q_i. Let d be the maximum number of edges in EIndex connected with one vertex in cand(u_x) or cand(u_y). CalculateVertexPair takes a time of $O(|\text{cand}'(u_x, u_y)| \times d^{|Q_i|})$ to calculate $m(Q_i, u_x, u_y)$. Here, cand$'(u_x, u_y)$ represents the different vertex pairs (v_x, v_y), where $v_x \in$ cand(u_x) and $v_y \in$ cand(u_y).

5 Experiment

In this section, we report and analyze experimental results. All the algorithms were implemented in JAVA

and run on a PC with an Intel i7 3.50 GHz CPU and 32 GB memory. Every quantitative test was repeated five times, and the average was reported.

Dataset. We use three real-life temporal networks:

•Offshore‡ is a social network of offshore entities and financial activities. It records 8.39×10^5 offshore entities (e.g., countries and companies), 3.6×10^6 relationships for offshore entities (e.g., close and establish), and 433 labels containing 40 years of offshore entities and financial activities, including 12 139 active days.

•Citation§ is a citation network about paper citation relationships. It contains 4.3×10^6 entities (e.g., papers and authors), 2.17×10^7 edges (e.g., published at and citation), and 273 labels (e.g., keywords) covering 80 years of information about the publication of the paper.

•MovieLens¶ is a movie recommendation network of rating data from multiple users for multiple movies. It records 1.0×10^7 ratings for 1.0×10^4 movies by 7.1×10^4 users. In addition to users and movies, the temporal graph also includes 20 other labels that show the category of each movie (e.g., romance and drama), and it has 1413 h of interactions.

The statistics of the three datasets are summarized in Table 2.

Algorithm. We implement and compare three algorithms:

•findAllTGARs: our baseline method for mining top-k TGARs;

•findTopkTGARs: our advanced algorithm for finding top-k TGARs; and

•findTopkTGARs$^+$: findTopkTGARs equipped with the proposed algorithm calculateVertexPair.

Parameter setting. We vary four parameters in our experiments, namely k (the top-k value), b (the size bound of Q), δ (the support threshold), and σ (the confidence threshold). k is selected from 5 to 25 with a default of 15; b, 2 to 5 with a default value of 4; δ, 0.05 to 0.21 with a default value of 0.13; and σ, 0.5 to 0.8 with a default value of 0.6. We fix $\tau = 3$ (duration threshold of instances), $\Delta t = 5$ (time window) and $\lambda = 0.5$ (for diversification) for all datasets. We found that TGARs mined in real-life datasets with infrequent edge labels usually denote unrelated facts. Therefore, we use the 15 most frequent edge patterns (with vertex and edge labels) to assign the triple pattern $q(u_x, u_y)$ to grow TGARs in all datasets. Unless otherwise specified, when varying a certain parameter, the values of the other parameters are set to their default values.

Table 2 Graph datasets.

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert T\rvert$	Time scale
Offshore	839 228	3 627 186	12 000	day
MovieLens	71 532	10 157 412	1413	hour
Citation	4 334 671	21 753 011	80	year

‡ https://offshoreleaks.icij.org/pages/database.

§ https://www.aminer.org/citation.

¶ https://grouplens.org/datasets/movielens.

EXP-1: efficiency testing. In the first set of tests, we compare the efficiency of algorithm findTopkTGARs$^+$ and its competitors under the default parameter settings and then report the running time of different algorithms with varying parameters.

EXP-1.1 Figure 6 shows the running time of findTopkTGARs$^+$ and its competitors on different datasets with default parameters. Similar results can be observed under other parameter settings. From Fig. 6, we see that findTopkTGARs$^+$ is much faster than its competitors. For example, on MovieLens, findTopkTGARs$^+$ takes approximately 20 min to calculate diversified top-k TGARs for all consequents $q(u_x, u_y)$, while other algorithms cannot obtain results in 3 h. We also find that, on each dataset, the time decreases sharply from findAllTGARs to findTopkTGARs. In particular, findTopkTGARs$^+$ outperforms findTopkTGARs by up to 2.1 fold, and findTopkTGARs outperforms findAllTGARs by 5.2 fold over Citation. This is because findTopkTGARs enforces a new threshold dive^*_{min} with $\text{dive}^*_{min} \gg \delta$, which helps to filter rules that cannot make top-k TGARs in the early stage in each round. findTopkTGARs$^+$ maintains a concise auxiliary data structure that can help to avoid repeated subgraph matching. These results indicate that our proposed algorithms findTopkTGARs and calculateVertexPair can help shrink the search space dramatically as the mining process progresses.

