RESEARCH-ARTICLE

An Edge-Based Decomposition Framework for Temporal Networks

LUTZ OETTERSHAGEN, University of Liverpool, Liverpool, Merseyside, U.K.

ATHANASIOS L KONSTANTINIDIS, University of Ioannina, Ioannina, Epirus, Greece

GIUSEPPE F. ITALIANO, LUISS University, Rome, Italy

Published: 10 March 2025

Citation in BibTeX format

WSDM '25: The Eighteenth ACM International Conference on Web Search and Data Mining
March 10 - 14, 2025
Hannover, Germany

Conference Sponsors:
SIGKDD
SIGMOD
SIGIR
SIGWEB

WSDM '25: Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining (March 2025)
https://doi.org/10.1145/3701551.3703556
ISBN: 9798400713293

An Edge-Based Decomposition Framework for Temporal Networks

Lutz Oettershagen
lutz.oettershagen@liverpool.ac.uk
University of Liverpool
Liverpool, UK

Athanasios L. Konstantinidis
a.konstantinidis@uoi.gr
University of Ioannina
Ioannina, Greece

Giuseppe F. Italiano
gitaliano@luiss.it
LUISS University
Rome, Italy

Abstract

A temporal network is a dynamic graph where every edge is assigned an integer time label that indicates at which discrete time step the edge is available. We consider the problem of hierarchically decomposing the network and introduce an edge-based decomposition framework that unifies the core and truss decompositions for temporal networks while allowing us to consider the network's temporal dimension. Based on our new framework, we introduce the (k, Δ)-core and (k, Δ)-truss decompositions, which are generalizations of the classic k-core and k-truss decompositions for multigraphs. Moreover, we show how (k, Δ)-cores and (k, Δ)-trusses can be efficiently further decomposed to obtain spatially and temporally connected components. We evaluate the characteristics of our new decompositions and the efficiency of our algorithms. Moreover, we demonstrate how our (k, Δ)-decompositions can be applied to analyze malicious content in a Twitter network to obtain insights that state-of-the-art baselines cannot obtain.

CCS Concepts

• **Theory of computation → Graph algorithms analysis**; • **Information systems → Social networks**.

Keywords

Temporal graphs, Core decomposition, Truss decomposition

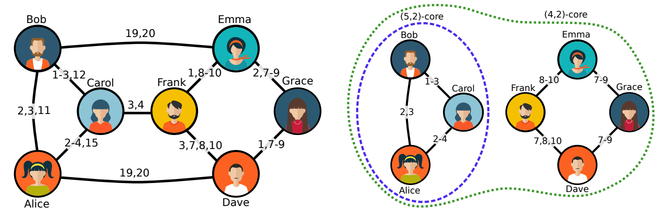
ACM Reference Format:
Lutz Oettershagen, Athanasios L. Konstantinidis, and Giuseppe F. Italiano. 2025. An Edge-Based Decomposition Framework for Temporal Networks. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining (WSDM '25), March 10–14, 2025, Hannover, Germany*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3701551.3703556

1 Introduction

Temporal networks are ubiquitous and have gained increasing attention in recent years due to their ability to capture the dynamic nature of real-world systems [16, 17, 29, 32, 34, 45, 50]. A temporal network consists of a set of nodes and a set of temporal edges. Each temporal edge exists only at a specific point in time, and the network's topology usually changes over time [13, 16]. In many applications, the temporal aspect is crucial for understanding the evolution and properties of the considered systems [10, 12, 15, 20, 28, 36, 38, 44, 48]. A common task in data mining is to decompose a network into cohesive components, and the k-core and k-truss decompositions are useful and popular primitives for this task [22, 27]. A *k-core* in a static graph G is a maximal subgraph G_k of G, such that every node in G_k has at least k neighbors in G_k [47]. Moreover, a *k-truss* in a static network is an edge-induced subgraph in which each edge is part of at least $k - 2$ triangles [8]. Both the k-core and k-truss decompositions find many applications in, e.g., social network analysis [48, 49], community detection [18, 43], and network visualization [1]. However, the static decompositions fail to decompose temporal networks in a useful way as they ignore the temporal aspects. For example, a node or edge that is at one point in time in an inner core or truss can be later at the periphery, i.e., in an outer core or truss, respectively.

WSDM '25, March 10–14, 2025, Hannover, Germany

ACM ISBN 979-8-4007-1329-3/25/03
https://doi.org/10.1145/3701551.3703556



(a) A temporal communication network. The edge labels denote the times of communication. **(b) The edge-induced subgraph containing the two most inner cores for $\Delta = 2$.**

Figure 1: Example of our edge-based core decomposition.

In this work, we consider the problem of decomposing a given temporal network into hierarchically arranged *temporal components* (cores or trusses). To this end, we first propose an edge-based decomposition framework to capture the temporal dynamics determined by the timestamps of the temporal edges. Based on our framework, we introduce the (k, Δ)-core and (k, Δ)-truss decompositions for temporal networks. Each edge in a (k, Δ)-core appears in the context of k temporal edges with a temporal distance of at most Δ. Specifically, each edge appearing at time t is at both endpoints incident to at least k temporal edges whose timestamps are at most Δ timesteps before or after time t. Similarly, in an (k, Δ)-truss, each edge appears in the temporal context of k temporal triangles in which the pairwise temporal distances of the edges are upper bounded by Δ. For example, Figure 1a shows a toy social network in which the timestamps of interactions between users are shown at each edge. Figure 1b shows the hierarchical decomposition into two edge-induced temporal components, i.e., the $(5, 2)$- and $(4, 2)$-cores of the network.

Many previously proposed variants of temporal network decompositions [2, 11, 25, 26, 31, 36, 51–53] show limitations in different ways, e.g., by focusing only on determining static k-cores of

k-trusses in temporal intervals, being not efficient enough for large-scale networks, or being not designed to work on highly dynamic temporal networks. Our work aims to overcome these restrictions.

1.1 Our Contributions

1. Edge-based decomposition framework: We introduce a framework for edge-based decompositions of temporal networks. Based on this framework, we introduce our new (k, Δ)-core and (k, Δ)-truss decompositions to decompose temporal networks into hierarchically organized temporal cores or trusses, respectively. We provide efficient algorithms for computing the (k, Δ)-decompositions.
2. Connected components: A temporal (k, Δ)-core or (k, Δ)-truss is, in general, not "connected". However, there is no standard definition of *connectedness* in temporal graphs [5, 7, 9, 23, 33]. We show that a variant of temporal reachability by Kovanen et al. [23] can be used to identify Δ*-connected components* of (k, Δ)-cores and (k, Δ)-trusses. The decomposition can be computed in linear time and is useful because the connected components of a (k, Δ)-core, or (k, Δ)-truss, are themselves (non-maximal) (k, Δ)-cores, or (k, Δ)-trusses.
3. Evaluation: We analyze our algorithms, compare them to existing state-of-the-art baselines, and show that our algorithms can efficiently handle large-scale and highly dynamic temporal networks for which the baselines fail. Finally, we demonstrate in a use-case how the (k, Δ)-core decomposition can be applied to analyze malicious content in a Twitter network to obtain insights that state-of-the-art baselines cannot obtain.

Please refer to [35] for the appendix and the omitted proofs.

1.2 Related Work

Comprehensive overviews of temporal graphs are provided by [16, 50]. Moreover, there are thorough surveys and introductions of core decompositions [22, 27]. Our decomposition framework is inspired by Batagelj and Zaveršnik [4] who generalized the static k-core decomposition using monotone vertex property functions. Recently, there has been an increasing interest in decompositions of temporal networks— Table 1 shows an overview of related decompositions. Wu et al. [51] propose a (k, h)-core decomposition where each vertex in a (k, h)-core has at least k neighbors, and there are at least h temporal edges to each neighbor. The authors of [2] extend the previous approach for maintaining the temporal (k, h)-cores under dynamic updates. Galimberti et al. [11] introduced temporal (k, I)-span-cores where each core is associated with a time interval I for which the coreness property holds in each time step of the interval I. There is a quadratic number of time intervals for which a temporal span-core can exist, and the asymptotic running times of the proposed algorithms are in $O(|\mathcal{T}|^2 \cdot |E|)$ where $\mathcal{T}$ is the interval spanned by the temporal graph. Hung et al. [19] define a temporal community as a (L, K)-lasting core. A (L, K)-lasting core is a set of vertices that forms a k-core with $k \geq K$ that lasts for time L. The definition is similar to the one of the (k, I)-spanning core; however, here, the length of the interval I is specified by L. Note that for both the (L, K)-lasting core and the (k, I)-spanning cores, the requirement that the k-core has to exist in each time step of the interval is often too restrictive as many real-world temporal networks are layer-wise sparse. Qin et al. [42] address temporal communities as (l, δ)-maximal dense core,

Table 1: Overview of related decompositions, with n and m being the number of nodes and temporal edges, resp., m_I (m_A) being the temporal edges in the interval I (the aggregated graph, resp.), $\mathcal{T}$ is the interval spanned by the temporal graph, δ_m is the max. degree.

