Noname manuscript No.
(will be inserted by the editor)

Temporal Hypergraph Motifs

Geon Lee · **Kijung Shin**

Received: 29 Jan 2022 / Revised: 16 Dec 2022 / Accepted: 06 Jan 2023

Abstract Group interactions arise in our daily lives (email communications, on-demand ride sharing, and comment interactions on online communities, to name a few), and they together form hypergraphs that evolve over time. Given such temporal hypergraphs, how can we describe their underlying design principles? If their sizes and time spans are considerably different, how can we compare their structural and temporal characteristics?

In this work, we define 96 *temporal hypergraph motifs* (TH-motifs) and propose the relative occurrences of their instances as an answer to the above questions. TH-motifs categorize the relational and temporal dynamics among three connected hyperedges that appear within a short time. For scalable analysis, we develop THYME$^+$, a fast and exact algorithm for counting the instances of TH-motifs in massive hypergraphs, and we show that THYME$^+$ is up to 2,163$\times$ *faster* while requiring less space than baseline approaches. In addition to exact counting algorithms, we design three versions of sampling algorithms for approximate counting. We theoretically analyze the accuracy of the proposed methods, and we empirically show that the most advanced algorithm, THYME-A*, is up to 11.1$\times$ more accurate than baseline approaches. Using the algorithms, we investigate 11 real-world temporal hypergraphs from various domains. We demonstrate that TH-motifs provide important information useful for downstream tasks and reveal interesting patterns, including the striking similarity between temporal hypergraphs from the same domain.

Keywords Hypergraph · Temporal Hypergraph Motif · Network Motif · Hypergraphlet · Graphlet

1 Introduction

Interactions in real-world systems are complex, and in many cases, they are beyond pairwise: email communications, on-demand ride sharing, and comment interactions on online

Geon Lee
Kim Jaechul Graduate School of AI, KAIST, Seoul, South Korea
E-mail: geonlee0325@kaist.ac.kr

Kijung Shin (Corresponding Author)
Kim Jaechul Graduate School of AI and School of Electrical Engineering, KAIST, Seoul, South Korea
E-mail: kijungs@kaist.ac.kr

communities, to name a few. These group interactions together form a *hypergraph*, which consists of a set of nodes and a set of hyperedges (see Fig. 1(a) for an example). Each *hyperedge* is a subset of *any* number of nodes, and by naturally representing a group interaction among multiple individuals or objects, it contributes to the powerful expressiveness of hypergraphs.

Recently, several empirical studies have revealed structural and temporal properties of real-world hypergraphs. Pervasive structural patterns include (a) heavy-tailed distributions of degrees, edge sizes, and intersection sizes (Kook et al., 2020); (b) giant connected components (Do et al., 2020), and small diameters (Do et al., 2020); and (c) substantial overlaps of hyperedges with homophily (Lee et al., 2021). Temporal properties observed commonly in various time-evolving hypergraphs include (a) significant overlaps between temporally adjacent hyperedges (Benson et al., 2018b); and (b) diminishing overlaps, densification, and shrinking diameters (Kook et al., 2020).

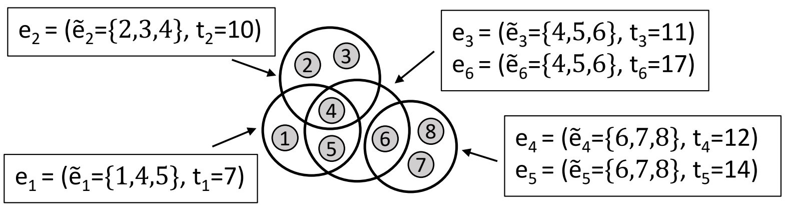
In addition to these macroscopic properties, local connectivity and dynamics in real-world hypergraphs have been studied. Benson et al. (2018a) examined the interactions among a fixed number of nodes, with a focus on their relations with the emergence of a hyperedge containing all the nodes. Lee et al. (2020) inspected the overlaps between three hyperedges, which they categorize into 26 patterns called hypergraph motifs (h-motifs). Comparing the relative counts of each h-motif's instances revealed that local structures are particularly similar between hypergraphs from the same domain but different across domains. In h-motifs, however, temporal dynamics are completely ignored.

This line of research has also revealed that specialized analysis tools (e.g., h-motifs (Lee et al., 2020) and multi-level decomposition (Do et al., 2020)) are useful for extracting unique high-order information that hypergraphs convey and also for coping with additional complexity due to the flexibility in the size of hyperedges. Simply utilizing graph analysis tools (e.g., network motifs (Milo et al., 2002)) after converting hypergraphs into pairwise graphs is often limited in addressing the above challenges (Lee et al., 2020; Yoon et al., 2020).

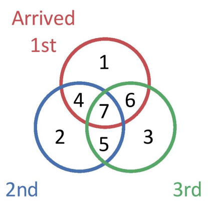
Motivated by interesting patterns that temporal network motifs revealed in ordinary graphs (Paranjape et al., 2017; Li et al., 2018; Kovanen et al., 2011; Gurukar et al., 2015; Redmond and Cunningham, 2013), we define 96 *temporal hypergraph motifs* (TH-motifs) for local pattern analysis of time-evolving hypergraphs. TH-motifs generalize the notion of static h-motifs, which completely ignore temporal information, and describe both relational and temporal dynamics among three connected temporal hyperedges that arrive within a short time. Specifically, given three connected hyperedges e_i, e_j, and e_k, all of which arrive within δ time units, TH-motifs describe their connectivity based on the emptiness of the seven subsets of them shown in Fig. 1(b). In the temporal perspective, the relative arrival orders of e_i, e_j, and e_k are taken into account, and thus patterns that are indistinguishable using static h-motifs can be characterized using TH-motifs.

Given a temporal hypergraph, where a timestamp is attached to each hyperedge (see Fig. 1(a) for an example), we summarize its local structural and temporal characteristics using the relative occurrence of 96 TH-motifs' instances. That is, we obtain a vector of length 96 regardless of the sizes and time spans of hypergraphs, and thus local characteristics of different hypergraphs can be easily compared.

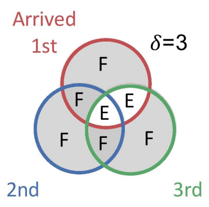
Another focus of this paper is the problem of counting TH-motifs' instances. Since the number of three connected hyperedges can be orders of magnitude larger than the number of hyperedges, directly enumerating all of them is computationally prohibitive, especially for massive hypergraphs. We develop THYME^{+} (**T**emporal **Hy**pergraph **M**otif C**e**nsus), which exactly counts each TH-motif's instances while avoiding direct enumeration. In our experi-



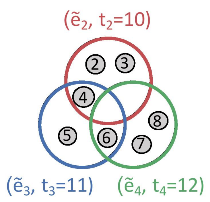
(a) An example temporal hypergraph



(b) 7 regions for defining TH-motifs



(c) The definition of TH-motif 77



(d) An instance of TH-motif 77

Fig. 1: **(a)** A temporal hypergraph with 8 nodes and 6 temporal hyperedges. **(b)** The 7 regions in the Venn diagram representation for defining TH-motifs. **(c)** The definition of TH-motif 77. 'F' and 'E' stand for 'filled' and 'empty', respectively. **(d)** The sequence $\langle e_2, e_3, e_4 \rangle$ is an instance of TH-motif 77.

ments, THYME$^+$ is up to **2,163× faster** than the direct extension of a recent exact temporal network motif counting algorithm (Paranjape et al., 2017), which enumerates every static h-motif in the induced static hypergraph. THYME$^+$ makes the best use of our two findings in real-world hypergraphs that temporal hyperedges tend to be (1) repetitive and (2) temporally local. These findings about duplicated (i.e., completely overlapped) hyperedges complement the findings in (Benson et al., 2018b), which focus mainly on partial overlaps.

In addition to exact counting algorithms, we design three different versions of sampling algorithms: THYME-A, THYME-A$^+$, and THYME-A* for approximately counting the instances of each TH-motif. All versions yield unbiased estimates of the counts by partially exploring the input temporal hypergraph by sampling time intervals. Specifically, we propose THYME-A*, the most accurate sampling algorithm by reducing the variances of THYME-A and THYME-A$^+$. According to our empirical analyses, THYME-A* is not only faster than the direct extension of a recent sampling method for counting temporal networks motifs in pairwise graphs (Liu et al., 2019) but also up to **11.1× more accurate** than it. These experimental results are consistent with our theoretical analyses.

Using TH-motifs, THYME$^+$, and THYME-A*, we investigate 11 real-world hypergraphs from 5 distinct domains. Our empirical study demonstrates that TH-motifs are informative, capturing both structural and temporal characteristics. Specifically, using the counts of incident TH-motifs' instances as features brings up to **25.7% improvement in the accuracy** of a hyperedge prediction task, compared to when static h-motifs are used instead of TH-motifs. Moreover, TH-motifs reveal interesting patterns, including the striking similarity between hypergraphs from the same domain.

In summary, our contributions are as follows:

1. **New concept:** We define 96 temporal hypergraph motifs (TH-motifs) for characterizing local structures and dynamics in hypergraphs of various sizes.
2. **Fast and exact algorithms:** We develop fast algorithms for exactly counting the instances of TH-motifs, and they are up to 2,163× faster than baseline.

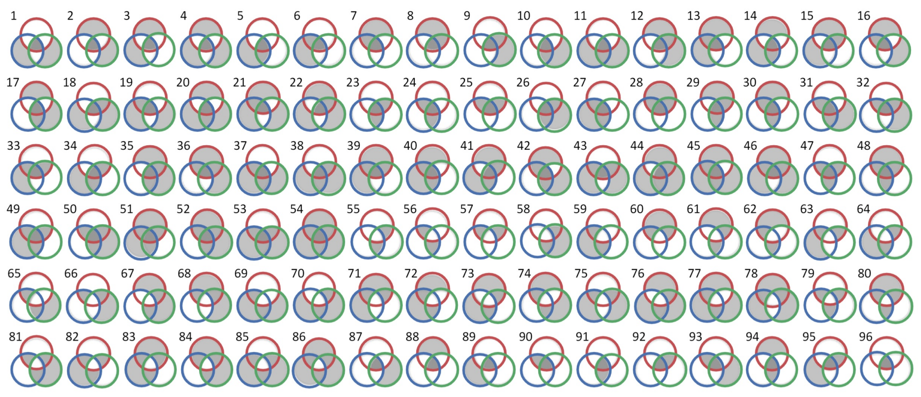


Fig. 2: **The 96 temporal hypergraph motifs (TH-motifs).** In each TH-motif, the red hyperedge arrives first followed by the blue one and then the green one. Each of the 7 distinct regions in the Venn diagram representation is colored white if it is empty, and it is colored grey if it is filled with at least one node. See Fig. 1(d) for an instance of TH-motif 77.

3. **Accurate approximate algorithms**: We develop fast but accurate sampling algorithms for counting the instances of each TH-motif, and they are at most $11.1\times$ more accurate than baseline.
4. **Empirical discoveries:** We demonstrate the usefulness of TH-motifs by uncovering the design principles of 11 real-world temporal hypergraphs from 5 different domains.

Reproducibility: The source code and datasets used in this work are available at `https://github.com/geon0325/THyMe`.

This work is an extended version of (Lee and Shin, 2021), with (a) new sampling algorithms for approximate counting of instances of TH-motifs, (b) theoretical analyses of exact and approximate algorithms, and (c) additional experimental results.