EXP-1.2 Figure 7 shows the running time of findTopkTGARs and findTopkTGARs$^+$ with varying parameters on Offshore and Citation datasets. Note that,

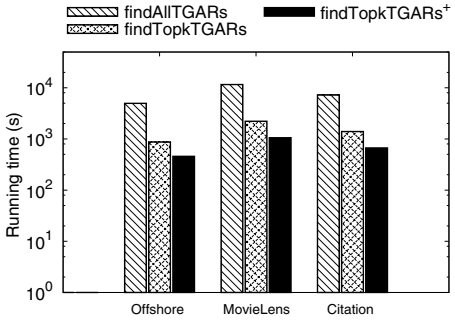
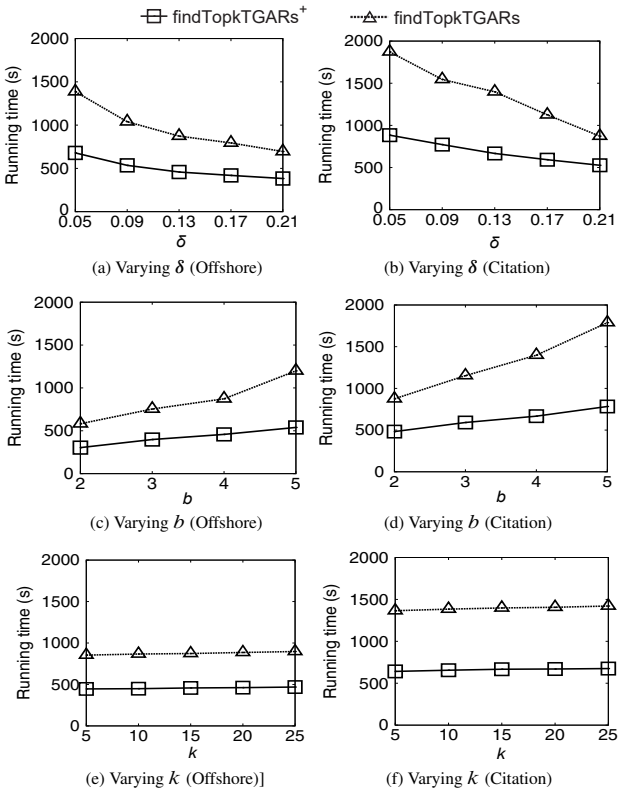


Fig. 6 Running time with default parameters.



(a) Varying δ (Offshore) (b) Varying δ (Citation)

(c) Varying b (Offshore) (d) Varying b (Citation)

(e) Varying k (Offshore)] (f) Varying k (Citation)

Fig. 7 Efficiency of varying parameters.

