Discovery of Temporal Network Motifs

Hanqing Chen[1], Shuai Ma[1], Junfeng Liu[1], Lizhen Cui[2]
[1]*SKLSDE Lab, Beihang University, Beijing, China*
[2]*School of Software & C-FAIR, Shandong University, Shandong, China*
{chenhanqing, mashuai, liujunfeng}@buaa.edu.cn clz@sdu.edu.cn

***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]–[7], social networks [4], [5], [7]–[15], electrical circuits [16], software architectures [17], and transport networks [5], [7], [18]. 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 attentions from both industry and academic communities under various terms [19], such as temporal networks, dynamic graphs, evolving networks, evolutionary networks, and graph streams [19]–[53].

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. [54] 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 in temporal networks [34]–[44], [50]–[53], [55]. [34]–[37], [51] impose a partial time order on edges and a local time window on adjacent edges in a temporal motif whose edges are attached with beginning timestamps and durations or labeled with only timestamps, where [51] further introduces a time duration bound on edges. [38]–[43], [53] impose a total time order and a global time window on all edges in a temporal motif, where [43] further introduces a minimum flow bound on edges. [44] imposes a global time window on all edges and nodes, while [55] imposes both a local time window and a global time window on edges in their temporal motifs. [52] focuses on the triangles counting problem and imposes a local time window on its temporal triangles. [50] focuses on the graph matching problem and imposes a partial time order and a global time window on its query graph. Here, the difference between the total time order and the partial time order is whether the time order of all edges in its motif is defined or not. The local time window is the time difference bounded on adjacent edges, while the global time window is defined on all edges. Different from above mentioned, there are studies focusing more on time-dependent node weights or edge weights (labels) in temporal networks. Each of node weights or edge weights (labels) in their temporal motifs displays statics or similar dynamics over a user-defined period. [45], [46] impose an increasing or decreasing trend of weights on the nodes in a motif, *i.e.,* node weights keep evolving in a consistent trend over a period. [47] imposes time-persistent relations on the edge labels in a pattern called induced relational state (IRS), *i.e.,* edge labels and directions remain unchanged over a period. These studies 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. This implies the importance of proper definitions for temporal networks and temporal motifs.

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 [20], [23], [33], [56]. Essentially, such a temporal network with T timestamps can be viewed as T snapshots of a static network. Road networks and energy transmission networks typically fall into this category [23], [26], [27], [57], in which traffic and energy consumption analyses are significant for urban computing [58].

However, the study of motifs in such temporal networks needs to deal with three challenging issues. First, how to properly define temporal motifs that are useful for practical applications? [48] 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. Second, 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 [34]–[38], [43]–[45], [51], [52], [55]. While [46], [47] do not need isomorphism tests, they need enumerate an exponential number of induced subgraphs. There also exist methods adopting approximation techniques to make computation tractable or parallel techniques to achieve speedup, including sampling approaches [39]–[42] and an OpenMP-based parallel approach [53] in motif number counting problems, and dual simulation [50], [59] in graph matching problems. Third, temporal networks are naturally dynamic and continuously updated, and hence dynamic algorithms are needed to discover temporal motifs, which has not been considered by existing studies [34]–[47], [50]–[53], [55].

Contributions & roadmap. To this end, we propose a novel concept of temporal motifs for temporal networks, and develop efficient static and incremental approaches to finding temporal motifs in large temporal networks.

(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 [34]–[47], [50]–[52], [55], 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 FTM-EL is on average $(1.04, 1.02, 1.94)$ times faster than FTM-IC on datasets (BJData, EURData, SYNData), respectively; (b) our incremental algorithms DFTM-EL and DFTM-IC are on average $(2.03, 1.98, 2.09)$ and $(1.96, 1.94, 1.93)$ times faster than their static counterparts FTM-EL and FTM-IC on (BJData, EURData, SYNData), respectively; (c) our motifs are useful for practical applications through case studies on real-life datasets BJData and EURData.

Due to space limiations, detailed proofs, applications of our temporal motifs, extra examples and experimental studies are available at https://codesanddata.github.io/FTM/paper-full.pdf.

II. Preliminary

In this section, we introduce the basic definitions of temporal graphs, temporal motifs and the problem. In this study, we use the terms network and graph interchangeably.

Temporal graphs. A *temporal graph* $G(V, E, L, T_b, T_e)$ 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) $[T_b, T_e]$ is a time interval representing $(T_e - T_b + 1)$ timestamps, in which $T_b \leq T_e$ are the beginning and ending timestamps, respectively, and (4) L is a set of label functions such that for each timestamp $t \in [T_b, T_e]$, 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 $G(V, E, T_b, T_e)$ 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 [20], [23], [33], [56]. Intuitively, (1) a temporal graph $G(V,\ E,\ L)$ essentially denotes a sequence $< G_1(V, E, L^{T_b})$, $\ldots$, $G_t(V, E, L^t)$, $\ldots$, $G_{T_e - T_b + 1}(V, E, L^{T_e})\ >$ of $T = T_e - T_b + 1$ standard graphs, and (2) its edge labels $L^t(e)$ ($t \in [T_b, T_e]$) specify the distances, communication latencies or travelling duration [19], [23], [26], [27], or the affinity or collaborative compatibility [60] between the two corresponding nodes of 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 ($t \in [T_b, T_e]$) its *snapshot* at timestamp t, and we also use intervals instead of time intervals for simplicity.

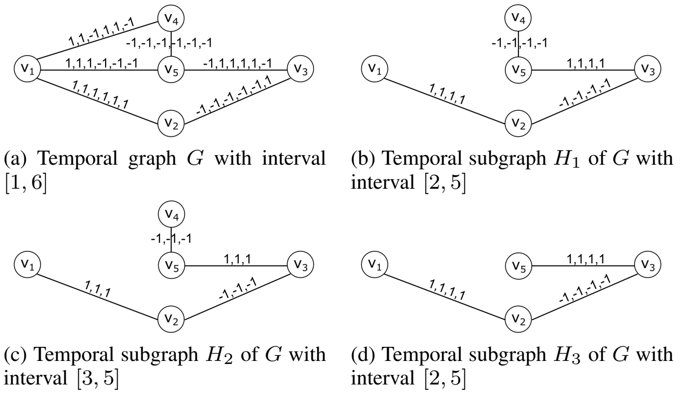
Temporal subgraphs. A temporal graph $G_s(V_s, E_s, T_{b_s}, T_{e_s}, L_s)$ is a *subgraph* of $G(V, E, T_b, T_e, L)$, if $V_s \subseteq V, E_s \subseteq V_s \times V_s$ and $E_s \subseteq E, T_{b_s} \leq T_{e_s} \in [T_b, T_e]$, and $L^t_s(e)$ = $L^t(e)$ for any timestamp $t \in [T_{b_s}, T_{e_s}]$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

That is, temporal subgraph $G_s(V_s, E_s, T_{b_s}, T_{e_s})$ only contains a subset of nodes and edges of temporal graph G, its edges have the same labels as G, and it is restricted within the interval $[T_{b_s}, T_{e_s}]$ that falls into the interval $[T_b, T_e]$ 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 $G_s(V_s, E_s, T_{b_s}, T_{e_s})$ 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, $L^{T_{b_s}}(e) = \cdots = L^{T_{e_s}}(e)$, *i.e.,* each edge has the same label at all the timestamps in interval $[T_{b_s}, T_{e_s}]$ and (3) $T_{e_s} - T_{b_s} + 1 \geq k$, *i.e.,* appear in at least k continuous snapshots.

Expandable temporal motifs. A temporal motif $G_s(V_s, E_s, T_{b_s}, T_{e_s})$ is left (resp. right) expandable if and only if for



(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 $[2, 5]$

Fig. 1: Running example

each edge e in E_s, $L^{T_{b_s}-1}(e) = L^{T_{b_s}}(e)$ (resp. $L^{T_{e_s}}(e) = L^{T_{e_s}+1}(e)$). That is, for each edge $e \in E_s$, its label keeps unchanged in interval $[T_{b_s}-1, T_{e_s}]$ (resp. $[T_{b_s}, T_{e_s}+1]$) 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 $T_{b_s} - 1 \geq T_b$ and $T_{e_s} + 1 \leq T_e$.

Maximal temporal motifs. A temporal motif $G_s(V_s, E_s, T_{b_s}, T_{e_s})$ is maximal if and only if there exist no edges e in E but not in E_s such that e is incident to G_s, $L^{T_{b_s}}(e) = \cdots = L^{T_{e_s}}(e)$. That is, e is connected to G_s, and has the same label in interval $[T_{b_s}, T_{e_s}]$.

We next illustrate these concepts with examples.

Example 1: (1) Fig. 1a depicts a temporal graph G with 5 nodes, 6 edges and 6 timestamps.
(2) Fig. 1b and Fig. 1d show two temporal subgraphs H_1 and H_3 of G with the same interval $[2, 5]$, and Fig. 1c 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 Fig. 1c is a left expandable temporal motif, as each its edge has the same label in interval $[2, 5]$. (c) Temporal motif H_3 in Fig. 1d 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 motifs H_1 in Fig. 1b 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 temporal graph $G(V, E, T_b, T_e, L)$ and frequency threshold k, the problem is to find all the maximal and non-expandable temporal motifs $G_s(V_s, E_s, T_{b_s}, T_{e_s})$.

Example 2: Given the temporal graph G in Fig. 1a and frequency threshold $k = 4$, the temporal subgraph H_1 in Fig. 1b 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 bottlenecks roads in road networks [61], [62], and energy consumption patterns with continuous high energy demand signal levels in transmission networks [57], [63], [64]. These applications typically concern some significant regular phenomenons from a temporal viewpoint, which is particularly important for urban computing [58].

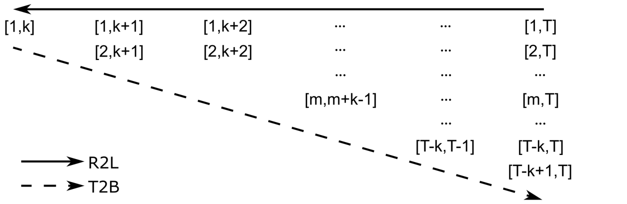


Fig. 2: TI-Table for temporal graph $G(V, E, 1, T, L)$ and k

Remarks. Different from existing studies on temporal motifs [34]–[47], [50]–[52], [55], we focus on a special class of temporal networks, where the nodes and edges are fixed, but edge labels vary regularly with timestamps [20], [23], [33], [56]. Moreover, the above mentioned applications cannot be solved by previous studies on temporal motifs. Indeed, the node weights or edge weights (labels) in the above mentioned applications display statics or similar dynamics over a period, and are more important than the network topology as mentioned in [45]–[47]. However, [45] and [46] define motifs with the node weights evolving in a consistent trend, which are unsuitable to analyze long time unchanged areas. [47] defines its induced relational states with the inducedness restrictions, which exhibits a bias towards certain types of motifs and is inappropriate for the need of discovering motifs in unknown situations (see a recent survey [48]). Further, our temporal motifs can be computed efficiently in polynomial time, as subgraph isomorphism tests 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 $G(V, E, T_b, T_e, L)$ and frequency threshold k. Without loss of generality, we assume that $T_b = 1$ and $T_e = T$ 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 is an algorithm to find all the maximal and non-expandable temporal motifs in $O(T|E|)$ time.* □

A. Analyses of Temporal Motifs

We start with the analyses of temporal motifs.