In Section 2 we review preliminaries and related prior works. In Section 3, we present the concept of TH-motifs. In Section 4, we develop algorithms for exactly counting the instances of TH-motifs. In Section 5, we develop algorithms for approximately counting the instances of TH-motifs. In Section 6, we empirically analyze real-world temporal hypergraphs through the lens of TH-motifs. Lastly, in Section 7, we offer conclusions.

2 Preliminaries and Related Works

In this section, we first review the concept of hypergraphs. Then, we introduce hypergraph motifs (h-motifs), which are designed for static hypergraphs. Lastly, we discuss other related works. Refer to Table 1 for the frequently-used notations.

2.1 Basic Concepts: Static and Temporal Hypergraphs

A *hypergraph* $G = (V, E)$ consists of a set of nodes $V = \{v_1, ..., v_{|V|}\}$ and a set of hyperedges $E = \{\tilde{e}_1, ..., \tilde{e}_{|E|}\}$. Each *hyperedge* $\tilde{e} \in E$ is a non-empty set of an arbitrary number of nodes. A *temporal hypergraph* $T = (V, \mathscr{E})$ on a node set V is an ordered sequence of *temporal*

Table 1: Frequently-used notations.

Notation	Definition
$T = (V, \mathscr{E})$	temporal hypergraph with temporal hyperedges $\mathscr{E}$
$G_T = (V, E_{\mathscr{E}})$	induced static hypergraph of the temporal hypergraph T
$e_i = (\tilde{e}_i, t_i)$	temporal hyperedge with nodes $\tilde{e}_i$ arrived at time t_i
$I(\tilde{e})$	set of temporal hyperedges whose nodes are $\tilde{e}$
$h(\tilde{e}_i, \tilde{e}_j, \tilde{e}_k)$	TH-motif corresponding to an instance $\langle e_i, e_j, e_k \rangle$
$P = (V_P, E_P)$	projected graph in THyMe
$Q = (V_Q, E_Q, t_Q)$	projected graph in THyMe$^+$

hyperedges. Each i^{th} temporal hyperedge $e_i = (\tilde{e}_i, t_i)$ where $\tilde{e}_i \subseteq V$ is the set of nodes and t_i is the time of arrival. Two distinct temporal hyperedges $e_i = (\tilde{e}_i, t_i)$ and $e_j = (\tilde{e}_j, t_j)$ are *duplicated* if they share exactly the same set of nodes, i.e., $\tilde{e}_i = \tilde{e}_j$. We assume the sequence is ordered and timestamps are unique, i.e., if $i < j$, then $t_i < t_j$. Let $t_{\min}$ and $t_{\max}$ be the first and the last timestamps of the temporal hypergraph, i.e., $t_{\min} = t_1$ and $t_{\max} = t_{|\mathscr{E}|}$. We denote the set of temporal hyperedges whose nodes are $\tilde{e}$ (i.e., those *inducing* $\tilde{e}$) by $I(\tilde{e}) := \{e_i = (\tilde{e}_i, t_i) \in \mathscr{E} : \tilde{e}_i = \tilde{e}\}$. The temporal hypergraph T induces a static hypergraph $G_T = (V, E_{\mathscr{E}})$ where timestamps and duplicated temporal hyperedges are ignored. That is, a hyperedge $\tilde{e} \in E_{\mathscr{E}}$ in G_T exists if and only if $I(\tilde{e}) \neq \emptyset$. Notably, the number of temporal hyperedges is typically much larger than that of static hyperedges in the induced hypergraph, i.e., $|\mathscr{E}| \gg |E_{\mathscr{E}}|$.

2.2 Static Hypergraph Motifs (h-motifs)

Hypergraph motifs (h-motifs) (Lee et al., 2020) are tools for understanding the local structural properties of static hypergraphs. Given three connected hyperedges, h-motifs describe their connectivity patterns by the emptiness of each of seven subsets: (1) $\tilde{e}_i \setminus \tilde{e}_j \setminus \tilde{e}_k$, (2) $\tilde{e}_j \setminus \tilde{e}_k \setminus \tilde{e}_i$, (3) $\tilde{e}_k \setminus \tilde{e}_i \setminus \tilde{e}_j$, (4) $\tilde{e}_i \cap \tilde{e}_j \setminus \tilde{e}_k$, (5) $\tilde{e}_j \cap \tilde{e}_k \setminus \tilde{e}_i$, (6) $\tilde{e}_k \cap \tilde{e}_i \setminus \tilde{e}_j$, and (7) $\tilde{e}_i \cap \tilde{e}_j \cap \tilde{e}_k$. While there can exist 2^7 possible cases of emptiness, 26 cases of them are considered after excluding symmetric, duplicated, and disconnected ones. Since non-pairwise interactions among the hyperedges (such as $\tilde{e}_i \cap \tilde{e}_j \cap \tilde{e}_k$) are taken into account, h-motifs effectively capture the high-order information of the overlapping patterns of the hyperedges. It is shown empirically that their occurrences in the real-world hypergraphs are significantly different from those in randomized hypergraphs. Moreover, the relative occurrences are particularly similar between hypergraphs from the same domain, while they are distinct between hypergraphs from different domains. Note that h-motifs, which are originally designed for static hypergraphs, completely ignore the temporal information.

2.3 Other Related Works

In this subsection, we review prior works on network motifs and empirical analyses of hypergraphs.

Network Motifs. Network motifs are fundamental building blocks of real-world graphs (Shen-Orr et al., 2002; Milo et al., 2002). Their relative occurrences in real-world graphs are significantly different from those in randomized ones (Milo et al., 2002) and unique within each domain (Milo et al., 2004). While they were originally defined on a static graph, they have been extended to temporal (Paranjape et al., 2017), heterogeneous (Rossi et al., 2020a; Li et al., 2018), and bipartite (Borgatti and Everett, 1997) graphs, as well as hypergraphs (Lee et al., 2020). Their usefulness has been demonstrated in a wide range of graph applications including community detection (Benson et al., 2016; Li et al., 2019; Tsourakakis et al., 2017; Yin et al., 2017; Arenas et al., 2008), ranking (Zhao et al., 2018), and embedding (Yu et al., 2019; Rossi et al., 2018a, 2020b; Lee et al., 2019; Rossi et al., 2018b).

Temporal Network Motifs: The notion of network motifs has been extended to temporal networks to describe patterns in sequences of temporal edges. Several definitions of temporal motifs have been used, and most of them consider the *temporal connectivity* between the edges. Kovanen et al. (2011) and Gurukar et al. (2015) consider δ-adjacency between temporal edges. That is, every consecutive edge should share a node and arrive within δ time units. Several counting algorithms for such patterns have been proposed (Redmond and Cunningham, 2013; Kovanen et al., 2011; Gurukar et al., 2015). Another definition of temporal motifs describes patterns of sequences of temporal edges where all edges arrive within δ time units (Paranjape et al., 2017) while taking their relative arrival orders into consideration. In this work, we define TH-motifs based on the notion of temporal motifs defined in (Paranjape et al., 2017) due to its simplicity and effectiveness.

Hypergraphs: Hypergraphs, which naturally represent group interactions among multiple individuals or objects, are useful in numerous application areas including computer vision (Yu et al., 2012), bioinformatics (Hwang et al., 2008), social network analysis (Yang et al., 2019), and circuit design (Karypis et al., 1999). Using hypergraphs, various analytical and predictive tasks have been performed, including classification (Yadati et al., 2018; Feng et al., 2019), clustering (Amburg et al., 2020; Li and Milenkovic, 2017), hyperedge prediction (Benson et al., 2018a; Yoon et al., 2020), and anomaly detection (Lee et al., 2022a). An extensive studies on local structural patterns (Benson et al., 2018a; Lee et al., 2020; Kim et al., 2022), global structural patterns (Chodrow, 2020; Do et al., 2020; Choe et al., 2022), and dynamic patterns (Kook et al., 2020; Benson et al., 2018b; Choo and Shin, 2022; Ko et al., 2022) have been discovered (see (Lee et al., 2022b)). In this work, we define temporal hypergraph motifs for describing dynamic and microscopic structural patterns of real-world temporal hypergraphs.

3 Proposed Concepts

In this section, we propose *temporal hypergraph motifs* (TH-motifs), which are tools for understanding the local structural and temporal characteristics of temporal hypergraphs. We introduce the definition and their relevant concepts.

Definition: TH-motifs describe structural and temporal patterns in sequences of three connected temporal hyperedges that are close in time. Note that three hyperedges are *connected* if and only if one among them overlaps with the others. Specifically, given three connected temporal hyperedges $\langle e_i = (\tilde{e}_i, t_i), e_j = (\tilde{e}_j, t_j), e_k = (\tilde{e}_k, t_k)\rangle$ where $t_i < t_j < t_k$ and $t_k - t_i \leq \delta$ (i.e., they arrive within a predefined time interval δ), TH-motifs describe the emptiness of the 7 subsets: (1) $\tilde{e}_i \setminus \tilde{e}_j \setminus \tilde{e}_k$, (2) $\tilde{e}_j \setminus \tilde{e}_k \setminus \tilde{e}_i$, (3) $\tilde{e}_k \setminus \tilde{e}_i \setminus \tilde{e}_j$, (4) $\tilde{e}_i \cap \tilde{e}_j \setminus \tilde{e}_k$, (5) $\tilde{e}_j \cap \tilde{e}_k \setminus \tilde{e}_i$, (6) $\tilde{e}_k \cap \tilde{e}_i \setminus \tilde{e}_j$, and (7) $\tilde{e}_i \cap \tilde{e}_j \cap \tilde{e}_k$. That is, in the structural aspect, TH-motifs describe the emptiness of the seven distinct regions in the Venn diagram representation (see Fig. 1(b)), effectively capturing the high-order connectivity among three hyperedges. In the temporal aspects, TH-motifs take the relative arrival orders of three hyperedges and their time interval into consideration. While there can exist 2^7 possible cases of emptiness, we consider 96 cases of them, which are called TH-motif 1 to TH-motif 96, after excluding those describing disconnected hyperedges. We visualize the 96 TH-motifs in Fig. 2. Recall that static h-motifs completely ignore temporal information, and also assume that every hyperedge is unique, while TH-motifs also describe the patterns among duplicated temporal hyperedges. Thus, while static h-motifs distinguish only 26 different patterns, TH-motifs distinguish 96 different patterns by considering temporal dynamics in addition to the connectivity.

Instance of TH-motifs: A sequence $\langle e_i, e_j, e_k \rangle$ of three temporal hyperedges is an *instance* of TH-motif t if its relational and temporal dynamics are described by TH-motif t (see Fig. 1(d) for an example). For each instance $\langle e_i, e_j, e_k \rangle$, we denote its corresponding TH-motif by $h(\tilde{e}_i, \tilde{e}_j, \tilde{e}_k)$. We let $\triangle(\langle e_i, e_j, e_k \rangle)$ be the *duration* (the time interval between the arrivals of the first and the last temporal hyperedge) of the instance, i.e., $\triangle(\langle e_i, e_j, e_k \rangle) = t_k - t_i$. In addition, let $\mathrm{t}(\langle e_i, e_j, e_k \rangle)$ be the *closure time* (the timestamp of the last temporal hyperedge) of the instance, i.e., $\mathrm{t}(\langle e_i, e_j, e_k \rangle) = t_k$. The time complexity of computing $h(\tilde{e}_i, \tilde{e}_j, \tilde{e}_k)$ is given in Lemma 1.

