Discovery of Temporal Network Motifs

Hanqing Chen, Shuai Ma, *Senior Member, IEEE,* Junfeng Liu, and Lizhen Cui

Abstract—Network motifs provide a deep insight into the network functional abilities, and have proven useful in various practical applications. Existing studies reveal that different definitions of motifs may be needed for different temporal networks. In this study, we focus on a class of temporal networks such that the nodes and edges keep fixed, but the edge labels vary regularly with timestamps. First, we propose a proper definition of temporal motifs, which appear continuously within sufficiently large time intervals, to properly reinterpret the recurrent and statistically significant nature of motifs in temporal networks. Second, we develop a low polynomial time solution to find temporal motifs for all possible time intervals with the top to bottom and right to left scheme, based on the analyses of the properties for temporal motifs. Third, we develop a theoretically faster incremental solution to efficiently find temporal motifs to support continuously updates of temporal networks, by identifying unaffected time intervals and unnecessary edges. Finally, we have conducted extensive experiments to verify the efficiency and usefulness of our static and incremental solutions.

Index Terms—Temporal graphs, temporal networks, network motifs, temporal motifs.

I. INTRODUCTION

NETWORK (or graph) motifs refer to recurrent and statistically significant subgraphs or patterns of a larger graph, which provide a deep insight into the network functional abilities [1], and have been applied widely, such as biological networks [2]–[6], social networks [4]–[9], electrical circuits [10], software architectures [11], and transport networks [5], [6]. It has been readily realized that dynamics have become an essential feature of the data analytic systems and applications that could be modeled as graphs or networks. In fact, dynamic networks have drawn significant attention from both industry and academia under various terms [12], such as temporal networks, dynamic graphs, evolving networks, evolutionary networks, and graph streams [12]–[20].

The study of temporal networks suggests the need of a significant reinterpretation of temporal network motifs. Most studies focus on time-dependent edges in temporal networks, *i.e.,* the dynamics come from the evolution/change of edges. [21] firstly studies the dynamics of motifs in networks with time-dependent edges, and reveals that motifs evolve over time. Subsequent studies begin to re-define motifs, where edges are attached with beginning timestamps and durations or only timestamps, and timestamps of the edges satisfy different time orders and time windows [22]–[34], *e.g.,* partial or total time orders, and local or global time windows, respectively.

The difference between the total and partial time orders lies in whether the time order of all the edges is defined or not. The local time window is the time difference bound on adjacent edges, while the global time window is defined on all the edges. [22], [23], [26], [30] impose a partial time order and a local time window on edges in a temporal motif, while [25] imposes a total time order and a local time window with a duration bound on edges. [27], [29] impose a total time order and a global time window on edges in a temporal motif, where [29] further introduces a flow bound on edges. [32] imposes both a local time window and a global time window on edges in a temporal motif. [24] focuses on the graph matching problem and imposes a partial time order and a global time window on its query graph. Some existing studies focus on the specific structure motifs, such as temporal triangles [31], temporal circles [28], temporal butterflies ($(2, 2)$-bicliques) [34] and temporal hypergraphs [33].

Different from the above mentioned, there are studies focusing more on persistent nodes or edges, or their time-dependent weights (labels) in temporal networks. Each of nodes, edges or their weights (labels) in their temporal motifs displays statics or similar dynamics over a user-defined period. [35], [36] impose an increasing or decreasing trend of weights on the nodes in a motif, *i.e.,* induced subgraphs with node weights evolving in a consistent trend over a period. [37] imposes time-persistent relations on the edges in a pattern called induced relational state, *i.e.,* induced subgraphs with edge labels and directions keeping unchanged over a period. [38] imposes time-persistent relations on the subgraph structure in a pattern called persistent community, *i.e.,* induced subgraphs of the node set preserving the k-core structures over a period.

The studies mentioned above reveal that different definitions of temporal motifs are essentially needed for different temporal networks due to the various application requirements. Moreover, most (if not any) existing methods of finding temporal motifs are all computationally intractable, *i.e.,* NP-hard, which may hinder their applications in practice.

In this study, we investigate an important class of temporal networks, where the nodes and edges are fixed, but the edge labels vary regularly with timestamps [13], [20], [39]. Essentially, such a temporal network with T timestamps can be viewed as T snapshots of a static network, where each timestamp of the temporal network corresponds to a static network, *i.e.,* a snapshot of the temporal network. Road networks and energy transmission networks typically fall into this category [16], [40], where roads or transmission lines are relatively fixed, and their traffic condition labels (e.g., congested, slow, and fast) or energy usage signal levels correspond to dynamic edge labels. Hence, investigating such a class of temporal networks is significant, as traffic and energy consumption analyses are significant for urban computing [41].

H. Chen, S. Ma (correspondence) and J. Liu are with the SKLSDE, Beihang University, Beijing 100191, China.
E-mail: {chenhanqing, mashuai, liujunfeng}@buaa.edu.cn
L. Cui is with the School of Software & C-FAIR, Shandong University, Shandong 250100, China.
E-mail: clz@sdu.edu.cn

However, the study of motifs in such temporal networks needs to deal with three challenging issues. (1) How to properly define temporal motifs that are useful for practical applications? [42] evaluates the impact of temporal inducedness and timing constraints (e.g., local and global time windows) in temporal motifs on social and communication networks. However, temporal motifs with other constraints have not been well considered. (2) An efficient solution to discover temporal motifs is needed, as temporal networks are typically large. Most existing methods of finding temporal motifs are all computationally intractable, as (subgraph) isomorphism tests are essentially an unavoidable step for the discovery of those temporal motifs [22], [23], [25]–[27], [29]–[32], [35]. While [35]–[38] do not need isomorphism tests, they need to enumerate induced subgraphs in exponential time. There also exist methods adopting approximation techniques to make computation tractable or parallel techniques to achieve speedup, including sampling approaches [33], [43]–[46] and parallel approaches [34], [47] to motif counting problems, and dual simulation [24], [48] in graph matching problems. (3) Temporal networks are naturally dynamic and continuously updated, and hence dynamic algorithms are needed, which has not been considered by existing studies.

Contributions & roadmap. To this end, we propose a concept of temporal motifs for temporal networks, whose nodes and edges are fixed, but edge labels vary regularly with timestamps, and develop efficient static and incremental approaches to finding temporal motifs.

(1) We propose a proper definition of (maximal and non-expandable) temporal motifs (motifs appearing continuously within sufficiently large time intervals) to properly reinterpret the recurrent and statistically significant nature of motifs in temporal networks (Section II). Different from existing studies on motifs in temporal networks, our temporal motifs can be computed efficiently in low polynomial time. Indeed, this is among the first works on discovering temporal motifs of temporal networks in polynomial time.

(2) We develop an efficient solution to compute maximal and non-expandable temporal motifs for all possible time intervals with the top to bottom and right to left scheme in low polynomial time (Section III). We design two data structures (the EL table and IC trees) to facilitate the finding of the edges having the same labels in time intervals, build the equivalence connection between maximal temporal motifs and connected components, and check whether a maximal temporal motif is expandable in constant time, based on the analyses of the properties for temporal motifs.

(3) We develop a theoretically faster incremental solution with a better time complexity than the static solution. It efficiently finds all the maximal and non-expandable temporal motifs for temporal networks with continuous updates (Section IV), by identifying unaffected time intervals and unnecessary edges.

(4) Using both real-life and synthetic datasets, we conduct an extensive experimental study (Section V). We find that (a) our algorithms FTM-EL and FTM-IC using the EL table and IC trees, respectively, are fast on large graphs, and our static algorithm FTM-EL is on average $(1.06, 1.13, 1.25, 1.75)$

(a) Temporal graph G with interval $[1, 6]$ (b) Temporal subgraph H_1 of G with interval $[2, 5]$

(c) Temporal subgraph H_2 of G with interval $[3, 5]$ (d) Temporal subgraph H_3 of G with interval $[2, 5]$
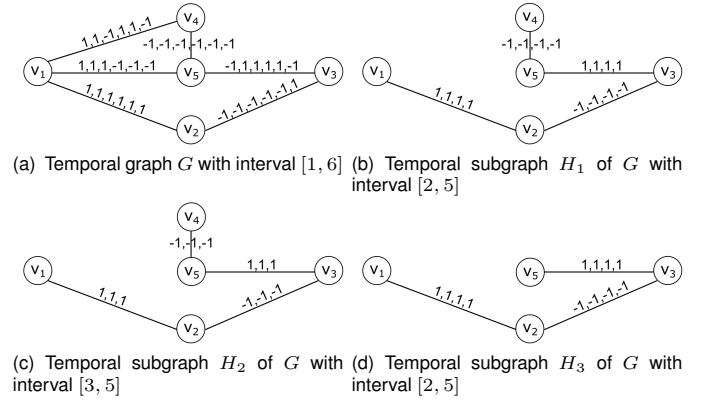
Figure 1. Running example

times faster than our baseline FTM-IC on datasets (BJ-DATA, EURDATA, LANDATA, SYNDATA), respectively; (b) our incremental algorithms DFTM-EL and DFTM-IC are $(2.09, 1.91, 1.55, 2.04)$ and $(2.04, 1.94, 1.63, 1.94)$ times faster than their static counterparts on (BJDATA, EURDATA, LANDATA, SYNDATA), respectively. (c) our temporal motifs are useful for practical applications through case studies on three real-life datasets. (d) The comparison with persistent community search [38] further justifies that existing methods are not suitable to identify our temporal motifs.

Due to space limitations, we give the details of existing temporal network motifs, the data structure (IC trees) in algorithms FTM-IC and DFTM-IC, the pseudo codes of algorithms FTM and DFTM, the algorithm complexity analyses, another alternative algorithm and the extra experimental studies in the supplemental material.

II. PRELIMINARY NOTATION AND PROBLEM DEFINITION

In this section, we introduce the basic concepts and the problem statement. Table I shows the main notations. In this study, we use the terms network and graph interchangeably.

Temporal graphs. A *temporal graph* $G(V, E, T_b, T_f, L)$ is a weighted undirected graph with edge labels varying with timestamps (positive integers), where (1) V is a finite set of nodes, (2) $E \subseteq V \times V$ is a finite set of edges, in which (u, v) or $(v, u) \in E$ denotes an undirected edge between nodes u and v, (3) $[T_b, T_f]$ is a time interval representing $(T_f - T_b + 1)$ timestamps, in which $T_b \leq T_f$ are the beginning and finishing timestamps, respectively, and (4) L is a set of label functions such that for each timestamp $t \in [T_b, T_f]$, L^t is a function that maps each edge $e \in E$ to a label (integers for simplicity). When it is clear from the context, we simply use $G(V, E, L)$ or $G(V, E, T_b, T_f)$ to denote a temporal graph.

We consider a special class of temporal graphs such as road networks and communication networks, where graph nodes and edges are fixed, but edge labels vary with respect to timestamps [13], [20], [39]. Intuitively, (1) a temporal graph $G(V, E, L)$ essentially denotes a sequence $\langle G_1(V, E, L^{T_b}), \ldots, G_{t-T_b+1}(V, E, L^t), \ldots, G_{T_f-T_b+1}(V, E, L^{T_f}) \rangle$ of $T = T_f - T_b + 1$ standard graphs, and (2) its edge labels $L^t(e)$ ($t \in [T_b, T_f]$) specify the distances, communication latencies or travelling duration [12], [16], or the affinity or collaborative compatibility [49] between the

nodes connecting edges at a timestamp in a discrete way with integers to denote the states, *e.g.,* fast/congested traffic conditions and friend/foe relationships.

We call the temporal graph $G(V, E, L)$ at timestamp t (*i.e.,* $G_t(V, E, L^t)$, $t \in [T_b, T_f]$) its *snapshot* at timestamp t, and we also use intervals instead of time intervals for simplicity.

Temporal subgraphs. A temporal graph $G_s(V_s, E_s, \hat{T}_b, \hat{T}_f, L_s)$ is a *subgraph* of $G(V, E, T_b, T_f, L)$, if $V_s \subseteq V, E_s \subseteq V_s \times V_s$ and $E_s \subseteq E, \hat{T}_b \leq \hat{T}_f \in [T_b, T_f]$, and $L_s^t(e) = L^t(e)$ for any timestamp $t \in [\hat{T}_b, \hat{T}_f]$ and edge $e \in E_s$.