TI-Table. For all maximal and non-expandable temporal motifs $G_s(V_s, E_s, T_{b_s}, T_{e_s})$, it is easy to see that $T_{e_s} - T_{b_s} + 1 \geq k$. Hence, all possible intervals can be arranged in a Triangularized Interval Table (TI-Table), as shown in Fig.2, where the intervals in the same row have the same beginning timestamp, and the intervals in the same column have the same ending timestamp. Note that no other intervals within $[1, T]$ satisfy the frequency threshold k.

The TI-Table essentially tells us three problems for finding all the maximal and non-expandable temporal motifs.

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

Let $\mathsf{S}[m, i]$ $(1 \leq m \leq T-k+1,\ m+k-1 \leq i \leq T)$ 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: $\mathsf{S}[m, T] \subseteq \mathsf{S}[m, T-1] \subseteq \cdots \subseteq \mathsf{S}[m, m+k-1]$. □

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.

$$\mathsf{R}[m,i] = \begin{cases} \mathsf{S}[m,i] - \mathsf{S}[m,i+1], m+k-1 \leq i < T; \\ \mathsf{S}[m,i], i = T. \end{cases}$$

By the definitions of S and R edge sets, it is easy to verify that $\mathsf{S}[m, i] = \mathsf{R}[m, i] \cup \mathsf{R}[m, i+1] \cup \cdots \cup \mathsf{R}[m, T]$. Further, R edge sets have the following non-redundancy property.

Proposition 1: $\mathsf{R}[m, i] \cap \mathsf{R}[m, j]$ *is empty for any* $i \neq j$, *where* $1 \leq m \leq T-k+1$ *and* $m+k-1 \leq i, j \leq T$. □

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

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

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

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

Connected components can be computed in linear time [65]. By Proposition 2, we have an efficient solution to generate maximal temporal motifs based on connected components.

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 $\mathsf{R}[m, i]$. (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 $[m, i+1], \ldots, [m, T]$ by adding edges in $\mathsf{R}[m, i]$.

Let $|E_m|$ and $|V_m|$ be the number of edges and nodes in temporal subgraph $G_s(\mathsf{S}[m, m+k-1])$, respectively. All maximal temporal motifs in the m-th row can be computed in an incremental way in $O(|E_m|)$ time when using R2L [66], while it is much slower in a decremental way when using L2R ($O(|E_m| \log |V_m|)$ time even for sparse graphs) [67], as the key step of generating maximal temporal motifs using R2L or L2R is maintaining connected components in an incremental or decremental way. As there exist no isolated nodes in $G_s(\mathsf{S}[m, m+k-1])$ $(E_m \geq \frac{V_m}{2})$, R2L is better than L2R for computing maximal temporal motifs.

Problem 3: Finally, we need to check whether a maximal temporal motif is expandable. When generating maximal temporal motifs using 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.* □

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.

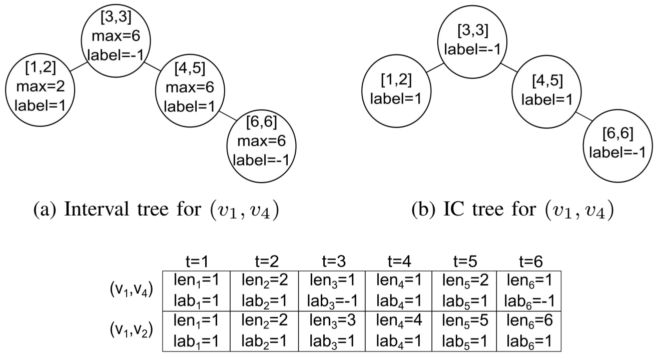
By Propositions 2 & 3, we know that the *top to bottom (T2B, for rows) and right to left (R2L, for columns)* scheme is a better choice to compute the maximal and non-expandable temporal motifs for all the intervals in the TI-Table.

B. Compute R *Edge Sets*

We first introduce how to compute edge sets $\mathsf{R}[m, i]$ $(1 \leq m \leq T-k+1$ and $m+k-1 \leq i \leq T)$. By the definition of R edge sets, any edge e in $\mathsf{R}[m, i]$ must have the same label in the left most interval $[m, m+k-1]$ of the m-th row in the TI-Table, as it is the minimum interval satisfying the frequency threshold k. Hence, we need to find the maximum interval $[m', i']$ containing $[m, m+k-1]$ (*i.e.,* $[m', i'] \supseteq [m, m+k-1]$) such that edge e has the same label in $[m', i']$, from which we know that e belongs to $\mathsf{R}[m, i']$. Note that if $m' < m$, we have already computed $[m', i']$ for e, as we follow the T2B scheme. There is no need to re-compute $[m', i']$ as $e \in \mathsf{R}[m, i']$ for $m' \leq m \leq i'-k+1$.

We design two data structures to efficiently identify the maximum interval $[m', i']$ containing $[m, m+k-1]$ such that edge e has the same label in interval $[m', i']$.

IC trees. The first data structure is a balanced tree for each edge, referred to as *interval containment trees* (or IC trees), inspired by interval trees [65]. Given an interval, interval trees are designed to efficiently identify overlapping intervals in $O(\log |V_{tree}| + h)$ time and $O(|V_{tree}|)$ space, where $|V_{tree}|$ is the number of nodes in the interval tree, and h is the number of overlapping intervals. Fig.3a depicts an interval tree for edge (v_1, v_4) in Fig.1a, where a node contains an interval, the edge label within the interval, and an extra value max that records the maximum right ending timestamp of all intervals in the subtree rooted at the node. However, interval trees are designed for *overlap* semantics, and if edge e does not have the same label in interval $[m, m+k-1]$, it returns multiple intervals overlapping with $[m, m+k-1]$.



(a) Interval tree for (v_1, v_4)

(b) IC tree for (v_1, v_4)

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

(c) The EL table for (v_1, v_2) and (v_1, v_4)

Fig. 3: Three data structures

IC trees are designed for *containment* semantics to address this issue. An IC tree for an edge e is a red-black balanced tree, in which (1) each of its node contains a maximum interval such that e keeps the same label, (2) any intersection of two different intervals is empty, (3) all intervals together cover the entire interval $[1, T]$. An IC tree identifies intervals containing a given interval in $O(\log|V_{tree}|)$ time and $O(|V_{tree}|)$ space. Both IC trees and interval trees can be created in $O(|V_{tree}| \log|V_{tree}|)$ time. Fig.3b depicts the IC tree for edge (v_1, v_4), and its interval tree counterpart is shown in Fig.3a.

The fundamental difference of IC trees and interval trees is that the query of an IC tree has better pruning conditions than interval trees, which leads to better query complexities. The query of an IC tree takes a query interval as input, and returns a single interval containing the query interval if exists. It starts from the root, checks whether the interval of the current node overlaps with the query interval, and then checks whether the current interval contains the query interval if the overlap relation exists. It returns $NULL$ if there only exists the overlap relation but no containment relations, as there are no intervals containing the query interval in the tree. If there exist no overlap relations, it traverses the right subtree when the query interval is later than the current interval, and traverses the left subtree otherwise until it reaches the leaf node or returns (based on the properties (2) and (3) of IC trees).

We next explain how to use the IC tree of edge e to find the maximum interval $[m', i']$ containing $[m, m+k-1]$ such that edge e has the same label in interval $[m', i']$ with an example.

Example 3: We consider the temporal graph G in Fig. 1a, frequency threshold $k = 3$, and interval $[2, 4]$ (the left most interval in the second row of the TI-Table).

(1) For the IC tree of edge (v_1, v_4) shown in Fig. 3b, the search process starts from the root node with interval $[3, 3]$. As $[3, 3]$ does not contain $[2, 4]$, it stops and finds no valid intervals. (v_1, v_4) does not belong to any R edge sets. For the interval tree of edge (v_1, v_4) shown in Fig. 3a, the search process starts from the root node with interval $[3, 3]$, then the nodes with intervals $[1, 2]$ and $[4, 5]$. As it returns three intervals, there are no valid intervals. This shows the benefit of using IC trees compared with using interval trees.

(2) For edge (v_1, v_2), its IC tree has a single node with interval $[1, 6]$, the search process returns the maximum interval $[1, 6]$ that contains $[2, 4]$. (v_1, v_2) belongs to R$[2, 6]$. □

EL table. To further improve the efficiency of computing R edge sets when an edge has no valid intervals as case (1) in Example 3 shows, we next design the second data structure 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 an EL table), in which for each edge e and timestamp t, it stores the label lab_t of e at timestamp t, and the number len_t of timestamps that e keeps the same label lab_t until timestamp t. Fig.3c depicts the EL table for edges (v_1, v_4) and (v_1, v_2) only in Fig.1a, where for edge (v_1, v_4), it keeps the same label in interval $[t - len_5 + 1, t] = [4, 5]$ at timestamp $t = 5$.

We next explain how to utilize the EL table to find the maximum interval $[m', i']$ containing $[m, m+k-1]$ for edge e such that e has the same label in $[m', i']$. Using the EL table for e, it starts from timestamp $t = m+k-1$ to T. The process continues until $len_t < k$ at timestamp t or $t = T$. If $len_t < k$, then the maximum interval is $[t-len_{t-1}, t-1]$ when $t > m+k-1$, and there are no valid intervals, otherwise. If $t = T$, then the maximum interval is $[T - len_{T-1}, T-1]$ when $len_T < k$, and is $[T - len_T + 1, T]$, otherwise.

Note that utilizing the EL table, it takes $O(T)$ time and $O(T)$ space to find the maximum interval containing a given interval for an edge, and the EL table for an edge can be created in $O(T)$ time.

We next use an example to show the above process.

Example 4: We consider the same setting as Example 3.