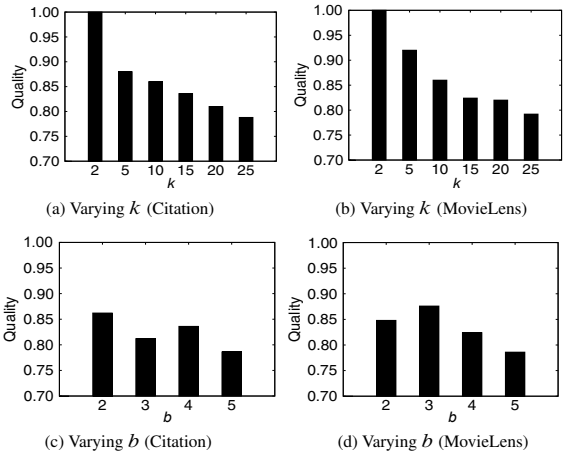
when varying a parameter, we keep other parameters at their default value. The results on the other datasets are consistent. In particular, we observe that (1) the increase in δ can decrease the running time of findTopkTGARs and findTopkTGARs$^+$; (2) the increase in b can increase the running time of findTopkTGARs and findTopkTGARs$^+$; (3) the increase in k has little impact on the running time of findTopkTGARs and findTopkTGARs$^+$; (4) the increase in σ can decrease the running time of findTopkTGARs and findTopkTGARs$^+$ (not shown). The reason could be that for a larger δ and σ, the TGARs in $ResultSet$ and $\mathcal{C}$ will be smaller, and it takes less time to verify each pair of TGARs from $ResultSet \times \mathcal{C}$. The increase in b will increase the number of the while loop and thus increase the time. Parameter k has little impact on the running time since we will keep track of (R, R') in $\mathcal{Q}$ with dive^*_{min} in each round, and incrementally update $\mathcal{Q}$ by comparing each pair of TGARs from $ResultSet \times \mathcal{C}$ with (R, R').

EXP-2: effectiveness testing. Since algorithm findTopkTGARs uses an approximate method to find the diversified top-k TGARs, there may exist inconsistency compared to the ground truth. That is, the TGARs in L_k returned by findTopkTGARs may not exist in that returned by findAllTGARs. As a result, we define the Precision to evaluate the quality of the proposed algorithm.

- **Precision:** For the set L_k calculated by findTopkTGARs, the Precision is defined as $\frac{|L_k \cap L_k^*|}{|L_k|}$, where L_k^* is the set calculated by algorithm findAllTGARs.

Figure 8 shows the values of Precision with varying parameters on the Citation and MovieLens datasets. Note that, when varying a parameter, we keep other parameters at their default value. The Precision is more than 80% on average with different parameters on both datasets, which justifies the effectiveness of the findTopkTGARs algorithm. In particular, the Precision is 100% when $k = 2$.

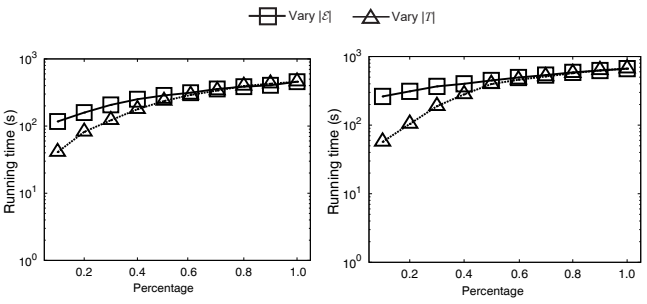
EXP-3: scalability. Figure 9 shows the scalability of algorithm findTopkTGARs$^+$ on the Offshore and Citation datasets. Similar results are observed on the other datasets. We generate ten temporal subgraphs by randomly picking 10%−100% of the temporal edges or 10%−100% of the timestamps and evaluate the running time of findTopkTGARs$^+$ on those subgraphs. As shown in Fig. 9, the running time increases smoothly



(a) Varying k (Citation) (b) Varying k (MovieLens)

(c) Varying b (Citation) (d) Varying b (MovieLens)

Fig. 8 Effectiveness of varying parameters.



(a) Percentage of $|\mathcal{E}|$ and $|T|$, Offshore (b) Percentage of $|\mathcal{E}|$ and $|T|$, Citation

Fig. 9 Scalability testings.

with the increasing number of $\mathcal{E}$ or increasing size of $|T|$, which suggests that findTopkTGARs$^+$ is scalable when handling large temporal networks.

EXP-4: case study. To demonstrate the effectiveness of algorithm findTopkTGARs$^+$ and its application, we perform a case study, as shown in Fig. 10, from Offshore.