That is, temporal subgraph $G_s(V_s, E_s, \hat{T}_b, \hat{T}_f)$ only contains a subset of nodes and edges of G, its edges have the same labels as G, and it is restricted within the interval $[\hat{T}_b, \hat{T}_f]$ that falls into the interval $[T_b, T_f]$ of G.

We next define temporal motifs in a temporal network G *w.r.t.* a frequency threshold k. Here k is a positive integer.

Temporal motifs. A temporal subgraph $G_s(V_s, E_s, \hat{T}_b, \hat{T}_f)$ is a temporal motif if and only if (1) G_s is a connected subgraph of G, (2) for each edge e in E_s, $L^{\hat{T}_b}(e) = \cdots = L^{\hat{T}_f}(e)$, *i.e.,* each edge has the same label at all the timestamps in interval $[\hat{T}_b, \hat{T}_f]$ and (3) $\hat{T}_f - \hat{T}_b + 1 \geq k$, *i.e.,* appear in at least k continuous snapshots.

Expandable temporal motifs. A temporal motif $G_s(V_s, E_s, \hat{T}_b, \hat{T}_f)$ is left (resp. right) expandable if and only if for each edge e in E_s, $L^{\hat{T}_b-1}(e) = L^{\hat{T}_b}(e)$ (resp. $L^{\hat{T}_f}(e) = L^{\hat{T}_f+1}(e)$). That is, for each edge $e \in E_s$, its label keeps unchanged in interval $[\hat{T}_b - 1, \hat{T}_f]$ (resp. $[\hat{T}_b, \hat{T}_f + 1]$) if G_s is left (resp. right) expandable.

We say that G_s is *non-expandable* if it is neither left nor right expandable. Note that $\hat{T}_b - 1 \geq T_b$ and $\hat{T}_f + 1 \leq T_f$.

Maximal temporal motifs. A temporal motif $G_s(V_s, E_s, \hat{T}_b, \hat{T}_f)$ is maximal if and only if there is no edge e in $E \setminus E_s$ such that e is incident to G_s, $L^{\hat{T}_b}(e) = \cdots = L^{\hat{T}_f}(e)$. That is, e is connected to G_s and has the same label in $[\hat{T}_b, \hat{T}_f]$.

We next illustrate these concepts with examples.

Example 1: (1) Figure 1(a) depicts a temporal graph G with 5 nodes, 6 edges and 6 timestamps.

(2) Figures 1(b) and 1(d) show two temporal subgraphs H_1 and H_3 of G with the same interval $[2, 5]$, and Figure 1(c) shows a temporal subgraph H_2 of G with interval $[3, 5]$.

(3) Consider frequency threshold $k = 2$. (a) Temporal subgraphs H_1, H_2 and H_3 are all temporal motifs, as they are all connected subgraphs, all edges keep their labels unchanged, and the lengths of their intervals are all not less than k. (b) Temporal motif H_2 in Figure 1(c) is a left expandable temporal motif, as each its edge has the same label in interval $[2, 5]$. (c) Temporal motif H_3 in Figure 1(d) is not a maximal temporal motif, as there exists edge (v_4, v_5) that is incident to H_3 and has the same label in interval $[2, 5]$. (d) Temporal motif H_1 in Figure 1(b) is maximal and non-expandable. Labels of its edges (v_1, v_2), (v_2, v_3), (v_3, v_5) and (v_4, v_5) change at timestamps 1 and 6, and there exist no adjacent edges in G that have the same label in interval $[2, 5]$. □

We now give the problem statement.

Problem statement. Given a temporal graph $G(V, E, T_b, T_f, L)$ and a frequency threshold k, the problem is to find all the maximal and non-expandable temporal motifs $G_s(V_s, E_s, \hat{T}_b, \hat{T}_f)$ in G.
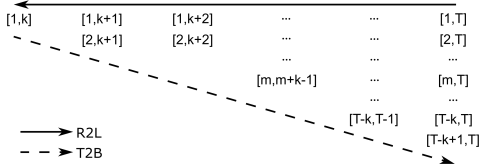
We next illustrate the problem with an example.

Example 2: Given the temporal graph G in Figure 1(a) and frequency threshold $k = 4$, the temporal subgraph H_1 in Figure 1(b) is one of the maximal and non-expandable temporal motifs in the final output. □

Applications. Our defined temporal motifs have certain potential applications, including discovering traffic patterns with long-time congested or bottleneck roads in road networks [50], [51], and energy consumption patterns with continuous high energy demand signal levels in transmission networks [40], [52]. These applications typically concern some significant regular phenomenons from a temporal viewpoint, which are significant for urban computing [41].

Our temporal motifs can also be adopted to other applications with proper preprocessing. (a) For temporal networks with varying edges, we can represent absent edges by virtual labels, and compute our temporal motifs without virtual labels. (b) To mine increasing or decreasing trends of the edge weights on the temporal networks with varying edge weights, we can transform the networks by using two labels to represent increasing or decreasing weights between two consecutive snapshots, and obtain the networks with varying edge labels. (c) For temporal networks with numerical data on edges, we can discretize them into ranges *i.e.,* labels determined by appropriate quantiles. (d) For edges with multiple properties, we can represent these properties by distinct label types, and further integrate these types into a single label type.

Remarks. Different from existing studies on temporal motifs [22]–[38], [43]–[47], we focus on a special but significant class of temporal networks, where the nodes and edges are fixed, but the edge labels vary regularly with timestamps [13], [20], [39]. Indeed, the nodes, edges or their weights (labels) in the above mentioned applications display statics or similar dynamics over a period, which are significant as mentioned in [35]–[38]. However, existing studies on temporal motifs, which are somehow similar to our motifs, are not suitable for our applications. [35], [36] define motifs with the node weights evolving in a consistent trend, which is not suitable to analyze long time unchanged areas. [35]–[38] define their motifs with the inducedness restrictions, which exhibit a bias towards certain types of motifs, and are not suitable for the need of discovering motifs in unknown situations (see a recent survey [42]). In particular, most existing methods of finding temporal

Table I
NOTATIONS AND THEIR DESCRIPTIONS

Notations	Descriptions
G / G_s	The temporal graph / The temporal subgraph
V,E / V_s,E_s	The node sets and edge sets in G / G_s
T_b, T_f / $\hat{T}_b, \hat{T}_f$	The beginning and finishing timestamps of G / G_s
G_t	The snapshot of G at timestamp t
L / L_s / L^t	The label function set of G / G_s / G_t
$\mathsf{S}[m,i]$	The set of edges whose labels keep unchanged in $[m,i]$
$\mathsf{R}[m,i]$	The difference set of edge sets $\mathsf{S}[m,i]$ and $\mathsf{S}[m,i+1]$
$G_s(\mathsf{S}[m,i])$	The temporal subgraph with edges and nodes in $\mathsf{S}[m,i]$
$\mathsf{EMaxIntvl}[e]$	The maximum interval $[m',i']$ for each edge e and row number m where the label of e keeps unchanged
$\mathsf{CC}[i,T]$	Connected components for edges in $\mathsf{S}[m,i]$
$\mathsf{TF}[m,i]$	The set of temporal motifs in interval $[m,i]$

Figure 2. TI-Table for temporal graph $G(V, E, 1, T, L)$ and k

motifs are all computationally intractable, such as [35]–[38], which enumerate any induced subgraphs by traversing all the nodes recursively in exponential time. However, our proposed temporal motifs can be computed in polynomial time, as subgraph isomorphism tests and enumeration of all the subgraphs are not necessary due to the definition of our temporal motifs and the type of temporal networks investigated.

III. FINDING TEMPORAL MOTIFS

In this section, we explain how to find all the maximal and non-expandable temporal motifs, given temporal graph $G(V, E, T_b, T_f, L)$ and frequency threshold k. Without loss of generality, we assume that $T_b = 1$ and $T_f = T$ in the sequel. The main result is stated below.

Theorem 1: *Given temporal graph $G(V, E, 1, T, L)$ and frequency threshold k, there exists an algorithm to find all the maximal and non-expandable temporal motifs in $O((T - k)|E|)$ time.* □

A. Analyses of Temporal Motifs

We start with the analyses of temporal motifs.

TI-Table. We arrange possible intervals of all the maximal and non-expandable temporal motifs $G_s(V_s, E_s, \hat{T}_b, \hat{T}_f)$ in a Triangularized Interval Table (TI-Table), as shown in Figure 2, where the intervals in the same row have the same beginning timestamp, and the intervals in the same column have the same finishing timestamp. Note that TI-Table is virtual and not physically constructed, which is provided just for the ease of understanding our algorithms.

Using TI-Table we see three challenges for finding all the maximal and non-expandable temporal motifs.

Challenge 1: The first one is how to efficiently identify the edges with the same labels in the intervals of the TI-Table.

Let $S[m, i]$ ($1 \le m \le T-k+1$, $m+k-1 \le i \le T$) be the set of edges in G whose labels are constant over the interval $[m, i]$. It is easy to verify the following observation.

Fact 1: $S[m, T] \subseteq S[m, T-1] \subseteq \cdots \subseteq S[m, m+k-1]$. □

The above observation reveals that there are heavy redundant edges for intervals in the same row. Hence, we define R edge sets based on the differences of S edge sets.

Let $R[m, i] = S[m, i] - S[m, i+1]$ for $m+k-1 \le i < T$, and $R[m, T] = S[m, T]$. By the definitions of S and R edge sets, it is easy to verify that $S[m, i] = \bigcup_{i \le j \le T} R[m, j]$. Further, R edge sets have the following non-redundancy property.

Proposition 1: $R[m, i] \cap R[m, j]$ *is empty for any $i \ne j$, where $1 \le m \le T - k + 1$ and $m + k - 1 \le i, j \le T$.* □

Proof: Assume that there exists an edge e in $R[m, i]$ and $R[m, j]$ ($m + k - 1 \le i < j \le T$). By the definition, we have $e \in S[m, i] - S[m, i+1]$ and $e \in S[m, j] - S[m, j+1]$ or

$S[m, T]$ if $j = T$. However, as $i < j$ and $S[m, j] \subseteq S[m, i+1]$, $e \in S[m, i + 1]$. This implies $e \notin R[m, i]$, which is a contradiction. Hence we have the conclusion. □

Proposition 1 tells us that the first challenge is converted to compute R edge sets. As will be seen shortly, we design two data structures to facilitate the computation.

Challenge 2: Once we obtain the edges having the same labels in intervals (*i.e.,* S edge sets assembled with R edge sets), the next challenge is how to generate maximal temporal motifs.

For edge set $S[m, i]$, we can derive temporal subgraph $G_s(V(S[m, i]), S[m, i], L_s)$ with interval $[m, i]$, denoted as $G_s(S[m, i])$, by keeping only the edges and nodes in $S[m, i]$. There is a close connection between maximal temporal motifs and connected components, as shown below.

Proposition 2: *Given edge set $S[m, i]$ ($1 \le m \le T-k+1$ and $m + k - 1 \le i \le T$), each connected component of temporal subgraph $G_s(S[m, i])$ is a maximal temporal motif.* □

Proof: Assume that there is a connected component that is not a maximal temporal motif $G_s(V_s, E_s, m, i)$. It implies that there is an extra edge $e \notin E_s$ adjacent to a node in G_s and having the same label in interval $[m, i]$, which contradicts with the fact that connected components are maximal [53]. Hence we have the conclusion. □

By Proposition 2, we have an efficient solution to generate maximal temporal motifs based on connected components, which can be computed in linear time [53].

We have two possible schemes to generate maximal temporal motifs for the m-th row in the TI-Table. (1) Left to right (L2R) scheme, *i.e.,* generate maximal temporal motifs from intervals $[m, m + k - 1]$ to $[m, T]$. The maximal temporal motifs with interval $[m, i + 1]$ are generated from those with interval $[m, i]$ by removing edges in $R[m, i]$. (2) Right to left (R2L) scheme, *i.e.,* generate maximal temporal motifs from intervals $[m, T]$ to $[m, m+k-1]$. The maximal temporal motifs with interval $[m, i]$ are generated from those with intervals $[m, i + 1], \dots, [m, T]$ by adding edges in $R[m, i]$.

Let $|E_m|$ and $|V_m|$ be the number of edges and nodes in temporal subgraph $G_s(S[m, m + k - 1])$, respectively. All maximal temporal motifs in the m-th row can be computed in an incremental way in $O(|E_m|)$ time by the R2L scheme [54], while it is much slower in a decremental way by the L2R scheme ($O(|E_m| \log |V_m|)$ time even for sparse graphs) [55], as the key step of computing maximal temporal motifs in the R2L or L2R scheme is maintaining connected components in an incremental or decremental way. As no isolated nodes exist in $G_s(S[m, m+k-1])$ ($E_m \ge \frac{V_m}{2}$), the R2L scheme is better than the L2R scheme for computing maximal temporal motifs.