Variant	Reference	Running Time	Description
Static k-core	[47]	$O(n+m)$	static cores
Static k-truss	[8]	$O(m^{1.5})$	static trusses
Historical k-core	[53]	$O(\log m + m_I)$	static cores spanning fixed interval
Time-range k-core	[52]	$O(\log m + \lvert I \rvert \cdot m_I)$	static cores in fixed interval
(k, h)-core	[51]	$O(n+m)$	parallel temporal edges
Span-core	[11]	$O(\lvert\mathcal{T}\rvert^2 \cdot m)$	cores spanning intervals
Span-truss	[26]	$O(\lvert\mathcal{T}\rvert^2 \cdot m^{1.5})$	trusses spanning intervals
(η, k)-pseudocore	[36]	$O(m\eta \cdot \delta_m)$	based on temporal H-index
(L, K)-lasting core	[19]	$O(nm^2 \cdot L)$	$k \geq K$-core lasting L steps
(l, δ)-dense core	[42]	$O(n \cdot \lvert\mathcal{T}\rvert)$	min. average degree in interval
(μ, τ, ϵ)-stable core	[41]	$O(m \cdot m_A)$	min. # similar neighb. in interval
(θ, τ)-persistent k-c.	[24]	NP-hard	persistence in sliding window
$(\mathbf{k}, \Delta)$-core	**This work**	$O(m \cdot \delta_m)$	based on temporal edge degree
$(\mathbf{k}, \Delta)$-truss		$O(m \cdot \delta_m^2)$	based on temporal edge support

which requires a core to maintain an average degree of at least δ throughout a time interval lasting no less than l units. However, their proposed solution is infeasible for networks spanning a long interval $\mathcal{T}$ due to the space complexity in $O(\alpha|\mathcal{T}| + |\mathcal{E}|)$ with α being the maximum number of nodes in a core. Oettershagen et al. [36] introduces a (non-hierarchical) decomposition of temporal networks into so-called (η, k)-pseudocores, describing components with high communication capabilities, where η is the depth of a recursive computation of a temporal H-index variant. Qin et al. [41] explore the concept of stable communities in temporal networks, introducing (μ, τ, ϵ)-stable cores. In this context, a node is considered a part of a (μ, τ, ϵ)-stable core if it maintains no fewer than μ neighbors, each exhibiting a similarity of at least ϵ, across at least τ snapshots within the temporal network. Yu et al. [53] discuss *historical k-cores* which are the k-cores in the aggregated graphs with respect to time intervals, i.e., given a time interval I the historical k-cores wrt. I are the k-cores of the aggregated graph spanned by I. Similarly, Yang et al. [52] extend the work of [53] and introduce *time-range k-cores queries* by allowing the resulting k-cores to be induced by any subinterval $I' \subseteq I$ of the time interval I. The works of [52, 53] focus on efficiently answering queries for the standard k-core definition in given time intervals, and the authors introduce indexing techniques to answer such queries efficiently. Zhong et al. [54] propose a framework to unify such k-core queries for temporal networks. In a different direction, the authors of [31] introduce the concept of identifying *core-invariant nodes* in temporal networks. A core invariant node keeps a core number above a given threshold within a certain time interval. Li et al. [24] define the (θ, τ)-persistent k-core, which requires an intricate *persistency* function to be maximized, leading to NP-hard optimization problem of finding the largest (θ, τ)-persistent k-core. Finally, a closely related is the concept of the truss-decomposition. A k-truss is the maximal edge-induced subgraph in which each edge is part of at least $(k-2)$ triangles [8]. The temporal k-truss [26] is an extension of [11] using the k-truss concept instead of k-cores, where a (k, I)-span truss is a truss that exists in each time step of the interval I. However, similar to the span-core, there is a quadratic number of

Table 2: Commonly used notations

Symbol	Definition
$\mathcal{G} = (V, \mathcal{E})$	Temporal graph $\mathcal{G}$ with nodes V and temporal edges $\mathcal{E}$
$e = (\{u, v\}, t)$	Temporal $\{u, v\}$-edge at time $t \in \mathbb{N}$
$m = \lvert\mathcal{E}\rvert$, $n = \lvert V\rvert$	Numbers of temporal edges and nodes
δ_m	Maximum degree in $\mathcal{G}$
$\mathcal{T}$	Time interval spanned by graph
$T(\mathcal{G})$	Set of timestamps in $\mathcal{G}$, i.e., $\{t \mid (u, v, t) \in \mathcal{E}\}$
$\varphi : \mathcal{E} \times 2^{\mathcal{E}} \to \mathbb{R}$	Temporal edge weight function
C_r^{φ}	Maximum edge-induced subgraph with $\varphi(e, \mathcal{E}(C_r^{\varphi})) \geq r$
$\Delta \in \mathbb{N}$	Temporal distance
$\Delta_m \in \mathbb{N}$	Max. temporal distance of two edges at a node
$d_\Delta(e, \mathcal{E}')$	Δ-degree of edge $e \in \mathcal{E}'$
$s_\Delta(e, \mathcal{E}')$	Δ-support of edge $e \in \mathcal{E}'$
$C_k^{d_\Delta}$	(k, Δ)-core
$C_k^{s_\Delta}$	(k, Δ)-truss
$c_\Delta(e)$	Core number of $e \in \mathcal{E}$
$\tau_\Delta(e)$	Truss number of $e \in \mathcal{E}$
ξ	Maximum of Δ-incident edges at any edge $e \in \mathcal{E}$

time intervals for which a temporal span-truss can exist leading to a asymptotic running time complexity in $O(|\mathcal{T}|^2 \cdot |E|^{1.5})$ rendering the approach infeasible for many real-world temporal networks.

2 Preliminaries

Table 2 in shows an overview of our notation. A static *(multi-)graph* $G = (V, E)$ consists of a finite set of nodes V and a finite (multi-)set $E \subseteq \{\{u, v\} \subseteq V \mid u \neq v\}$ of undirected edges. We say that an edge $e = \{u, v\}$ is *incident* to u and v. The degree $\delta(v)$ of a node $v \in V$ is the number of edges incident to v. Given a graph G and $k \in \mathbb{N}$, a subgraph H is a *k-core* of G if *(i)* each vertex $u \in V(H)$ has degree at least k in H and *(ii)* H is maximal.

An *temporal network* (or temporal graph) $\mathcal{G} = (V, \mathcal{E})$ consists of a finite set of nodes V and a finite set $\mathcal{E}$ of undirected *temporal edges* $e = (\{u, v\}, t)$ with u and v in V, $u \neq v$, and *timestamp* $t \in \mathbb{N}$. The timestamp specifies when the edge exists in the graph. We define $n = |V|$, $m = |\mathcal{E}|$, and $T(\mathcal{G})=\{t \mid (\{u, v\}, t) \in \mathcal{E}\}$. We use $\mathcal{E}(\mathcal{G})$ to denote the temporal edges of $\mathcal{G}$. We assume that $n \leq 2m$, which holds unless isolated vertices exist. We denote with δ_m the maximum degree in $\mathcal{G}$. For a subset $\mathcal{E}' \subseteq \mathcal{E}$, we define the *edge-induced subgraph* $\mathcal{G}' = (V', \mathcal{E}')$ with $V' = \{u, v \mid (\{u, v\}, t) \in \mathcal{E}'\}$; we may also write $\mathcal{G}' \subseteq \mathcal{G}$. Finally, all our new definitions and algorithms can be easily adapted to respect a restrictive time interval $I = [\alpha, \beta]$ with $\alpha, \beta \in \mathbb{N}$ such that only temporal edges $e = (\{u, v\}, t) \in \mathcal{E}$ with $t \in I$ are considered. We do not make the restrictive time interval explicit for ease of readability.

3 An Edge-based Decomposition Framework

In this section, we introduce our new decomposition framework. The motivation origins from the following observations: Given a temporal network $\mathcal{G} = (V, \mathcal{E})$, each temporal edge $e = (\{u, v\}, t_e) \in \mathcal{E}$ has its timestamp t_e determining the time of the existence of the edge. Hence, the temporal edges $\mathcal{E}$ fully define the temporal dimension and properties of the network, while the nodes V can be considered static objects. Hence, by defining a temporal edge-based decomposition framework, we can naturally account for the temporal dimension. To this end, we first define the temporal edge weight function $\varphi : \mathcal{E} \times 2^{\mathcal{E}} \to \mathbb{R}$ such that $\varphi(e, \mathcal{E}')$ assigns a weight to edge $e \in \mathcal{E}$ with respect to $\mathcal{E}' \subseteq \mathcal{E}$. Based on φ, we decompose the temporal network.