Lemma 1 (Complexity of computing $h(\tilde{e}_i, \tilde{e}_j, \tilde{e}_k)$) *Given an instance $\langle e_i, e_j, e_k \rangle$ of a TH-motif, the time complexity of computing $h(\tilde{e}_i, \tilde{e}_j, \tilde{e}_k)$ is $O(\min(|\tilde{e}_i|, |\tilde{e}_j|, |\tilde{e}_k|))$.*
Proof. *Once we compute $|\tilde{e}_i \cap \tilde{e}_j \cap \tilde{e}_k|$, by the assumptions above, the cardinalities of the remaining six subsets are obtained immediately in $O(1)$ time as follows:*

(1) $|\tilde{e}_i \setminus \tilde{e}_j \setminus \tilde{e}_k| = |\tilde{e}_i| - |\tilde{e}_i \cap \tilde{e}_j| - |\tilde{e}_k \cap \tilde{e}_i| + |\tilde{e}_i \cap \tilde{e}_j \cap \tilde{e}_k|$
(2) $|\tilde{e}_j \setminus \tilde{e}_k \setminus \tilde{e}_i| = |\tilde{e}_j| - |\tilde{e}_j \cap \tilde{e}_k| - |\tilde{e}_i \cap \tilde{e}_j| + |\tilde{e}_i \cap \tilde{e}_j \cap \tilde{e}_k|$
(3) $|\tilde{e}_k \setminus \tilde{e}_i \setminus \tilde{e}_j| = |\tilde{e}_k| - |\tilde{e}_k \cap \tilde{e}_i| - |\tilde{e}_j \cap \tilde{e}_k| + |\tilde{e}_i \cap \tilde{e}_j \cap \tilde{e}_k|$
(4) $|\tilde{e}_i \cap \tilde{e}_j \setminus \tilde{e}_k| = |\tilde{e}_i \cap \tilde{e}_j| - |\tilde{e}_i \cap \tilde{e}_j \cap \tilde{e}_k|$
(5) $|\tilde{e}_j \cap \tilde{e}_k \setminus \tilde{e}_i| = |\tilde{e}_j \cap \tilde{e}_k| - |\tilde{e}_i \cap \tilde{e}_j \cap \tilde{e}_k|$
(6) $|\tilde{e}_k \cap \tilde{e}_i \setminus \tilde{e}_j| = |\tilde{e}_k \cap \tilde{e}_i| - |\tilde{e}_i \cap \tilde{e}_j \cap \tilde{e}_k|$

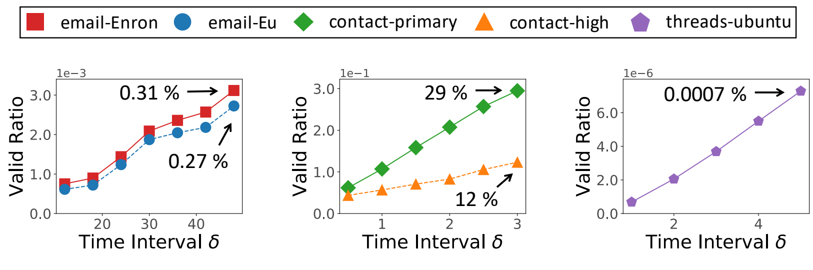
Let $s = \arg\min_{t \in \{i,j,k\}} |\tilde{e}_t|$. We can compute $|\tilde{e}_i \cap \tilde{e}_j \cap \tilde{e}_k|$ in $O(|\tilde{e}_s|)$ time by checking whether each node in $\tilde{e}_s$ is contained in $\tilde{e}_x$ and $\tilde{e}_y$ where $x \neq y \neq s$. Thus, the time complexity of computing $h(\tilde{e}_i, \tilde{e}_j, \tilde{e}_k)$ is $O(|\tilde{e}_s|) = O(\min(|\tilde{e}_i|, |\tilde{e}_j|, |\tilde{e}_k|))$.

Triple, Pair, and Single Inducing TH-motifs: The 96 TH-motifs can be categorized into three types based on the number of underlying static hyperedges. A TH-motif is *triple-inducing* if underlying hyperedges in its instance $\langle e_i, e_j, e_k \rangle$ are distinct (i.e., $\tilde{e}_i \neq \tilde{e}_j$, $\tilde{e}_j \neq \tilde{e}_k$, and $\tilde{e}_k \neq \tilde{e}_i$), as in TH-motifs 1-86. If two are duplicated while the remaining one is different, as in TH-motifs 87-95, it is *pair-inducing*. If all three hyperedges are duplicated, as in TH-motif 96, it is *single-inducing*.

4 Exact Counting Algorithms

In this section, we describe methodologies for exactly counting the instances of each TH-motif in the input temporal hypergraph. We first present DP, which extends a recent exact counting algorithm (Paranjape et al., 2017) for temporal network motifs. Then, we describe THyMe, a preliminary version of our proposed algorithm THyMe$^+$. Lastly, we propose THyMe$^+$ (**T**emporal **Hy**pergraph **M**otif C**e**nsus), a fast and efficient algorithm that addresses the limitations of the previous ones.

Remarks: The problem of counting TH-motifs has additional technical challenges while it bears some similarity with counting static h-motifs or temporal network motifs. First, the



(a) `email` (b) `contact` (c) `threads`

Fig. 3: Only a small fraction of static h-motifs' instances in the induced static hypergraphs are induced by any valid instance of TH-motifs. We report the results in small datasets where the instances of static h-motifs can be exactly counted.

number of temporal hyperedges is typically much larger than that of hyperedges in the underlying static hypergraph. For example, the 11 considered real-world temporal hypergraphs (see Section 6.1) have up to $1.2-22.0\times$ more temporal hyperedges than the underlying static ones. This incurs significant bottlenecks of enumeration methods, and thus fast algorithms are demanded. Temporal network motifs are defined only by pairwise interactions among a fixed number of nodes and their timestamps. However, TH-motifs are defined not just by pairwise interactions but also by non-pairwise interactions among three hyperedges, in addition to their timestamps.

4.1 Dynamic Programming (DP): Extension of Paranjape et al. (2017)

We present Dynamic Programming (DP), which is a baseline approach for counting the instances of each TH-motif in the input temporal hypergraph T.

Counting in DP: Given an input temporal hypergraph $T = (V, \mathscr{E})$, DP enumerates the instances of static h-motifs in the induced static hypergraph $G_T = (V, E_{\mathscr{E}})$. This step can be processed by using an existing algorithm provided in (Lee et al., 2020). For each instance $\{\tilde{e}_i, \tilde{e}_j, \tilde{e}_k\}$ of static h-motif in G_T, DP counts the instances of each TH-motif whose temporal hyperedges (a) induce the static h-motif instance and (b) arrive within δ time. To this end, we adapt the dynamic programming scheme provided by (Paranjape et al., 2017), as described in detail in Appendix A.

Limitations of DP: Using dynamic programming, DP avoids enumerating over all instances of TH-motifs. However, it still enumerates all instances of static h-motifs in the induced hypergraph G_T, most of which however are not induced by any valid instance of TH-motifs, as seen in Fig. 3. For example, in `threads-ubuntu`, only 0.0007% of the static h-motifs instances are induced by any valid instance of TH-motifs when δ is 5 hours. That is, DP enumerates every three connected hyperedges in G_T, ignoring any temporal information, while we are interested only in three connected temporal hyperedges that arrive within in a short period of time.

Algorithm 1: THYME: Preliminary Algorithm for Exact Counting of TH-motifs

```
Input  : (1) temporal hypergraph: $T = (V, \mathscr{E})$
         (2) time interval $\delta$
Output: # of each temporal h-motif $M$'s instances: $C[M]$
$C \leftarrow$ map initialized to 0
$P = (V_P = \varnothing, E_P = \varnothing)$
$w_s \leftarrow 1$
for each temporal hyperedge $e_i = (\tilde{e}_i, t_i) \in \mathscr{E}$ do
    insert($e_i$)
    while $t_{w_s} + \delta < t_i$ do
        remove($e_{w_s}$)
        $w_s \leftarrow w_s + 1$
    $S \leftarrow$set of three connected temporal hyperedges including $e_i$
    for each instance $\langle e_j, e_k, e_i \rangle \in S$ do
        $C[h(\tilde{e}_j, \tilde{e}_k, \tilde{e}_i)] \leftarrow C[h(\tilde{e}_j, \tilde{e}_k, \tilde{e}_i)] + 1$
return $M$
Procedure insert($e_i = (\tilde{e}_i, t_i)$)
    $V_P \leftarrow V_P \cup \{e_i\}$
    $N_{e_i} \leftarrow \{e : e \in V_P \setminus \{e_i\}$ and $\tilde{e}_i \cap \tilde{e} \neq \varnothing\}$
    $E_P \leftarrow E_P \cup \{(e_i, e) : e \in N_{e_i}\}$
Procedure remove($e_i = (\tilde{e}_i, t_i)$)
    $V_P \leftarrow V_P \setminus \{e_i\}$
    $N_{e_i} \leftarrow \{e : e \in V_P$ and $\tilde{e}_i \cap \tilde{e} \neq \varnothing\}$
    $E_P \leftarrow E_P \setminus \{(e_i, e) : e \in N_{e_i}\}$
```

4.2 THYME: Preliminary Version of the Proposed Algorithm

To address the limitations of DP, we present THYME, a preliminary version of our proposed algorithm THYME$^+$. THYME directly enumerates each instance of TH-motifs, instead of those of static h-motifs, to avoid the unnecessary search. To this end, THYME concisely considers the temporal hyperedges that occur in the δ-sized temporal window. In response to the arrival of a new temporal hyperedge e_i at time t_i, the temporal window moves to $[t_i - \delta, t_i]$. It maintains only a succinct projected graph $P = (V_P, E_P)$ that represents the connectivity between the temporal hyperedges that occur within the current temporal window. As the window moves, the projected graph P is incrementally updated, reflecting the changes of the current temporal hyperedges. Using P, THYME exhaustively enumerates the instances of TH-motifs.

Projected Graph in THYME: The projected graph $P = (V_P, E_P)$ is a graph where each node is a temporal hyperedge and two nodes are adjacent by an edge if their corresponding temporal hyperedges share any nodes. In THYME, P is maintained on the fly, with response to the temporal hyperedges that either enter or exit the sliding time window. The update schemes are described as `insert` and `remove`, respectively, in Algorithm 1. In `insert`, a temporal hyperedge e_i is added as a node (line 14) and its neighbors (i.e., those in V_P that are adjacent to e_i) are joined by edges (lines 15-16). In `remove`, a temporal hyperedge e_i, as well as its incident edges are removed from V_P and E_P, respectively (lines 18-20).

Counting in THYME: The counting procedure of THYME is described in Algorithm 1. The sets of nodes and edges of the projected graph P are initialized to empty maps, i.e., $V_P = \varnothing$ and $E_P = \varnothing$ (line 2). Once a temporal hyperedge $e_i = (\tilde{e}_i, t_i) \in \mathscr{E}$ arrives, the temporal window is moved to $[t_i - \delta, t_i]$ and the projected graph P is updated accordingly, as described above. Then, it enumerates the instances of three connected nodes in P, which corresponds to the instances of TH-motifs of T containing e_i (line 9). For each instance $\langle e_j, e_k, e_i \rangle$ of TH-motif M, the corresponding count $C[M]$ is incremented (line 11).

Complexity of THYME: We analyze the time complexity of THYME, and to this end, we consider two critical steps that are the bottlenecks of THYME: (1) updating the projected graph P and (2) searching for three connected temporal hyperedges. Here, we assume all sets and maps are implemented using hash tables, and the projected graph P is a weighted graph where the weight of edge $(e_i, e_j) \in E_P$ is $|\tilde{e}_i \cap \tilde{e}_j|$.