(1) For edge (v_1, v_4) and interval $[2, 4]$, we check the row for (v_1, v_4) in the EL table in Fig. 3c from timestamps 4 to 6 ($T = 6$ for G). The process stops at the timestamp 4. As $t = 4 \leq 2 + 3 - 1$ and $len_4 < k = 3$, there exist no valid maximum intervals containing $[2, 4]$. (v_1, v_4) does not belong to any R edge sets. This shows the benefit of using the EL table compared with using IC trees.

(2) For edge (v_1, v_2) and interval $[2, 4]$, we check the row for (v_1, v_2) in the EL table in Fig. 3c from timestamps 4 to 6. The process stops at the timestamp 6. As $len_T \geq k$ and $t = T$, it returns the maximum interval $[1, 6]$ containing $[2, 4]$. (v_1, v_2) belongs to R$[2, 6]$. □

Computing R edge sets. We next show the details of procedure computeRES to compute R edge sets for the intervals in a row of TI-Table, shown in Fig. 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.

Array EMaxIntvl is a data structure that dynamically maintains the current maximum intervals $[m', i']$ containing $[m, m+k-1]$ for all edges having the same label in $[m', i']$ in the T2B and R2L 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 maximal and non-expandable temporal motifs.

```
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 $[m', i'] \supseteq [m, m + k - 1]$ for $e$ such
          that $e$ has the same label using the $m$-th row in the
          EL table or IC tree for $e$;      /*cases (1) and (2)*/
6.      if $[m', i']$ exists then
7.        $R[m, i'] := R[m, i'] \cup \{e\}$;
8.        EMaxIntvl$[e] := [m', i']$;
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).
```

Fig. 4: Procedure computeRES

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 valid maximum interval for rows above m. In this case, we need to use the EL table or IC tree for e to identify the maximum interval $[m', i'] \supseteq [m, m + k - 1]$. (2) $i' < m$, which means that the label of edge e changes at timestamp $i' + 1$, and EMaxIntvl$[e]$ needs to be updated with the EL table or IC tree for e, as the intersection of intervals $[m, m + k - 1]$ and $[m', i']$ is empty. (3) $m \leq i' < m + k - 1$, which means that there exist no valid intervals containing $[m, m + k - 1]$, as the label of edge e changes at timestamp $i' + 1$. Hence, e does not belong to any R edge set 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, we know that e belongs to edge set $R[m, i']$. That is, cases (3) and (4) avoid the unnecessary use of the EL table or IC trees, by utilizing array EMaxIntvl.

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 $[m', i']$ containing $[m, m + k - 1]$, in which e keeps the same label. If $[m', i']$ exists, it puts e into $R[m, i']$ and updates EMaxIntvl$[e]$ with $[m', i']$ (lines 4-8). 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).

When handling an edge for a single interval, IC trees have better computational complexity than the EL table. However, the EL table may have better computational efficiency in some cases as shown in Examples 3 & 4. Indeed, the EL table shows better computational complexity than IC trees when handling an edge for all intervals in a row of TI-Table.

Proposition 4: *To determine* R *edge set in a row to which edge e belongs, it takes procedure* computeRES $O(1)$ *and* $O(\log|V_e|)$ *amortized time when using the EL table and IC trees, respectively. Here $|V_e|$ is the number of nodes in the IC tree of the edge e.* □

```
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 $CC[i, T]$.
```

Fig. 5: Procedure generateMaxTM

Time complexity. The construction of the EL table and IC trees takes $O(T|E|)$ and $O(|E| maxV_e \log maxV_e)$ time, respectively. By Proposition 4, it is easy to verify that procedure computeRES takes $O(|E|)$ and $O(|E| \log maxV_e)$ time for a row, when using the EL table and IC trees, respectively. Here $maxV_e$ is the largest $|V_e|$ for all $e \in E$.

C. Generate Maximal Temporal Motifs

We then introduce how to compute the maximal temporal motifs falling into 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 all rows from 1 to m in TI-Table, (2) the array EMaxIntvl of all edges for the m-th row, and (3) the maximal and non-expandable temporal motifs for all intervals in rows from 1 to $m - 1$ 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 $S[m, i] = R[m, i] \cup R[m, i + 1] \cup \cdots \cup R[m, T]$, where each cc in $CC[i, T]$ corresponds to a maximal temporal motifs with interval $[m, i]$. When computing the connected components $CC[i, T]$, we only need to add edges in $R[m, i]$ to the connected components $CC[i + 1, T]$.

We next present the details of procedure generateMaxTM shown in Fig. 5, which takes as input R edge sets, array EMaxIntvl, interval $[m, i]$, connected components $CC[i + 1, T]$, and returns connected components $CC[i, T]$ 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

```
Procedure generateExpTM
Input: Connected components CC[$i, T$] and interval [$m, i$].
Output: Set TF[$m, i$] of maximal and non-expandable temporal motifs in interval [$m, i$].
1. 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.         TF[$m, i$] := TF[$m, i$] $\cup$ {$G_s$};
6. return TF[$m, i$].
```

Fig. 6: Procedure generateExpTM

G_s in $\mathsf{CC}[i, T]$, it updates G_s by adding e to G_s and changing its interval with interval $G_s.intvl \cap \mathsf{EMaxIntvl}[e]$ (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 $G_s.intvl \cap G_{s'}.intvl \cap \mathsf{EMaxIntvl}[e]$, and updates $\mathsf{CC}[i, T]$ by replacing G_s and G'_s with $G_{ss'}$ (lines 8-10). Finally, it returns connected components $\mathsf{CC}[i, T]$ (line 11).

We next exemplify the procedure generateMaxTM.

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

Time complexity. Let $|E_m|$ be the number of edges in temporal subgraph $G_s(\mathsf{S}[m, m + k - 1])$. It takes procedure generateMaxTM $O(|E_m|)$ time to generate all the maximal temporal motifs for the intervals in the m-th row in TI-Table.

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) $\mathsf{CC}[i, T]$ 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* $i < i' \leq T$).

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 maximum interval containing $[m, m + k - 1]$ for each edge e in E_s, such that edge e has the same label in interval $[m', i']$, *i.e.,* the intersection of $\mathsf{EMaxIntvl}[e]$ for each edge e in E_s. Note that the interval $intvl$ is dynamically maintained when generating connected components in procedure generateMaxTM. That is, G_s is non-expandable iff $m' = m$, and is left expandable iff $m' < m$ such that $G_s[V_s, E_s, m, i]$ already appears in the maximal and non-expandable temporal motifs in $\mathsf{TF}[m', i]$.

We next present the details of procedure generateExpTM shown in Fig. 6, which takes as input connected components $\mathsf{CC}[i, T]$ 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 connected component G_s in $\mathsf{CC}[i, T]$ but not in $\mathsf{CC}[i + 1, T]$ (line 2), as other connected components 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 temporal motif G_s is non-expandable, it adds G_s 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 exemplify the procedure generateExpTM.

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 Fig. 1b). □

Time complexity. Let $|E_m|$ be the number of edges in temporal subgraph $G_s(\mathsf{S}[m, m + k - 1])$. It takes procedure generateExpTM $O(|E_m|)$ time to generate all the maximal and non-expandable temporal motifs for the intervals in the m-th row in TI-Table.

E. The Complete Algorithm

Based on three procedures as above mentioned, 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.

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 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 $m + k - 1 \leq i \leq T$ and the updated array EMaxIntvl. (2) Then it initializes connected components $\mathsf{CC}[i, T]$ for $i \in [m + k - 1, T + 1]$. Note that we use $\mathsf{CC}[T + 1, T]$ to represent an empty set just 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 connected component $\mathsf{CC}[i, T]$ for interval $[m, i]$. Each connected component $\mathsf{CC}[i, T]$ 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 $[m, i]$, and finally returns TF.

Proposition 5: FTM *correctly finds all the maximal and non-expandable temporal motifs.* □

We next exemplify the algorithm FTM.

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 $\mathsf{R}[2,6] = \{(v_1, v_2), (v_4, v_5)\}$ and $\mathsf{R}[2,5] = \{(v_2, v_3), (v_3, v_5)\}$, and updates $\mathsf{EMaxIntvl}[(v_3, v_5)] = [2,5]$. (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 $\mathsf{R}[2,5]$ 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 Fig. 1b).

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

Time and space complexities. 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 $O(T|E|)$ and $O(T|E| \log maxV_e)$ time, respectively. Note that it only needs this time to obtain TF, though it takes $O(T^2|E|)$ worst-case time to output all temporal motifs from TF; Algorithms FTM-EL and FTM-IC take $O(T|E| + TmaxE_m)$ and $O($ $maxV_e|E| + TmaxE_m)$ space, respectively, which is dominated by key data structures IC trees, the EL table and TF.

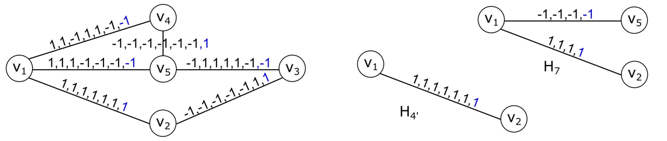
Remarks. The above analysis reveals that algorithm FTM-EL has a better time complexity than algorithm FTM-IC, while the latter has a better space complexity than the former. Further, both the time and space complexities of FTM-EL and FTM-IC have no connections with the number of edge labels. By Proposition 5 and the above complexity analysis, we have proved Theorem 1. Finally, our method can find temporal motifs with (or without) specific edge labels, as it can easily check the types of edge labels, and deals with temporal networks with real number edge weights, by discretizing these numbers into ranges *i.e.,* labels.

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 $O(TmaxE_{TF} + \Delta T|E|)$ time. Here $maxE_{TF}$ is the largest edge number in* $\mathsf{TF}^\circ[m, T]$ *for $m \in [1, T - k]$, which is smaller than $|E|$ in practice.* □

We first build the connections between TF° and TF, which is the basis for the design of our incremental algorithm.



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

(b) Temporal motifs $H_{4'}$ and H_7 of G' with interval $[1,7]$ and $[4,7]$, respectively

Fig. 7: Running example for incremental algorithm

Proposition 6: $\mathsf{TF}[m, i] = \mathsf{TF}^\circ[m, i]$ *for all $m \in [1, T - k]$ and $i \in [m + k - 1, T - 1]$.* □

Proposition 6 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 7: *We only need to use the edges in* $\mathsf{TF}^\circ[m, T]$ *instead of* $\mathsf{S}[m, T]$ *to compute the maximal and non-expandable motifs for intervals $[m, i]$ when $m \in [1, T - k + 1]$ and $i \in [T, T + \Delta T]$.* □

Proposition 7 reveals that we can further reduce the computation cost by utilizing the set of edges in $\mathsf{TF}^\circ[m, T]$ ($m \in [1, T - k + 1]$), which is a subset of $\mathsf{S}[m, T]$.