Definition 1. *Given $r \in \mathbb{R}$ and $\varphi : \mathcal{E} \times 2^{\mathcal{E}} \to \mathbb{R}$, the (r, φ)-component of a temporal graph $\mathcal{G} = (V, \mathcal{E})$ is the inclusion-maximal edge-induced subgraph $C_r^{\varphi} \subseteq \mathcal{G}$ such that for each temporal edge $e \in \mathcal{E}(C_r^{\varphi})$ it holds $\varphi(e, \mathcal{E}(C_r^{\varphi})) \geq r$.*

The value $c_\varphi(e)$ of $e \in \mathcal{E}$ is the maximum $r \in \mathbb{R}$ such that e is in a C_r^{φ} component of $\mathcal{G}$ but not in a $C_{r'}^{\varphi}$ component with $r' > r$. We call the function φ *monotone* if for $\mathcal{E}_1 \subseteq \mathcal{E}_2 \subseteq \mathcal{E}$ and for $e \in \mathcal{E}$, $\varphi(e, \mathcal{E}_1) \leq \varphi(e, \mathcal{E}_2)$. For a monotone function φ, assigning $c_\varphi(e)$ to the edges $e \in \mathcal{E}$ induces a hierarchical decomposition of $\mathcal{E}$. Moreover, monotonicity allows us to apply a simple edge peeling strategy to compute the decomposition as shown in Algorithm 1.

Algorithm 1: Decomposition Framework

Input: Temporal graph $\mathcal{G} = (V, \mathcal{E})$ and φ
Output: $c_\varphi(e)$ for all $e \in \mathcal{E}$
Initialize $c[e] = \varphi(e, \mathcal{E})$ for all $e \in \mathcal{E}$ and $\mathcal{E}' = \mathcal{E}$
while $\mathcal{E}' \neq \emptyset$ **do**
  Choose $e \in \mathcal{E}'$ with $c[e]$ minimal
  $\mathcal{E}' \leftarrow \mathcal{E}' \setminus \{e\}$
  for *all affected* $f \in \mathcal{E}'$ *with* $c[f] > c[e]$ **do**
    Update $c[f] \leftarrow \max(c[e], \varphi(f, \mathcal{E}'))$
return $c[e]$ for all $e \in \mathcal{E}$

Theorem 1. *Given a temporal graph $\mathcal{G} = (V, \mathcal{E})$ and a monotone function φ, Algorithm 1 correctly computes $c_\varphi(e)$ of all edges $e \in \mathcal{E}$.*

In the following, we define instances of the function φ to develop our new core and truss decompositions as new essential primitives for temporal network analysis. We also provide efficient corresponding variants of Algorithm 1. It is important to note that our Definition 1 is a general framework as any temporal and non-temporal edge property can be used to potentially define additional variants of decompositions, allowing for the exploration of more advanced methods, such as distance-based [6] or motif-based decompositions [46] in future work.

3.1 Temporal (k, Δ)-Cores

Next, we establish our specific temporal edge weight function to identify cohesive temporal subgraphs. To this end, consider a single temporal edge representing an interaction between two vertices. A necessary condition for a temporal edge to be part of an inner core is that it occurs in the context of many other spatially and temporally local interactions. To capture this context, we define the degree of a temporal edge as follows.

Definition 2. *Let $\mathcal{G} = (V, \mathcal{E})$ be a temporal graph, $\Delta \in \mathbb{N}$, we define the Δ-degree $d_\Delta : \mathcal{E} \times 2^{\mathcal{E}} \to \mathbb{N}$ as*

$$d_\Delta(e, \mathcal{E}') = \min(\left|\left\{(\{u, w\}, t') \in \mathcal{E}' \mid |t - t'| \leq \Delta\right\}\right|, \left|\left\{(\{v, w\}, t') \in \mathcal{E}' \mid |t - t'| \leq \Delta\right\}\right|)$$

for $e = (\{u, v\}, t)$. We denote two edges $e = (\{u, v\}, t)$ and $f = (\{u, w\}, t')$ with $|t - t'| \leq \Delta$ as Δ-incident each other.

Note that each edge counts in its own Δ-degree as it is considered incident to itself.

Lemma 1. *The Δ-degree d_Δ is monotone.*

The Δ-degree d_Δ as the temporal edge weight function φ together with Definition 1 leads our new (k,Δ)-*core decomposition.* And with Lemma 1 and Theorem 1, Algorithm 1 results in the *core numbers* c_Δ for all edges and the temporal cores $C_k^{d_\Delta} \subseteq \mathcal{G}$. We will discuss a more efficient implementation in Section 3.1.1. In the following, we call $C_k^{d_\Delta}$ a (k,Δ)-core and the edge-induced subgraph containing only edges with core number exactly k the (k,Δ)-shell of $\mathcal{G}$.

The (k,Δ)-core decomposition is a generalization of the classical static k-core decomposition for multigraphs. To see this, we first define the *edge k-core* for static multigraphs as follows. Given $k \in \mathbb{N}$, the edge k-core of a multigraph G is the inclusion-maximal edge-induced subgraph G_k of G such that each endpoint u, v of each edge $\{u,v\}$ in G_k has at least $k+1$ neighbors. Hence, for $k > 1$, a subgraph H is a k-core if and only if H is an edge $(k-1)$-core.

Theorem 2. *Let $\mathcal{G} = (V,\mathcal{E})$ be a temporal network and $D(u) = \max\{|t_1 - t_2| \mid (\{u,v\},t_1),(\{u,w\},t_2) \in \mathcal{E}\}$ be the maximum time difference over edges incident to the vertex $u \in V$. Moreover, let $\Delta_m = \max_{u \in V} D(u)$. For all $e \in \mathcal{E}$, the core number $c_{\Delta_m}(e)$ equals the static edge core number $c_s(e)$.*

Hence, our new (k,Δ)-core generalizes the conventional static edge k-core (and hence the node-based static k-core) if we choose $\Delta \geq \Delta_m$. However, note that by using smaller values of $\Delta < \Delta_m$, increasingly higher time resolutions can be considered, which is impossible for the static (and other temporal) core decompositions.

Moreover, the (k,Δ)-core allows us to identify the hierarchical organization of temporal edges within the network due to the following containment property.

Theorem 3. *Let $\mathcal{G} = (V,\mathcal{E})$ be a temporal network, $k, k', \Delta, \Delta' \in \mathbb{N}$ with $k \leq k'$, and $\Delta \geq \Delta'$. Furthermore, let $C_k^{d_\Delta}$ and $C_{k'}^{d_{\Delta'}}$ be a (k,Δ)- and a (k',Δ')-core, respectively. Then $C_k^{d_\Delta} \subseteq C_{k'}^{d_{\Delta'}}$.*

We give an example of (k,Δ)-cores in Section 4.

3.1.1 Efficient Computation. Based on the containment property (Theorem 3), our (k,Δ)-core decomposition can be efficiently computed for a fixed value of Δ by adapting the greedy *peeling* algorithm introduced by Batagelj and Zaversnik [3]. Algorithm 2 shows our edge-peeling algorithm for computing the (k,Δ)-cores. The algorithm removes a temporal edge with the lowest Δ-degree in each iteration. To this end it uses three arrays $a_u[e], a_v[e]$, and $d[e]$ to store for each temporal edge $e = (\{u,v\},t) \in \mathcal{E}$ the current numbers of Δ-incident edges at the endpoints u and v and their minimum, respectively, i.e., after initialization (line 1-6), $d[e]$ equals the minimum number of Δ-incident edges at the endpoints of edge e. Let $\{e_1,\ldots,e_m\}$ be the sequence in which the edges are processed by the for loop in line 8. In the i-th round, Algorithm 2 processes e_i with $d[e_i]$, i.e., the edge with the lowest Δ-degree currently remaining in the graph. Now, because for e_i the value of $d[e_i]$ is the lowest, each edge e remaining in $\mathcal{G}$, has at least $d[e] \geq d[e_i]$ Δ-incident edges at both endpoints. Therefore, e_i is part of a maximal edge-induced subgraph in which each edge has at both endpoints at least $d[e_i]$ Δ-incident edges, i.e., $c_\Delta(e_i) = d[e_i]$. The loop in line 8 processes each temporal edge e_i once in order of minimal Δ-degree, and $d[e_i]$ will not be changed after e_i is processed due to line 10. After the loop in line 8 ends, the algorithm returns d, i.e., the core numbers for all $e \in \mathcal{E}$. Now let ξ be the maximum of Δ-incident edges at any edge $e \in \mathcal{E}$. In each iteration, we may have to update the value $d[f]$ for each of the at most ξ Δ-incident edges of e. Determining these edges is possible in $O(\log \delta_m)$ by storing the edges at each vertex in chronologically ordered incidence lists. Finally, updating the bin position of f takes only constant time.