(1) Updating the projected graph P: The bottleneck of updating the projected graph in THYME is finding all neighbors of the target hyperedge. Specifically, as described in Algorithm 8 in Appendix C, given a temporal hyperedge e_i, the set of temporal hyperedges in P that share any number of nodes in e_i is retrieved in THYME. The time complexity of Algorithm 8 is given in Lemma 2.

Lemma 2 (Complexity of Algorithm 8 in THYME) *The time complexity of Algorithm 1 is $O(\sum_{e_j \in N_{e_i}} |\tilde{e}_i \cap \tilde{e}_j|)$.*
Proof. *By the assumptions above, line 4 takes $O(1)$ time, and it is executed $|\tilde{e}_i \cap \tilde{e}_j|$ times for each $e_j \in N_{e_i}$. Thus, the time complexity of Algorithm 8 is $O(\sum_{e_j \in N_{e_i}} |\tilde{e}_i \cap \tilde{e}_j|)$.*

(2) Searching three connected temporal hyperedges: Once the projected graph is updated, THYME searches for new instances of TH-motifs that contain e_i. To this end, as described in Algorithm 10 in Appendix C, the 1-hop and 2-hop neighbors of the target hyperedge in the projected graph are explored. The time complexity of Algorithm 10 is given in Lemma 3.

Lemma 3 (Complexity of Algorithm 10 in THYME) *The time complexity of Algorithm 10 is $O(\sum_{e_j \in N_{e_i}} |N_{e_i} \cup N_{e_j}|)$.*
Proof. *By the assumptions above, given a temporal hyperedge $e_i = (\tilde{e}_i, t_i)$, computing $N_{e_i} \cup N_{e_j}$ for every neighbor $e_j \in N_{e_i}$ takes $O(\sum_{e_j \in N_{e_i}} |N_{e_i} \cup N_{e_j}|)$ time.*

Based on the analyses above, we sum up the time complexity of THYME in Theorem 1.

Theorem 1 (Complexity of THYME) *The time complexity of* THYME *to process a temporal hyperedge e_i is*

$$O\left(\sum_{e_j \in N_{e_i}} \sum_{e_k \in (N_{e_i} \cup N_{e_j})} \min(|\tilde{e}_i|, |\tilde{e}_j|, |\tilde{e}_k|)\right). \quad (1)$$

Proof. *Each temporal hyperedge e_i is processed through three steps: (a) insertion into P, (b) removal from P, and (c) enumeration and counting TH-motifs. The time complexity of each step is summarized as follows.*

(a) The time complexity of inserting the node e_i and its adjacent edges to P is $O(\sum_{e_j \in N_{e_i}} |\tilde{e}_i \cap \tilde{e}_j|)$, as stated in Lemma 2.
(b) The time complexity of removing edges that are adjacent to e_i is $O(|N_{e_i}|)$, i.e., proportional to the number of its neighbors.
(c) The time complexity of enumerating three connected nodes that contain e_i is $O(\sum_{e_j \in N_{e_i}} |N_{e_i} \cup N_{e_j}|)$, as stated in Lemma 3, and for each triple of nodes, it takes $O(\min(|\tilde{e}_i|, |\tilde{e}_j|, |\tilde{e}_k|))$ time to identify its TH-motif. Thus, the time complexity of this step is bounded by Eq. (1).

Algorithm 2: THYME$^+$: Proposed Algorithm for Exact Counting of TH-motifs

```
Input : (1) temporal hypergraph: $T = (V, \mathscr{E})$
        (2) time interval $\delta$
Output: # of each temporal h-motif $M$'s instances: $C[M]$
$C \leftarrow$ map initialized to 0
$Q = (V_Q = \varnothing, E_Q = \varnothing, t_Q = \varnothing)$
$w_s \leftarrow 1$
for each temporal hyperedge $e_i = (\tilde{e}_i, t_i) \in \mathscr{E}$ do
    insert($e_i$)
    while $t_{w_s} + \delta < t_i$ do
        remove($e_{w_s}$)
        $w_s \leftarrow w_s + 1$
    $S \leftarrow$ set of 3 connected static hyperedges including $\tilde{e}_i$
    for each instance $\{\tilde{e}_i, \tilde{e}_j, \tilde{e}_k\} \in S$ do
        comb3($\tilde{e}_i, \tilde{e}_j, \tilde{e}_k$)
    for each pair $(\tilde{e}_i, \tilde{e}_j) \in N_{\tilde{e}_i}$ do
        comb2($\tilde{e}_i, \tilde{e}_j$)
    comb1($\tilde{e}_i$)
return $M$

Procedure insert($e_i = (\tilde{e}_i, t_i)$)
    if $\tilde{e}_i \notin V_Q$ then
        $V_Q \leftarrow V_Q \cup \{\tilde{e}_i\}$
        $N_{\tilde{e}_i} \leftarrow \{\tilde{e} : \tilde{e} \in V_Q \setminus \{\tilde{e}_i\}$ and $\tilde{e}_i \cap \tilde{e} \neq \varnothing\}$
        $E_Q \leftarrow E_Q \cup \{(\tilde{e}_i, \tilde{e}) : \tilde{e} \in N_{\tilde{e}_i}\}$
    $t_Q(\tilde{e}_i) \leftarrow t(\tilde{e}_i) \cup \{t_i\}$
Procedure remove($e_i = (\tilde{e}_i, t_i)$)
    $t(\tilde{e}_i) \leftarrow t(\tilde{e}_i) \setminus \{t_i\}$
    if $t_Q(\tilde{e}_i) = \varnothing$ then
        $V_Q \leftarrow V_Q \setminus \{\tilde{e}_i\}$
        $N_{\tilde{e}_i} \leftarrow \{\tilde{e} : \tilde{e} \in V_Q$ and $e_i \cap e \neq \varnothing\}$
        $E_Q \leftarrow E_Q \setminus \{(\tilde{e}_i, \tilde{e}) : \tilde{e} \in N_{\tilde{e}_i}\}$
Procedure comb3($\tilde{e}_i, \tilde{e}_j, \tilde{e}_k$)
    $C[h(\tilde{e}_j, \tilde{e}_k, \tilde{e}_i)] \leftarrow C[h(\tilde{e}_j, \tilde{e}_k, \tilde{e}_i)] + \sum_{t \in t_Q(\tilde{e}_j), t' \in t_Q(\tilde{e}_k)} \mathbb{1}[t < t']$
    $C[h(\tilde{e}_k, \tilde{e}_j, \tilde{e}_i)] \leftarrow C[h(\tilde{e}_k, \tilde{e}_j, \tilde{e}_i)] + \sum_{t \in t_Q(\tilde{e}_j), t' \in t_Q(\tilde{e}_k)} \mathbb{1}[t' < t]$
Procedure comb2($\tilde{e}_i, \tilde{e}_j$)
    $C[h(\tilde{e}_i, \tilde{e}_j, \tilde{e}_i)] \leftarrow C[h(\tilde{e}_i, \tilde{e}_j, \tilde{e}_i)] + \sum_{t \in t_Q(\tilde{e}_i) \setminus \{t_i\}, t' \in t_Q(\tilde{e}_j)} \mathbb{1}[t < t']$
    $C[h(\tilde{e}_j, \tilde{e}_i, \tilde{e}_i)] \leftarrow C[h(\tilde{e}_j, \tilde{e}_i, \tilde{e}_i)] + \sum_{t \in t_Q(\tilde{e}_i) \setminus \{t_i\}, t' \in t_Q(\tilde{e}_j)} \mathbb{1}[t' < t]$
    $C[h(\tilde{e}_j, \tilde{e}_j, \tilde{e}_i)] \leftarrow C[h(\tilde{e}_j, \tilde{e}_j, \tilde{e}_i)] + \binom{|t_Q(\tilde{e}_j)|}{2}$
Procedure comb1($\tilde{e}_i$)
    $C[h(\tilde{e}_i, \tilde{e}_i, \tilde{e}_i)] \leftarrow C[h(\tilde{e}_i, \tilde{e}_i, \tilde{e}_i)] + \binom{|t_Q(\tilde{e}_i) - \{t_i\}|}{2}$
```

Limitations of THYME: Though THYME avoids the redundant search in the induced static hypergraph G_T, it directly enumerates every instance of TH-motifs in T. Since the size of the temporal hypergraph is much larger than that of induced static hypergraphs, counting the instances in temporal hypergraphs can be more computationally challenging, especially when the time interval δ is large. Each temporal hyperedge within the temporal window corresponds to a unique node in the projected graph P even when many temporal hyperedges are highly duplicated, as in real-world hypergraphs (see Section 6.4).

4.3 THyMe^{+}: Advanced Version of the Proposed Algorithm

We present THyMe^{+}, our proposed algorithm for exactly counting the instances of TH-motifs. THyMe^{+} is faster and more efficient than DP and THyMe, as shown empirically in Section 6, by addressing their limitations as follows.

- DP enumerates all instances of static h-motifs in the induced hypergraph G_T, where most of them are redundant, not induced by any instance of TH-motifs of the temporal hypergraph T. THyMe^{+} selectively enumerates the h-motif instances and thus reduces the redundancy.
- THyMe exhaustively enumerates all instances of TH-motifs. THyMe^{+} reduces the enumeration by introducing an effective counting scheme.
- The projected graph P maintained by THyMe can be large since each temporal hyperedge is represented as a unique node. THyMe^{+} maintains a projected graph Q that is typically smaller than P. In Q, the same node can be shared by multiple temporal hyperedges. The motivation behind Q is empirically demonstrated in Section 6.4.

Projected Graph in THyMe^{+}: THyMe^{+} maintains a projected graph $Q = (V_Q, E_Q, t_Q)$ composed of a set of nodes V_Q, a set of edges E_Q, and a map t_Q. Each node and edge represent a static hyperedge and a pair of static hyperedges that share any nodes, respectively. In addition, t_Q maps a set of timestamps of temporal hyperedges inducing a particular static hyperedge. Notably, while each node in the projected graph P used in THyMe is a unique temporal hyperedge, Q represents the connectivity between hyperedges in the induced static hypergraph G_T. That is, duplicated temporal hyperedges can share the same node in Q, and thus the size of the graph can be much smaller than P, i.e., $|E_Q| < |E_P|$.

The update schemes of Q, `insert` and `remove` in Algorithm 2 add or delete nodes and their adjacent edges, respectively. More specifically, in `insert`, given a new temporal hyperedge $e_i = (\tilde{e}_i, t_i)$, its set of nodes $\tilde{e}_i$ is inserted as a new node, only if there do not exist any temporal hyperedges in the current temporal window whose nodes are $\tilde{e}_i$ (line 3). Once the new node is inserted, their incident edges are created as well (lines 4-5). Finally, the timestamp t_i is added in $t_Q(\tilde{e}_i)$ (line 6). In `remove`, given a temporal hyperedge e_i to be removed, it first deletes its timestamp t_i from $t_Q(\tilde{e}_i)$ (line 2). If e_i is the only temporal hyperedge in the current window whose node set is $\tilde{e}_i$, then $\tilde{e}_i$ and its incident edges are removed from V_Q and E_Q, respectively (lines 4-6).