Algorithm DFTM. Based on Propositions 6 & 7, 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 $\mathsf{TF}[m, i] = \mathsf{TF}^\circ[m, i]$ for $i \in [m + k - 1, T - 1]$ by Proposition 6.

(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 $\mathsf{TF}^\circ[m, T]$ are considered when computing edge set $\mathsf{R}[m, i]$ using the EL table or IC trees, by Propositions 6 & 7.

(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 next exemplify the algorithm DFTM.

Example 8: Consider the updated temporal graph G' in Fig. 7a of temporal graph G in Fig. 1a 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 in Example 7. The updated TI-Table with $T + \Delta T = 7$ 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 $m \in [1,2]$, $\mathsf{TF}[m, i] = \mathsf{TF}^\circ[m, i]$ 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 $\mathsf{TF}^\circ[m, 6]$ 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 Fig. 7b, $H_{4'}$ with interval $[1,7]$ belongs to TF. Note that (v_1, v_2) belongs to a temporal motif in $\mathsf{TF}^\circ[1,6]$ 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 Fig. 7b.

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

Correctness. The correctness of algorithm DFTM follows easily from Propositions 6, 7 & 5.

Time and space complexities. Incremental algorithms DFTM-EL and DFTM-IC take $O(TmaxE_{TF} + \Delta T|E|)$ and $O((TmaxE_{TF} + \Delta T|E|)\log maxV_e)$ time, respectively. As algorithms FTM-EL and FTM-IC take $O((T + \Delta T)|E|)$ and $O((T + \Delta T)|E|\log maxV_e)$ time, respectively, incremental algorithms are obviously better than their static counterparts.

The only extra cost in the space complexity of algorithm DFTM is TF°$[m, T]$ for $m \in [1, T - k + 1]$, which is $O(TmaxE_{TF})$, as static algorithms FTM-EL and FTM-IC take $O((T+\Delta T)|E|+(T+\Delta T)maxE_m)$ and $O(maxV_e|E|+(T + \Delta T)maxE_m)$ space, respectively, for $G(V, E, 1, T + \Delta T)$. $maxE_{TF}$ is typically much smaller than E, and hence DFTM-EL and DFTM-IC have the same space complexities as their static counterparts FTM-EL and FTM-IC, respectively.

Remarks. Incremental algorithms have theoretically better time complexities and the same space complexities with a reasonable extra space cost, compared with their static counterparts. Further, incremental algorithms support temporal networks evolving with continuous updates. In addition, we have proved Theorem 2 by the above correctness and complexity analyses. Finally, there also exists an incremental method for the increase of frequency threshold k. It is based on the anti-monotone property for the maximal and non-expandable motif *w.r.t.* the value of k. It can select all motifs with larger interval lengths than the new k from the previous results.

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

We first illustrate our experimental settings.

Datasets. We gathered three datasets.

(1) BJData [33] is a real-life dataset that records three road traffic conditions (*i.e.,* congested, slow and fast) in Beijing. We extracted the data with 288 snapshots, $69,416$ nodes and $88,396$ edges, *i.e.,* the dataset has $19,991,808$ nodes and $25,458,048$ edges in total.

(2) EURData [57] 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 hourly energy demand signal. We transformed the graph by switching edges to nodes and nodes to edges, and used 10 labels to represent different energy demand signal levels (1-10, from lowest to highest). The dataset has $26,304$ snapshots, $2,706$ nodes and $7,334$ edges, *i.e.,* $71,178,624$ nodes and $192,913,536$ edges in total.

(3) SYNData [20] 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 their copies 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 [34]–[39], [43], [44] have time-dependent edges, *i.e.,* all edges appear at part of timestamps in motifs. However, our temporal motifs need edges appearing at all timestamps. Temporal motifs in [45]–[47] are somehow similar to our motifs, as their nodes or edges of motifs appear at all timestamps of the motif interval, respectively. However, [45]–[47] both limit motifs with induced subgraphs, and enumerate induced subgraphs by traversing all nodes recursively. That is, enumeration of all subgraphs with any size to find our temporal motifs takes exponential time. Therefore, their methods cannot be used to find our temporal motifs, and there is no need to compare these methods with our algorithms.

Exp-1: Tests of static algorithms. In the first set of tests, we test the running time and space cost of static algorithms FTM-EL and FTM-IC *w.r.t.* the graph sizes in terms of the number $|E|$ of edges, the number T of snapshots, and the frequency threshold k.There is no need to test the number of labels by the time and space complexity analyses.

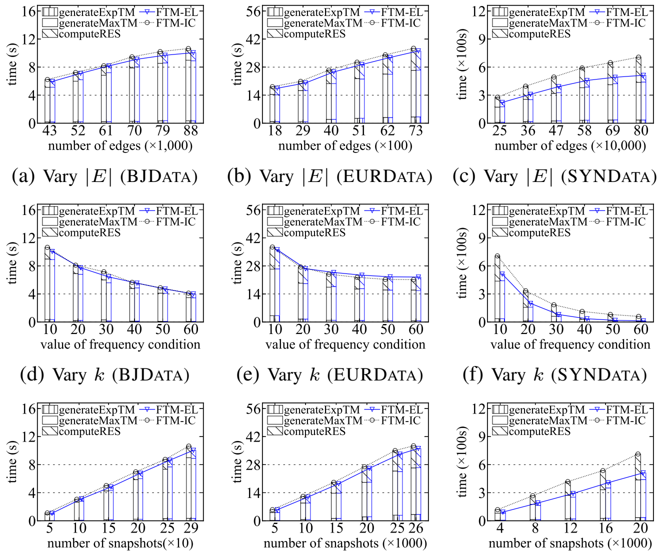
Exp-1.1. To evaluate the impacts of the graph size in terms of the number $|E|$ of edges, we varied $|E|$ from $43,000$ to $88,396$ for BJData, from $1,800$ to $7,334$ for EURData, and from $250,000$ to $800,000$ for SYNData, respectively. We fixed $k = 10$ and randomly selected edges by fixing $|V|=$ (69416, 2706, 400000) for (BJData, EURData, SYNData), respectively. The results are reported in Figs. 8a-8c.

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 (1.04, 1.04, 1.30) times faster than FTM-IC on (BJData, EURData, SYNData) on average, respectively.

When varying the number $|E|$ of edges, three procedures in FTM all increase with the increment of $|E|$. generateMaxTM takes the most time, on average (85%, 70%, 77%) for FTM-EL and (80%, 68%, 59%) for FTM-IC on (BJData, EURData, SYNData), respectively. computeRES using the EL table is on average (1.42, 1.16, 2.93) times faster than using the IC trees on (BJData, EURData, SYNData), respectively.

TABLE I: Space cost of static algorithms ($k = 10$)

Datasets	Aver. $\lvert E_m \rvert$	$maxE_m$	Aver.$\lvert V_e \rvert$	$maxV_e$	Memory cost (GB)	
					FTM-EL	FTM-IC
BJDATA (457 MB, $\lvert E \rvert = 88,396$)	$56,749$	$74,729$	28	172	0.466	0.391
EURDATA (3.15 GB, $\lvert E \rvert = 7,334$)	$4,966$	$6,636$	$1,442$	$6,683$	10.832	9.866
SYNDATA (30.6 GB, $\lvert E \rvert =$ 800K)	$195,377$	$275,813$	479	727	22.082	27.348



(a) Vary $|E|$ (BJDATA) (b) Vary $|E|$ (EURDATA) (c) Vary $|E|$ (SYNDATA)

(d) Vary k (BJDATA) (e) Vary k (EURDATA) (f) Vary k (SYNDATA)

(g) Vary T (BJDATA) (h) Vary T (EURDATA) (i) Vary T (SYNDATA)

Fig. 8: Algorithms FTM-EL vs. FTM-IC

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, and from 400 to $2,000$ for SYNDATA, respectively. We used the entire graphs for all datasets, and fixed $k = 10$. The results are reported in Figs. 8g-8i.

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 FTM-IC. This is consistent with the time complexity analysis. Moreover, FTM-EL is (1.03, 1.03, 1.36) times faster than FTM-IC on (BJDATA, EURDATA, SYNDATA) on average, respectively. It reveals that the larger T is, the faster FTM-EL is.

When varying the number T of timestamps, all three procedures in FTM increase with the increment of T. generateMaxTM takes the most time, on average (86%, 68%, 79%) for FTM-EL and (81%, 65%, 59%) for FTM-IC on (BJDATA, EURDATA, SYNDATA), respectively. computeRES using the EL table is on average (1.39, 1.12, 3.54) times faster than using IC trees on (BJDATA, EURDATA, SYNDATA), respectively. Hence, the time difference of FTM-EL and FTM-IC is mainly caused by computeRES.

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

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 (1.05, 0.98, 3.05) times faster than FTM-IC on (BJDATA, EURDATA, SYNDATA) on average, respectively. Note that FTM-EL takes more time than FTM-IC when $k \geq 30$ on EURDATA, as IC trees have small node number relative to the large timestamp on EURDATA (as shown in Table I), and FTM-IC is faster when there exists no maximum interval for an edge of the smaller temporal graph with larger k.

Exp-1.4. To evaluate the space cost, we tested the maximum and average value of $|E_m|$ and $|V_e|$ 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 I. Note that $|E_m|$ is the number of edges in temporal subgraph $G_s(\mathsf{S}[m, m + k - 1])$, and $|V_e|$ is the number of nodes in the IC tree of edge $e \in E$; $maxE_m$ is the largest $|E_m|$, and $maxV_e$ is the largest $|V_e|$.

$|E_m|$ is less than $|E|$ to various degrees, as edge labels in different datasets change with different frequencies. $|E|$ is (1.18, 1.11, 2.90) times of $maxE_m$ on (BJDATA, EURDATA, SYNDATA), respectively. In addition, $|V_e|$ is less than T to various degrees for the same reason. T is (10.17, 18.24, 4.17) times of $maxV_e$ on (BJDATA, EURDATA, SYNDATA), respectively. Moreover, FTM-EL uses 19% less memory than FTM-IC on SYNDATA, but uses (19%, 10%) more memory than FTM-IC on (BJDATA, EURDATA), respectively. The reason is that, $maxV_e$ is not much smaller than T on BJDATA and SYNDATA. These are consistent with the space complexity analysis. Finally, the memory cost of algorithms is rational, as SYNDATA is large enough.