Algorithm 2: (k,Δ)-core decomposition

Input: Temporal graph $\mathcal{G} = (V,\mathcal{E})$ and $\Delta \in \mathbb{N}$
Output: Core number $c_\Delta(e)$ for all $e \in \mathcal{E}$
1 Initialize $a_u[e] = 0$ and $a_v[e] = 0$ for all $e = (\{u,v\},t) \in \mathcal{E}$
2 Initialize $d[e] = 0$ for all $e \in \mathcal{E}$
3 **for** $e = (\{u,v\},t) \in \mathcal{E}$ **do**
4     $a_u[e] \leftarrow |\{(\{u,w\},t') \in \mathcal{E} \mid |t-t'| \leq \Delta\}|$
5     $a_v[e] \leftarrow |\{(\{v,w\},t') \in \mathcal{E} \mid |t-t'| \leq \Delta\}|$
6     $d[e] = \min(a_u[e], a_v[e])$
7 Bin sort edges $\mathcal{E}$ in increasing order of $d[e]$
8 **for** $e = (\{u,v\},t) \in \mathcal{E}$ *in sorted order* **do**
9   **for** $f = (\{x,w\},t') \in \mathcal{E}$ *with* $x \in \{u,v\}$ **do**
10   **if** $|t-t'| \leq \Delta$ **and** $d[f] > d[e]$ **then**
11             $a_x[f] \leftarrow a_x[f] - 1$
12             $d[f] \leftarrow \min(a_x[f], a_w[f])$
13   Update the bin position of f
14   remove e from $\mathcal{G}$
15 **return** d

Theorem 4. *Given a temporal graph $\mathcal{G} = (V,\mathcal{E})$ and $\Delta \in \mathbb{N}$, Algorithm 2 computes the (k,Δ)-core numbers of all $e \in \mathcal{E}$ correctly in $O(m \cdot \max(\log \delta_m, \xi))$ time and $O(m)$ space.*

3.2 Temporal (k,Δ)-Trusses

The k-truss in a static network is the maximal subgraph where each edge is part of at least $k-2$ triangles, aiming to enhance cohesiveness compared to the k-core by requiring stronger local connectivity. We now introduce our temporal truss variant by first defining an edge weighting function counting the number of temporally local triangles in which a temporal edge participates.

Definition 3. *Let $\mathcal{G} = (V,\mathcal{E})$ be a temporal graph, $\Delta \in \mathbb{N}$, we define the Δ-support $s_\Delta : \mathcal{E} \times 2^{\mathcal{E}} \to \mathbb{N}$ as*

$$\begin{aligned} s_\Delta(e,\mathcal{E}') = |\{\{e_i,e_j\} \mid\ & e_i = (\{u,w\},t_1), e_j = (\{v,w\},t_2) \in \mathcal{E}' \\ & \text{with } u \neq v \neq w, |t-t_1| \leq \Delta, |t-t_2| \leq \Delta \\ & \text{and } |t_1 - t_2| \leq \Delta\}|. \end{aligned}$$

Lemma 2. *The Δ-support s_Δ is monotone.*

By using the Δ-support as the temporal edge weight function φ in our decomposition framework (Definition 1), we obtain our new (k,Δ)-*truss decomposition.* Following Lemma 2 and Theorem 1, Algorithm 1 with function s_Δ results in the *truss numbers* τ_Δ for all edges and the *temporal trusses* $C_k^{s_\Delta} \subseteq \mathcal{G}$. In the following, we call $C_k^{s_\Delta}$ a (k,Δ)-truss. The (k,Δ)-truss decomposition is a generalization of the conventional static k-truss decomposition (with the truss numbers shifted by two) in multigraphs by setting $\Delta \geq \Delta_m$ as defined in Theorem 2. Furthermore, Theorem 3 holds analogously for (k,Δ)-trusses (replace d_Δ with s_Δ) allowing for a hierarchical decomposition over k as well as Δ. The (k,Δ)-truss decomposition can be efficiently computed (we provide an algorithm in Appendix B).

Theorem 5. *Given $\mathcal{G} = (V,\mathcal{E})$, $\Delta \in \mathbb{N}$, and $s_\Delta^{\max} = \max_{e\in\mathcal{E}} s_\Delta(e,\mathcal{E})$, we can compute the (k,Δ)-truss numbers of all $e \in \mathcal{E}$ correctly in $O(m \cdot \max(\log \delta_m, \xi^2))$ time and $O(\max(m, s_\Delta^{\max}))$ space.*

3.3 Identifying Connected Components

After computing the (k,Δ)-cores or (k,Δ)-trusses of a temporal graph $\mathcal{G}$, we consider the (k,Δ)-core or (k,Δ)-truss $C_k^\star$, where $\star \in \{d_\Delta, s_\Delta\}$, as the subgraph of $\mathcal{G}$ induced by the edges e with a core number $c_\Delta(e) \geq k$ or a truss number $\tau_\Delta(e) \geq k$, respectively. A natural question that arises is if $C_k^\star$ is *connected*. However, connectivity in temporal graphs is less clearly defined than in conventional static graphs [5, 7, 9, 33]. Commonly, temporal connectivity is based on temporal reachability, which itself is defined using *temporal walks* [5, 33]. A temporal walk is a sequence of connected temporal edges with (strictly) increasing timestamps to capture the possible *flow of information*. The requirement of increasing timestamps leads to inherent non-symmetric temporal walks, reflecting the fact that information cannot flow backward in time. This non-symmetry (together with non-transitivity) often makes determining temporal connected components a hard problem. For this reason, in the context of temporal motif mining, Kovanen et al. [23] loosened the restriction of increasing timestamps and defined Δ-*walks* in terms of *temporal locality* to ensure symmetry and transitivity.

Definition 4 (Kovanen et al. [23]). *A Δ-walk ω in a temporal graph $\mathcal{G} = (V,\mathcal{E})$ is a sequence of $\ell \in \mathbb{N}$ temporal edges $\omega = (e_1 = (\{v_1,v_2\},t_1), e_2 = (\{v_2,v_3\},t_2)\ldots, e_\ell = (\{v_\ell,v_{\ell+1}\},t_\ell))$ for which $|t_i - t_{i+1}| \leq \Delta$ for all $1 \leq i < \ell$, i.e., e_i and e_{i+1} for $1 \leq i < \ell$ are Δ-incident. Moreover, we say that an edge $e_j \in \mathcal{E}$ is Δ-reachable from edge $e_i \in \mathcal{E}$ if there exists a Δ-walk from e_i to e_j.*

For $\Delta \geq \Delta_m$ as defined in Theorem 2, Δ-reachability equals conventional reachability. For smaller Δ values, Δ-reachability captures temporal locality motivated by the fact that in many real-world scenarios, the significance of past or future events, e.g., human interactions or information dissemination, tends to diminish with time passing by. For example, in a social network, the impact of interactions from several years ago is usually less influential on an individual's current preferences and decisions compared to recent interactions. Similarly, events in the far future are usually less relevant to the current situation than events that will happen soon.

It is easy to see that the Δ-reachability leads to a decomposition of the temporal network that is an equivalence relation, i.e., satisfying reflexivity, symmetry, and transitivity. We define the Δ-connected components of a temporal network $\mathcal{G} = (V,\mathcal{E})$ (or a subgraph like a (k,Δ)-core or (k,Δ)-truss) as the maximal subset $\mathcal{E}' \subseteq \mathcal{E}$ of temporal edges such that all edges in $\mathcal{E}'$ are pairwise Δ-reachable.

Using Δ-connectedness, we can further decompose (k,Δ)-cores or (k,Δ)-trusses into Δ-connected (k,Δ)-cores or (k,Δ)-trusses.

Theorem 6. *Let $C_k^\star$ with $\star \in \{d_\Delta, s_\Delta\}$ be a (k,Δ)-core or (k,Δ)-truss. The Δ-connected components of $C_k^\star$ are non-inclusion-maximal (k,Δ)-cores or (k,Δ)-trusses, respectively.*

Figure 2e shows the two 2-components of the $(2,2)$-core shown in Figure 2b. The Δ-connected components can be computed using a recursive algorithm in linear time [23]. In Appendix C, we propose a new and simple linear-time algorithm based on transforming the temporal graph into a static representation.