Challenge 3: Finally, we need to check whether a maximal temporal motif is expandable. When generating maximal temporal motifs in the R2L scheme, we find that there is an efficient way to do this, as shown below.

Proposition 3: *Each dynamically generated maximal temporal motif in the R2L scheme is not right expandable.* □

Proof: Assume that we generate the maximal temporal motifs for interval $[m, i]$ with set $R[m, i]$ ($i \in [m + k - 1, T]$). By the definition of R edge sets, any edge in $R[m, i]$ changes its label at timestamp $i + 1$. Hence, we have the conclusion. □

	t=1	t=2	t=3	t=4	t=5	t=6
(v_1,v_4)	len_1=-2 lab_1=1	len_2=2 lab_2=1	len_3=-1 lab_3=-1	len_4=-2 lab_4=1	len_5=2 lab_5=1	len_6=-1 lab_6=-1
(v_1,v_2)	len_1=-6 lab_1=1	len_2=2 lab_2=1	len_3=3 lab_3=1	len_4=4 lab_4=1	len_5=5 lab_5=1	len_6=6 lab_6=1

Figure 3. The EL table for (v_1,v_4) and (v_1,v_2)

Proposition 3 reveals that for a dynamically generated motif with interval $[m, i]$, we only need to check whether it is left expandable. That is, to check whether it has already appeared in the maximal temporal motifs with intervals $[1, i], \cdots, [m - 1, i]$, i.e., the intervals above $[m, i]$ in the same column of the TI-Table. It implies that a top to bottom (T2B) scheme, which computes motifs with intervals from the first row to the last row in the TI-Table, is a proper choice.

By Propositions 2 & 3, we know that the *top to bottom (T2B, for rows) and right to left (R2L, for columns)* scheme is a better choice to compute the maximal and non-expandable temporal motifs for all the intervals in the TI-Table.

B. Compute R Edge Sets

We first introduce how to compute R edge sets. To reduce the space cost of R edge sets, we choose to compute R edge sets by rows of TI-Table in the T2B scheme, i.e., edge sets $R[m, i]$ $(m + k − 1 \leq i \leq T)$ for each m from 1 to $T − k + 1$. Assume that an edge e has the same label in intervals $[H_1 = 1, U_1], [H_2, U_2], \cdots, [H_n, U_n = T]$ (referred to as *maximum intervals*), where $H_{j+1} = U_j + 1$ for $1 \leq j < n$, and that the edge e in any two adjacent intervals has different labels. In order to compute R edge sets for m-th row, we need to find the maximum interval $[H_j, U_j]$ containing $[m, m + k − 1]$ (the left most interval of the m-th row in the TI-Table), as it is the minimum interval satisfying the frequency threshold k. If there exists a maximum interval $[H_j, U_j]$ containing $[m, m+k−1]$, the edge e belongs to $R[m, U_j]$, and the edge e does not belong to any edge sets $R[m, i]$ $(m + k − 1 \leq i \leq T)$, otherwise.

We next use an example to show how the above process.

Example 3: We consider the edge (v_1, v_5) in temporal graph G in Figure 1(a), frequency threshold $k = 3$. (1) For the row number $m = 1$, we find the maximum interval $[1, 3]$, and know that $(v_1, v_5) \in R[1, 3]$. (2) For the row number $m = 2$, we find no maximum intervals containing $[2, 4]$, and hence $(v_1, v_5) \notin R[2, i]$ for $i \geq 4$. □

To efficiently identify these maximum intervals, we present our baseline solution and our optimized solution with data structures called IC trees and the EL table, respectively.

Baseline. Our baseline solution is a red-black balanced tree for each edge, referred to as *interval containment tree* (or IC tree), inspired by interval trees [53]. IC trees are designed for *containment* semantics to address this issue. They can be created in $O(|V_{tree}| \log|V_{tree}|)$ time, and identify intervals containing a given interval in $O(\log|V_{tree}|)$ time and $O(|V_{tree}|)$ space. Here, $|V_{tree}|$ is the number of tree nodes. Note that an IC tree returns $NULL$ if no intervals in the tree contain the given interval. Detailed introductions are available in our supplemental material due to space limitations.

To further improve the efficiency of identifying maximum intervals, we next design a data structure called EL table.

Procedure computeRES
Input: The EL table or IC trees for temporal graph $G(V, E, 1, T, L)$, frequency threshold k, array EMaxIntvl, and row number m.
Output: R edge sets and array EMaxIntvl.
1. **let** $R[m, i] := \emptyset$ for all $i \in [m + k − 1, T]$;
2. **for each** edge e **in** E
3. $[m', i'] :=$ EMaxIntvl$[e]$;
4.   **if** $[m', i']$ is null **or** $i' < m$ **then**
5.   find the maximum interval $[m, i']$ for edge e and row m in the EL table or IC tree for e;   /*cases (1) and (2)*/
6.   EMaxIntvl$[e] := [m, i']$ if $[m, i']$ exists;
7.   **if** $i' \geq m + k − 1$ **then**
8. $R[m, i'] := R[m, i'] \cup \{e\}$;
9.   **if** $[m', i']$ is not null **and** $i' \geq m + k − 1$ **then**
10. $R[m, i'] := R[m, i'] \cup \{e\}$;   /*case (4) */
11. **return** (R, EMaxIntvl).

Figure 4. Procedure computeRES

EL table. EL table is a $|E| \times T$ table, where each row records the label perseveration information of an edge *w.r.t.* timestamps (referred to as the EL table), in which for each edge e and timestamp t, it stores the label lab_t of edge e at timestamp t, and the number len_t of timestamps that edge e keeps the same label lab_t until timestamp t in the time order (positive numbers) or in the reverse order (negative numbers). We store negative numbers len_t at timestamps t when edge e changes its label or $t = 1$, and positive numbers at other timestamps. Figure 3 depicts the EL table for edges (v_1, v_4) and (v_1, v_2) only in Figure 1(a), where for edge (v_1, v_4), it has four negative numbers len_t, as it changes its label at timestamps $t = 3, 4, 6$.

The EL table takes $O(T|E|)$ space and can be created in $O(T|E|)$ time, as it can store the label lab_t and the number len_t for each edge e and timestamp t in constant time. Note that the negative number len_t can be stored at the timestamps when the label of edge e changes, while the positive number len_t can be stored at other timestamps. In addition, the EL table is irrelevant to the specific k, hence it can be created in preprocessing to fit different settings of k.

We next explain how to utilize the EL table to find the maximum interval containing $[m, m+k−1]$ for edge e. Using the EL table for e, we can get the beginning timestamp of one maximum interval $t = m − len_m + 1$ if $len_m > 0$, and $t = m$ otherwise. Then we know that the maximum interval containing $[m, m + k − 1]$ is $[t, t − len_t − 1]$ (here $len_t < 0$) if $t − len_t − 1 \geq m + k − 1$, and it does not exist otherwise.

We next use an example to show the above process.

Example 4: We consider the temporal graph G in Figure 1(a), frequency threshold $k = 3$, and the current row $m = 3$.
(1) For edge (v_1, v_4), we can check the EL table for (v_1, v_4) in Figure 3, get the beginning timestamp 3 of the maximum interval ($len_3 < 0$), and know that the maximum interval $[t, t − len_t − 1] = [3, 3]$ does not contain $[m, m + k − 1] = [3, 5]$. Hence, no maximum intervals contain $[3, 5]$.
(2) For edge (v_1, v_2), we can check the EL table for (v_1, v_2), get the beginning timestamp 1 $(3 − len_3 + 1)$ of the maximum interval, and know that the maximum interval $[t, t−len_t−1] = [1, 6]$ contains $[3, 5]$, as $1 − len_1 − 1 \geq 5$. □

We use the EL table to find the maximum interval containing $[m, m + k − 1]$ in constant time by the maximum interval containing m, which has better time complexity than IC trees.

Procedure generateMaxTM
Input: R edge sets, array EMaxIntvl, interval $[m, i]$, connected components $CC[i + 1, T]$ for edges in $R[m, i + 1] \cup \cdots \cup R[m, T]$.
Output: Updated connected components $CC[i, T]$.

1. $CC[i, T] := CC[i + 1, T]$;
2. **for each** e **in** $R[m, i]$
3.   **if** e is disconnected from any cc in $CC[i, T]$
4. $G_s :=$ a new cc having edge e only
            with interval EMaxIntvl$[e]$;
5. $CC[i, T] := CC[i, T] \cup \{G_s\}$;   /*case (1)*/
6.   **if** e is connected to only one cc G_s in $CC[i, T]$
7.   Update G_s by adding e with
            interval $G_s.intvl \cap$ EMaxIntvl$[e]$;   /*case (2)*/
8.   **if** e is connected to two ccs G_s and $G_{s'}$ in $CC[i, T]$
9. $G_{ss'} :=$ the cc consisting of $G_s, G_{s'}$ and edge e
            with interval $G_s.intvl \cap G_{s'}.intvl \cap$ EMaxIntvl$[e]$;
10. $CC[i, T] := CC[i, T] \setminus \{G_s, G_{s'}\} \cup \{G_{ss'}\}$;   /*case (3)*/
11. **return** updated $CC[i, T]$.

Figure 5. Procedure generateMaxTM

Computing R **edge sets**. We next show the details of procedure computeRES to compute R edge sets for one row of intervals in TI-Table, shown in Figure 4. It takes as input the EL table or IC trees for temporal graph $G(V, E, 1, T, L)$, frequency threshold k, array EMaxIntvl and row number m, and returns R edge sets for the m-th row intervals and updated EMaxIntvl. Here, array EMaxIntvl is a data structure that dynamically maintains the maximum interval $[m', i']$ found by the EL table or IC trees for each edge and row number m in the T2B scheme. It reduces the cost to compute the R edge sets by avoiding unnecessary use of the EL table or IC trees, and facilitates the generation of non-expandable temporal motifs.

When computing the R edge set to which edge e belongs for the m-th row in TI-Table, there are four cases to consider. Assume that the current EMaxIntvl$[e] = [m', i']$. (1) EMaxIntvl$[e]$ is null, which means that there is no maximum intervals found for rows above m. In this case, we need to use the EL table or IC trees for e to identify the maximum interval $[m, i']$, update EMaxIntvl$[e]$ with $[m, i']$ if it exists, and save e to $R[m, i']$ if $i' \geq m + k - 1$. Note that the beginning timestamp of the new identified maximum interval is m if exists in this case, otherwise it can be identified for the $(m-1)$-th row. (2) $i' < m$, which means that the label of edge e changes at timestamp $i' + 1$, and EMaxIntvl$[e]$ needs to be updated with EL table or IC trees, as the intersection of intervals $[m, m + k - 1]$ and $[m', i']$ is empty. (3) $m \leq i' < m + k - 1$, which means that no maximum intervals containing $[m, m + k - 1]$ exist, as the label of edge e changes at timestamp $i' + 1$. Hence, e does not belong to any R edge sets for the intervals in the m-th row. (4) $i' \geq m + k - 1$, which means that edge e has the same label in $[m, i']$. Hence, edge e belongs to $R[m, i']$. That is, cases (3) and (4) avoid the unnecessary use of the EL table or IC trees by array EMaxIntvl. Note that the array EMaxIntvl can further avoid the unnecessary use of the EL table when no maximum intervals contain $[m, m + k - 1]$, as EMaxIntvl can be updated by the maximum interval containing m.

Procedure computeRES first initializes all R edge sets for intervals in the m-th row to be empty (line 1). For each edge e, it fetches the current EMaxIntvl$[e]$ (line 3). For cases (1) and (2), it uses the EL table and IC tree for e to find the maximum interval $[m, i']$ for row m, in which e keeps the label unchanged (lines 4-5). It updates EMaxIntvl$[e]$ with $[m, i']$ if

$[m, i']$ exists (line 6). If $i' \geq m + k - 1$, it puts e into $R[m, i']$ (lines 7-8). For case (4), it simply puts e into $R[m, i']$ (lines 9-10). For case (3), it does nothing. Finally, R edge sets and updated array EMaxIntvl are returned (line 11).