Counting in THyMe^{+}: The counting procedure of THyMe^{+} is described in Algorithm 2. The sets of nodes and edges of the projected graph Q are initialized to empty maps, i.e., $V_Q = \varnothing$ and $E_Q = \varnothing$ (line 2). For each temporal hyperedge $e_i = (\tilde{e}_i, t_i)$, it moves the temporal window to $[t_i - \delta, t_i]$ as described above. Once Q is updated, THyMe^{+} counts the instances of TH-motifs that contain e_i and any two previous temporal hyperedges. To minimize enumeration, THyMe^{+} adapts effective counting schemes, `comb3`, `comb2`, and `comb1`, which compute the number of instances of triple-inducing, pair-inducing, and single-inducing TH-motifs, respectively, as follows:

- **Triple-inducing TH-motifs (lines 9-11):** THyMe^{+} first enumerates the instances of three connected hyperedges in Q that contains $\tilde{e}_i$ (line 9). For each set $\{\tilde{e}_i, \tilde{e}_j, \tilde{e}_k\}$ of three connected hyperedges, the number of instances of TH-motifs that contain e_i is counted by timestamp combinations using `comb3`. That is, since e_i is the latest temporal hyperedge, the set $\{\tilde{e}_i, \tilde{e}_j, \tilde{e}_k\}$ can be induced by sequences of either $\langle e_x, e_y, e_i \rangle$ or $\langle e_y, e_x, e_i \rangle$ where $\tilde{e}_x = \tilde{e}_j$ and $\tilde{e}_y = \tilde{e}_k$. Since $t_x \in t_Q(\tilde{e}_j)$ and $t_y \in t_Q(\tilde{e}_k)$, the number of such instances can be computed by the number of timestamp combinations of $t_Q(\tilde{e}_j)$ and $t_Q(\tilde{e}_k)$ (lines 2-3).

- **Pair-inducing TH-motifs (lines 12-13):** THYME$^+$ enumerates each edge $(\tilde{e}_i, \tilde{e}_j)$ in Q that is adjacent to $\tilde{e}_i$, which can be induced by three different orders of sequences, $\langle e_x, e_y, e_i \rangle$, $\langle e_y, e_x, e_i \rangle$, and $\langle e_y, e_y, e_i \rangle$ where $\tilde{e}_x = \tilde{e}_i$ and $\tilde{e}_y = \tilde{e}_j$. Since $t_x \in t_Q(\tilde{e}_i) \setminus \{t_i\}$ and $t_y \in t_Q(\tilde{e}_j)$, the number of the sequences can be computed by the number of combinations of the set of these timestamps (lines 2-3).
- **Single-inducing TH-motifs (line 14):** Single-inducing TH-motifs, each of which consists of three duplicated temporal hyperedges, can be immediately counted using `comb1`. That is, a sequence $\langle e_x, e_y, e_i \rangle$ where $\tilde{e}_x = \tilde{e}_i$ and $\tilde{e}_y = \tilde{e}_i$ can be an instance of a single-inducing TH-motif. Since $t_x \in t_Q(\tilde{e}_i) \setminus \{t_i\}$, $t_y \in t_Q(\tilde{e}_i) \setminus \{t_i\}$, and $t_x < t_y$, the number of such instances is computed immediately (line 2).

In Section 6.4, we share empirical observations supporting the intuition behind THYME$^+$.

Complexity of THYME$^+$: We analyze the time complexity of THYME$^+$, and to this end, we consider three steps that are bottlenecks of THYME$^+$: (1) updating the projected graph Q, (2) searching for three connected static hyperedges, and (3) combinatorial counting. We assume all sets and maps are implemented using hash tables and the projected graph Q is a weighted graph where the weight of edge $(\tilde{e}_i, \tilde{e}_j)$ is $|\tilde{e}_i \cap \tilde{e}_j|$.
(1) Updating the projected graph Q: As described in Algorithm 9 in Appendix C, given a temporal hyperedge e_i, the set of induced hyperedges in Q that share any number of nodes in $\tilde{e}_i$ is retrieved in THYME$^+$. The time complexity of Algorithm 9 is given in Lemma 4.

Lemma 4 (Complexity of Algorithm 9 in THYME$^+$) *The time complexity of Algorithm 9 is $O(\sum_{\tilde{e}_j \in N_{\tilde{e}_i}} |\tilde{e}_i \cap \tilde{e}_j|)$.*
Proof. *By the assumptions above, line 4 takes $O(1)$ time, and it is executed $|\tilde{e}_i \cap \tilde{e}_j|$ times for each $\tilde{e}_j \in N_{\tilde{e}_i}$. Thus, the time complexity of Algorithm 9 is $O(\sum_{\tilde{e}_j \in N_{\tilde{e}_i}} |\tilde{e}_i \cap \tilde{e}_j|)$.*

(2) Searching three connected static hyperedges: Once the projected graph is updated, THYME$^+$ searches for new instances of TH-motifs. To this end, THYME$^+$ searches for the set of three connected induced static hyperedges that contain $\tilde{e}_i$. As described in Algorithm 11 in Appendix C, the 1-hop and 2-hop neighbors of the target hyperedge in the projected graph are explored. The time complexity of Algorithm 11 is given in Lemma 5.

Lemma 5 (Complexity of Algorithm 11 in THYME$^+$) *The time complexity of Algorithm 11 is $O(\sum_{\tilde{e}_j \in N_{\tilde{e}_i}} |N_{\tilde{e}_i} \cup N_{\tilde{e}_j}|)$.*
Proof. *By the assumptions above, given a temporal hyperedge $e_i = (\tilde{e}_i, t_i)$, computing $N_{\tilde{e}_i} \cup N_{\tilde{e}_j}$ for every neighbor $\tilde{e}_j \in N_{\tilde{e}_i}$ takes $O(\sum_{\tilde{e}_j \in N_{\tilde{e}_i}} |N_{\tilde{e}_i} \cup N_{\tilde{e}_j}|)$ time.*

(3) Combinatorial counting: In THYME$^+$, we introduce `comb3`, `comb2`, and `comb1` (described in Algorithm 2), which efficiently count the number of triple-inducing, pair-inducing, and single-inducing TH-motifs, respectively, without directly enumerating instances of them. The time complexities of `comb3`, `comb2`, and `comb1` are given in Lemma 6, Lemma 7, and Lemma 8, respectively. We assume that, for each hyperedge $\tilde{e}_i \in V_Q$ in the projected graph Q, $t_Q(\tilde{e}_i)$ is maintained sorted in increasing order, and $|t_Q(\tilde{e}_i)|$ is maintained and thus immediately obtainable.

Lemma 6 (Complexity of `comb3` in THYME$^+$) *The time complexity of `comb3` is $O(|t_Q(\tilde{e}_j)| + |t_Q(\tilde{e}_k)|)$.*
Proof. *Line 2 and line 3 in Algorithm 2 can be computed by sorting $t_Q(\tilde{e}_j) \cup t_Q(\tilde{e}_k)$ in increasing order and summing up the ranks of the elements of $t_Q(\tilde{e}_j)$ and those of $t_Q(\tilde{e}_k)$, respectively. By the assumptions above, $t_Q(\tilde{e}_j)$ and $t_Q(\tilde{e}_k)$ are sorted, and thus sorting $t_Q(\tilde{e}_j) \cup t_Q(\tilde{e}_k)$ takes $O(|t_Q(\tilde{e}_j)| + |t_Q(\tilde{e}_k)|)$ time.*

Lemma 7 (Complexity of `comb2` in THYME$^+$) *The time complexity of* `comb2` *is* $O(|t_Q(\tilde{e}_i)| + |t_Q(\tilde{e}_j)|)$.
Proof. *Line 2 and line 3 in Algorithm 2 can be computed by sorting* $t_Q(\tilde{e}_i) \cup t_Q(\tilde{e}_j)$ *in increasing order and summing up the ranks of the elements of* $t_Q(\tilde{e}_i)$ *and those of* $t_Q(\tilde{e}_j)$*, respectively. By the assumptions above,* $t_Q(\tilde{e}_i)$ *and* $t_Q(\tilde{e}_j)$ *are sorted, and thus sorting* $t_Q(\tilde{e}_i) \cup t_Q(\tilde{e}_j)$ *takes* $O(|t_Q(\tilde{e}_i)| + |t_Q(\tilde{e}_j)|)$ *time. In addition, by the assumptions above, the size of* t_Q *is immediately obtainable, and thus line 4 is computed in constant time. Hence,* `comb2` *takes* $O(|t_Q(\tilde{e}_i)| + |t_Q(\tilde{e}_j)|)$ *time.*

Lemma 8 (Complexity of `comb1` in THYME$^+$) *The time complexity of* `comb1` *is* $O(1)$.
Proof. *By the assumptions above, the size of* t_Q *is immediately obtainable. From* $|t_Q(\tilde{e}_i) - \{t_i\}| = |t_Q(\tilde{e}_i)| - 1$*, we can compute* $\binom{|t_Q(\tilde{e}_i) - \{t_i\}|}{2}$ *in constant time.*

Based on the analyses above, we sum up the time complexity of THYME$^+$ in Theorem 2.

Theorem 2 (Complexity of THYME$^+$) *The time complexity of* THYME$^+$ *to process a temporal hyperedge* e_i *is*

$$O\left(\sum_{\tilde{e}_j \in N_{\tilde{e}_i}} \sum_{\tilde{e}_k \in (N_{\tilde{e}_i} \cup N_{\tilde{e}_j})} \Big(\max(|t_Q(\tilde{e}_i)|, |t_Q(\tilde{e}_j)|, |t_Q(\tilde{e}_k)|) + \min(|\tilde{e}_i|, |\tilde{e}_j|, |\tilde{e}_k|)\Big)\right). \quad (2)$$

Proof. *Each temporal hyperedge* e_i *is processed through three steps: (a) insertion into* Q*, (b) removal from* Q*, and (c) enumeration of the three connected nodes in* Q *that contain* $\tilde{e}_i$*. The time complexity of each step is summarized as follows.*

(a) The time complexity of inserting the node $\tilde{e}_i$ *and its adjacent edges to* Q *is* $O(\sum_{\tilde{e}_j \in N_{\tilde{e}_i}} |\tilde{e}_i \cap \tilde{e}_j|)$*, as stated in Lemma 4.*
(b) The time complexity of removing edges that are adjacent to $\tilde{e}_i$ *is* $O(|N_{\tilde{e}_i}|)$*, i.e., proportional to the number of its neighbors.*
(c) The time complexity of enumerating three connected nodes that contain $\tilde{e}_i$ *is* $O(\sum_{\tilde{e}_j \in N_{\tilde{e}_i}} |N_{\tilde{e}_i} \cup N_{\tilde{e}_j}|)$*. For each instance* $(\tilde{e}_i, \tilde{e}_j, \tilde{e}_k)$*, it takes* $O(\max(|t_Q(\tilde{e}_i)|, |t_Q(\tilde{e}_j)|, |t_Q(\tilde{e}_k)|))$ *time from Lemmas 6, 7, and 8. Additionally, identifying the corresponding TH-motif takes* $O(\min(|\tilde{e}_i|, |\tilde{e}_j|, |\tilde{e}_k|))$ *time. Thus, the time complexity of this step is bounded by Eq.* (2)*.*

5 Approximate Counting Algorithms

In this section, we describe three versions of THYME-A (THYME Approximate), which approximately count the instances of each TH-motif in the input temporal hypergraph. Specifically, all versions repeatedly sample time intervals of length $\triangle$ from the considered time period. Then, they exactly count the number of instances of each TH-motif whose closure time is within the sampled time interval. Lastly, based on the counts, they estimate the counts over the entire time interval. By partially exploring the given temporal hypergraph, all versions rapidly yield unbiased estimates of the number of TH-motifs.