Figure 10 shows a TGAR $R(u_2, u_3, \Delta t = 5 \text{ years})$: $Q_8(u_2, u_3) \overset{5 \text{ years}}{\Rightarrow} q_8(u_2, u_3)$ with parameters ($b = 4$, $\tau = 3$, $\delta = 0.13$, $\sigma = 0.6$, and $\Delta t = 5$ years), which identifies a temporal association among offshore entities that indicates a business shifting operation. It states that (1) if two companies u_2 and u_5 share the same shareholder, (2) company u_5 is active in "Panama", and (3) company u_2 is active in "BVI", then company u_2 will change jurisdiction from "British Virgin Islands (BVI)" to "Panama" in 5 years. Interestingly, R corresponds to the fact that when BVI cracks down on the bearer shares, many companies move bearer share clients to Panama.

If the same threshold of support is set, interesting rules, such as the one above, will be ignored using the concept of "minimal occurrence[14]" since the TGAR's support metric is a subset of our support metric.

Summary. We summarize the major results as follows:

- FindTopkTGARs can significantly reduce the search space, which brings about empirically a 6 fold speedup compared to the baseline method;
- By adding our proposed incremental subgraph matching technique into the bounding framework, we can avoid repeated subgraph matching in the exploring phase, which brings about empirically a 3 fold speedup over findTopkTGARs.

6 Related Work

The following section discusses related work in four directions.

Temporal graph analysis. Our work is related to the studies on temporal graph analysis. Huang et al.[20] investigated the minimum spanning tree problem in temporal graphs. Yang et al.[5] studied the problem of mining a set of diversified temporal subgraph patterns from a temporal graph. Ma et al.[6] proposed a data-driven approach to finding dense subgraphs in large temporal networks with T timestamps. Namaki et al.[14] first extended graph-pattern association rules with temporal constraints over a temporal graph. However, they did not consider the diversity feature such that most of the rules are highly overlapping. Li et al.[7] investigated the problem of finding persistent communities in a temporal network. Chu et al.[21] found density bursting subgraphs online in a temporal weighted graph consisting of a potentially endless stream of updates. Wen et al.[22] proposed a new reachability model that is called span-reachability and is designed to relax the time order dependency and identify the relationship between entities in a given period. Ma et al.[23] investigated temporal bounded simulation on temporal graphs. To the best of our knowledge, our work is the first to research the problem of mining the top-k most diversified TGARs from a temporal graph.

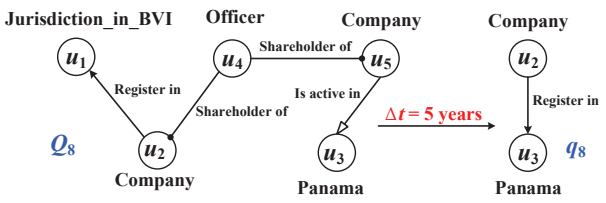


Fig. 10 Real-life TGAR.

Graph association rules. The studies of graph association rules related to our work are Refs. [1, 3, 24, 25]. Fan et al.[1] proposed association rules that are extended with graph patterns over static graphs. To detect data inconsistency, (conditional) functional dependencies are extended[25] to specify value dependencies on clustered values via path patterns and graph patterns. Wang et al.[3] extended the work in Ref. [1] and proposed generalized graph-pattern association rules to capture complex social relations.

Close to our work is Ref. [14] that proposed and studied the graph temporal association rules mining problem. Namaki et al.[14] studied the problem of mining b-maximal graph temporal association rules (GTARs) over a temporal graph, rather than studying the most diversified top-k problem, leading to the mining of many similar rules. Furthermore, the antecedents and consequents of GTARs have only one common vertex, and the antecedents of GTARs only consider their matching instances at each timestamp and do not consider the duration feature. Moreover, the definition of the support metric in this work underestimates the actual support since it only considers each match once of the common vertex in antecedents and consequents at each timestamp and use the number of minimal occurrences.