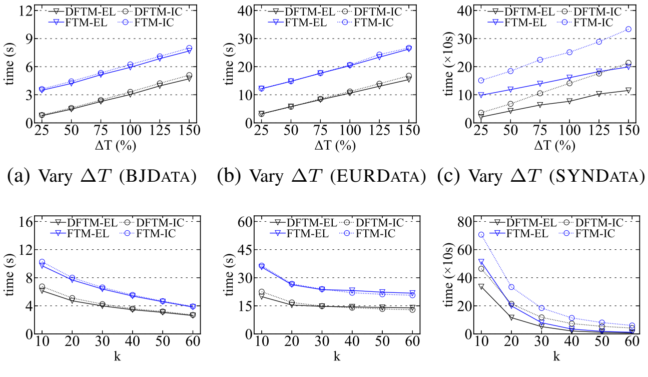
Exp-2: Tests of incremental algorithms. In the second set of tests, we test the running time and memory cost of incremental algorithms DFTM-EL and DFTM-IC *w.r.t.* the number ΔT of increased timestamps and the frequency threshold k, compared with our static algorithms FTM-EL and FTM-IC.

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

When varying the sizes of ΔT, the running time of all four algorithms increases with the increment of ΔT. Moreover, DFTM-EL is always the best, and becomes faster for larger temporal graphs. Incremental algorithms are always faster than their static counterparts, as well. DFTM-EL is (2.50, 2.31, 2.55) times faster than FTM-EL, and DFTM-IC is $(2.40, 2.29, 2.34)$ times faster than FTM-IC on (BJDATA,

TABLE II: Space cost of incremental algorithms ($k = 10$ and $\Delta T = 150\%$)

Datasets	Aver. $\lvert E_{m,TF}\rvert$	$maxE_{TF}$	Aver. $\lvert\mathsf{S}[m,T]\rvert$	Max. $\lvert\mathsf{S}[m,T]\rvert$	Memory cost (GB)	
					(DFTM-EL, FTM-EL)	(DFTM-IC, FTM-IC)
BJDATA (457 MB, $\lvert E\rvert = 88,396$)	$3,731$	$32,454$	$28,437$	$50,761$	(0.440, 0.452)	(0.367, 0.378)
EURDATA (3.15 GB, $\lvert E\rvert = 7,334$)	7	$4,887$	$2,553$	$5,254$	(10.235, 10.621)	(9.272, 9.667)
SYNDATA (30.6 GB, $\lvert E\rvert =$ 800K)	$1,045$	$147,542$	$2,953$	$197,092$	(22.076, 22.082)	(27.336, 27.348)



(a) Vary ΔT (BJDATA) (b) Vary ΔT (EURDATA) (c) Vary ΔT (SYNDATA)

(d) Vary k (BJDATA) (e) Vary k (EURDATA) (f) Vary k (SYNDATA)

Fig. 9: Incremental algorithms DFTM-EL and DFTM-IC

EURDATA, 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 Figs. 9d-9f.

When varying k, the running time of all algorithms decreases with the increment of k, along the same reason as Exp-1.3. Moreover, DFTM-EL is generally the best, but DFTM-IC becomes faster when given the smaller temporal graph and larger k (*i.e.,* $k = 60$ on BJDATA and EURDATA), which is consistent with their static counterparts. Incremental algorithms are always faster than their static counterparts.DFTM-EL is (1.58, 1.66, 1.62) times faster than FTM-EL, and DFTM-IC is (1.53, 1.60, 1.54) times faster than FTM-IC on (BJDATA, EURDATA, SYNDATA) on average, respectively.

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

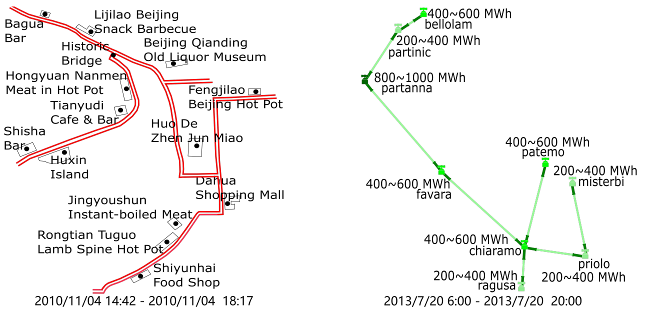
Parameter $|E_{m,TF}|$ is less than $|\mathsf{S}[m,T]|$ to various degrees, and the maximum $|\mathsf{S}[m,T]|$ is (1.56, 1.08, 1.34) times larger than $maxE_{TF}$ on (BJDATA, EURDATA, SYNDATA), respectively. The average of $|\mathsf{S}[m,T]|$ is (7.62, 364.71, 2.83) times larger than the average of $|E_{m,TF}|$ on (BJDATA, EURDATA, SYNDATA), respectively. It justifies the use of $\mathsf{TF}^{\circ}[m,T]$ instead of $\mathsf{S}[m,T]$ for $1 \leq m \leq T-k+1$. Moreover, DFTM-EL uses 19% less memory than DFTM-IC on SYNDATA, but uses (20%, 10%) more memory than DFTM-IC on (BJDATA, EURDATA), 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 may use 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 further conducted case studies on BJDATA and EURDATA. The following shows two cases from the result of FTM with $k = 10$.

Case 1. For BJDATA, we used the traffic data management system from [32] to visualize our temporal motifs and found corresponding roads in Google Maps. We use red, yellow and green colors to represent edges with congested, slow and fast traffic condition labels in the motif, respectively. 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 [61], [62], [68]–[70]. As Fig. 10a depicts, the motif has 19 nodes, 18 edges and the time interval length is 44. It corresponds to the traffic conditions in the *Shichahai scenic area at Beijing* during time interval [14:42, 18:17]. It is the large area with a long congested time involving with roads, *e.g.,* Houhai Beiyan, Qianhai Beiyan, Qianhai Dongyan, Di'anmen Outer Street, Baimi Byway and Fangzhuanchang Hutong. Observe that there are many restaurants or shops along these roads in Fig.10a, 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. We use dark green, green and light green colors to represent merging points of transmission lines with energy demand signals from 800 MWh to 1000 MWh, from 400 MWh to 600 MWh and from 200 MWh to 400 MWh in the motif, respectively. We use our temporal motifs to discover hybrid energy consumption patterns. As Fig. 10b depicts, the motif has 9 nodes, 8 edges and the time interval length is 15 (numbers on nodes represent dynamic energy demand signal levels). It corresponds to 9 merging points of transmission lines and their links in Italy during [2013/7/20 6:00, 2013/7/20 20:00]. It reveals the area with the staggering electricity usage in summer, as each merging point needs more than 200 MWh energy in 14 hours (the average energy needed for each merging point in EURDATA is only about 180 MWh). Some proper energy preservation and transmission strategies can be adopted to maintain good load balance of energy systems, such as energy transmission from the partinic region to the partanna region.

Summary. We have following findings. (1) Static algorithms FTM-EL and FTM-IC both run fast on large temporal graphs.



(a) Temporal motif with all congested labels at Beijing on BJData

(b) Temporal motif in Italy on EURData

Fig. 10: Case studies on real-life datasets

(2) FTM-EL is on average $(1.04, 1.02, 1.94)$ times faster than FTM-IC on datasets (BJData, EURData, SYNData), respectively, but when given a small temporal graph and a large frequency threshold k, FTM-IC may become faster because IC trees are faster when there exists no maximum interval for an edge of the small temporal graph with larger k. (3) Incremental algorithms DFTM-EL and DFTM-IC are $(2.03, 1.98, 2.09)$ and $(1.96, 1.94, 1.93)$ times faster than their static counterparts, respectively. In addition, DFTM-EL is generally faster than DFTM-IC, except for small temporal graphs and large frequency thresholds along the same reason as (2). (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.

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]–[7], [71], social networks [4], [5], [7]–[15], [71]–[73], electrical circuits [16], software architectures [17], and transport networks [5], [7], [18], [71]. These focus on network motifs in static networks. In this study, we focus on network motifs in temporal networks.

Network motifs in temporal networks. Network motifs in temporal networks have attracted growing attentions. These studies re-define temporal motifs to meet application demands, including social networks [38]–[43], [46], [47], [50], [52], [53], [55], communication networks [34]–[39], [47], [50], [53], transaction networks [38]–[43], [45], [47], [52], [53], transport networks [43], [46], [50] and biological networks [45], [51].

Some of these studies focus on temporal motifs with time-dependent edges. [54] firstly studies the dynamics of motifs in networks with time-dependent edges. Subsequent studies begin to define motifs with different time orders and different time windows on edges, *e.g.,* a partial time order [34]–[37], [50], a total time order [38]–[43], [53], [55], a local time window [34]–[37], [51], [52], [55], and a global time window [38]–[44], [50], [53], [55]. A recent survey [48] systematically evaluates the impact of temporal inducedness and timing constraints (*i.e.,* local or global time windows) in some of these temporal motifs.

Other these studies focus on temporal motifs with node weights or edge weights (labels) displaying statics or similar dynamics over a user-defined period. [45], [46] define motifs as induced subgraphs with node weights evolving in a consistent trend over the time interval. [47] defines induced relational states (IRS) as induced subgraphs with edge labels and directions keeping constant over the time interval. However, it uses subgraph enumeration to identify IRS in exponential time.

In conclusion, most these works focus on small motifs, and the discovery of these motifs is computationally intractable. Further, no incremental solutions have been investigated to meet the need of the dynamic nature of temporal networks.

Different definitions of temporal motifs 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 and precisely in polynomial time. In addition, we also develop incremental algorithms 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 [49], lasting dense subgraphs [74], graph association rules [75], graph temporal association rules [76], dense temporal graphs [20], [33], [56], frequent graph patterns [77] and frequent pattern mining [78].

The concepts of temporal graphs and subgraphs that we use essentially follow [33], [56], which were used to study the dense temporal graphs problem. Slightly different from their definition, we use edge labels, instead of edge weights, to represent the dynamic properties of temporal graphs.

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 topics need a further study. One is to relax the continuousness requirement of edges that keep the same labels in at least k snapshots, and the other is to investigate the discovery of temporal motifs in a distributed environment.

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] S. Gu, J. Johnson, F. E. Faisal, and T. Milenković, "From homogeneous to heterogeneous network alignment via colored graphlets," *Scientific reports*, vol. 8, no. 1, pp. 12 524:1–12 524:16, 2018.

[7] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield, "Efficient graphlet counting for large networks," in *ICDM*, 2015.

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

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

[10] R. A. Rossi, N. K. Ahmed, A. Carranza, D. Arbour, A. Rao, S. Kim, and E. Koh, "Heterogeneous graphlets," *TKDD*, vol. 15, no. 1, pp. 9:1–9:43, 2020.