4 Comparison of Decompositions

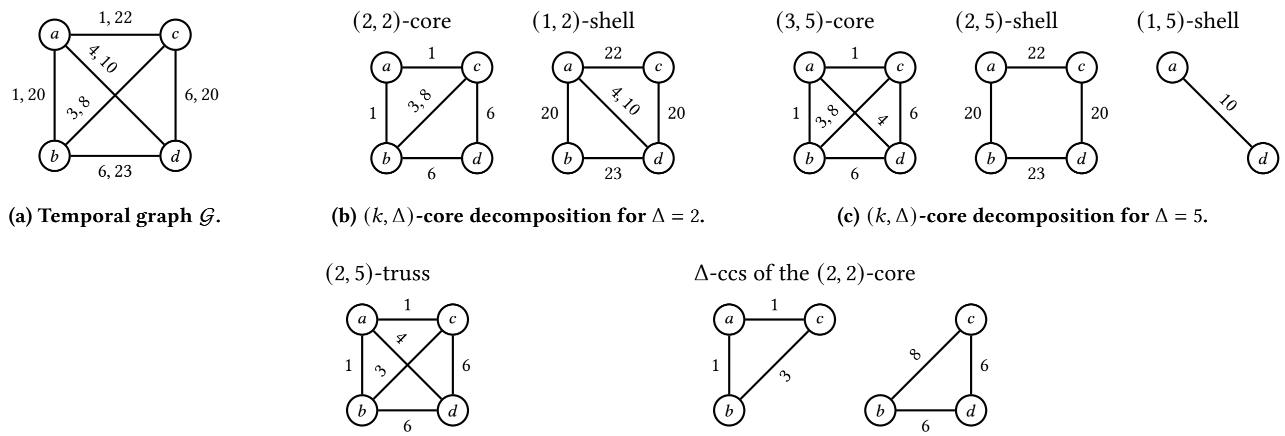
We use a toy communication network to compare our new (k,Δ)-decompositions to the static variants and state-of-the-art temporal core and truss decompositions. Figure 2a shows the temporal graph in which nodes communicate at different times, shown at the edges. Each timestamp corresponds to a temporal edge.

Figure 2b shows the $(2,2)$-core and the $(1,2)$-shell. Similarly, Figure 2c shows the $(3,5)$-core and the $(2,5)$- and $(1,5)$-shells. Each temporal edge in the $(3,5)$-core has a 5-degree of at least three. For example, both endpoints of the temporal edge $(\{a,b\},1)$, are incident to at least three edges $(\{u,v\},t)$ such that $|t-1| \leq 5$. In the case of a, these edges are $(\{a,b\},1)$, $(\{a,c\},1)$, and $(\{a,d\},4)$. And for endpoint b, the Δ-incident edges are $(\{a,b\},1)$, $(\{b,c\},3)$, and $(\{b,d\},6)$. Similarly, in Figure 2d, the $(2,5)$-truss of $\mathcal{G}$ is shown in which each edge has a 5-support of at least two. The $(3,5)$-core reflects pair-wise communications of all four nodes occurring in the interval $[1,8]$, whereas the $(2,5)$-shell shows (non-pairwise) communications occurring in the later interval $[20,23]$. Note that the $(2,5)$-*core* consist of the $(3,5)$-core together with the $(2,5)$-shell. Figure 2e shows the Δ-connected components of the $(2,2)$-core, highlighting the communications between the nodes $\{a,b,c\}$ and $\{b,c,d\}$ in the two distinct intervals $[1,3]$ and $[6,8]$. It is essential to mention that for our (k,Δ)-core decomposition, one of the key differences to traditional static and temporal core decompositions is that our approach decomposes the network on an edge basis instead of node-wise, often leading to a more fine-grained decomposition. We now compare our approach to other decompositions.

Static cores and trusses: A natural question is if the temporal information in the temporal network $\mathcal{G}$ is necessary or if a purely static network decomposition suffices. Using the conventional static k-core decomposition computed on the underlying aggregated graph (or underlying multigraph) results in each node having a core number of three (or six, resp.), showing that the static approach ignores the network's temporal dimension and does not suffice. Similar arguments hold for the static truss decomposition; for example, the underlying aggregated graph is a static 4-truss.

Interval queries: The approaches that query a static k-core during an interval [52, 53] can, e.g., identify the $(3,5)$-core shown in Figure 2c by carefully choosing the interval $[1,8]$. However, it is not possible to obtain the, e.g., $(2,2)$-core using the interval $[1,8]$ as this would lead to the inclusion of the temporal edge $(\{a,d\},4)$ as temporal (non-)locality is not considered in the subgraph.

$(\mathbf{k},\mathbf{h})$**-cores:** For the (k,h)-core decomposition [51], each vertex in a (k,h)-core has at least k neighbors, and there are at least h temporal edges to each neighbor. Note that the actual timestamps or their relative distances to each other are not considered. Hence the temporal network $\mathcal{G}$ itself is a (k,h)-core for $k=3$ and $h=2$.



(a) Temporal graph $\mathcal{G}$. **(b) (k,Δ)-core decomposition for $\Delta = 2$.** **(c) (k,Δ)-core decomposition for $\Delta = 5$.**

(d) (k,Δ)-truss decomposition for $\Delta = 5$. **(e) 2-connected components of the $(2,2)$-core.**

Figure 2: Examples of the (k,Δ)-core and (k,Δ)-truss decompositions and Δ-connected components.

Span cores and trusses: For the spanning decompositions (i.e., span-core [11], span-truss [26], (L,K)-lasting core [19]) the cores (trusses, resp.) need to exist for some interval I requiring all edges of the core to exist at each timestamp of I. This requirement is, in many cases, too restrictive, as we see in the example graph (Figure 2a), which does not contain any non-trivial spanning core (truss) even for the minimum interval length of $|I| = 1$.

In conclusion, using the simple temporal network $\mathcal{G}$ shown in Figure 2a, we can already verify that the discussed baselines lead to significantly different or trivial decompositions. Similar arguments or examples are also possible for other approaches. As most baselines are node-based, they are usually unable to achieve the fine-grained decomposition on an edge basis as our approach does. Finally, in Section 5.4, we discuss an empirical use-case of analyzing malicious retweets in a subnetwork of the Twitter graph, and we show that our (k,Δ)-decompositions lead to insights that cannot be obtained using the state-of-the-art baselines.

5 Experiments

We compare our new decompositions with state-of-the-art baselines and discuss an use-case using the (k,Δ)-core and (k,Δ)-truss decompositions for analyzing malicious tweets.

5.1 Experimental Setup

All experiments run on a computer cluster. Each experiment had an exclusive node with an Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz and 96 GB of RAM. We used a time limit of 12 hours. We implemented our algorithms in C++ using GNU CC Compiler 11.4.0 with the flag `-O3`[1]. We denote our implementation of Algorithm 2 as (k,Δ)-`Core` and our implementation of our (k,Δ)-truss algorithm (see Appendix B) as (k,Δ)-`Truss`.

5.1.1 Baselines. We use the following state-of-the-art core decomposition baselines:

Table 3: Statistics of the data sets.

	Data Set	$\lvert V(\mathcal{G})\rvert$	$\lvert E(\mathcal{G})\rvert$	Span	Domain	Ref.
Small	*FacebookMsg*	1.9K	59.8K	194 days	social network	[39]
	Enron	86.8K	1.1M	4 years	email network	[20]
	AskUbuntu	134.0K	257.3K	7 years	question answering	[40]
	Twitter	346.1K	2.1M	176 days	retweets	[48]
Large	*Wikipedia*	1.9M	40.0M	6 years	co-editing	[40]
	StackOverflow	2.6M	48.0M	8 years	question answering	[30]
	Reddit	3.0M	84.3M	9 years	social network	[14]
	Bitcoin	48.1M	111.0M	7 years	financial	[21]

- `Stat-`k`-C` is the static k-core decomposition algorithm [3] in which we ignore all time stamps and $\mathcal{G}$ is interpreted as undirected multilayer graph.
- (k,h)`-C` is the (k,h)-core decomposition [51]. We use the implementation provided by [37] and $h \in \{2,4,8\}$.
- `PC-`η is the (η,k)-pseudocore decomposition [36]. We use the implementation provided by the authors and $\eta \in \{8,16,32\}$.
- `SpanC` and `SpanT` are the maximal span-core and span-truss decompositions [11, 26]. We use the implementations provided by the authors.
- (L,K)`-C` is the algorithm for the lasting k-core with $k \geq K$ [19]. This algorithm only returns the k-cores with $k \geq K$ and a maximal number of nodes lasting for exactly a time duration of length L. We set $L = 1$ minute and $k = 2$. We use the implementation provided by [37].
- (l,δ)`-C` is the algorithm for the maximal dense core [42]. We use the implementation provided by the authors and the proposed default values for the parameters of $l = 3$ and $\delta = 3$.

5.1.2 Data sets. We use eight real-world network data sets of different sizes and from various domains. Table 3 gives an overview. Further details are provided in Appendix D.