Complexity. The construction of the EL table and IC trees takes $O(T|E|)$ and $O(|E|(T + \max V_e \log \max V_e))$ time, respectively. The procedure computeRES takes $O(|E|)$ and $O(|E| \log \max V_e)$ time, when using the EL table and IC trees, respectively. Here, $\max V_e$ is the largest $|V_e|$ for all $e \in E$. As for space complexity, the EL table and IC trees take $O(T|E|)$ and $O(|E| \max V_e)$ space, and R edge sets and array EMaxIntvl both take $O(|E|)$ space. Note that if we compute R edge sets for other rows of TI-Table, we can reuse the $O(|E|)$ space for R edge sets and array EMaxIntvl.

C. Generate Maximal Temporal Motifs

We then introduce how to compute the maximal temporal motifs with interval $[m, i]$. As we adopt the T2B and R2L scheme, we already have the following ready when dealing with interval $[m, i]$: (1) the R edge sets for rows from the first to m-th in TI-Table, (2) the array EMaxIntvl of all the edges for the m-th row, and (3) the maximal and non-expandable temporal motifs for all the intervals in rows from the first to $(m - 1)$-th and for all intervals $[m, j]$ ($j \in [i + 1, T]$) in the m-th row. Note that once we generate the maximal temporal motifs for an interval, we immediately check whether they are expandable or not (will be introduced in Section III-D).

By Proposition 2 (Section III-A), we dynamically maintain the connected components (or simply ccs) $CC[i, T]$ for edges in $S[m, i] = \bigcup_{i \leq j \leq T} R[m, j]$, where each cc in $CC[i, T]$ corresponds to a maximal temporal motifs with interval $[m, i]$. When computing the ccs $CC[i, T]$, we only need to add edges in $R[m, i]$ to the ccs $CC[i + 1, T]$.

We next present the details of procedure generateMaxTM shown in Figure 5, which takes as input R edge sets, array EMaxIntvl, interval $[m, i]$ and ccs $CC[i + 1, T]$, and returns ccs $CC[i, T]$ for interval $[m, i]$. It first initializes $CC[i, T]$ to $CC[i + 1, T]$ (line 1). Then it updates $CC[i, T]$ by adding edges in $R[m, i]$ (lines 2-10). For each edge e in $R[m, i]$, it checks the connectivity between e and all ccs in $CC[i, T]$. There are three cases. (1) When e is disconnected from any cc in $CC[i, T]$, it creates a new cc G_s having a single edge e with interval EMaxIntvl$[e]$, and adds G_s to $CC[i, T]$ (lines 3-5). (2) When e is connected to only one cc G_s in $CC[i, T]$, it updates G_s by adding e to G_s and changing its interval with interval $G_s.intvl \cap$ EMaxIntvl$[e]$ (lines 6-7). (3) When e is connected to two ccs G_s and $G_{s'}$, it creates a new cc $G_{ss'}$ by combining e with G_s and $G_{s'}$, with interval $G_s.intvl \cap G_{s'}.intvl \cap$ EMaxIntvl$[e]$, and updates $CC[i, T]$ by replacing G_s and $G_{s'}$ with $G_{ss'}$ (lines 8-10). Finally, it returns ccs $CC[i, T]$ (line 11).

We next illustrate generateMaxTM with an example.

Example 5: We consider the temporal graph G in Figure 1(a) and frequency threshold $k = 4$. Assume that the procedure generateMaxTM executes for row $m = 2$ and has the following ready: $R[2, 6] = \{(v_1, v_2), (v_4, v_5)\}$, $R[2, 5] = \{(v_2, v_3), (v_3, v_5)\}$, and EMaxIntvl$[(v_1, v_2), (v_4, v_5), (v_2, v_3), (v_3, v_5)]$ $= [[1, 6], [1, 6], [1, 5], [2, 5]]$.

Procedure generateExpTM
Input: Connected components $\mathsf{CC}[i,T]$ and interval $[m,i]$.
Output: Set $\mathsf{TF}[m,i]$ of maximal and non-expandable temporal motifs in interval $[m,i]$.

1. $\mathsf{TF}[m,i] := \emptyset$;
2. **for each** newly generated cc G_s
       /*any other cc has already been checked*/
3. $[m',i] := G_s.intvl$;
4.   **if** $m' = m$ **then** /*not left expandable cc */
5. $\mathsf{TF}[m,i] := \mathsf{TF}[m,i] \cup \{G_s\}$;
6. **return** $\mathsf{TF}[m,i]$.

Figure 6. Procedure generateExpTM

(1) For interval $[2,6]$: it updates $\mathsf{CC}[6,6] = \emptyset$ by adding edges in $\mathsf{R}[2,6]$, and generates two ccs $G_{s_1}(\{(v_1,v_2)\})$ and $G_{s_2}(\{(v_4,v_5)\})$ both with interval $[1,6]$ (case (1)).
(2) For interval $[2,5]$: it updates $\mathsf{CC}[5,6] = \{G_{s_1}, G_{s_2}\}$ by adding edges in $\mathsf{R}[2,5]$, combines G_{s_1}, G_{s_2} and edges in $\mathsf{R}[2,5]$, and generates a cc $G_{s_3}(\{(v_1,v_2),(v_4,v_5),(v_2,v_3),(v_3,v_5)\})$ with interval $[1,6] \cap [2,5]=[2,5]$ (case (3)). $\quad\square$

Complexity. Let $|E_m|$ be the number of edges in temporal subgraph $G_s(\mathsf{S}[m,m+k-1])$. Procedure generateMaxTM takes $O(|E_m|)$ time to generate all the maximal temporal motifs for the intervals in the m-th row in the TI-Table. As for space complexity, connected components $\mathsf{CC}[i,T]$ ($i \in [m+k-1,T]$) take $O(|E_m|)$ space.

D. Generate Non-expandable Temporal Motifs

We next introduce how to check whether a maximal temporal motif is expandable or not, once we generate the maximal temporal motifs (*i.e.,* connected components or simply ccs) $\mathsf{CC}[i,T]$ for interval $[m,i]$ using procedure generateMaxTM, with the T2B and R2L scheme. That is, when we generate non-expandable temporal motifs for interval $[m,i]$, we already have maximal and non-expandable temporal motifs in $\mathsf{TF}[m',i']$ for intervals $[m',i']$, where $m' < m$ or ($m' = m$ and $i < i' \le T$).

For a maximal temporal motif $G_s(V_s,E_s,m,i)$ in $\mathsf{CC}[i,T]$, we only need to check whether it is left expandable by Proposition 3 (Section III-A). For each maximal motif $G_s(V_s,E_s,m,i)$, it is associated with an interval $intvl = [m',i']$, which equals to the intersection of the current found maximum interval containing $[m,m+k-1]$, *i.e.,* $\mathsf{EMaxIntvl}[e]$, for each edge e in E_s. Note that the interval $intvl$ is dynamically maintained in procedure generateMaxTM. Hence, each edge e in E_s has the same label in interval $[m',i']$. That is, G_s is non-expandable iff $m' = m$, and is left expandable iff $m' < m$, as $G_s(V_s,E_s,m',i')$ already appears in $\mathsf{TF}[m',i']$.

We next present the details of procedure generateExpTM shown in Figure 6, which takes as input ccs $\mathsf{CC}[i,T]$ and interval $[m,i]$, and returns set $\mathsf{TF}[m,i]$ of maximal and non-expandable temporal motifs in interval $[m,i]$. It first initializes the set $\mathsf{TF}[m,i]$ to empty (line 1), and then checks each newly generated cc G_s in $\mathsf{CC}[i,T] \setminus \mathsf{CC}[i+1,T]$ (line 2), as other ccs have already been checked before. Then it fetches the interval $[m',i']$ of G_s, where $i' = i$ is guaranteed by Proposition 3 (line 3). Only when $m' = m$, the motif G_s is non-expandable, and added to $\mathsf{TF}[m,i]$ (lines 4-5). Finally, it returns the set $\mathsf{TF}[m,i]$ (line 6).

Note that we do not remove any connected components when their corresponding motifs are left expandable, as their corresponding motifs might become non-expandable after updating connected components in $\mathsf{CC}[i',T]$ for $i' < i$.

We next illustrate generateExpTM with an example.

Example 6: We consider the same setting as Example 5. Assume that generateExpTM executes for row $m = 2$.
(1) For interval $[2,6]$: $\mathsf{CC}[6,6] = \{G_{s_1}(\{(v_1,v_2)\}), G_{s_2}(\{(v_4,v_5)\})\}$ both with interval $[1,6]$ (left expandable as $1 < m$).
(2) For interval $[2,5]$: $\mathsf{CC}[5,6] = \{G_{s_3}(\{(v_1,v_2),(v_4,v_5),(v_2,v_3),(v_3,v_5)\})$ with interval $[2,5]\}$. G_{s_3} is non-expandable ($2 = m$) and belongs to $\mathsf{TF}[2,5]$ (H_1 in Figure 1(b)). $\quad\square$

Complexity. Procedure generateExpTM takes $O(|E_m|)$ time to generate all the non-expandable temporal motifs for the intervals in the m-th row of the TI-Table.

E. The Complete Algorithm

Based on the previous three procedures, we finally show our algorithm FTM, which takes as input temporal graph $G(V,E,1,T,L)$ and frequency threshold k, and returns set TF of all the maximal and non-expandable temporal motifs.

The algorithm FTM first creates the EL table or IC trees, and initializes the array EMaxIntvl to be empty. Then it adopts the T2B scheme to deal with all the rows m from 1 to $T - k + 1$. For each row m, (1) it invokes computeRES to obtain the edge set $\mathsf{R}[m,i]$ for $m + k - 1 \le i \le T$ and the updated array EMaxIntvl. (2) Then it initializes ccs $\mathsf{CC}[i,T]$ for $i \in [m+k-1,T+1]$. Note that we use $\mathsf{CC}[T+1,T]$ to represent an empty set for convenience. (3) After that, it adopts the R2L scheme with columns i from T to $m+k-1$. For each i, (a) it invokes generateMaxTM to obtain the updated ccs $\mathsf{CC}[i,T]$ for interval $[m,i]$. Each cc in $\mathsf{CC}[i,T]$ corresponds to a maximal temporal motif. (b) Then it invokes generateExpTM to obtain the set $\mathsf{TF}[m,i]$ of maximal and non-expandable temporal motifs with interval $[m,i]$. Finally, it returns the set TF.

Proposition 4: FTM *correctly finds all the maximal and non-expandable temporal motifs.* $\quad\square$

Proof: We show this by a loop invariant.

Loop invariant: before the start of each row m, algorithm FTM finds all the maximal and non-expandable temporal motifs for all the intervals in rows from 1 to $(m-1)$ in the TI-Table shown in Figure 2.

For row $m = 1$, it is easy to verify that the loop invariant holds. Assume that the loop invariant holds for row $m > 1$, and we show that the loop invariant holds for row $m + 1$.

(1) By Proposition 2, procedure generateMaxTM correctly generates all maximal temporal motifs for each interval at the $(m+1)$-th row in the TI-Table. The frequency threshold k is assured from the arrangement of the TI-Table. (2) By Proposition 3, procedure generateExpTM correctly checks whether a maximal temporal motif is left expandable or not, and generates maximal and non-expandable temporal motifs. This guarantees that FTM correctly finds the maximal and non-expandable temporal motifs for all intervals in the $(m+1)$-th row. This shows that the loop invariant holds for row $m+1$.

Putting these together, and we have the conclusion. $\quad\square$

We next illustrate algorithm FTM with an example.

(a) Updated temporal graph G' with interval $[1,7]$

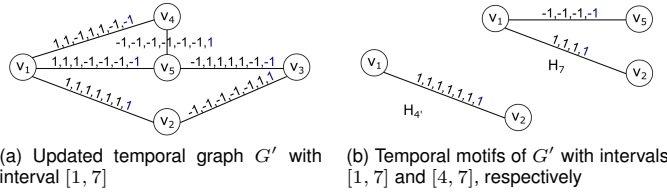(b) Temporal motifs of G' with intervals $[1,7]$ and $[4,7]$, respectively

Figure 7. Running example for incremental algorithm

Example 7: We consider the same setting as Example 5. The corresponding TI-Table consists of 6 intervals in total, *i.e.*, $\{([1,4],[1,5],[1,6]),([2,5],[2,6]),([3,6])\}$. Assume that FTM executes for row $m=2$.