[11] X. Hu, Y. Tao, and C.-W. Chung, "Massive graph triangulation," in *SIGMOD*, 2013.

[12] A. Pinar, C. Seshadhri, and V. Vishal, "Escape: Efficiently counting all 5-vertex subgraphs," in *WWW*, 2017.

[13] M. Bressan, F. Chierichetti, R. Kumar, S. Leucci, and A. Panconesi, "Counting graphlets: Space vs time," in *WSDM*, 2017.

[14] X. Chen, Y. Li, P. Wang, and J. C. Lui, "A general framework for estimating graphlet statistics via random walk," *PVLDB*, vol. 10, no. 3, pp. 253–264, 2016.

[15] 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.

[16] 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.

[17] 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.

[18] P. Kaluza, A. Kölzsch, M. T. Gastner, and B. Blasius, "The complex network of global cargo ship movements," *Journal of the Royal Society Interface*, vol. 7, no. 48, pp. 1093–1103, 2010.

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

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

[21] J. Chan, J. Bailey, C. Leckie, and M. Houle, "ciforager: Incrementally discovering regions of correlated change in evolving graphs," *TKDD*, vol. 6, no. 3, pp. 11:1–11:50, 2012.

[22] P. Bogdanov, C. Faloutsos, M. Mongiovì, E. E. Papalexakis, R. Ranca, and A. K. Singh, "Netspot: Spotting significant anomalous regions on dynamic networks," in *SDM*, 2013.

[23] M. Mongiovì, P. Bogdanov, and A. K. Singh, "Mining evolving network processes," in *ICDM*, 2013.

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

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

[26] 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.

[27] L. Foschini, J. Hershberger, and S. Suri, "On the complexity of time-dependent shortest paths," *Algorithmica*, vol. 68, no. 4, pp. 1075–1097, 2014.

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

[29] K. Semertzidis and E. Pitoura, "Durable graph pattern queries on historical graphs," in *ICDE*, 2016.

[30] S. Huang, A. W. Fu, and R. Liu, "Minimum spanning trees in temporal graphs," in *SIGMOD*, 2015.

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

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

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

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

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

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

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

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

[44] Y. Li, Z. Lou, Y. Shi, and J. Han, "Temporal motifs in heterogeneous information networks," in *MLG Workshop@ KDD*, 2018.

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

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

[47] 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.

[48] P. Liu, V. Guarrasi, and A. E. Sariyuce, "Temporal network motifs: Models, limitations, evaluation," *TKDE*, 2021, to be published.

[49] 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.

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

[51] 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.

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

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

[54] 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.

[55] 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.

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

[57] 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.

[58] 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.

[59] 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.

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

[61] C. Li, W. Yue, G. Mao, and Z. Xu, “Congestion propagation based bottleneck identification in urban road networks,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 4827–4841, 2020.

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

[63] N. Hutcheon and J. W. Bialek, “Updated and validated power flow model of the main continental european transmission network,” in *2013 IEEE Grenoble Conference*, 2013.

[64] 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.

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

[66] R. E. Tarjan, “Efficiency of a good but not linear set union algorithm,” *Journal of the ACM*, vol. 22, no. 2, pp. 215–225, 1975.

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

[68] V. W. Chu, R. K. Wong, W. Liu, and F. Chen, “Causal structure discovery for spatio-temporal data,” in *DASFAA*, 2014.

[69] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xing, “Discovering spatio-temporal causal interactions in traffic data streams,” in *SIGKDD*, 2011.

[70] R. Inoue, A. Miyashita, and M. Sugita, “Mining spatio-temporal patterns of congested traffic in urban areas from traffic sensor data,” in *ITSC*, 2016.

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

[72] Z. Zhao, G. Wang, A. R. Butt, M. Khan, V. A. Kumar, and M. V. Marathe, “Sahad: Subgraph analysis in massive networks using hadoop,” in *IPDPS*, 2012.

[73] L. Lai, L. Qin, X. Lin, and L. Chang, “Scalable subgraph enumeration in mapreduce: a cost-oriented approach,” *VLDB Journal*, vol. 26, no. 3, pp. 421–446, 2017.

[74] 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.

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

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

[77] 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.

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

A. Appendix: application of our temporal motifs

There are certain potential applications of our defined temporal motifs in temporal networks, where the nodes and edges are fixed, but the edge labels vary regularly with timestamps [1]–[4].

(1) Road traffic analyses. There are many sensors, cameras and floating cars in road networks collecting road traffic information periodically [5]. In contrast to dynamic traffic information, roads are relatively fixed, which can be naturally represented as structurally-static temporal networks with edge labels varying regularly with timestamps [2], [6], [7]. Here, edge labels can represent real-time reports of traffic condition on roads, such as congestion levels (*e.g.,* congested, slow, and fast). In such a temporal network, our temporal motifs can discover different kinds of traffic patterns, including long-time congestion patterns with all congested roads, congestion propagation patterns with bottlenecks and influenced roads and hybrid traffic condition patterns [8], [9]. Understanding these traffic patterns helps to improve the traffic conditions by rearranging traffic policemen or redesigning tidal lanes and traffic lights.

(2) Energy consumption analyses. There are many open energy systems collecting information of transmission networks, energy generation and demand signals [10]–[12]. A structurally-static temporal network can be built to represent the dynamics of those energy signals, where each transmission line corresponds to an edge, and the merging point of transmission lines corresponds to a node. Each edge or node can be labeled by the corresponding series of energy signal levels in the time interval of the temporal network. In such a temporal network, our temporal motifs can discover different kinds of energy consumption patterns, including long-term high energy consumption patterns and hybrid energy consumption patterns [5], [11], [12]. Long-term high energy consumption patterns correspond to regions with continuous high energy demand signal levels, and hybrid energy consumption patterns correspond to regions with both continuous high and low energy demand signal levels. Understanding these energy consumption patterns helps to reveal the implied correlation among energy demand, regions and seasons, and to design proper energy preservation and transmission strategies.

(3) Flood risk analyses. There are many wireless sensor network systems with different types of sensors recording significant environment information, including the water discharge from dam, rainfall, humidity and temperature [13]–[15]. With the combination of the geographical information system [14], [15], a structurally-static temporal network can be built, where the sub-region corresponds to a node and each adjacency relation between sub-regions corresponds to an edge. Each edge can be labeled by the corresponding series of water peak discharge levels, storm levels, etc, which are typically averaged over each sub-region [13]. In such a temporal network, our temporal motifs can discover different kinds of environment patterns with continuous low or high water and weather signal levels. Understanding these patterns helps to identify flood risk areas in order to take necessary actions before the flood occurs [15], and to alleviate the risk by redesigning the water discharge from dam.

B. Appendix: correctness proofs of temporal motif analyses

Proof of Proposition 1: Assume that there exists an edge e in R[m, i] and R[m,j] ($m+k-1 \leq i < j \leq T$). By the definition, we have $e \in \mathsf{S}[m,i] - \mathsf{S}[m,i+1]$ and $e \in \mathsf{S}[m,j] - \mathsf{S}[m,j+1]$ or $\mathsf{S}[m,T]$ if $j = T$. However, as $i < j$ and $\mathsf{S}[m,j] \subseteq \mathsf{S}[m,i+1]$, $e \in \mathsf{S}[m,i+1]$. This implies $e \notin \mathsf{R}[m,i]$, which is a contradiction. Hence we have the conclusion. □

Proof of Proposition 2: 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 [16]. Hence we have the conclusion. □

Proof of Proposition 3: Assume that we are generating the maximal temporal motifs for interval $[m,i]$ with edge set $\mathsf{R}[m,i]$ ($i \in [m+k-1, T]$). By the definition of R edge sets, any edge in $\mathsf{R}[m,i]$ changes its label at timestamp $i+1$. Hence, we have the conclusion. □

C. Appendix: correctness proofs of procedure computeRES

Proof of Proposition 4: We assume that edge e has the same label in each of the intervals $[H_1 = 1, U_1], \cdots, [H_n, U_n = T]$ such that $H_{i+1} = U_i+1$ for $1 \leq i < n$, and that e has different labels in any two adjacent intervals. The length of $[H_i, U_i]$ is also denoted as l_i for $1 \leq i \leq n$. Note that $n = |V_e|$ is the number of nodes for the IC tree of e.

There are two cases for $[H_i, U_i]$ during the computation of R edge sets.

(1) Case $l_i \geq k$. For the H_i-th row, the EL table or the IC tree of e needs to be used (cases (1) and (2)), which takes $l_i - k$ and $\log n$ time, respectively. For rows from $H_i + 1$ to U_i, $\mathsf{EMaxIntvl}[e]$ is used once (case (4)), which takes constant time 1 for each row.

(2) Case $l_i < k$. For rows from H_i to U_i, the EL table or IC tree of e needs to be used (cases (1) and (2)), which takes constant time 1 and $\log n$ time for each row, respectively.

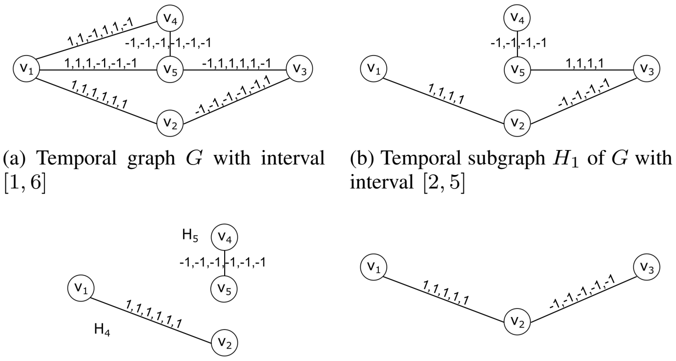
For the EL table, the worst case for dealing with $[H_i, U_i]$ is $2l_i - k - 1$ time, and the following holds for T rows: $\sum_{i=1}^{n}(2l_i - k - 1) < \sum_{i=1}^{n}(2l_i) = 2T$. For IC trees, the worst case for dealing with $[H_i, U_i]$ is $l_i \log n$ time, and the following holds for T rows: $\sum_{i=1}^{n}(l_i \log n) = T \log n$. As there are in total T rows, we have the conclusion. □