5.2 Choosing the Parameter Δ

We choose the values of Δ based on the node-based inter-event times (IETs). The IETs are defined as the set $\mathcal{I} = \{t_2 - t_1 \mid e_1 =$

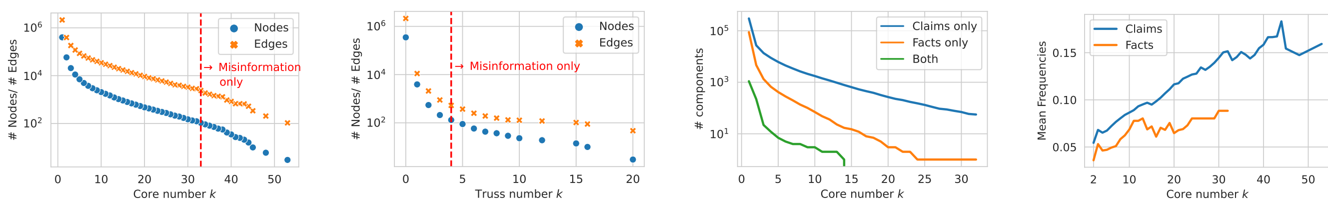
[1]The source code is available at https://gitlab.com/tgpublic/tgkd.

Table 4: Statistics for the chosen values of Δ.

Data Set	Duration $\Delta_{10\%}$	Duration $\Delta_{25\%}$	Duration $\Delta_{50\%}$	Duration $\Delta_{75\%}$	Avg. Δ-degree $\Delta_{10\%}$	Avg. Δ-degree $\Delta_{25\%}$	Avg. Δ-degree $\Delta_{50\%}$	Avg. Δ-degree $\Delta_{75\%}$	Max. Δ-degree $\Delta_{10\%}$	Max. Δ-degree $\Delta_{25\%}$	Max. Δ-degree $\Delta_{50\%}$	Max. Δ-degree $\Delta_{75\%}$	Avg. Δ-support $\Delta_{10\%}$	Avg. Δ-support $\Delta_{25\%}$	Avg. Δ-support $\Delta_{50\%}$	Avg. Δ-support $\Delta_{75\%}$	Max. Δ-support $\Delta_{10\%}$	Max. Δ-support $\Delta_{25\%}$	Max. Δ-support $\Delta_{50\%}$	Max. Δ-support $\Delta_{75\%}$
FacebookMsg	34.0s	1.6m	9.6m	3.4h	1.12	1.41	3.13	10.46	28	28	30	141	0.0003	0.004	0.12	2.11	2	12	137	1285
Enron	9.0m	1.0h	10.0h	2.8d	1.20	1.73	4.16	12.42	229	554	727	1255	0.07	0.35	2.81	22.96	452	1237	1962	13079
AskUbuntu	12.4m	1.5h	1.1d	16.9d	1.02	1.13	1.54	3.52	13	27	31	272	0.0002	0.003	0.02	0.26	1	7	14	145
Twitter	5.0s	21.0s	6.4m	18.7h	1.03	1.17	1.79	5.54	7	22	218	14459	0.0000	0.0001	0.01	2.83	1	15	531	16017
Wikipedia	1.0d	2.0d	7.0d	31.0d	5.23	6.37	9.82	17.78	1043	1638	4472	16881	0.19	0.23	0.35	0.75	731	731	804	1530
StackOverflow	1.8m	5.9m	39.4m	18.2h	1.07	1.33	2.41	5.35	13	21	37	183	0.0049	0.04	0.25	0.60	9	30	203	600
Reddit	1.1m	4.4m	32.0m	10.2h	1.03	1.23	2.26	6.22	53	63	240	3146	0.0004	0.0066	0.10	0.78	10	86	1260	9564
Bitcoin	8.3m	21.9m	1.8h	19.8h	1.79	2.32	5.16	27.01	611	1572	3917	17289	0.12	0.25	1.59	68.87	213	711	5259	111904

Table 5: The running times in seconds (OOT–out of time (12h), OOM–out of memory (96GB)).

	Baselines											(k,Δ)-Core				(k,Δ)-Truss			
Data Set	Stat-k-C	$(k,2)$-C	$(k,4)$-C	$(k,8)$-C	PC-8	PC-16	PC-32	SpanC	SpanT	(L,K)-C	(l,δ)-C	$\Delta_{10\%}$	$\Delta_{25\%}$	$\Delta_{50\%}$	$\Delta_{75\%}$	$\Delta_{10\%}$	$\Delta_{25\%}$	$\Delta_{50\%}$	$\Delta_{75\%}$
FacebookMsg	0.02	0.01	0.01	0.01	0.28	0.59	1.23	OOM	81.28	0.22	OOM	0.04	0.05	0.05	0.07	0.04	0.04	0.05	0.17
Enron	0.78	0.32	0.27	0.25	37.21	79.64	166.57	OOM	39398.65	6.93	OOM	1.76	2.01	2.71	4.88	2.16	2.70	5.90	29.06
AskUbuntu	0.25	0.10	0.09	0.09	0.90	1.78	3.53	OOM	OOT	1.43	OOM	0.21	0.23	0.27	0.40	0.21	0.22	0.25	0.60
Twitter	2.23	1.19	1.06	1.05	684.04	3521.95	7931.67	OOM	42039.00	12.45	OOM	2.85	3.02	4.10	99.74	2.83	3.06	5.27	310.26
Wikipedia	98.10	50.00	45.18	44.64	1758.70	3807.09	8367.10	OOM	OOT	1605.45	OOM	222.14	280.05	466.25	899.00	306.85	429.42	914.14	3192.34
StackOverflow	122.23	73.10	55.18	52.23	2146.92	4518.03	9694.15	OOM	OOT	263.71	OOM	83.29	92.65	111.19	152.80	83.86	90.31	103.18	161.23
Reddit	255.62	145.96	126.69	121.05	1696.92	3593.85	OOT	OOM	OOT	498.25	OOM	159.74	178.69	226.34	515.22	158.80	167.79	209.99	769.91
Bitcoin	291.67	119.20	108.12	107.36	OOT	OOT	OOT	OOM	OOT	2410.94	OOM	222.35	271.93	550.86	3867.92	205.20	282.98	1342.53	OOT



(a) Number of users (nodes) and retweets (edges) in the $(k,\Delta_{50\%})$-cores.

(b) Number of users (nodes) and retweets (edges) in the $(k,\Delta_{50\%})$-trusses.

(c) Numbers of Δ-connected components of the $(k,\Delta_{50\%})$-cores with only claims, facts, or both.

(d) Mean frequencies (s^{-1}) of retweets in $\Delta_{50\%}$-connected components.

Figure 3: Statistics of the $(k,\Delta_{50\%})$-cores, $(k,\Delta_{50\%})$-trusses, and $\Delta_{50\%}$-connected components of the $(k,\Delta_{50\%})$-cores.

$(\{u,v\},t_1), e_2 = (\{u,w\},t_2) \in \mathcal{E}, t_1 \leq t_2$ and e_1, e_2 are consecutive$\}$ where two temporal edges are consecutive iff. there is no other edge $e' = (\{u,x\},t')$ with $t_1 < t' < t_2$. Let $i_1 < i_2 \in \mathcal{I}$ such that there exists no other $j \in \mathcal{I}$ with $i_1 < j < i_2$. Then, for $\Delta_1, \Delta_2 \in \mathbb{N}$ with $i_1 < \Delta_1 \leq \Delta_2 < i_2$ the (k,Δ_1)-core and (k,Δ_2)-core (or (k,Δ_1)-truss and (k,Δ_2)-truss, resp.) are isomorphic for all k because the Δ-degrees (or Δ-supports) do not change going from Δ_1 to Δ_2. Based on this observation, we cover the range of the most relevant Δ values by choosing Δ to be the 10, 25, 50, and 75-percentiles of the IETs, denoted with $\Delta_{10\%}$, $\Delta_{25\%}$, $\Delta_{50\%}$, and $\Delta_{75\%}$, respectively. Table 4 shows the durations as well as the average and maximum Δ-degree and Δ-support, respectively. As expected, with increasing Δ, the maximum and average Δ-degrees and Δ-supports increase.

5.3 Efficiency

Table 5 shows the running times in seconds. For low values of Δ, the running times of our algorithms are comparable, or even less, than the static-k-core. The running times for computing the (k,Δ)-core and (k,Δ)-truss decompositions increase with increasing Δ because the numbers of Δ-incident edges (triangles, resp.) increase (see Table 4) leading to higher numbers of updates of the Δ-degree, or Δ-support, in the peeling-step of the algorithms. This effect is particularly pronounced for the *Twitter* data set and the increase from $\Delta_{50\%}$ to $\Delta_{75\%}$ where the maximum Δ-degree and the maximum Δ-support increase in two orders of magnitude leading also to a significant increase in the running time. Similarly, for *Bitcoin*, the massive increase of the average and maximum Δ-support from $\Delta_{50\%}$ to $\Delta_{75\%}$ causes (k,Δ)-Truss to exceed the time limit.