Temporal subgraph matching. Many temporal queries have been studied to detect events over dynamic networks. An event is denoted as a match of the query. These queries include subgraph isomorphism[26, 27], continuous patterns[28] that incrementally find subgraph

matches over evolving graphs, and durable queries[27] that isomorphism matches that last for the longest period. Mining algorithms are also introduced to discover communication motifs in dynamic networks[16, 29]. Subsequence mining has been leveraged to identify patterns over sequence representation of temporal graphs[16]. Motifs in temporal networks as induced subgraphs on sequences of temporal ordered edges are discovered over unlabeled networks[29], and fast algorithms for specific 3-nodes and 3-edges patterns are proposed to mine topologically frequent motifs. These tasks do not consider events captured by temporal graph patterns and use mining models that are very different from TGARs discovery.

Frequent subgraph pattern mining. The studies of frequent subgraph pattern mining are related to our work. Kuramochi and Karypis[30] proposed an algorithm for exact mining of all frequent subgraphs in a given static graph that enumerates all the isomorphisms of the given graph and relies on the maximum-independent set (MIS) metric whose computation is NP-complete. Elseidy et al.[31] proposed an apriori-like algorithm for exact mining of all frequent subgraphs based on the MIS metric from a given static graph. Apart from the exact mining algorithms, a line of work focused on approximate mining of frequent subgraphs in a given static graph. Kuramochi and Karypis[32] proposed a heuristic approach that largely prunes the search space but discovers only a small subset of frequent subgraphs without provable guarantees. Chen et al.[33] used an approximate version of the MIS metric, allowing approximate matches during the pruning.

7 Conclusion

We have proposed TGARs and viable support, confidence, and diversification measures for the discovery of diversified top-k TGARs. Compared with graph rules introduced in Ref. [1], the TGARs can model temporal information and constraints over temporal graphs. We show the impracticality of computing all TGARs corresponding to a particular event $q(u, u')$ before computing the diversified top-k TGARs. Therefore, we devise a new algorithm to maintain a list L_k of diversified top-k TGARs in each round during the rule generation process. Our algorithm can achieve a guaranteed approximation ratio. We further construct an auxiliary data structure, namely, EIndex, and dynamically maintain EIndex during pattern growth. On the basis of EIndex, we can avoid conducting repeated subgraph matching in the exploring process. We conduct extensive performance studies on large real graphs to demonstrate the efficiency and effectiveness of our approach. We also find that TGARs can be used for activity prediction, among other applications.

We are exploring other quality metrics for TGARs, and experimenting with larger-scale real-world graphs. Another topic is to develop fast online discovery of TGARs over graph streams.

Acknowledgment

This work was partially supported by the National Key Research and Development Program (No. 2018YFB1800203), National Natural Science Foundation of China (No. U19B2024), and Postgraduate Scientific Research Innovation Project of Hunan Province (No. CX20210038).

References

[1] W. Fan, X. Wang, Y. Wu, and J. Xu, Association rules with graph patterns, *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1502–1513, 2015.

[2] X. Wang, Y. Xu, R. Zhao, J. Lin, and H. Zhan, Gparminer: A system to mine graph pattern association rules, in *Proc. International Conference on Database Systems for Advanced Applications* (*DASFAA 2019*), Chiang Mai, Thailand, 2019, pp. 547–552.

[3] X. Wang, Y. Xu, and H. Zhan, Extending association rules with graph patterns, *Expert Syst. Appl.*, vol. 141, no. 10, p. 112897, 2020.

[4] P. Lin, Q. Song, and Y. Wu, Fact checking in knowledge graphs with ontological subgraph patterns, *Data Sci. Eng.*, vol. 3, no. 4, pp. 341–358, 2018.

[5] Y. Yang, D. Yan, H. Wu, J. Cheng, S. Zhou, and J. C. S. Lui, Diversified temporal subgraph pattern mining, in *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 1965–1974.

[6] S. Ma, R. Hu, L. Wang, X. Lin, and J. Huai, Fast computation of dense temporal subgraphs, in *Proc. 33rd IEEE International Conference on Data Engineering* (*ICDE 2017*), San Diego, CA, USA, 2017, pp. 361–372.

[7] R. Li, J. Su, L. Qin, J. X. Yu, and Q. Dai, Persistent community search in temporal networks, in *Proc. 34th IEEE International Conference on Data Engineering* (*ICDE 2018*), Paris, France, 2018, pp. 797–808.