D. Appendix: detailed process of algorithm FTM

The complete algorithm FTM is shown in Fig.1, which takes as input a temporal graph $G(V, E, 1, T, L)$ and a frequency

```
Algorithm FTM
Input: Temporal graph $G(V, E, 1, T, L)$ and frequency threshold $k$.
Output: Set TF of all the maximal and non-expandable temporal motifs.
1.  let ds be the EL table or IC trees for $G$;
2.  EMaxIntvl[$e$] := null for each $e$;
3.  for $m := 1$ to $T - k + 1$        /*T2B*/
4.    (R, EMaxIntvl) := computeRES(ds, $k$, EMaxIntvl, $m$);
5.    CC[$i, T$] := $\emptyset$ for $i \in [m + k - 1, T + 1]$;
6.    for $i := T$ to $m + k - 1$       /*R2L/
7.      CC[$i, T$] := generateMaxTM(R, EMaxIntvl, [$m, i$],
          CC[$i + 1, T$]);    /*compute maximal
          motifs with the interval [$m, i$]*/
8.      TF[$m, i$] := generateExpTM(CC[$i, T$], [$m, i$]);
9.  return TF.
```

Fig. 1: Algorithm FTM



(a) Temporal graph G with interval $[1, 6]$

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

(c) Temporal subgraphs H_4 and H_5 of G with interval $[1, 6]$

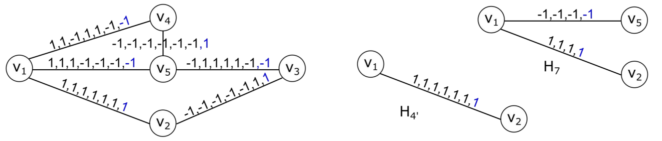
(d) Temporal subgraph H_6 of G with interval $[1, 5]$

Fig. 2: Running example

threshold k, and returns the set TF of all the maximal and non-expandable temporal motifs. It first creates the EL table or IC trees (referred to as ds) (line 1), and initializes the array EMaxIntvl to be empty (line 2). Then it adopts the T2B scheme to deal with all rows m from 1 to $T - k + 1$ (line 3). For each row m, it invokes computeRES to obtain the edge set R[m, i] for $m + k - 1 \leq i \leq T$ and the updated array EMaxIntvl (line 4), and initializes connected components CC[i, T] for $i \in [m+k-1, T+1]$ (line 5). Note that we use CC[$T+1, T$] to represent an empty set for convenience. After that, it adopts the R2L scheme with columns i from T to $m+k-1$ (line 6). For each i, it invokes generateMaxTM to obtain the updated connected component CC[i, T] for interval $[m, i]$ (line 7). Each connected component CC[i, T] corresponds to a maximal temporal motif. Then it invokes generateExpTM to obtain the set TF[m, i] of maximal and non-expandable temporal motifs with interval $[m, i]$ (line 8). Finally, it returns the set TF of all the maximal and non-expandable temporal motifs.

E. Appendix: detailed examples of algorithms FTM and DFTM

Example of algorithm FTM. We consider the temporal graph G in Fig. 2a and frequency threshold $k = 4$. The corresponding TI-Table consists of 6 intervals in total, *i.e.,* $\{([1, 4], [1, 5], [1, 6]), ([2, 5], [2, 6]), ([3, 6])\}$.



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

(b) Temporal motifs $H_{4'}$ and H_7 of G' with interval $[1, 7]$ and $[4, 7]$, respectively

Fig. 3: Running example for incremental algorithm

(1) For row $m = 1$, computeRES returns R$[1, 4] = \emptyset$, R$[1, 5] = \{(v_2, v_3)\}$, R$[1, 6] = \{(v_1, v_2), (v_4, v_5)\}$, and EMaxIntvl$[(v_1, v_2), (v_4, v_5), (v_2, v_3)] = [[1, 6], [1, 6], [1, 5]]$. (a) For $i = 6$: generateMaxTM fetches edges from R$[1, 6]$, generates ccs $G_{s_1}(\{(v_1, v_2)\})$ and $G_{s_2}(\{(v_4, v_5)\})$ both with interval $[1, 6]$ (case (1)), and generateExpTM adds them to TF (H_4 and H_5 in Fig. 2c, respectively) as the beginning timestamp of $[1, 6]$ is equal to $m = 1$. (b) For $i = 5$: generateMaxTM adds edges from R$[1, 5]$ to G_{s_1}, and generates cc G_{s_3} with interval $[1, 5]$ (case (2)), which is added to TF with generateExpTM (H_6 in Fig. 2d) as $m = 1$. (c) For $i = 4$: the algorithm does nothing as R$[1, 4] = \emptyset$.

(2) For row $m = 2$, computeRES 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} and G_{s_2} with interval $[1, 6]$ are generated, which are left expandable ($1 < m = 2$). (b) For $i = 5$: generateMaxTM combines G_{s_1}, G_{s_2} and edges from R$[2, 5]$ to generate a cc G_{s_4}, and updates its interval with $[2, 5]$ (case (3)). As the beginning timestamp of $[2, 5]$ is $2 = m$, generateExpTM adds the cc G_{s_4} to TF (H_1 in Fig.2b).

(3) For row $m = 3$, computeRES returns R$[3, 6] = \{(v_1, v_2), (v_4, v_5)\}$ and keeps EMaxIntvl unchanged. G_{s_1} and G_{s_2} with interval $[1, 6]$ are generated again, which are left expandable ($1 < m = 3$).

Finally, algorithm FTM returns the set TF = $\{H_1, H_4, H_5, H_6\}$ of maximal and non-expandable temporal motifs.

Example of algorithm DFTM**.** Consider the updated temporal graph G' in Fig. 3a of temporal graph G in Fig. 2a with $T = 6$ and $\Delta T = 1$, frequency threshold $k = 4$, and the set TF° of maximal and non-expandable temporal motifs for G computed by FTM in Example E. The updated TI-Table with $T + \Delta T = 7$ 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 $m \in [1, 2]$, TF$[m, i]$ = TF°$[m, i]$ for $i \in [m+3, 5]$, *i.e.,* the maximal and non-expandable temporal motifs keep unchanged for intervals $\{[1, 4], [1, 5], [2, 5]\}$.

(2) For row $m \in [1, 3]$ and $i \in [6, 7]$, sets TF°$[m, 6]$ are used to compute edge sets R$[m, 6]$ and R$[m, 7]$. Then DFTM invokes procedures generateMaxTM and generateExpTM to obtain maximal and non-expandable temporal motifs $H_{4'}$ with interval $[1, 7]$ and H_5 with interval $[1, 6]$ shown in Figs. 3b & 2c, respectively. Note that H_4 in TF°$[1, 6]$ is right expandable after updates.

(3) For row $m = 4$, there is only one interval $[4, 7]$, and DFTM finds maximal and non-expandable temporal motif H_7 with interval $[4, 7]$ in Fig. 3b, which follows exactly the same way as FTM.

Finally, algorithm DFTM returns the set TF = $\{H_1, H_{4'}, H_5, H_6, H_7\}$.

F. Appendix: correctness proofs of algorithm FTM

Proof of Proposition 5: We show this by a loop invariant.

Loop invariant: before the start of each row m (line 3), algorithm FTM finds all the maximal and non-expandable temporal motifs for all intervals in rows from 1 to $(m-1)$ in the TI-Table.

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 also 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. □

G. Appendix: correctness proofs of algorithm DFTM

Proof of Proposition 6: We show this by proof by induction.

(1) For row $m = 1$, some edges of any temporal motif G_s in $\mathsf{TF}^\circ[1, T-1]$ 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 $\mathsf{S}[1, T]$ and $\mathsf{R}[1, T-1]$ keep unchanged after update, 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 $\mathsf{TF}^\circ[m+1, T-1]$ 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 $\mathsf{S}[m+1, T]$ and $\mathsf{R}[m+1, T-1]$ keep unchanged after update, the conclusion holds for $m+1$.

Putting these together, we have the conclusion. □

Proof of Proposition 7: This is proved by showing that for $m \in [1, T-k+1]$, if edge $e \in \mathsf{S}[m, T]$ and $e \notin \mathsf{TF}^\circ[m, T]$, then $e \notin \mathsf{TF}[m, i]$ for $i \geq T$.

(1) It is obvious that the edges in $\mathsf{TF}[m, i]$ must belong to $\mathsf{S}[m, T]$. (2) Assume that edge $e \in \mathsf{TF}[m, i]$ for $i \geq T$. As $e \in \mathsf{TF}[m, i]$, e belongs to a maximal and non-expandable motif in $\mathsf{TF}[m, i]$ that consists of edges e, $\cdots$, e_h $(h \geq 0)$. Then we know e, $\cdots$, $e_h \in \mathsf{S}[m, i] \subseteq \mathsf{S}[m, T]$. It implies that $e, \cdots, e_h$ belong to a maximal and non-expandable motif in $\mathsf{TF}^\circ[m, T]$. Therefore, edge $e \in \mathsf{TF}^\circ[m, T]$, which is a contradiction.

Putting these together, we have the conclusion. □

H. Appendix: detailed algorithm complexity analyses

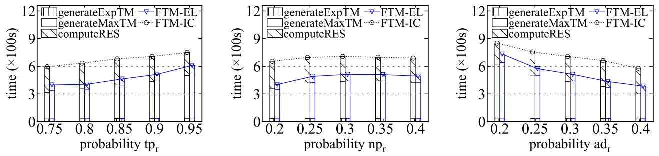
(1) Time complexity of the procedure generateMaxTM. Let $|E_m|$ be the number of edges in temporal subgraph $G_s(\mathsf{S}[m, m+k-1])$. Note that the dynamic connectivity checking for an edge with the ccs in $\mathsf{CC}[i, T]$ $(i \in [m+k-1, T])$ utilizes disjoint sets [17]. For each e, cases (1) and (2) take $O(1)$ constant time, and case (3) takes $O(\min(|E_s|, |E_{s'}|))$ time, where $|E_s|$ and $|E_{s'}|$ are the numbers of edges in ccs G_s and $G_{s'}$, respectively. The worse case is that all edges in $\mathsf{S}[m, m+k-1]$ form a single connected component. This implies that cases (1), (2) and (3) all take $O(|E_m|)$ time. Therefore, it takes generateMaxTM $O(|E_m|)$ time to generate all the maximal temporal motifs for the intervals in the m-th row in the TI-Table.

(2) Time complexity of the procedure generateExpTM. Let $|E_m|$ be the number of edges in temporal subgraph $G_s(\mathsf{S}[m, m+k-1])$. In the worse case, each edge leads to a newly generated connected component. Hence, there are at most $|E_m|$ newly generated connected components for the m-th row. As the checking takes $O(1)$ constant time for each newly generated connected component, we know that procedure generateExpTM takes $O(|E_m|)$ time to generate all maximal and non-expandable temporal motifs for the intervals in the m-th row.