Remarks: The problem of sampling temporal hypergraphs has additional challenges compared to the exact counting problem. A straightforward approach is to partition the considered period of time into intervals and count the number of instances of TH-motifs that occurred within each interval, as in sampling temporal network motifs in pairwise graphs (Liu et al., 2019). This approach, however, requires careful consideration of the instances that cross interval boundaries, which can degrade the scalability of the method, as described below. Instead, in this work, we sample time intervals and use THYME^{+} to exactly count the number of instances whose closure time is within the interval, which effectively addresses the above challenges.

5.1 DISCRETE TIME PARTITIONING (DTP): *Extension of Liu et al. (2019)*

We present DISCRETE TIME PARTITIONING (DTP), which is a baseline approach for approximately counting the instances of each TH-motif in the input temporal hypergraph.

Counting in DTP: Given an input temporal hypergraph $T = (V, \mathscr{E})$, DTP partitions the period of considered time (i.e., from $t_{\min}$ to $t_{\max}$) into discrete time intervals. For each interval, whose length is $c\delta$ ($c > 0$ is an integer), DTP exactly counts the instances that occurred within the interval. Since it may omit a large number of instances that cross interval boundaries, counts are incremented by the inverse of the probability that the instance is completely contained in the interval. To this end, the duration of every enumerated instance should be traced. Furthermore, DTP incorporates an importance sampling to enhance the estimator by probabilistically sampling intervals and thus reduce the variance. That is, DTP selects an interval with probability proportional to the number of temporal hyperedges arriving within the interval. Refer to Appendix B for details.

Limitations of DTP: While DTP yields unbiased estimation by carefully considering the duration of TH-motifs' instances, it has limited scalability due to its instance-level enumeration. That is, DTP traces the duration of every instance that occurs within the sampled time intervals, and this can be inefficient in temporal hypergraphs whose size is typically much larger than that of induced static hypergraphs. Even worse, this limits the utilization of our most advanced exact counting method, THYME^{+}, which avoids direct enumeration of the instances but counts the number of instances of TH-motifs based on combinations of a set of timestamps.

5.2 THYME-A: Preliminary Version of Approximate Counting Algorithm

We present THYME-A, a preliminary version of our approximate counting algorithm. Instead of partitioning the period of time into discrete time intervals, THYME-A samples $\triangle$-sized time intervals and exactly counts the number of each TH-motif's instances whose closure times are within the intervals. It yields unbiased estimation based on our fastest and most scalable exact counting method, THYME^{+}.

Counting in THYME-A: The counting procedure of THYME-A is described in Algorithm 3. It samples a $\triangle$-sized time interval by selecting an initial timestamp t_j of the interval from an ordered set $\mathscr{T}_{\text{naive}} = \{t_{\min} - \triangle + 1, \cdots, t_{\max}\}$ uniformly at random (line 3). Then using THYME^{+}, it counts the instances of each TH-motif whose closure time is within the sampled time interval $\{t_j, \cdots, t_j + \triangle - 1\}$ (line 4), and these are added to the total counts $\widetilde{C}$ (line 6). Specifically, THYME^{+} scans temporal hyperedges that arrived within $\{t_j - \delta + 1, \cdots, t_j + \triangle - 1\}$ and selectively counts the number of instances of TH-motifs whose closure time is within $\{t_j, \cdots, t_j + \triangle - 1\}$. These procedures are repeated s times to reduce the variance of the estimation. Lastly, each estimate $\widetilde{C}[M]$ of TH-motif M is rescaled by multiplying $\frac{|\mathscr{T}_{\text{naive}}|}{s\triangle}$ (line 8). This rescaling step ensures that each estimate $\widetilde{C}[M]$ is unbiased, as formalized in Theorem 3.

Algorithm 3: THYME-A: Preliminary Version of the Proposed Algorithm for Approximate Counting of TH-motifs

```
Input  : (1) temporal hypergraph: $T = (V, \mathscr{E})$
         (2) time interval $\delta$
         (3) number of samples $s$
         (4) sampling time interval $\triangle$
         (5) sampling space $\mathscr{T}_{\text{naive}}$
Output: approximated # of each temporal h-motif $M$'s instances: $\widetilde{C}[M]$
1 $\widetilde{C}$ ← map initialized to zero
2 for each sample $n = 1, \cdots, s$ do
3     $t_j$ ← sample a uniformly random timestamp from $\mathscr{T}_{\text{naive}}$
4     $C_j$ ← exact counts of instances of TH-motifs whose closure time is within $\{t_j, \cdots, t_j + \triangle - 1\}$
        using THYME+
5     for each TH-motif $M$ do
6         $\widetilde{C}[M] \leftarrow \widetilde{C}[M] + C_j[M]$
7 for each TH-motif $M$ do
8     $\widetilde{C}[M] \leftarrow \widetilde{C}[M] \cdot \frac{|\mathscr{T}_{\text{naive}}|}{s\triangle}$
9 return $\widetilde{C}$
```

Theorem 3 (Bias and variance of THYME-A) *For every TH-motif M,* THYME-A *provides an unbiased estimate $\widetilde{C}[M]$ of the count $C[M]$ of its instances, i.e.,*

$$\mathbb{E}[\widetilde{C}[M]] = C[M]. \tag{3}$$

The variance of the estimate is

$$\mathbb{V}ar[\widetilde{C}[M]] = \frac{1}{s\triangle^2}\left\{(|\mathscr{T}_{naive}| - 1)\sum_{t_j \in \mathscr{T}_{naive}} X_j[M]^2 - \sum_{j \neq j'} X_j[M]X_{j'}[M]\right\} \tag{4}$$

where $X_j[M]$ is the number of instances of TH-motif M whose closure time is within $\{t_j, \cdots, t_j + \triangle - 1\}$.

Proof. Let W_{ij} be a random variable indicating whether the i^{th} ($i \in \{1, \cdots, s\}$) sampled time interval is the j^{th} time interval (out of $|\mathscr{T}_{\text{naive}}|$ possible samples). That is, $W_{ij} = 1$ if the j^{th} time interval is sampled for the i^{th} sample, and $W_{ij} = 0$ otherwise. Let $\tilde{\mathrm{c}}[M]$ be the number of times that TH-motif M's instances are counted while processing s sampled time intervals. That is,

$$\tilde{\mathrm{c}}[M] := \sum_{i=1}^{s} \sum_{t_j \in \mathscr{T}_{\text{naive}}} W_{ij} X_j[M].$$

Then, by the scaling scheme,

$$\widetilde{C}[M] = \tilde{\mathrm{c}}[M] \cdot \frac{|\mathscr{T}_{\text{naive}}|}{s\triangle}.$$

Proof of the Bias of $\widetilde{C}[M]$ (Eq. (3)): Since any time interval is sampled from $\mathscr{T}_{\text{naive}}$ uniformly at random,

$$P[W_{ij} = 1] = \mathbb{E}[W_{ij}] = \frac{1}{|\mathscr{T}_{\text{naive}}|}.$$

In addition, since each instance of TH-motif M is included in $\triangle$ different time intervals,

$$\sum_{t_j \in \mathscr{T}_{\text{naive}}} X_j[M] = \triangle C[M].$$

From linearity of expectation,

$$\mathbb{E}[\widetilde{\mathrm{c}}[M]] = \sum_{i=1}^{s} \sum_{t_j \in \mathcal{T}_{\text{naive}}} \mathbb{E}\left[W_{ij} X_j[M]\right] = \sum_{i=1}^{s} \sum_{t_j \in \mathcal{T}_{\text{naive}}} X_j[M] \cdot \mathbb{E}[W_{ij}] = \sum_{i=1}^{s} \sum_{t_j \in \mathcal{T}_{\text{naive}}} \frac{X_j[M]}{|\mathcal{T}_{\text{naive}}|} = s \triangle C[M].$$

Then, by the scaling term,

$$\mathbb{E}[\widetilde{C}[M]] = \frac{|\mathcal{T}_{\text{naive}}|}{s\triangle} \mathbb{E}\left[\widetilde{\mathrm{c}}[M]\right] = C[M].$$

Proof of the Variance of $\widetilde{C}[M]$ (Eq. (4)): Since $W_{ij} = W_{ij}^2$, the variance of W_{ij} is

$$\mathbb{V}ar[W_{ij}] = \mathbb{E}[W_{ij}^2] - \mathbb{E}[W_{ij}]^2 = \frac{1}{|\mathcal{T}_{\text{naive}}|} - \frac{1}{|\mathcal{T}_{\text{naive}}|^2}.$$

Consider the covariance between W_{ij} and $W_{i'j'}$. If $i = i'$, then

$$\mathbb{C}ov(W_{ij}, W_{i'j'}) = \mathbb{E}[W_{ij}, W_{ij'}] - \mathbb{E}[W_{ij}] \cdot \mathbb{E}[W_{ij'}] = -\frac{1}{|\mathcal{T}_{\text{naive}}|^2}.$$

Since time intervals are sampled independently, if $i \neq i'$, then $\mathbb{C}ov(W_{ij}, W_{i'j'}) = 0$.
These imply

$$\begin{aligned}
\mathbb{V}ar[\widetilde{\mathrm{c}}[M]] &= \mathbb{V}ar\left[\sum_{i=1}^{s} \sum_{t_j \in \mathcal{T}_{\text{naive}}} W_{ij} X_j[M]\right] \\
&= \sum_{i=1}^{s} \sum_{t_j \in \mathcal{T}_{\text{naive}}} \mathbb{V}ar\left[W_{ij} X_j[M]\right] + \sum_{i=1}^{s} \sum_{j \neq j'} \mathbb{C}ov\left(W_{ij} X_j[M], W_{ij'} X_{j'}[M]\right) \\
&= \sum_{i=1}^{s} \sum_{t_j \in \mathcal{T}_{\text{naive}}} X_j[M]^2 \mathbb{V}ar[W_{ij}] + \sum_{i=1}^{s} \sum_{j \neq j'} X_j[M] X_{j'}[M] \mathbb{C}ov(W_{ij}, W_{ij'}) \\
&= \frac{s(|\mathcal{T}_{\text{naive}}| - 1)}{|\mathcal{T}_{\text{naive}}|^2} \sum_{t_j \in \mathcal{T}_{\text{naive}}} X_j[M]^2 - \frac{s}{|\mathcal{T}_{\text{naive}}|^2} \sum_{j \neq j'} X_j[M] X_{j'}[M].
\end{aligned}$$

Thus, from $\widetilde{C}[M] = \widetilde{\mathrm{c}}[M] \cdot \frac{|\mathcal{T}_{\text{naive}}|}{s\triangle}$,

$$\mathbb{V}ar[\widetilde{C}[M]] = \frac{|\mathcal{T}_{\text{naive}}|^2}{s^2 \triangle^2} \mathbb{V}ar[\widetilde{\mathrm{c}}[M]] = \frac{1}{s\triangle^2} \left\{ (|\mathcal{T}_{\text{naive}}| - 1) \sum_{t_j \in \mathcal{T}_{\text{naive}}} X_j[M]^2 - \sum_{j \neq j'} X_j[M] X_{j'}[M] \right\}.$$

Limitations of THyMe-A: Despite the simplicity of THyMe-A, it samples time intervals from $\mathcal{T}_{\text{naive}} = \{t_{\min} - \triangle + 1, \cdots, t_{\max}\}$, which covers the entire period of time. This can be particularly inefficient in temporal hypergraphs whose arrival times of temporal hyperedges are extremely sparse, i.e., there are many time units when none of the temporal hyperedges arrive. In such cases, THyMe-A rarely counts any instances of TH-motifs even if it samples a large number of intervals and thus gives inaccurate estimation.