The algorithm invokes computeRES and returns $R[2,6] = \{(v_1,v_2),(v_4,v_5)\}$ and $R[2,5] = \{(v_2,v_3),(v_3,v_5)\}$, and updates $EMaxIntvl[(v_3,v_5)] = [2,5]$. (a) For $i=6$: G_{s_1} ($\{(v_1,v_2)\}$) and $G_{s_2}(\{(v_4,v_5)\})$ with interval $[1,6]$ are generated, which are left expandable ($1<m$). (b) For $i=5$: generateMaxTM combines G_{s_1}, G_{s_2} and edges from $R[2,5]$ to generate a cc G_{s_3} with interval $[2,5]$. generateExpTM adds the cc G_{s_3} to TF as $m=2$ (H_1 in Figure 1(b)).

Finally, H_1 is one of the maximal and non-expandable temporal motifs in set TF returned. □

Complexity. For convenience, we denote the algorithm FTM using the EL table and IC trees as FTM-EL and FTM-IC, respectively. Algorithms FTM-EL and FTM-IC take $O((T-k)|E|)$ and $O((T-k)|E|\log\max V_e)$ time, respectively. Note that it only needs this time to obtain set TF, though it takes $O((T-k)^2|E|)$ worst-case time to output all the temporal motifs from set TF. Algorithms FTM-EL and FTM-IC take $O((T-k)|E|+(T-k)\max E_m)$ and $O(\max V_e|E|+(T-k)\max E_m)$ space, which is dominated by data structures IC trees, the EL table and set TF. Here, $\max E_m$ is the largest number $|E_m|$ for $m\in[1,T-k+1]$.

Remarks. The above analysis reveals that algorithm FTM-EL has a better time complexity than FTM-IC, while the latter has a better space complexity than the former. Further, both the time and space complexities of FTM-EL and FTM-IC are irrelevant to the number of edge labels. By Proposition 4 and the complexity analysis, we have proved Theorem 1. Finally, our method can extend to find temporal motifs with (or without) specific edge labels or intervals, as it can check the types of edge labels or compute specific R edge sets.

IV. INCREMENTAL ALGORITHM

Temporal networks are naturally dynamic and updated continuously. In this section, we present an incremental method to efficiently find all the maximal and non-expandable temporal motifs, given temporal graph $G(V,E,1,T+\Delta T,L)$, frequency threshold k, and set TF of all the maximal and non-expandable temporal motifs for $G(V,E,1,T,L)$. That is, temporal graph $G(V,E,1,T,L)$ evolves with ΔT new graph snapshots. The main result is stated below.

Theorem 2: *Given temporal graph $G(V,E,1,T+\Delta T,L)$, frequency threshold k and set TF of maximal and non-expandable temporal motifs for $G(V,E,1,T,L)$, there is an algorithm that finds set TF^+ of updated maximal and non-expandable temporal motifs in $O((T-k)\max E_{TF}+\Delta T|E|)$*

time. Here $\max E_{TF}$ is the largest edge number in $TF[m,T]$ for $m\in[1,T-k+1]$ ($\max E_{TF}<|E|$ in practice). □

We now illustrate how to incrementally maintain the EL table only (the incremental maintenance of IC trees is omitted, due to space limitations), which can be incrementally maintained in $O(\Delta T|E|)$ time. We can store the label lab_t and the number len_t for each edge and timestamp t in $[T+1,T+\Delta T]$, and update the number len_t at the beginning timestamp of the last maximal interval in $[1,T]$ for each edge. Consider the updated temporal graph G' in Figure 7(a) of temporal graph G in Figure 1(a) with $T=6$ and $\Delta T=1$. For the EL table of edge (v_1,v_4) in Figure 3, we need to store the label $lab_7=-1$ and the number $len_7=2$, and update the number $len_6=-2$.

We next build the connections between TF and TF^+, which is the design basis of our incremental algorithm.

Proposition 5: $TF^+[m,i]=TF[m,i]$ *for all $m\in[1,T-k]$ and $i\in[m+k-1,T-1]$.* □

Proof: We show this by proof by induction.

(1) For row $m=1$, some edges of any temporal motif G_s in $TF[1,T-1]$ must change their labels at the timestamp T, hence G_s is not right expandable. They are obviously not left expandable as $m=1$. It is the same for $i\in[k,T-2]$. As sets $S[1,T]$ and $R[1,T-1]$ keep unchanged after updates, the conclusion holds for $m=1$.

(2) Assume that the conclusion holds for row $m>1$, we then show that it holds for row $m+1$. For row $m+1$, similarly any temporal motif G_s in $TF[m+1,T-1]$ is non-expandable, as the maximal and non-expandable temporal motifs keep unchanged for the rows above $m+1$. It is the same for $i\in[m+k,T-2]$. As sets $S[m+1,T]$ and $R[m+1,T-1]$ keep unchanged after updates, the conclusion holds for $m+1$.

Putting these together, we have the conclusion. □

Proposition 5 reveals that the maximal and non-expandable temporal motifs for intervals $[m,i]$ ($m\in[1,T-k]$ and $i\in[m+k-1,T-1]$) are unaffected after updates.

Proposition 6: *We only need to use the edges in $TF[m,T]$ instead of $S[m,T]$ to compute the maximal and non-expandable motifs for intervals $[m,i]$ when $m\in[1,T-k+1]$ and $i\in[T,T+\Delta T]$.* □

Proof: It is proved by showing that for $m\in[1,T-k+1]$, if edge $e\in S[m,T]$ and $e\notin TF[m,T]$, then $e\notin TF^+[m,i]$ for $i\geq T$. It is obvious that the edges in $TF^+[m,i]$ must belong to $S[m,T]$. Assume that edge $e\in TF^+[m,i]$ for $i\geq T$. As $e\in TF^+[m,i]$, e belongs to a maximal and non-expandable motif in $TF^+[m,i]$ that consists of edges $e,\cdots,e_h$ ($h\geq 0$). Then we know $e,\cdots,e_h\in S[m,i]\subseteq S[m,T]$. It implies that $e,\cdots,e_h$ belong to a maximal and non-expandable motif in $TF[m,T]$. Therefore, edge $e\in TF[m,T]$, which is a contradiction.

Putting these together, we have the conclusion. □

Proposition 6 reveals that we can further reduce the computation cost by utilizing the set of edges in $TF[m,T]$ ($m\in[1,T-k+1]$), which is a subset of $S[m,T]$. In addition, we can slightly change the procedure computeRES by replacing E with $TF[m,T]$ in line 2 of Figure 4.

Algorithm DFTM. Based on Propositions 5 & 6, now we can easily design our incremental algorithm DFTM by slightly revising our static algorithm FTM. Similar to FTM, we denote

the algorithm DFTM using the EL table and IC trees as DFTM-EL and DFTM-IC, respectively.

(1) For row $m \in [1, T - k]$, DFTM simply assigns $\mathsf{TF}^+[m, i] = \mathsf{TF}[m, i]$ for $i \in [m + k - 1, T - 1]$ to maintain the unchanged temporal motifs by Proposition 5.

(2) For row $m \in [1, T - k + 1]$, DFTM is essentially the same as the static algorithm FTM, except that it only deals with intervals $[m, i]$ for $i \in [T, T + \Delta T]$, and only the edges in $\mathsf{TF}[m, T]$ are considered when computing edge set $\mathsf{R}[m, i]$ in procedure computeRES by Propositions 5 & 6.

(3) For row $m \in [T - k + 2, T + \Delta T - k + 1]$, DFTM handles the intervals $[m, i]$ for $i \in [m + k - 1, T + \Delta T]$ along the same lines as FTM.

We illustrate algorithm DFTM with an example.

Example 8: Consider the updated temporal graph G' in Figure 7(a) of graph G in Figure 1(a) with $T = 6$ and $\Delta T = 1$, frequency threshold $k = 4$, and set TF of maximal and non-expandable temporal motifs for G computed by FTM. The updated TI-Table consists of 10 intervals, *i.e.,* $\{([1, 4], [1, 5], [1, 6], [1, 7]), ([2, 5], [2, 6], [2, 7]), ([3, 6], [3, 7]), ([4, 7])\}$.

(1) For row $m \in [1, 2]$, $\mathsf{TF}^+[m, i] = \mathsf{TF}[m, i]$ for $i \in [m + 3, 5]$, *i.e.,* the maximal and non-expandable temporal motifs keep unchanged for intervals $\{[1, 4], [1, 5], [2, 5]\}$, such as H_1.

(2) For row $m \in [1, 3]$ and $i \in [6, 7]$, sets $\mathsf{TF}[m, 6]$ are used to compute edge sets $\mathsf{R}[m, 6]$ and $\mathsf{R}[m, 7]$. Then DFTM obtains maximal and non-expandable temporal motifs by $\mathsf{R}[m, 6]$ and $\mathsf{R}[m, 7]$. As shown in Figure 7(b), $H_{4'}$ with interval $[1, 7]$ belongs to TF^+. Note that (v_1, v_2) belongs to a temporal motif in $\mathsf{TF}[1, 6]$ and is right expandable after updates.

(3) For row $m = 4$, DFTM finds maximal and non-expandable temporal motif H_7 with interval $[4, 7]$ in Figure 7(b).

Finally, $H_1, H_{4'}$ & H_7 belong to the set TF^+ returned. □

Correctness. The correctness of algorithm DFTM follows easily from Propositions 5, 6 & 4.

Complexity. Incremental algorithms DFTM-EL and DFTM-IC take $O((T - k) \max E_{TF} + \Delta T|E|)$ and $O(((T - k) \max E_{TF} + \Delta T|E|) \log \max V_e)$ time, respectively. As algorithms FTM-EL and FTM-IC take $O((T + \Delta T - k)|E|)$ and $O((T + \Delta T - k)|E| \log \max V_e)$ time, respectively, incremental algorithms are better than their static counterparts. The only extra cost in the space complexity of DFTM is $\mathsf{TF}[m, T]$ for $m \in [1, T - k + 1]$, which is $O((T - k) \max E_{TF})$, as static algorithms FTM-EL and FTM-IC take $O((T + \Delta T)|E| + (T + \Delta T - k) \max E_m)$ and $O(\max V_e|E| + (T + \Delta T - k) \max E_m)$ space, respectively, for $G(V, E, 1, T + \Delta T)$. Hence, incremental algorithms have the same space complexities as their static counterparts ($\max E_{TF} < |E|$).

Remarks. (1) We have proved Theorem 2 following the above correctness and complexity analyses. (2) The incremental algorithm has theoretically better time complexity and the same space complexity with a reasonable extra space cost, compared with its static counterpart, which supports temporal networks evolving with continuous updates.

V. EXPERIMENTAL STUDY

Using real-life and synthetic datasets, we conducted an extensive experiment of static algorithms FTM-EL and FTM-IC, and incremental algorithms DFTM-EL and DFTM-IC.

A. Experimental Settings

Datasets. We gathered four datasets.

(1) BJDATA [20] is a real-life dataset that records three road traffic conditions (*i.e.,* congested, slow and fast) in Beijing. We extracted the day level data (Nov. 4, 2010) with 288 snapshots (5 minutes a snapshot), $69,416$ nodes and $88,396$ edges, *i.e.,* $19,991,808$ nodes and $25,458,048$ edges in total.

(2) EURDATA [40] is a real-life dataset for a renewable European electric power system. Each edge represents a transmission line, and each node represents a merging point of transmission lines with a dynamic energy demand signal. We transformed the graph by switching edges to nodes and nodes to edges, and used 10 labels to discretize energy demand signals (1-10: from lowest to highest, such as '1' means '$0 \sim 200$ MWh'). The dataset has $26,304$ snapshots (1 hour a snapshot), $2,706$ nodes and $7,334$ edges, *i.e.,* $71,178,624$ nodes and $192,913,536$ edges in total.

(3) LANDATA [56] is a real-life computer event dataset from the Los Alamos National Laboratory enterprise network. Each node represents a computer, and each edge represents the communication of two computers with properties of the start time, the duration, and the number of bytes sent. We used 6 labels to discretize the number of bytes sent on average (0: the virtual label means no bytes sent, 1-5: from lowest to highest, such as '1' means '< 4 KB'). We transformed five-day data (Day 3-7) to the network with 1440 snapshots (5 minutes a snapshot), $155,252$ nodes and $359,450$ edges, *i.e.,* $223,562,880$ nodes and $517,608,000$ edges in total.