(3) Time complexity of the algorithm FTM. Let $|E_m|$ be the number of edges in temporal subgraph $G_s(\mathsf{S}[m, m+k-1])$, and let $|V_e|$ be the number of nodes in the IC tree of edge $e \in E$. We also denote $maxE_m$ as the largest number of edges in all temporal subgraphs $G_s(\mathsf{S}[m, m+k-1])$, $maxV_e$ as the largest number of nodes in all IC trees. Note that here $|E_m| \leq |E|$ for all $m \in [1, T-k+1]$ and $maxE_m$ is obviously smaller than $|E|$.

For each row m, (1) procedure computeRES takes $O(|E|)$ and $O(|E| \log maxV_e)$ time, respectively, when using the EL table and IC trees, (2) procedure generateMaxTM takes $O(|E_m|)$ time, and (3) procedure generateExpTM takes $O(|E_m|)$ time. There are in total $O(T-k+1) = O(T)$ rows in the TI-Table. Hence, algorithms FTM-EL and FTM-IC take $O(T|E|)$ and $O(T|E| \log maxV_e)$ time, respectively. Note that the motifs in the TF have $O(T^2)$ distinct intervals, and the motifs with the same interval contain $O(|maxE_m|)$ edges in total. Therefore, though we only need $O(T|E|)$ or $O(T|E| \log maxV_e)$ time to obtain TF, we take $O(T^2|maxE_m|) = O(T^2|E|)$ worst-case time to output all temporal motifs from TF.

(4) Space complexity of the algorithm FTM. The space cost of FTM is dominated by its key data structures. (1) For IC



(a) Vary tp_r (SYNDATA) (b) Vary np_r (SYNDATA) (c) Vary ad_r (SYNDATA)

Fig. 4: Algorithms FTM-EL vs. FTM-IC

trees, they cost $O(maxV_e|E|)$ space, as each IC tree costs $O(maxV_e)$ space. (2) For the EL table, it costs $O(T|E|)$ space. (3) For array EMaxIntvl, it costs $O(|E|)$ space for all edges. (4) For connected components, they cost $O(|E|)$ space in total, as only one copy is maintained for all CC$[i,T]$ ($i \in [m+k-1,T]$). (5) For the set TF of maximal and non-expandable temporal motifs, it can only cost $O(TmaxE_m)$ space. For each row m, we can save the motifs with the interval $[m,T]$ as all edges from ccs in CC$[T,T]$ that need to be saved (*i.e.,* edges in R$[m,T]$), but save the motifs with the interval $[m,i]$ as all the unsaved edges from newly generated ccs in CC$[i,T]$, $i \in [m+k-1,T-1]$ that need to be saved (*i.e.,* edges in R$[m,i']$, $i' \in [m+k-1,T]$) and the pointers to the motifs with already saved edges. As there are in total $O(maxE_m)$ edges in S$[m,m+k-1]$, there are at most $O(maxE_m)$ pointers to distinct saved motifs. Therefore, TF only cost $O(maxE_m)$ space to save edges and pointers for each row.

From the above mentioned, FTM-EL and FTM-IC take $O(T|E| + TmaxE_m)$ and $O(maxV_e|E| + TmaxE_m)$ space, respectively.

(5) Time complexity of the algorithm DFTM. (a) For each row $m \in [1, T+k-1]$, procedure computeRES takes $O(|E_{m,TF}|)$ and $O(|E_{m,TF}| \log\ maxV_e)$ time when using the EL table and IC trees, respectively, and procedures generateMaxTM and generateExpTM take $O(|E_{m,TF}|)$ time. Hence, this part in total takes $O(TmaxE_{TF})$ and $O(T\ maxE_{TF} \log maxV_e)$ time, respectively, when using the EL table or IC trees. (b) For each row $m \geq T-k+2$, the time complexity is the same as the static algorithm. Hence, this part in total takes $O(\Delta T|E|)$ and $O(\Delta T|E| \log maxV_e)$ time, respectively, when using the EL table or IC trees.

Therefore, incremental algorithms DFTM-EL and DFTM-IC take $O(TmaxE_{TF} + \Delta T|E|)$ and $O((TmaxE_{TF} + \Delta T|E|) \log\ maxV_e)$ time, respectively.

I. Appendix: extra experimental tests

Exp-1.5. We varied tp_r from 0.75 to 0.95 while fixed $k = 10$, and used the entire temporal graphs for SYNDATA. Note that tp_r controls the probability of each activated edge that activates its copy in the next snapshot, which means that it controls the frequency of label changing for each edge. The result is reported in Fig. 4a.

When varying the probability tp_r, the running time of two algorithms increases with the increment of tp_r. The reason is that for larger probability tp_r, edges change their labels less frequently, and larger temporal motifs are generated due to larger $|E_m|$. Moreover, FTM-EL is 1.43 times faster than FTM-IC on average.

When varying the probability tp_r, all three procedures in FTM-EL, generateMaxTM and generateExpTM in FTM-IC increase with the increment of tp_r, as larger temporal motifs are generated. However, computeRES in FTM-IC decreases with the increment of tp_r, as IC trees have fewer nodes and are faster when edges change their labels less frequently. Further, generateMaxTM takes the most time which occupies 79% and 54% time in FTM-EL and FTM-IC on average, respectively.

Exp-1.6. To evaluate the impacts of probability np_r, we varied np_r from 0.2 to 0.4, and used the same setting as Exp-1.5. Note that np_r controls the probability of each activated edge that activates its neighboring edges in the next snapshot, which means that it has indirect influence on the frequency of label changing for edge. The result is reported in Fig. 4b.

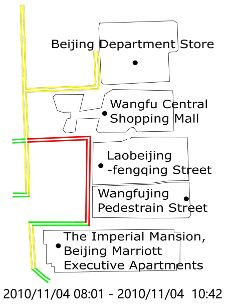
When varying the probability np_r, the running time of two algorithms increases with the increment of np_r when $np_r \leq 30$, and slightly decreases when np_r is larger. The reason is that for larger probability np_r, edges are more likely to activate their neighboring edges in the next snapshot, and help each activated neighbor edge to activate its copy in the next snapshot, which results in larger $|E_m|$ for each row m and more temporal motifs. However, when np_r is enough large (*i.e.,* $np_r > 30$), each edge exerts more influence on its neighboring edges in the next snapshot, which results in larger but less temporal motifs. Moreover, FTM-EL is 1.44 times faster than FTM-IC on average.

When varying the probability np_r, all three procedures in FTM-EL, generateMaxTM and generate- ExpTM in FTM-IC increase, and computeRES in FTM-IC decreases with the increment of np_r, along the same reason as Exp-1.5. Further, generateMaxTM takes the most time which occupies 79% and 55% time in FTM-EL and FTM-IC on average, respectively.

Exp-1.7. To evaluate the impacts of activation density ad_r, we varied ad_r from 0.2 to 0.4, and used the same setting as Exp-1.5. Note that ad_r controls the percentage of activated edges (labels are 1), which means it controls the proportion of different labels. The result is reported in Fig. 4c.

When varying the probability ad_r, the running time of two algorithms decreases with the increment of ad_r. The reason is that for larger percentage ad_r of activated edges, the generator needs to activate more edges in the first snapshot, which causes edges to change their labels more frequently and smaller temporal motifs to be generated due to more activated neighbor edges and decayed probabilities np_r and tp_r. Moreover, FTM-EL is 1.38 times faster than FTM-IC on average.

When varying the probability ad_r, all three procedures in FTM-EL, generateMaxTM and generate- ExpTM in FTM-IC decrease with the increment of ad_r, as smaller temporal motifs are generated. However, computeRES in FTM-IC increases with the increment of ad_r, as IC trees have more nodes and are slower when edge labels change more frequently. Further, generateMaxTM takes the most time which occupies 79% and 57% time in FTM-EL and FTM-IC on average, respectively.



(a) Temporal motif with hybrid traffic condition labels at Beijing on BJDATA

Fig. 5: Case studies on real-life datasets

Case study: temporal motif with hybrid labels.

We use our temporal motifs to discover hybrid traffic condition patterns from the result of the algorithm FTM with $k = 10$. As Fig. 5a depicts, the motif has 15 nodes, 14 edges and the time interval length is 33. It corresponds to the traffic conditions in the *Wangfujing shopping area at Beijing* during time interval [08:01, 10:42]. It is the large area with unchanged traffic conditions for a long time involving with roads, *e.g.,* Chenguang Street, Datianshuijing Hu Tong, Wangfujing West Street, South Koudai Hu Tong, Daruanfu Hu Tong and Xiagongfu Street. Observe that there are two congested bottleneck roads near Wangfu Central Shopping Mall and Laobeijingfengqing Street in Fig.5a, which cause a traffic jam due to their popularity. Other roads with slow labels are influenced by the traffic jam. Note that, there are some roads with fast labels in the pattern in Fig.5a, which indicates potential solutions to improve the traffic conditions, such as arranging traffic policemen to guide vehicles to roads near The Imperial Mansion, Beijing Marriott Executive Apartments.

REFERENCES

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

[2] M. Mongiovì, P. Bogdanov, and A. K. Singh, "Mining evolving network processes," in *ICDM*, 2013.

[3] 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.

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

[5] 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.

[6] 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.

[7] L. Foschini, J. Hershberger, and S. Suri, "On the complexity of time-dependent shortest paths," *Algorithmica*, vol. 68, no. 4, pp. 1075–1097, 2014.

[8] C. Li, W. Yue, G. Mao, and Z. Xu, "Congestion propagation based bottleneck identification in urban road networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 4827–4841, 2020.

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

[10] N. Hutcheon and J. W. Bialek, "Updated and validated power flow model of the main continental european transmission network," in *2013 IEEE Grenoble Conference*, 2013.

[11] 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.

[12] 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.

[13] S. Rozalis, E. Morin, Y. Yair, and C. Price, "Flash flood prediction using an uncalibrated hydrological model and radar rainfall data in a mediterranean watershed under changing hydrological conditions," *Journal of hydrology*, vol. 394, no. 1-2, pp. 245–255, 2010.

[14] V. Seal, A. Raha, S. Maity, S. K. Mitra, A. Mukherjee, and M. K. Naskar, "A real time multivariate robust regression based flood prediction model using polynomial approximation for wireless sensor network based flood forecasting systems," in *International Conference on Computer Science and Information Technology*, 2012.

[15] N. Ahmad, M. Hussain, N. Riaz, F. Subhani, S. Haider, K. S. Alamgir, and F. Shinwari, "Flood prediction and disaster risk analysis using gis based wireless sensor networks, a review," *Journal of Basic and Applied Scientific Research*, vol. 3, no. 8, pp. 632–643, 2013.

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

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