[8] D. Guo, B. Ren, and G. Cheng, Minimum-cost forest for uncertain multicast with delay constraints, *Tsinghua Science and Technology*, vol. 24, no. 2, pp. 147–159, 2019.

[9] Y. Qin, D. Guo, X. Lu, and G. Cheng, Design and optimization of VLC enabled data center network, *Tsinghua Science and Technology*, vol. 25, no. 1, pp. 82–92, 2020.

[10] Z. Hu, X. Teng, D. Guo, B. Ren, P. Lv, and Z. Liu, Comparing set reconciliation methods based on bloom filters and their variants, *Tsinghua Science and Technology*,

vol. 21, no. 2, pp. 157–167, 2016.
[11] Q. Yuan, G. Cong, and A. Sun, Graph-based point-of-interest recommendation with geographical and temporal influences, in *Proc. 23rd ACM International Conference on Conference on Information and Knowledge Management* (*CIKM 2014*), Shanghai, China, 2014, pp. 659–668.
[12] S. Amer-Yahia, L. V. S. Lakshmanan, S. Vassilvitskii, and C. Yu, Battling predictability and overconcentration in recommender systems, *IEEE Data Eng. Bull.*, vol. 32, no. 4, pp. 33–40, 2009.
[13] D. Xin, H. Cheng, X. Yan, and J. Han, Extracting redundancy-aware top-k patterns, in *Proc. 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, USA, 2006, pp. 444–453.
[14] M. H. Namaki, Y. Wu, Q. Song, P. Lin, and T. Ge, Discovering graph temporal association rules, in *Proc. 2017 ACM on Conference on Information and Knowledge Management* (*CIKM 2017*), Singapore, 2017, pp. 1697–1706.
[15] S. Gollapudi and A. Sharma, An axiomatic approach for result diversification, in *Proc. 18th International Conference on World Wide Web* (*WWW 2009*), Madrid, Spain, 2009, pp. 381–390.
[16] S. Gurukar, S. Ranu, and B. Ravindran, COMMIT: A scalable approach to mining communication motifs from dynamic networks, in *Proc. 2015 ACM SIGMOD International Conference on Management of Data*, Melbourne, Australia, 2015, pp. 475–489.
[17] M. Fiedler and C. Borgelt, Subgraph support in a single large graph, in *Proc. 7th IEEE International Conference on Data Mining* (*ICDM 2007*), Omaha, NE, USA, 2007, pp. 399–404.
[18] X. Yan and J. Han, GSpan: Graph-based substructure pattern mining, in *Proc. 2002 IEEE International Conference on Data Mining* (*ICDM 2002*), Maebashi City, Japan, 2002, pp. 721–724.
[19] Q. Zhang, D. Guo, X. Zhao, and A. Guo, On continuously matching of evolving graph patterns, in *Proc. 28th ACM International Conference on Information and Knowledge Management* (*CIKM 2019*), Beijing, China, 2019, pp. 2237–2240.
[20] S. Huang, A. W. Fu, and R. Liu, Minimum spanning trees in temporal graphs, in *Proc. 2015 ACM SIGMOD International Conference on Management of Data*, Melbourne, Australia, 2015, pp. 419–430.
[21] L. Chu, Y. Zhang, Y. Yang, L. Wang, and J. Pei, Online density bursting subgraph detection from temporal graphs, *Proc. VLDB Endow.*, vol. 12, no. 13, pp. 2353–2365, 2019.
[22] D. Wen, Y. Huang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, Efficiently answering span-reachability queries in large temporal graphs, in *Proc. 36th IEEE International Conference on Data Engineering* (*ICDE 2020*), Dallas, TX, USA, 2020, pp. 1153–1164.
[23] Y. Ma, Y. Yuan, M. Liu, G. Wang, and Y. Wang, Graph simulation on large scale temporal graphs, *GeoInformatica*, vol. 24, no. 1, pp. 199–220, 2020.
[24] M. Berlingerio, F. Bonchi, B. Bringmann, and A. Gionis, Mining graph evolution rules, in *Proc. 2009th European Conference on Machine Learning and Knowledge Discovery in Databases* (*ECMLPKDD 2009*), Bled, Slovenia, 2009, pp. 115–130.
[25] W. Fan, Y. Wu, and J. Xu, Functional dependencies for graphs, in *Proc. 2016 International Conference on Management of Data*, San Francisco, CA, USA, 2016, pp. 1843–1857.
[26] H. Hsieh and C. Li, Mining temporal subgraph patterns in heterogeneous information networks, in *Proc. 2010 IEEE Second International Conference on Social Computing*, Minneapolis, MN, USA, 2010, pp. 282–287.
[27] K. Semertzidis and E. Pitoura, Durable graph pattern queries on historical graphs, in *Proc. 2016 IEEE 32nd International Conference on Data Engineering* (*ICDE 2016*), Helsinki, Finland, 2016, pp. 541–552.
[28] J. Gao, C. Zhou, and J. X. Yu, Toward continuous pattern detection over evolving large graph with snapshot isolation, *VLDB J.*, vol. 25, no. 2, pp. 269–290, 2016.
[29] A. Paranjape, A. R. Benson, and J. Leskovec, Motifs in temporal networks, in *Proc. Tenth ACM International Conference on Web Search and Data Mining* (*WSDM 2017*), Cambridge, UK, 2017, pp. 601–610.
[30] M. Kuramochi and G. Karypis, Finding frequent patterns in a large sparse graph*, *Data Min. Knowl. Discov.*, vol. 11, no. 3, pp. 243–271, 2005.
[31] M. Elseidy, E. Abdelhamid, S. Skiadopoulos, and P. Kalnis, GraMi: Frequent subgraph and pattern mining in a single large graph, *Proc. VLDB Endow.*, vol. 7, no. 7, pp. 517–528, 2014.
[32] M. Kuramochi and G. Karypis, GREW—A scalable frequent subgraph discovery algorithm, in *Proc. 4th IEEE International Conference on Data Mining* (*ICDM2004*), Brighton, UK, 2004, pp. 439–442.
[33] C. Chen, X. Yan, F. Zhu, and J. Han, GApprox: Mining frequent approximate patterns from a massive network, in *Proc. 7th IEEE International Conference on Data Mining* (*ICDM 2007*), Omaha, NE, USA, 2007, pp. 445–450.