(4) SYNDATA [13] is produced by the synthetic data generator. All edge labels are -1 at first, and then some of them are activated (transformed to 1) at random. Each activated edge activates its neighboring edges and its copy in the next snapshot with decayed probabilities np_r and tp_r respectively. The process stops when the percentage of activated edges reaches the activation density ad_r. We fixed tp_r, ad_r and np_r to 0.9, 0.3 and 0.3, respectively, by default. We generated a dataset with $2,000$ snapshots, $400,000$ nodes and $800,000$ edges, *i.e.,* 0.8 billion nodes and 1.6 billion edges in total.

Implementation. All algorithms were implemented with C++. Experiments were conducted on a PC with 2 Intel Xeon E5-2640 2.6GHz CPUs, 64GB RAM, and a Windows 7 operating system. All tests were repeated over 3 times and the average is reported. Algorithm codes and datasets are available at https://github.com/CodesandData/FTM.

B. Experimental Results

We next present our findings. Note that the definitions of temporal networks and motifs in previous studies are different from ours. Temporal motifs in [22]–[34], [43]–[47] have time-dependent edges, *i.e.,* all edges appear at part of timestamps in motifs. However, our temporal motifs need edges appearing at all the timestamps. Temporal motifs in [35]–[38] are somehow similar to ours, as their nodes or edges of motifs appear at all the timestamps of the motif interval. However, [35]–[38] all limit motifs with induced subgraphs, and enumerate
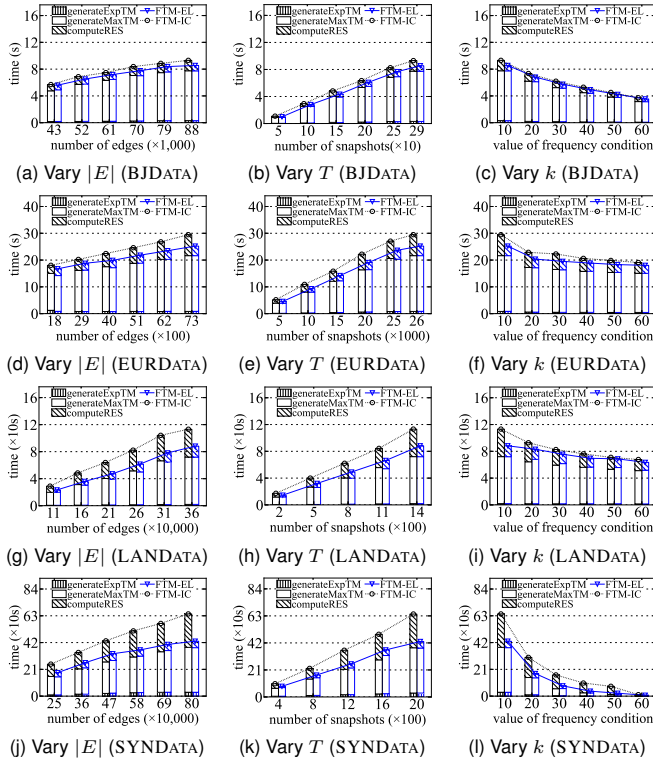
Figure 8. Test of the static algorithm FTM

Table II
PARAMETERS AND SPACE COSTS OF THE STATIC ALGORITHMS ($k = 10$)

Datasets	Average $|E_m|$	maxE_m	Memory cost (GB)	
			FTM-EL	FTM-IC
BJDATA (457 MB)	56,749	74,729	0.423	0.347
EURDATA (3.15 GB)	4,966	6,636	2.438	1.472
LANDATA (8.83 GB)	253,124	276,594	7.094	5.736
SYNDATA (30.60 GB)	195,377	275,813	22.064	27.328

EURDATA, LANDATA, SYNDATA), respectively. Hence, the time difference of FTM-EL and FTM-IC is mainly caused by procedure computeRES.

Exp-1.2. To evaluate the impacts of the number T of snapshots, we varied T from 50 to 288 for BJDATA, from 5,000 to 26,304 for EURDATA, from 288 to 1,440 for LANDATA, and from 400 to 2,000 for SYNDATA, respectively. We used the entire graphs for all the datasets, and fixed $k = 10$. The results are reported in Figures 8b & 8e & 8h & 8k.

When varying the number T of timestamps, the running time of both FTM-EL and FTM-IC increases with the increment of T, and FTM-EL is faster than baseline FTM-IC. This is consistent with the time complexity analysis. Moreover, FTM-EL is on average (1.05, 1.15, 1.25, 1.36) times faster than baseline FTM-IC on (BJDATA, EURDATA, LANDATA, SYNDATA), respectively. It reveals that the larger T is, the faster FTM-EL is.

When varying the number T of timestamps, the running time of all three procedures in FTM increases with the increment of T. Procedure generateMaxTM takes the most time, on average (87%, 83%, 80%, 81%) for FTM-EL on (BJDATA, EURDATA, LANDATA, SYNDATA), respectively. Procedure computeRES using the EL table is on average (1.28, 1.68, 2.35, 4.83) times faster than using IC trees on (BJDATA, EURDATA, LANDATA, SYNDATA), respectively.

Exp-1.3. To evaluate the impacts of frequency threshold k, we varied k from 10 to 60, fixed $T = (288, 26304, 1440, 2000)$ for (BJDATA, EURDATA, LANDATA, SYNDATA), respectively, and used the entire temporal graphs for all the datasets. The result is reported in Figures 8c & 8f & 8i & 8l.

When varying the frequency threshold k, the running time of both FTM-EL and FTM-IC decreases with the increment of k. The reason is that, there are fewer edges having the same labels in intervals and fewer temporal motifs for larger k. Moreover, FTM-EL is on average (1.07, 1.11, 1.11, 2.44) times faster than baseline FTM-IC on (BJDATA, EURDATA, LANDATA, SYNDATA), respectively.

When varying the frequency threshold k, the running time of all three procedures in FTM decreases with the increment of k. Procedure generateMaxTM takes the most time, on average (87%, 82%, 79%, 69%) for FTM-EL, respectively. Procedure computeRES using the EL table is on average (1.64, 1.78, 1.75, 5.38) times faster than using IC trees on (BJDATA, EURDATA, LANDATA, SYNDATA), respectively.

Exp-1.4. To evaluate the space cost, we tested the maximum and average value of $|E_m|$ and memory cost in practice (using function GetProcessMemoryInfo), along the same setting as Exp-1.3, while fixed $k = 10$. The result is reported in Table II. Note that $|E_m|$ is the number of edges in temporal subgraph $G_s(\mathsf{S}[m, m + k - 1])$, and max$E_m$ is the largest $|E_m|$.

any induced subgraphs by traversing all the nodes recursively. That is, enumeration of all the subgraphs to find our temporal motifs takes exponential time. Therefore, their methods are not suitable to find our temporal motifs, and there is essentially no need to compare these methods with our algorithms.

Exp-1: Tests of the static algorithm. In the first set of tests, we test the running time and space cost of the static algorithm FTM *w.r.t.* the graph sizes in terms of the number $|E|$ of edges, the number T of snapshots, and the frequency threshold k. The number of labels is not needed by the complexity analyses.

Exp-1.1. To evaluate the impacts of the graph size in terms of the number $|E|$ of edges, we varied $|E|$ from 43,000 to 88,396 for BJDATA, from 1,800 to 7,334 for EURDATA, from 110,000 to 359,450 for LANDATA, and from 250,000 to 800,000 for SYNDATA, respectively. We fixed $k = 10$ and randomly selected edges by fixing $|V| = (69416, 2706, 155252, 400000)$ for (BJDATA, EURDATA, LANDATA, SYNDATA), respectively. The results are reported in Figures 8a & 8d & 8g & 8j.

When varying the number $|E|$ of edges, the running time of FTM-EL and FTM-IC increases with the increment of $|E|$, which is consistent with the complexity analysis. Moreover, FTM-EL is (1.06, 1.12, 1.26, 1.36) times faster than baseline FTM-IC on (BJDATA, EURDATA, LANDATA, SYNDATA) on average, respectively.

When varying the number $|E|$ of edges, the running time of three procedures in FTM all increases with the increment of $|E|$. Procedure generateMaxTM takes the most time, on average (86%, 82%, 80%, 79%) for FTM-EL on (BJDATA, EURDATA, LANDATA, SYNDATA), respectively. Procedure computeRES using the EL table is on average (1.64, 1.79, 2.59, 4.09) times faster than using IC trees on (BJDATA,

$|E_m|$ is less than $|E|$ to various degrees, as edge labels in different datasets change with different frequencies. $|E|$ is (1.18, 1.11, 1.30, 2.90) times of $\max E_m$ on (BJDATA, EURDATA, LANDATA, SYNDATA), respectively. Moreover, FTM-EL uses 19% less memory than baseline FTM-IC on SYNDATA, but uses ($22\%, 66\%, 24\%$) more memory than FTM-IC on (BJDATA, EURDATA, LANDATA), respectively. The reason is that the duration of labels keeping constant in SYNDATA is less than other datasets, which results in more tree nodes of the IC trees. These are consistent with the space complexity analysis. Finally, the memory cost of algorithms is rational, as SYNDATA is large enough.

Exp-2: Tests of the incremental algorithm. In the second set of tests, we test the running time and memory cost of incremental algorithms DFTM-EL and baseline DFTM-IC *w.r.t.* the number ΔT of increased timestamps and the frequency threshold k, compared with our static algorithms FTM-EL and FTM-IC. Note that the incremental maintenance time of query answers (*i.e.,* the motif set TF) is considered in algorithms, and the incremental maintenance time of the data structures (*i.e.,* the EL table or IC trees) is separately reported in the supplementary material due to space limitations.

Exp-2.1. To evaluate the impacts of the number ΔT of increased timestamps, we used the entire graphs and varied ΔT from 25% to 150% for all the datasets, while fixed $k = 20$. We fixed the original timestamps $T = (112, 10400, 576, 800)$ for (BJDATA, EURDATA, LANDATA, SYNDATA), respectively. The results are reported in Figures 9a-9d.

When varying the number ΔT, the running time of all four algorithms increases with the increment of ΔT. Moreover, DFTM-EL is always the best, and incremental algorithms are always faster than their static counterparts. DFTM-EL is (2.60, 2.25, 1.75, 2.55) times faster than FTM-EL, and DFTM-IC is (2.47, 2.32, 1.85, 2.38) times faster than FTM-IC on (BJDATA, EURDATA, LANDATA, SYNDATA) on average, respectively. These are consistent with the time complexity analysis.

Exp-2.2. To evaluate the impacts of frequency threshold k, we varied k from 10 to 60, while fixed $\Delta T = 150\%$ and the same T setting as Exp-2.1. The results are reported in Figures 9e-9h.

When varying k, the running time of all algorithms decreases with the increment of k, along the same reason as Exp-1.3. Moreover, incremental algorithms are faster than their static counterparts, and DFTM-EL is always the best. DFTM-EL is (1.58, 1.56, 1.34, 1.53) times faster than FTM-EL, and DFTM-IC is (1.63, 1.57, 1.40, 1.54) times faster than FTM-IC on (BJDATA, EURDATA, LANDATA, SYNDATA) on average, respectively.

Exp-2.3. To evaluate the space cost, we tested the maximum and average value of $|E_{m,TF}|$ and $|S[m,T]|$ and the memory cost in practice, along the same setting as Exp-2.1, while fixed $\Delta T = 150\%$ and $k = 10$. The result is reported in Table III. Note that $|E_{m,TF}|$ is the number of edges in TF$[m,T]$, and $\max E_{TF}$ is the largest $|E_{m,TF}|$.