Algorithm 4: THYME-A$^+$: Advanced Version of the Proposed Algorithm for Approximate Counting of TH-motifs

```
Input  : (1) temporal hypergraph: $T = (V, \mathscr{E})$
         (2) time interval $\delta$
         (3) number of samples $s$
         (4) sampling time interval $\triangle$
         (5) sampling space $\mathscr{T}_{\text{adv}}$
Output: approximated # of each temporal h-motif $M$'s instances: $\widehat{C}$[M]
1 $\widehat{C} \leftarrow$ map initialized to zero
2 for each sample $n = 1, \cdots, s$ do
3     $t_j \leftarrow$ sample a uniformly random timestamp from $\mathscr{T}_{\text{adv}}$
4     $C_j \leftarrow$ exact counts of instances of TH-motifs whose closure time is within $\{t_j, \cdots, t_j + \triangle - 1\}$
        using THYME$^+$
5     for each TH-motif $M$ do
6         $\widehat{C}[M] \leftarrow \widehat{C}[M] + C_j[M]$
7 for each TH-motif $M$ do
8     $\widehat{C}[M] \leftarrow \widehat{C}[M] \cdot \frac{|\mathscr{T}_{\text{adv}}|}{s\triangle}$
9 return $\widehat{C}$
```

5.3 THYME-A$^+$: Advanced Version of Approximated Counting Algorithm

In order to reduce the variance of THYME-A, we propose THYME-A$^+$, which yields more accurate estimates of the number of instances of TH-motifs by reducing the sampling space of the time intervals. Intuitively speaking, sampling informative intervals and knowing more instances are helpful to accurately estimate the count. To this end, THYME-A$^+$ samples $\triangle$-sized time interval from $\mathscr{T}_{\text{adv}} = \bigcup_{e=(\tilde{e},t)} \{t - \triangle + 1, \cdots, t\}$, which is the union of the *valid* intervals of each timestamp attached to hyperedges, and in this way, we can ensure that at least one temporal hyperedge is associated with the sampled interval.

Counting in THYME-A$^+$: The counting procedure of THYME-A$^+$ is described in Algorithm 4. The only difference from THYME-A is the sampling space of the intervals; it samples time intervals from $\mathscr{T}_{\text{adv}}$, which is empirically much smaller than $\mathscr{T}_{\text{naive}}$ in real-world temporal hypergraphs. Once the exact counts of TH-motifs during the s sampled intervals are obtained, they are rescaled by multiplying $\frac{|\mathscr{T}_{\text{adv}}|}{s\triangle}$ (line 8), which makes each estimate $\widehat{C}[M]$ of TH-motif M unbiased, as formalized in Theorem 4.

Theorem 4 (Bias and variance of THYME-A$^+$) *For every TH-motif M,* THYME-A *provides an unbiased estimate $\widehat{C}[M]$ of the count $C[M]$ of its instances, i.e.,*

$$\mathbb{E}[\widehat{C}[M]] = C[M]. \tag{5}$$

The variance of the estimate is

$$\mathbb{V}ar[\widehat{C}[M]] = \frac{1}{s\triangle^2}\left\{(|\mathscr{T}_{adv}| - 1)\sum_{t_j \in \mathscr{T}_{adv}} Y_j[M]^2 - \sum_{j \neq j'} Y_j[M]Y_{j'}[M]\right\} \tag{6}$$

where $Y_j[M]$ is the number of instances of TH-motif M whose closure time is within $\{t_j, \cdots, t_j + \triangle - 1\}$

Proof. We can prove Theorem 4 by substituting $\mathscr{T}_{\text{naive}}$, $X_j[M]$, and $\widetilde{C}$ in the proof of Theorem 3 with $\mathscr{T}_{\text{adv}}$, $Y_j[M]$, and $\widehat{C}$, respectively.

<u>Comparison with THYME-A:</u> Empirically, THYME-A$^+$ provides a better trade-off between speed and accuracy than THYME-A, as presented in Section 6. This is theoretically supported by comparing their variances. First, since the sampling space $\mathscr{T}_{\text{adv}}$ of THYME-A$^+$ is a subset of $\mathscr{T}_{\text{naive}}$ of THYME-A (specifically it consists of only *valid* time intervals),

$$\sum_{j\in\mathscr{T}_{\text{naive}}} X_j[M]^2 = \sum_{j\in\mathscr{T}_{\text{adv}}} Y_j[M]^2 \quad \text{and} \quad \sum_{j\neq j'} X_j[M]X_{j'}[M] = \sum_{j\neq j'} Y_j[M]Y_{j'}[M]$$

hold. The difference between the variance of THYME-A (Eq.(4)) and that of THYME-A$^+$ (Eq.(6)) is in the scalar terms $(|\mathscr{T}_{\text{naive}}|-1)$ and $(|\mathscr{T}_{adv}|-1)$. Since $|\mathscr{T}_{\text{naive}}|-1 \geq |\mathscr{T}_{\text{adv}}|-1$, the variance of THYME-A$^+$ is smaller than or equal to that of THYME-A, i.e., $\mathbb{V}ar[\widehat{C}[M]] \leq \mathbb{V}ar[\widetilde{C}[M]]$. In Section 6.5, we empirically confirm our theoretical analyses.

5.4 THYME-A*: Even Better Version of THYME-A$^+$

In this subsection, we propose THYME-A* (described in Algorithm 5), which is even better than our advanced sampling algorithm, THYME-A$^+$. THYME-A* incorporates weighted sampling into THYME-A$^+$ to reduce the variance of the unbiased estimator. That is, each time interval is sampled with probability proportional to the number of temporal hyperedges arriving within the interval.

<u>Counting in THYME-A*:</u> While THYME-A$^+$ samples an initial timestamp t_j of the interval $\{t_j,\cdots,t_j+\triangle-1\}$ from $\mathscr{T}_{\text{adv}}$ uniformly at random, THYME-A* makes use of the temporal distribution of the input temporal hypergraph by sampling t_j with the probability q_j, which is proportional to the number n_j of temporal hyperedges that arrived within the interval. That is, THYME-A* first samples a temporal hyperedge $e=(\tilde{e},t)\in\mathscr{E}$ uniformly at random (line 3). This is followed by sampling an initial timestamp t_j of the interval from $\{t-\triangle+1,\cdots,t\}$ uniformly at random (line 4). These processes sample t_j with a bias, specifically with the probability proportional to n_j, where n_j is additionally obtainable from THYME$^+$ without affecting its time complexity. Specifically, the probability q_j of the timestamp t_j being sampled is $\frac{n_j}{\triangle|\mathscr{E}|}$, as shown in Lemma 9.

Lemma 9 (Sampling probability of time interval in THYME-A*) *The sampling probability q_j of time interval $\{t_j,\cdots t_j+\triangle-1\}$ in* THYME-A* *(lines 3 and 4 in Algorithm 5) is $\frac{n_j}{\triangle|\mathscr{E}|}$.*

Proof. *In line 3, a temporal hyperedge is sampled uniformly at random, and thus the probability of any temporal hyperedge whose arrival time is within $\{t_j,\cdots,t_j+\triangle-1\}$ being sampled is $\frac{n_j}{|\mathscr{E}|}$. Once such a temporal hyperedge $e=(\tilde{e},t)$ is sampled, in line 4, t_j is sampled from $\{t-\triangle+1,\cdots,t\}$ uniformly at random, and thus the probability of t_j being sampled is $\frac{1}{\triangle}$. These imply $q_j=\frac{n_j}{|\mathscr{E}|}\cdot\frac{1}{\triangle}=\frac{n_j}{\triangle|\mathscr{E}|}$.*

Since t_j is sampled with probability q_j, the number $\overline{C}[M]$ of instances counted in the sampled interval is rescaled by multiplying it with the inverse of q_j (line 8). In addition, the total number of instances $\overline{C}[M]$ of TH-motif M is rescaled once more by multiplying $\frac{1}{s\triangle}$ (line 10). These make each estimate $\overline{C}[M]$ of TH-motif M unbiased, as formalized in Theorem 5.

Algorithm 5: THYME-A*: Even Better Version of the Proposed Algorithm for Approximate Counting of TH-motifs

```
Input  : (1) temporal hypergraph: $T = (V, \mathscr{E})$
         (2) time interval $\delta$
         (3) number of samples $s$
         (4) sampling time interval $\triangle$
Output: approximated # of each temporal h-motif $M$'s instances: $\overline{C}[M]$
1  $\overline{C} \leftarrow$ map initialized to zero
2  for each sample $n = 1, \cdots, s$ do
3      $e = (\tilde{e}, t) \leftarrow$ sample a uniformly random temporal hyperedge from $\mathscr{E}$
4      $t_j \leftarrow$ sample a uniformly random timestamp from $\{t - \triangle + 1, \cdots t\}$
5      $C_j \leftarrow$ exact counts of instances of TH-motifs whose closure time is within $\{t_j, \cdots, t_j + \triangle - 1\}$
         using THYME$^+$
6      $q_j \leftarrow \frac{n_j}{\triangle|\mathscr{E}|}$ where $n_j$ is the number of temporal hyperedges arriving within $\{t_j, \cdots, t_j + \triangle - 1\}$
7      for each TH-motif $M$ do
8          $\overline{C}[M] \leftarrow \overline{C}[M] + C_j[M] \cdot \frac{1}{q_j}$
9  for each TH-motif $M$ do
10     $\overline{C}[M] \leftarrow \overline{C}[M] \cdot \frac{1}{s\triangle}$
11 return $\overline{C}$
```

Theorem 5 (Bias and variance of THYME-A*) *For every TH-motif M,* THYME-A* *provides an unbiased estimate $\overline{C}[M]$ of the count $C[M]$ of its instances, i.e.,*

$$\mathbb{E}[\overline{C}[M]] = C[M]. \tag{7}$$

The variance of the estimate is

$$\mathbb{V}ar[\overline{C}[M]] = \frac{1}{s\triangle^2}\left\{ \sum_{t_j \in \mathscr{T}_{adv}} \frac{Y_j[M]^2(1-q_j)}{q_j} - \sum_{j \neq j'} Y_j[M]Y_{j'}[M]. \right\} \tag{8}$$

where $Y_j[M]$ is the number of instances of TH-motif M whose closure time is within $\{t_j, \cdots, t_j + \triangle - 1\}$, and q_j is the probability that t_j is sampled.