For the (k,h)-core baseline ((k,h)-C), the running times decrease for increasing h because only temporal edges with at least h parallel temporal edges are considered in the computation. The (η,k)-pseudocore (PC-η) has in most cases (significant) higher running times compared to our algorithm and cannot finish the computations in the given time limit of twelve hours for the *Bitcoin* data set and *Reddit* for $\eta = 32$. The span-truss implementation (SpanT) is for all data sets significantly slower than (k,Δ)-Truss and can only finish the computation in the time limit for the three data sets with the shortest total span *FacebookMsg*, *Enron*, and *Twitter*. The reason is that the running time is in $O(|\mathcal{T}|^2 \cdot m^{1.5})$, i.e., has a quadratic term in the interval spanned by the network. The (L,K)-C algorithm performs very well in terms of running time. However, (L,K)-C *fails to identify any (non-trivial) cores (or trusses, resp.)* because it is too strict by requiring a core to exist in each time step of the interval, which can be impossible in temporal networks that are sparse in each time step. For example, the average number of edges in a time step of *FacebookMsg* is only slightly above one. Due to their high memory demands SpanC and (l, δ)-C fail the computations due to out-of-memory error for all data sets. The reason is that the implementations rely on representing the temporal networks with data structures that need $O(|V|^2 \cdot x)$ space, where x is the number of timestamps $x = |T(\mathcal{G})|$ in the case of SpanC or duration of the interval spanned by the network, i.e., $x = |\mathcal{T}|$, for (l, δ)-C. In contrast, computing the (k, Δ)-cores and (k, Δ)-trusses is memory-efficient—Table 6 shows the maximum memory usage of our algorithms for the large data sets. The memory usage of (k, Δ)-Core and (k, Δ)-Truss were equal for all data sets and in order of the numbers of temporal edges. The algorithms needed the most memory for the *Bitcoin* data set with 42.7 GB. For the small data sets the memory usage is far below one gigabyte, reaching a maximum for *Twitter* of 769 MB.

Table 6: Memory usage of (k, Δ)-Core and (k, Δ)-Truss.

Data Set	Max. memory usage in GB
Wikipedia	13.7
StackOverflow	16.7
Reddit	29.3
Bitcoin	42.7

5.4 Use Case: Analyzing Malicious Retweets

We use the (k, Δ)-core and (k, Δ)-truss decompositions to analyze malicious tweets in the X network (formerly Twitter). Recent works showed that spreaders of misinformation and fake news often can be found in the inner cores of the static k-core decomposition [48, 49]. Shao et al. [48] analyzed a subgraph of the Twitter graph (data set *Twitter* in Table 3) representing users and retweets in the critical period of six months before the 2016 US Presidential Elections. The network uses a fine-grained time scale of seconds, and all edges are either labeled as *fact-checking* or *misinformation* (or claims), where *misinformation* constitutes 81.9% of the edges. Shao et al. [48] showed that the users with a high (static) core value exhibit a strong tendency to disseminate misinformation and fake news. Here, we extend and improve the results of Shao et al. [48] by analyzing the network using our (k, Δ)-decompositions: We computed the $(k, \Delta_{50\%})$-core and -truss decompositions and for each core or truss the $\Delta_{50\%}$-connected components. Figure 3a shows for each $(k, \Delta_{50\%})$-core the numbers of nodes and edges; similarly, Figure 3b shows for each $(k, \Delta_{50\%})$-truss the numbers of nodes and edges. The dashed vertical lines mark the value of k from which on the $(k, \Delta_{50\%})$-cores and $(k, \Delta_{50\%})$-trusses contain only claims and misinformation but no fact-checking. We observe that edges in the inner cores only represent misinformation, which not only aligns with Shao et al. [48] observation that users in inner cores tend to spread misinformation but strengthens it with the observation that the communication in inner $(k, \Delta_{50\%})$-cores, for $k \geq 33$, only consists of misinformation and does not contain any fact-checking. We see this behavior even stronger for the (k, Δ)-trusses: For $k \geq 4$, the (k, Δ)-trusses only consist of misinformation retweets. Furthermore, the $\Delta_{50\%}$-connected components of the cores and trusses are highly edge-homophilic, i.e., for most cores and trusses all edges in a Δ-connected component are either misinformation or fact-checking. Figure 3c shows the numbers of the $\Delta_{50\%}$-connected components containing only claims, fact-checking, or both. The majority of $\Delta_{50\%}$-connected components only contain claims and significantly less components only contain fact-checking or both. Moreover, for $k \geq 14$, there are no $\Delta_{50\%}$-connected components containing both.

Figure 3a shows that the numbers of users decreases for increasing k. Furthermore, many of the Δ-connected components in the inner (k, Δ)-cores and -trusses, i.e., with large k, consist of only a few users. In these user groups that spread misinformation, highly frequent retweeting appears. Figure 3d shows the mean frequencies (in s^{-1}) of retweets in the Δ-connected components that either only contain fact-checking or misinformation spreading. For the latter, the mean frequencies are significantly higher and show a stronger increase for increasing $k \geq 12$.

In comparison, using the baselines, we cannot to obtain these insights. The static k-cores in the graph are single connected components for $k \geq 3$ preventing insights into smaller components of the network. SpanC and (l, δ)-C run out of memory. For SpanT and (L, K)-C, the results are empty because of the high time-granularity of the network, resulting in too sparse time steps for these approaches. Using the (k, h)-C baseline, we can obtain inner cores that have a higher ratio of misinformation. But we cannot achieve a strict distinction of finding only misinformation-spreading cores, as with the (k, Δ)-core decomposition, because all (k, h)-cores contain misinformation and fact-checking for $h \in \{2, 4, 8\}$. In the case of PC-η baseline, the highest core values belong to nodes that are mainly involved in spreading misinformation. However, the subgraphs corresponding to the (η, k)-cores span the complete time interval of the network, so we cannot obtain any information about the temporal properties as we do with the new (k, Δ)-decompositions.

In conclusion, our analysis underscores the intrinsic value of (k, Δ)-decompositions, especially when combined with Δ-connected components, in identifying temporal structures and dynamic patterns within fine-grained temporal networks. Finally, see Appendix E for an additional comparison of the cohesiveness of the (k, Δ)-core and (k, Δ)-truss decompositions.

6 Conclusion and Future Work

We addressed the hierarchical decomposition of temporal networks by introducing a novel edge-based framework that incorporates the temporal dimension, leading to the development of the (k, Δ)-core and (k, Δ)-truss decompositions. Our highly efficient algorithms successfully handled large-scale, dynamic temporal networks where existing methods failed, and we demonstrated their effectiveness in a real-world use case of analyzing malicious Twitter content.

Currently, our algorithms require a predefined temporal distance Δ. Although it's straightforward to compute decompositions for all relevant Δ values, future work will focus on developing more efficient algorithms for simultaneous decomposition in both k and Δ. Additionally, we plan to explore size-restricted (k, Δ)-cores or trusses and consider advanced edge-weighting functions.

Acknowledgments: Giuseppe F. Italiano was partially supported by the Italian Ministry of University and Research under PRIN Project n. 2022TS4Y3N - EXPAND: scalable algorithms for EXPloratory Analyses of heterogeneous and dynamic Networked Data.

Ethical Considerations

Our algorithm is not expected to have any greater negative societal impact than comparable network decomposition algorithms, considering aspects of fairness, privacy, security, safety, or potential misuse by malicious actors.

References

[1] J Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. 2006. How the k-core decomposition helps in understanding the internet topology. In *ISMA Works. on Int. Topol.*, Vol. 1.

[2] Wen Bai, Yadi Chen, and Di Wu. 2020. Efficient temporal core maintenance of massive graphs. *Information Sciences* 513 (2020), 324–340.

[3] Vladimir Batagelj and Matjaz Zaversnik. 2003. An O(m) algorithm for cores decomposition of networks. *arXiv cs/0310049* (2003).

[4] Vladimir Batagelj and Matjaž Zaveršnik. 2011. Fast algorithms for determining (generalized) core groups in social networks. *Advances in Data Analysis and Classification* 5, 2 (2011), 129–145.

[5] Sandeep Bhadra and Afonso Ferreira. 2003. Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks. In *Ad-Hoc, Mobile, and Wireless Networks: ADHOC-NOW2003*. Springer, 259–270.

[6] Francesco Bonchi, Arijit Khan, and Lorenzo Severini. 2019. Distance-generalized core decomposition. In *proceedings of the 2019 international conference on management of data*. 1006–1023.

[7] Arnaud Casteigts, Timothée Corsini, and Writika Sarkar. 2022. Simple, strict, proper, happy: A study of reachability in temporal graphs. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*. Springer, 3–18.

[8] Jonathan Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis. *National security agency technical report* 16, 3.1 (2008), 1–29.

[9] Isnard Lopes Costa, Raul Lopes, Andrea Marino, and Ana Silva. 2023. On Computing Large Temporal (Unilateral) Connected Components. In *IWOCA*. Springer, 282–293.

[10] Jessica Enright and Rowland Raymond Kao. 2018. Epidemics on dynamic networks. *Epidemics* 24 (2018), 88–97.