Parameter $|E_{m,TF}|$ is less than $|S[m,T]|$ to various degrees, and the maximum $|S[m,T]|$ is (1.44, 1.08, 1.00, 1.24) times larger than $\max E_{TF}$ on (BJDATA, EURDATA, LANDATA, SYNDATA), respectively. The average of $|S[m,T]|$ is (5.14, 364.71, 1.01, 2.41) times larger than the average of $|E_{m,TF}|$
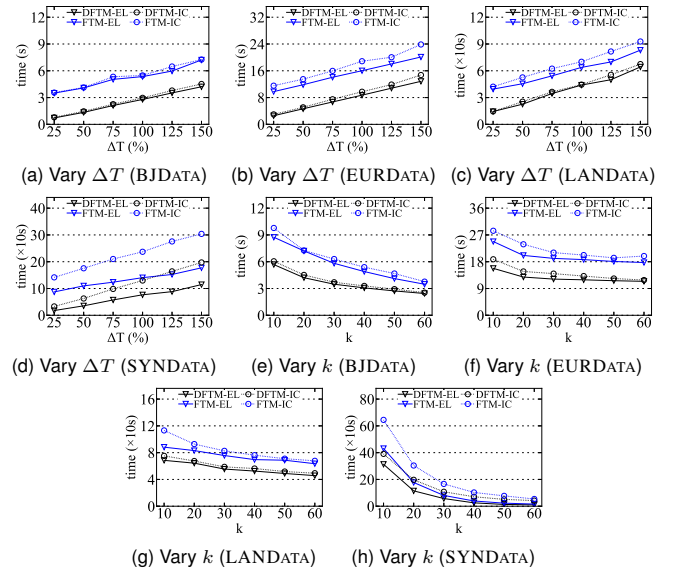

Figure 9. Test of the incremental algorithm DFTM

on (BJDATA, EURDATA, LANDATA, SYNDATA), respectively. It justifies the use of TF$[m,T]$ instead of S$[m,T]$ for $1 \leq m \leq T - k + 1$. Moreover, DFTM-EL uses 19% less memory than baseline DFTM-IC on SYNDATA, but uses ($22\%, 72\%, 11\%$) more memory than DFTM-IC on (BJDATA, EURDATA, LANDATA), respectively. The reason is along the same lines as Exp-1.4. Finally, the memory cost of incremental algorithms is comparable to their static counterparts, where the former uses less memory than the latter, as the computation for rows $m \in [1, T - k + 1]$ only considers intervals $[T, T + \Delta T]$, and may cause less memory for generateMaxTM.

Exp-3. Case studies of temporal motifs. To justify the practical usability of our temporal motifs, we conducted case studies on BJDATA, EURDATA and LANDATA. The following shows three cases from the result of FTM with $k = 10$.

Case 1. For BJDATA, we used the traffic data management system from [19] to visualize our temporal motifs and found corresponding roads in Google Maps. The red color is used to represent edges with 'congested' traffic condition labels. We use our temporal motifs to discover long-time congestion patterns with all congested roads, which can be regarded as a kind of spatio-temporal congestion patterns [50], [51]. As Figure 10(a) depicts, the motif corresponds to the traffic conditions in the *Shichahai scenic area* during [14:42, 18:17]. Observe that there are many restaurants or shops along those roads, which indicates the congestion may be caused by these hot spots or popular areas for tourists in the afternoon.

Case 2. For EURDATA, we transformed the obtained temporal motifs to their original graphs, hence dynamic properties in this case are on nodes, instead of edges. The dark green, green, and light green colors are used to represent energy demand signals '$800 \sim 1000$ MWh', '$400 \sim 600$ MWh', and '$200 \sim 400$ MWh', respectively. We use our temporal motifs to discover hybrid energy consumption patterns. As Figure 10(b) depicts, the motif corresponds to 9 merging points of transmission lines and their links in Italy during [6:00, 20:00] on 20/07/2013. It reveals the area with the staggering electricity usage in summer, as each merging point needs more

Table III
SPACE COSTS OF THE INCREMENTAL ALGORITHMS ($k = 10$ AND $\Delta T = 150\%$)

Datasets	Average $|E_{m,TF}|$	$\max E_{TF}$	Average $|S[m,T]|$	Max. $|S[m,T]|$	Memory cost (GB)	
					(DFTM-EL, FTM-EL)	(DFTM-IC, FTM-IC)
BJDATA (457 MB)	5,534	35,224	28,437	50,761	(0.407, 0.410)	(0.334, 0.337)
EURDATA (3.15 GB)	7	4,887	2,553	5,254	(2.273, 2.411)	(1.318, 1.456)
LANDATA (8.83 GB)	54,205	191,366	54,522	191,377	(7.092, 7.094)	(5.735, 5.736)
SYNDATA (30.60 GB)	1,225	159,525	2,953	197,092	(22.058, 22.064)	(27.318, 27.328)

than 200 MWh energy in 14 hours (the average energy needed in EURDATA is only about 180 MWh). Some proper strategies can be adopted to maintain good load balance, such as energy transmission from the partinic region to the partanna region.

Case 3. For LANDATA, we discover temporal motifs with frequent byte communications. The red color is used to represent the highest level of bytes sent (_i.e.,_ more than 16 KB sent in 5 miniutes, only 3% number of labels reach this level). We use our temporal motifs to discover patterns with large numbers of bytes sent on the enterprise network. As Figure 10(c) depicts, the motif corresponds to 10 computers and their communications during $[11:10, 11:55]$ on Day 3. This reveals the long-time working scheme of the computers. Some proper strategies can be adopted to achieve good computer management, such as adding backup computers to prevent communication failures (_e.g.,_ backups for Comp 210831).

Exp-4. Comparison with persistent community search. As discussed before, there essentially exist no suitable methods that solve our problem. However, we take a try of algorithm BB-All that searches the largest maximal persistent community (PC) [38] called (θ, τ)-persistent $\widetilde{k}$-core, which is a maximal induced subgraph that keeps the $\widetilde{k}$-core structures in any θ-length subinterval of all persistent intervals, and its core persistence (aggregate the length of all the maximal persistent intervals) is $\geq \tau$ (please refer to [38] for more details). We extended algorithm BB-All to obtain all the PCs as suggested by [38], and transformed the input graph by removing those edges labelled with ('fast', $0 \sim 200$ MWh, $0, -1$) on (BJDATA, EURDATA, LANDATA, SYNDATA), respectively, as BB-All does not consider labels and cannot handle fixed graph structures. We fixed $\widetilde{k} = 1$, $\theta = 1$ (_i.e.,_ keep the 1-core structure at all the timestamps of all the persistent intervals), and $\tau = 10$ (the same as $k = 10$ in our algorithm if the PC has only one persistent interval).

The running time of BB-All on BJDATA, EURDATA and LANDATA all exceeds one hour, which is much slower than our algorithm (10s on BJDATA, 36s on EURDATA, and 101s on LANDATA), and BB-All runs for out of memory on SYNDATA. The reason is that the algorithm has at most $O(2^{\widetilde{n}})$ search subspaces [38], where $\widetilde{n}$ denotes the size of the largest connected component of the temporal graph, which is typically large in graphs. In addition, in order to enumerate all PCs, some of its pruning rules have to be abandoned.

We next analyze the result that BB-All finds in one hour, _i.e.,_ the percentage of valid persistent intervals (the length is large enough to show its persistence, here we set ≥ 5 is valid). The percentage of valid persistent intervals is only (34.51%, 67.88%, 1.87%) on (BJDATA, EURDATA, LANDATA), respectively. It means each PC has a large number of short-length persistent intervals. The reason is that the parameter τ
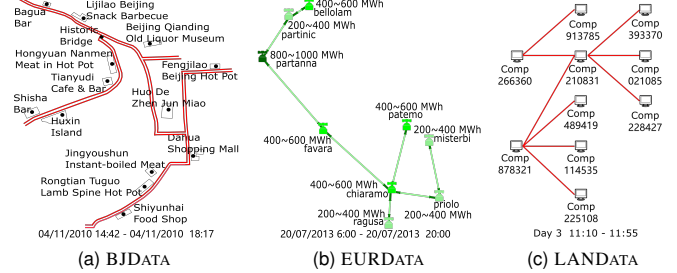

Figure 10. Case studies on real-life datasets

only restricts the sum of all the interval length. Not to mention that BB-All does not consider edge labels.

This shows that existing methods, including BB-All, are not suitable to identify our temporal motifs.

Summary. We have following findings. (1) Static algorithms FTM-EL and FTM-IC both run fast on large temporal graphs. (2) FTM-EL is on average $(1.06, 1.13, 1.25, 1.75)$ times faster than our baseline FTM-IC on (BJDATA, EURDATA, LANDATA, SYNDATA), respectively. (3) Incremental algorithms DFTM-EL and DFTM-IC are $(2.09, 1.91, 1.55, 2.04)$ and $(2.04, 1.94, 1.63, 1.94)$ times faster than their static counterparts on (BJDATA, EURDATA, LANDATA, SYNDATA), respectively. (4) The space cost of incremental algorithms is comparable to their static counterparts, and the former may even use less memory than the latter in practice. (5) Our proposed temporal motifs are useful for practical applications. (6) The comparison with persistent community search [38] further justifies that existing methods are not suitable to identify our temporal motifs.

VI. RELATED WORK

Network motifs in static networks. Network motifs, which mean the recurrent and statistically significant subgraphs or patterns of a large graph, have been applied to various fields, _e.g.,_ biological networks [2]–[6], social networks [4]–[9], electrical circuits [10], software architectures [11], and transport networks [5], [6]. These focus on network motifs in static networks, while we focus on network motifs in temporal networks in this study.

Network motifs in temporal networks. Network motifs in temporal networks have attracted growing attentions. These studies re-define temporal motifs to meet application demands. Most studies focus on temporal motifs with time-dependent edges. [21] firstly studies the dynamics of motifs in networks with time-dependent edges. [22], [23], [26], [30] impose a partial time order and a local time window on edges in a temporal motif, while [25] imposes a total time order and a local time window with a duration bound on edges. [27], [29] impose a total time order and a global time window on

edges in a temporal motif, where [29] further introduces a flow bound on edges. Based on the definition of [27], four sampling approaches [43]–[46] are designed, and [47] further designs a parallel approach. [32] imposes both a local time window and a global time window on edges in a temporal motif. [24] focuses on the graph matching problem and imposes a partial time order and a global time window on its query graph. Some existing studies focus on the specific structure motifs, such as temporal triangles [31], temporal circles [28], temporal butterflies $((2, 2)$-bicliques) [34] and temporal hypergraphs [33]. A recent survey [42] systematically evaluates the impact of temporal inducedness and timing constraints (*i.e.,* time windows) in some of these temporal motifs. Other studies focus on temporal motifs with nodes, edges or their weights (labels) displaying statics or similar dynamics in a user-defined period. [35], [36] define motifs as induced subgraphs with node weights evolving in a consistent trend over a period. [37] defines induced relational states as induced subgraphs with edge labels and directions keeping unchanged over a period. [38] defines persistent communities as induced subgraphs that keep the k-core structures in any given length subintervals of all the persistent intervals.

These studies cannot handle fixed graph structures, and most these works focus on small motifs. The discovery of these motifs is computationally intractable, and no incremental solutions have been investigated to handle the dynamic nature of temporal networks. Different temporal motif definitions are needed for different temporal networks due to the various needs of applications. That is, to properly reinterpret the recurrent and statistically significant nature of motifs in temporal networks is needed in the first place. This is the first study on temporal motifs for an important class of temporal networks, where nodes and edges are fixed, but edge labels vary regularly with timestamps, that can be computed efficiently in polynomial time. We also develop the incremental algorithm to handle the dynamic nature of temporal networks.

Similar but different concepts. There are also studies on graphs that bear similarities but are different from motifs, such as persistent connected components [57], lasting dense subgraphs [58], graph association rules [59], graph temporal association rules [60], dense temporal graphs [13], [20], [39], frequent graph patterns [61] and frequent patterns [62].

The concepts of temporal graphs and subgraphs that we use essentially follow [20], [39], which were used to study the dense temporal graphs problem. Slightly different from their definitions, we use edge labels, instead of edge weights, to represent the dynamic properties of temporal graphs. In addition, the maximal and non-expandable properties in our definitions have certain connections to maximal or closed frequent patterns [60]–[62], and they both serve for reducing redundant patterns.

VII. CONCLUSIONS

We have proposed a proper notion of temporal motifs for a class of temporal networks, where the nodes and edges are fixed, but the edge labels vary regularly with timestamps. We have developed algorithms FTM-EL and FTM-IC to efficiently find all the maximal and non-expandable temporal network motifs using the EL table and IC trees, respectively. Further, we have also developed incremental algorithms DFTM-EL and DFTM-IC to deal with continuous updates of temporal networks. Finally, we have experimentally verified that both algorithms FTM-EL and FTM-IC run fast on large temporal graphs, and that incremental algorithms DFTM-EL and DFTM-IC outperform their static counterparts. By case studies on real-life datasets, we have also verified the practical usability of our proposed temporal motifs.

A couple of interesting topics need a further study, such as relax the continuousness requirement of edges that keep the same labels in at least k snapshots, and investigate the periodicity and distributed settings of temporal motifs.