Qianzhen Zhang received the BSc and MSc degrees in computer science from Zhengzhou University and Guangxi University in 2014 and 2018, respectively. Currently, he is pursing the PhD degree at the College of Systems Engineering, National University of Defense Technology, China. His research interests include continuous subgraph matching, graph data analytics, and knowledge graph.

Chu Huang received the BSc degree in information management and information system from Guangxi University in 2020. Currently, she is pursuing the MSc degree at the College of Systems Engineering, National University of Defense Technology, China. Her research interests include graph mining and graph data analytics.

Deke Guo received the BSc degree from Beihang University, and the PhD degree from National University of Defense Technology. He is currently a professor with the College of Systems Engineering, National University of Defense Technology. His research interests include distributed systems, software-defined networking, data center networking, wireless and mobile systems, and interconnection networks. He is a senior member of the IEEE and a member of the ACM. He was a visiting scholar at the Department of Computer Science and Engineering in Hong Kong University of Science and Technology from 2007 to 2009, under supervision of Dr. Yunhao Liu.

Xiang Zhao received the PhD degree of computer science and engineering from University of New South Wales (UNSW), Australia, under the supervision of Prof. Xuemin Lin in 2014. He is currently an associate professor at the College of Systems Engineering, National University of Defense Technology, China. Since 2015, he has also been a concurrent associate research fellow at Collaborative Innovation Center of Geospatial Technology. His research interests include knowledge graphs, graph data analytics, and natural language processing.

Xi Wang received the BSc degree in management science and engineering from Dalian Maritime University in 2019. Currently, she is pursuing the MSc degree at the College of Systems Engineering, National University of Defense Technology, China. Her research interests include continuous subgraph matching and graph data analytics.