[11] Edoardo Galimberti, Martino Ciaperoni, Alain Barrat, Francesco Bonchi, Ciro Cattuto, and Francesco Gullo. 2020. Span-core decomposition for temporal networks: Algorithms and applications. *TKDD* 15, 1 (2020), 1–44.

[12] Michel Gendreau, Gianpaolo Ghiani, and Emanuela Guerriero. 2015. Time-dependent routing problems: A review. *Computers & operations research* 64 (2015), 189–197.

[13] Aristides Gionis, Lutz Oettershagen, and Ilie Sarpe. 2024. Mining temporal networks. In *Companion Proceedings of the ACM on Web Conference 2024*. 1260–1263.

[14] Jack Hessel, Chenhao Tan, and Lillian Lee. 2016. Science, askscience, and badscience: On the coexistence of highly related communities. In *Proceedings of the international AAAI conference on web and social media*, Vol. 10. 171–180.

[15] Tad Hogg and Kristina Lerman. 2012. Social dynamics of digg. *EPJ Data Science* 1, 1 (2012), 5.

[16] Petter Holme. 2015. Modern temporal network theory: a colloquium. *The European Physical Journal B* 88, 9 (2015), 234.

[17] Petter Holme, Christofer R Edling, and Fredrik Liljeros. 2004. Structure and time evolution of an Internet dating community. *Social Networks* 26, 2 (2004), 155–174.

[18] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. 2014. Querying k-truss community in large and dynamic graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1311–1322.

[19] Wei-Chun Hung and Chih-Ying Tseng. 2021. Maximum (L, K)-Lasting Cores in Temporal Social Networks. In *DASFAA Intl Workshops*. Cham, 336–352.

[20] Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *ECML*. Springer, 217–226.

[21] Dániel Kondor, István Csabai, János Szüle, Márton Pósfai, and Gábor Vattay. 2014. Inferring the interplay between network structure and market effects in Bitcoin. *New Journal of Physics* 16, 12 (2014), 125003.

[22] Yi-Xiu Kong, Gui-Yuan Shi, Rui-Jie Wu, and Yi-Cheng Zhang. 2019. k-core: Theories and applications. *Physics Reports* 832 (2019), 1–32.

[23] Lauri Kovanen, Márton Karsai, Kimmo Kaski, János Kertész, and Jari Saramäki. 2011. Temporal motifs in time-dependent networks. *Journal of Statistical Mechanics: Theory and Experiment* 2011, 11 (2011), P11005.

[24] Rong-Hua Li, Jiao Su, Lu Qin, Jeffrey Xu Yu, and Qiangqiang Dai. 2018. Persistent community search in temporal networks. In *ICDE*. IEEE, 797–808.

[25] Longlong Lin, Pingpeng Yuan, Rong-Hua Li, and Hai Jin. 2021. Mining Diversified Top-r Lasting Cohesive Subgraphs on Temporal Networks. *IEEE Transactions on Big Data* 8, 6 (2021), 1537–1549.

[26] Quintino Francesco Lotito and Alberto Montresor. 2020. Efficient Algorithms to Mine Maximal Span-Trusses From Temporal Graphs. *arXiv preprint arXiv:2009.01928* (2020).

[27] Fragkiskos D Malliaros, Christos Giatsidis, Apostolos N Papadopoulos, and Michalis Vazirgiannis. 2020. The core decomposition of networks: Theory, algorithms and applications. *The VLDB J.* 29, 1 (2020), 61–92.

[28] Rossana Mastrandrea, Julie Fournet, and Alain Barrat. 2015. Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys. *PloS one* 10, 9 (2015).

[29] Othon Michail. 2016. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics* 12, 4 (2016), 239–280.

[30] Alan E. Mislove. 2009. *Online social networks: measurement, analysis, and applications to distributed information systems.* Ph. D. Dissertation. Rice University.

[31] Mahdihusain Momin, Raj Kamal, Shantwana Dixit, Sayan Ranu, and Amitabha Bagchi. 2023. KWIQ: Answering k-core Window Queries in Temporal Networks. (2023).

[32] Vincenzo Nicosia, John Tang, Cecilia Mascolo, Mirco Musolesi, Giovanni Russo, and Vito Latora. 2013. Graph metrics for temporal networks. *Temporal networks* (2013), 15–40.

[33] Vincenzo Nicosia, John Tang, Mirco Musolesi, Giovanni Russo, Cecilia Mascolo, and Vito Latora. 2012. Components in time-varying graphs. *Chaos: An interdisc. journal of nonlinear science* 22, 2 (2012), 023101.

[34] Lutz Oettershagen, Athanasios L Konstantinidis, and Giuseppe F Italiano. 2022. Inferring Tie Strength in Temporal Networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 69–85.

[35] Lutz Oettershagen, Athanasios L. Konstantinidis, and Giuseppe F. Italiano. 2023. An Edge-Based Decomposition Framework for Temporal Networks. arXiv:2309.11843 [cs.SI]

[36] Lutz Oettershagen, Nils M. Kriege, and Petra Mutzel. 2023. A Higher-Order Temporal H-Index for Evolving Networks. In *SIGKDD*. ACM, 1770–1782. https://doi.org/10.1145/3580305.3599242

[37] L. Oettershagen and P. Mutzel. 2022. Tglib: an open-source library for temporal graph analysis. In *ICDM Workshops*. IEEE, 1240–1245.

[38] Lutz Oettershagen, Petra Mutzel, and Nils M Kriege. 2022. Temporal walk centrality: ranking nodes in evolving networks. In *Proceedings of the ACM Web Conference 2022*. 1640–1650.

[39] Pietro Panzarasa, Tore Opsahl, and Kathleen M. Carley. 2009. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *J. Am. Soc. for Inf. Sc. and Tech.* 60, 5 (2009), 911–932.

[40] Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. 2017. Motifs in temporal networks. In *Proceedings of ACM WSDM*. 601–610.

[41] Hongchao Qin, Rong-Hua Li, Guoren Wang, Xin Huang, Ye Yuan, and Jeffrey Xu Yu. 2020. Mining stable communities in temporal networks by density-based clustering. *IEEE Transactions on Big Data* 8, 3 (2020), 671–684.

[42] Hongchao Qin, Rong-Hua Li, Ye Yuan, Guoren Wang, Lu Qin, and Zhiwei Zhang. 2022. Mining Bursting Core in Large Temporal Graphs. *Proceedings of the VLDB Endowment* (2022).

[43] Giulio Rossetti and Rémy Cazabet. 2018. Community discovery in dynamic networks: a survey. *ACM computing surveys (CSUR)* 51, 2 (2018), 1–37.

[44] Diego Santoro and Ilie Sarpe. 2022. ONBRA: Rigorous Estimation of the Temporal Betweenness Centrality in Temporal Networks. In *Proceedings of the ACM Web Conference 2022*. 1579–1588.

[45] Ilie Sarpe and Fabio Vandin. 2021. OdeN: simultaneous approximation of multiple motif counts in large temporal networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1568–1577.

[46] Ilie Sarpe, Fabio Vandin, and Aristides Gionis. 2024. Scalable Temporal Motif Densest Subnetwork Discovery. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2536–2547.

[47] Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks* 5, 3 (1983), 269–287.

[48] Chengcheng Shao, Pik-Mai Hui, Lei Wang, Xinwen Jiang, Alessandro Flammini, Filippo Menczer, and Giovanni Luca Ciampaglia. 2018. Anatomy of an online misinformation network. *Plos one* 13, 4 (2018).

[49] Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos. 2016. Corescope: Graph mining using k-core analysis—patterns, anomalies and algorithms. In *IEEE ICDM*. IEEE, 469–478.

[50] Yishu Wang, Ye Yuan, Yuliang Ma, and Guoren Wang. 2019. Time-dependent graphs: Definitions, applications, and algorithms. *Data Sci. and Engin.* 4, 4 (2019), 352–366.

[51] Huanhuan Wu, James Cheng, Yi Lu, Yiping Ke, Yuzhen Huang, Da Yan, and Hejun Wu. 2015. Core decomposition in large temporal graphs. In *Big Data*. IEEE, 649–658.

[52] Junyong Yang, Ming Zhong, Yuanyuan Zhu, Tieyun Qian, Mengchi Liu, and Jeffery Xu Yu. 2023. Scalable Time-Range k-Core Query on Temporal Graphs. *arXiv preprint arXiv:2301.03770* (2023).

[53] Michael Yu, Dong Wen, Lu Qin, Ying Zhang, Wenjie Zhang, and Xuemin Lin. 2021. On querying historical k-cores. *Proceedings of the VLDB Endowment* (2021).

[54] Ming Zhong, Junyong Yang, Yuanyuan Zhu, Tieyun Qian, Mengchi Liu, and Jeffery Xu Yu. 2024. A Unified and Scalable Algorithm Framework of User-Defined Temporal $(k, \mathcal{X})$-Core Query. *IEEE Transactions on Knowledge and Data Engineering* (2024).