REFERENCES

[1] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon, "Network motifs in the transcriptional regulation network of escherichia coli," *Nature Genetics*, vol. 31, pp. 64–68, 2002.

[2] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[3] N. H. Tran, K. P. Choi, and L. Zhang, "Counting motifs in the human interactome," *Nature communications*, vol. 4, no. 1, pp. 2241:1–2241:8, 2013.

[4] X. Li, R. Cheng, K. C.-C. Chang, C. Shan, C. Ma, and H. Cao, "On analyzing graphs with motif-paths," *PVLDB*, vol. 14, no. 6, pp. 1111–1123, 2021.

[5] C. Ma, R. Cheng, L. V. Lakshmanan, T. Grubenmann, Y. Fang, and X. Li, "Linc: a motif counting algorithm for uncertain graphs," *PVLDB*, vol. 13, no. 2, pp. 155–168, 2019.

[6] E. R. Ateskan, K. Erciyes, and M. E. Dalkilic, "Parallelization of network motif discovery using star contraction," *Parallel Comput.*, vol. 101, pp. 102 734:1–102 734:12, 2021.

[7] P. Li, H. Dau, G. Puleo, and O. Milenkovic, "Motif clustering and overlapping clustering for social network analysis," in *INFOCOM*, 2017.

[8] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, "Local higher-order graph clustering," in *KDD*, 2017.

[9] M. Bressan, S. Leucci, and A. Panconesi, "Motivo: fast motif counting via succinct color coding and adaptive sampling," *PVLDB*, vol. 12, no. 11, pp. 1651–1663, 2019.

[10] S. Itzkovitz, R. Levitt, N. Kashtan, R. Milo, M. Itzkovitz, and U. Alon, "Coarse-graining and self-dissimilarity of complex networks," *Physical Review E*, vol. 71, no. 1, pp. 016 127:1–016 127:10, 2005.

[11] S. Valverde and R. V. Solé, "Network motifs in computational graphs: A case study in software architecture," *Physical Review E*, vol. 72, no. 2, pp. 026 107:1–026 107:8, 2005.

[12] P. Holme and J. Saramäki, "Temporal networks," *Physics reports*, vol. 519, no. 3, pp. 97–125, 2012.

[13] P. Bogdanov, M. Mongiovì, and A. K. Singh, "Mining heavy subgraphs in time-evolving networks," in *ICDM*, 2011.

[14] W. Yu, C. C. Aggarwal, S. Ma, and H. Wang, "On anomalous hotspot discovery in graph streams," in *ICDM*, 2013.

[15] C. C. Aggarwal and K. Subbian, "Evolutionary network analysis: A survey," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 10:1–10:36, 2014.

[16] H. Wu, J. Cheng, S. Huang, Y. Ke, Y. Lu, and Y. Xu, "Path problems in temporal graphs," *PVLDB*, vol. 7, no. 9, pp. 721–732, 2014.

[17] J. Mondal and A. Deshpande, "EAGr: supporting continuous ego-centric aggregate queries over large dynamic graphs," in *SIGMOD*, 2014.

[18] A. Epasto, S. Lattanzi, and M. Sozio, "Efficient densest subgraph computation in evolving graphs," in *WWW*, 2015.

[19] H. Huang, J. Song, X. Lin, S. Ma, and J. Huai, "TGraph: A temporal graph data management system," in *CIKM*, 2016.

[20] S. Ma, R. Hu, L. Wang, X. Lin, and J. Huai, "An efficient approach to finding dense temporal subgraphs," *TKDE*, vol. 32, no. 4, pp. 645–658, 2020.

[21] D. Braha and Y. Bar-Yam, "Time-dependent complex networks: Dynamic centrality, dynamic motifs, and cycles of social interactions," in *Adaptive Networks*, T. Gross and H. Sayama, Eds. Berlin, Heidelberg: Springer, 2009, pp. 39–50.

[22] L. Kovanen, M. Karsai, K. Kaski, J. Kertész, and J. Saramäki, "Temporal motifs in time-dependent networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2011, no. 11, pp. P11 005:1–P11 005:18, 2011.

[23] L. Kovanen, K. Kaski, J. Kertész, and J. Saramäki, "Temporal motifs reveal homophily, gender-specific patterns, and group talk in call sequences," *PNAS*, vol. 110, no. 45, pp. 18 070–18 075, 2013.

[24] C. Song, T. Ge, C. Chen, and J. Wang, "Event pattern matching over graph streams," *PVLDB*, vol. 8, no. 4, pp. 413–424, 2014.

[25] Y. Hulovatyy, H. Chen, and T. Milenković, "Exploring the structure and function of temporal networks with dynamic graphlets," *Bioinformatics*, vol. 31, no. 12, pp. i171–i180, 2015.

[26] S. Gurukar, S. Ranu, and B. Ravindran, "Commit: A scalable approach to mining communication motifs from dynamic networks," in *SIGMOD*, 2015.

[27] A. Paranjape, A. R. Benson, and J. Leskovec, "Motifs in temporal networks," in *WSDM*, 2017.

[28] R. Kumar and T. Calders, "2scent: An efficient algorithm to enumerate all simple temporal cycles," *Proceedings of the VLDB Endowment*, vol. 11, no. 11, pp. 1441–1453, 2018.

[29] C. Kosyfaki, N. Mamoulis, E. Pitoura, and P. Tsaparas, "Flow motifs in interaction networks," in *EDBT*, 2019.

[30] Q. Zhao, Y. Tian, Q. He, N. Oliver, R. Jin, and W.-C. Lee, "Communication motifs: a tool to characterize social communications," in *CIKM*, 2010.

[31] N. Pashanasangi and C. Seshadhri, "Faster and generalized temporal triangle counting, via degeneracy ordering," in *KDD*, 2021.

[32] P. Liu, N. Masuda, T. Kito, and A. E. Sarıyüce, "Temporal motifs in patent opposition and collaboration networks," *Scientific reports*, vol. 12, no. 1, pp. 1917:1–1917:11, 2022.

[33] G. Lee and K. Shin, "Temporal hypergraph motifs," *KAIS*, vol. 65, no. 4, pp. 1549–1586, 2023.

[34] X. Cai, X. Ke, K. Wang, L. Chen, T. Zhang, Q. Liu, and Y. Gao, "Efficient temporal butterfly counting and enumeration on temporal bipartite graphs," *Proc. VLDB Endow.*, vol. 17, no. 4, p. 657–670, 2024.

[35] R. Jin, S. McCallen, and E. Almaas, "Trend motif: A graph mining approach for analysis of dynamic complex networks," in *ICDM*, 2007.

[36] E. Desmier, M. Plantevit, C. Robardet, and J.-F. Boulicaut, "Trend mining in dynamic attributed graphs," in *ECML/PKDD*, 2013.

[37] R. Ahmed and G. Karypis, "Algorithms for mining the evolution of conserved relational states in dynamic networks," *KAIS*, vol. 33, no. 3, pp. 603–630, 2012.

[38] R. Li, J. Su, L. Qin, J. X. Yu, and Q. Dai, "Persistent community search in temporal networks," in *ICDE*, 2018.

[39] S. Ma, R. Hu, L. Wang, X. Lin, and J. Huai, "Fast computation of dense temporal subgraphs," in *ICDE*, 2017.

[40] T. V. Jensen and P. Pinson, "Re-europe, a large-scale dataset for modeling a highly renewable european electricity system," *Scientific data*, vol. 4, pp. 170 175:1–170 175:18, 2017.

[41] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM TIST*, vol. 5, no. 3, pp. 38:1–38:55, 2014.

[42] P. Liu, V. Guarrasi, and A. E. Sariyuce, "Temporal network motifs: Models, limitations, evaluation," *TKDE*, vol. 35, no. 1, pp. 945–957, 2023.

[43] P. Liu, A. R. Benson, and M. Charikar, "Sampling methods for counting temporal motifs," in *WSDM*, 2019.

[44] J. Wang, Y. Wang, W. Jiang, Y. Li, and K.-L. Tan, "Efficient sampling algorithms for approximate temporal motif counting," in *CIKM*, 2020.

[45] I. Sarpe and F. Vandin, "oden: Simultaneous approximation of multiple motif counts in large temporal networks," in *CIKM*, 2021.

[46] I. Sarpe and F. Vandin, "Presto: Simple and scalable sampling techniques for the rigorous approximation of temporal motif counts," in *SDM*, 2021.

[47] Z. Gao, C. Cheng, Y. Yu, L. Cao, C. Huang, and J. Dong, "Scalable motif counting for large-scale temporal graphs," in *ICDE*, 2022.

[48] S. Ma, Y. Cao, W. Fan, J. Huai, and T. Wo, "Strong simulation: Capturing topology in graph pattern matching," *TODS*, vol. 39, no. 1, pp. 4:1–4:46, 2014.

[49] A. Gajewar and A. D. Sarma, "Multi-skill collaborative teams based on densest subgraphs," in *SDM*, 2012.

[50] C. Li, W. Yue, G. Mao, and Z. Xu, "Congestion propagation based bottleneck identification in urban road networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 4827–4841, 2020.

[51] H. Nguyen, W. Liu, and F. Chen, "Discovering congestion propagation patterns in spatio-temporal traffic data," *IEEE Trans. Big Data*, vol. 3, no. 2, pp. 169–180, 2016.

[52] S. Pfenninger and I. Staffell, "Long-term patterns of european pv output using 30 years of validated hourly reanalysis and satellite data," *Energy*, vol. 114, pp. 1251–1265, 2016.

[53] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. Cambridge, Mass: MIT press, 2009.

[54] R. E. Tarjan, "Efficiency of a good but not linear set union algorithm," *JACM*, vol. 22, no. 2, pp. 215–225, 1975.

[55] M. Thorup, "Decremental dynamic connectivity," *Journal of Algorithms*, vol. 33, no. 2, pp. 229–243, 1999.

[56] M. J. M. Turcotte, A. D. Kent, and C. Hash, *Unified Host and Network Data Set*. World Scientific, Nov. 2018, ch. Chapter 1, pp. 1–22.

[57] M. Vernet, y. Pigne, and E. Sanlaville, "A Study of Connectivity on Dynamic Graphs: Computing Persistent Connected Components," *4OR: A Quarterly Journal of Operations Research*, 2022, to be published.

[58] K. Semertzidis, E. Pitoura, E. Terzi, and P. Tsaparas, "Finding lasting dense subgraphs," *Data Mining and Knowledge Discovery*, vol. 33, no. 5, pp. 1417–1445, 2019.

[59] W. Fan, X. Wang, Y. Wu, and J. Xu, "Association rules with graph patterns," *PVLDB*, vol. 8, no. 12, pp. 1502–1513, 2015.

[60] M. H. Namaki, Y. Wu, Q. Song, P. Lin, and T. Ge, "Discovering graph temporal association rules," in *CIKM*, 2017.

[61] H. Cheng, X. Yan, and J. Han, "Mining graph patterns," in *Frequent Pattern Mining*, C. C. Aggarwal and J. Han, Eds. Cham: Springer, 2014, pp. 307–338.

[62] C. C. Aggarwal and J. Han, Eds., *Frequent Pattern Mining*. Cham: Springer, 2014.

Hanqing Chen received the B.S. degree in software engineering from Beihang University, China, in 2019. He is a Ph.D. candidate in the School of Computer Science and Engineering, Beihang University, China. His research interests include big data and graph analytics.

Shuai Ma received the Ph.D. degrees in computer science from Peking University, China, in 2004, and from The University of Edinburgh, England, in 2010, respectively. He is a professor with the School of Computer Science and Engineering, Beihang University, China. He was a postdoctoral research fellow with the Database Group, University of Edinburgh, a summer intern at Bell Labs, Murray Hill, NJ, and a visiting researcher of MSRA. His current research interests include big data, database, theory and systems, graph and social data analysis, data cleaning and data quality.

Junfeng Liu received the B.S. degree in computer science from Hefei University of Technology, China, in 2018. He is a Ph.D. candidate in the School of Computer Science and Engineering, Beihang University, China. His research interests include big data and graph analytics.

Lizhen Cui received the Ph.D. degree in computer science from the Shandong University, China, in 2005. He is a professor with the School of Software & C-FAIR, Shandong University, China. He is the chair of the School of Software & C-FAIR, Shandong University, a visiting researcher of Georgia Institute of Technology, and a visiting professor of Nanyang Technological University. His current research interests include recommender systems and data mining.