Proof. Let Q_{ij} be a random variable indicating whether the i^{th} sampled time interval is the j^{th} time interval (out of $|\mathscr{T}_{\text{adv}}|$ possible samples). That is, $Q_{ij} = 1$ if the j^{th} time interval is sampled for the i^{th} sample, and $Q_{ij} = 0$ otherwise. Let $\bar{\text{c}}[M]$ be the number of weighted counts of TH-motif M's instances while processing s sampled time intervals. That is,

$$\bar{\text{c}}[M] = \sum_{i=1}^{s} \sum_{t_j \in \mathscr{T}_{\text{adv}}} Q_{ij} \frac{Y_j[M]}{q_j}.$$

Then, by the scaling scheme,

$$\overline{C}[M] = \bar{\text{c}}[M] \cdot \frac{1}{s\triangle}.$$

<u>**Proof of the Bias of $\overline{C}[M]$ (Eq. (7)):**</u> From Lemma 9, since the j^{th} time interval is sampled with probability q_j,

$$P[Q_{ij} = 1] = \mathbb{E}[Q_{ij}] = q_j.$$

In addition, since each instance of TH-motif M is included in $\triangle$ time intervals,

$$\sum_{t_j \in \mathscr{T}_{\text{adv}}} Y_j[M] = \triangle C[M].$$

From linearity of expectation,

$$\mathbb{E}[\bar{c}[M]] = \sum_{i=1}^{s}\sum_{t_j\in\mathscr{T}_{\text{adv}}} \mathbb{E}\left[Q_{ij}\frac{Y_j[M]}{q_j}\right] = \sum_{i=1}^{s}\sum_{t_j\in\mathscr{T}_{\text{adv}}} \frac{Y_j[M]}{q_j}\mathbb{E}[Q_{ij}] = \sum_{i=1}^{s}\sum_{t_j\in\mathscr{T}_{\text{adv}}} Y_j[M] = s\triangle C[M].$$

Then, by the rescaling term,

$$\mathbb{E}[\overline{C}[M]] = \frac{1}{s\triangle}\mathbb{E}[\bar{c}[M]] = C[M].$$

Proof of the Variance of $\overline{C}[M]$ (Eq. (8)): Since $Q_{ij} = Q_{ij}^2$, the variance of Q_{ij} is

$$\mathbb{V}ar[Q_{ij}] = \mathbb{E}[Q_{ij}^2] - \mathbb{E}[Q_{ij}]^2 = q_j - q_j^2.$$

Consider the covariance between Q_{ij} and $Q_{i'j'}$. If $i = i'$, then

$$\mathbb{C}ov(Q_{ij}, Q_{i'j'}) = \mathbb{E}[Q_{ij}, Q_{ij'}] - \mathbb{E}[Q_{ij}]\cdot\mathbb{E}[Q_{ij'}] = -q_j q_{j'}.$$

Since time intervals are sampled independently, if $i \neq i'$, then $\mathbb{C}ov(Q_{ij}, Q_{i'j'}) = 0$.
These imply

$$\begin{aligned}
\mathbb{V}ar[\bar{c}[M]] &= \mathbb{V}ar\left[\sum_{i=1}^{s}\sum_{t_j\in\mathscr{T}_{\text{adv}}} Q_{ij}\frac{Y_j[M]}{q_j}\right] \\
&= \sum_{i=1}^{s}\sum_{t_j\in\mathscr{T}_{\text{adv}}} \mathbb{V}ar\left[Q_{ij}\frac{Y_j[M]}{q_j}\right] + \sum_{i=1}^{s}\sum_{j\neq j'} \mathbb{C}ov\left(Q_{ij}\frac{Y_j[M]}{q_j}, Q_{ij'}\frac{Y_{j'}[M]}{q_{j'}}\right) \\
&= \sum_{i=1}^{s}\sum_{t_j\in\mathscr{T}_{\text{adv}}} \frac{Y_j[M]^2}{q_j^2}\mathbb{V}ar[Q_{ij}] + \sum_{i=1}^{s}\sum_{j\neq j'} \frac{Y_j[M]Y_{j'}[M]}{q_j q_{j'}}\mathbb{C}ov(Q_{ij}, Q_{ij'}) \\
&= s\left\{\sum_{t_j\in\mathscr{T}_{\text{adv}}} \frac{Y_j[M]^2(1-q_j)}{q_j} - \sum_{j\neq j'} Y_j[M]Y_{j'}[M]\right\}.
\end{aligned}$$

Thus, from the scaling term $\overline{C}[M] = \frac{1}{s\triangle}\bar{c}[M]$,

$$\mathbb{V}ar[\overline{C}[M]] = \frac{1}{s^2\triangle^2}\mathbb{V}ar[\bar{c}[M]] = \frac{1}{s\triangle^2}\left\{\sum_{t_j\in\mathscr{T}_{\text{adv}}} \frac{Y_j[M]^2(1-q_j)}{q_j} - \sum_{j\neq j'} Y_j[M]Y_{j'}[M]\right\}.$$

Comparison with THyMe-A$^+$: The variance of THyMe-A* (Eq. (8)) is a generalization of that of THyMe-A$^+$ (Eq. (6)), and if the sampling probability of each interval is set to $q_j = \frac{1}{|\mathscr{T}_{\text{adv}}|}$ consistently for $j = 1, \cdots, |\mathscr{T}_{\text{adv}}|$ (i.e., sampling uniformly at random), then the variances of THyMe-A$^+$ and THyMe-A* become the same. However, different from THyMe-A$^+$, THyMe-A* prioritizes intervals where many temporal hyperedges are closed. This is expected to reduce the variance by multiplying smaller weights $\frac{1-q_j}{q_j}$ to intervals with many instances (i.e., intervals with large $Y_j[M]$) and larger weights to those with fewer instances. One might hypothesize that directly utilizing $\frac{1}{Y_j[M]}$ as the weight can be better. However, $Y_j[M]$ is not known in advance, and computing $Y_j[M]$ for every j is computationally expensive. Note that q_j is obtained during the process of THyMe-A* without incurring additional computational cost, and THyMe-A* even does not require q_j in advance for sampling intervals. Moreover, as shown in Fig. 4, $\frac{1-q_j}{q_j}$ tends to be negatively

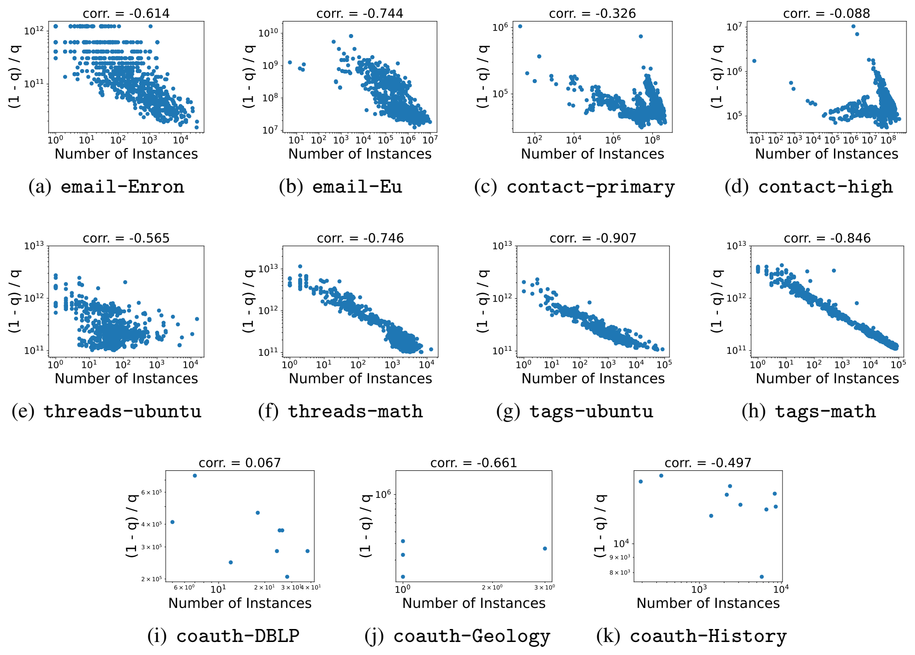


Fig. 4: The weighted sampling scheme in THYME-A* is effective. The number of instances in each interval $[t_j, t_j + \triangle)$ and the term $\frac{1-q_j}{q_j}$ tend to be negatively correlated, which theoretically reduces the variance of the estimate of THYME-A*. We set δ to 96 hours, 8 hours, 4 hours, 4 hours, and 3 years for the datasets from the `email`, `contact`, `threads`, `tags`, and `coauthorship` domains, respectively.

correlated with $Y_j[M]$ (i.e., positively correlated with $\frac{1}{Y_j[M]}$), and this supports the claim that using q_j for weighted sampling can reduce the variance of THYME-A*. In Section 6.5, we demonstrate that empirical results from real-world temporal hypergraphs meet our theoretical analyses.

6 Empirical Studies

In this section, we review experiments to answer Q1-Q4.

Q1. **Discoveries:** Which findings do TH-motifs bring?
Q2. **Comparison with Static H-motifs:** Are TH-motifs more informative than static hypergraph motifs (Lee et al., 2020)?
Q3. **Performance of Exact Algorithms:** How fast and efficient is THYME$^+$? Why is THYME$^+$ fast and efficient?
Q4. **Performance of Approximate Algorithms:** How fast and accurate are THYME-A, THYME-A$^+$, and THYME-A*? Do they perform better than the baseline approach?

We first describe the settings where the experiments are conducted. Then, we provide some empirical observations using the proposed concepts and algorithms. Next, we test the

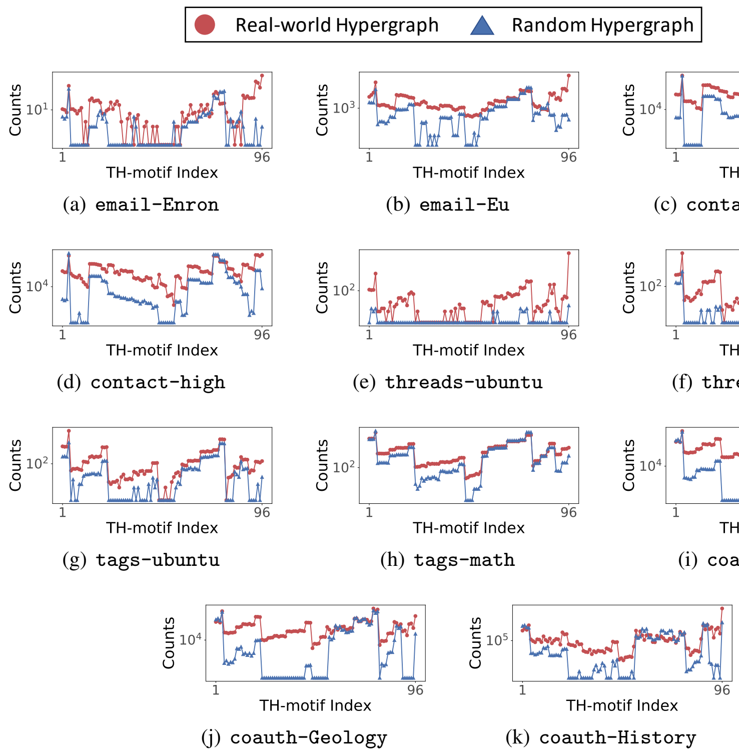


Fig. 5: The distribution of the numbers of TH-motifs' instances in real-world temporal hypergraphs and that in randomized temporal hypergraphs are significantly different. We set δ to 1 hour in all datasets from the `email`, `contact`, `threads`, and `tags` domains. For all datasets from the `coauth` domain, we set δ to 2 years.

scalability of the methods. Finally, we provide possible reasons why THYME^{+} is efficient based on the observations on real-world temporal hypergraphs.

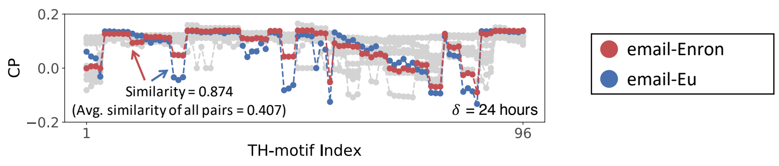
6.1 Experimental Settings

Machines: We conducted all the experiments on a machine with i9-10900K CPU and 64GB RAM.

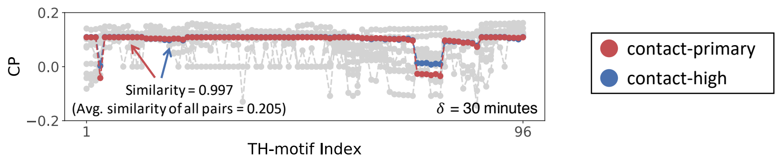
Implementation: We implemented DP, THYME, and THYME^{+} commonly in C++.

Datasets: We used eleven real-world temporal hypergraphs from five different domains. Refer to Table 2 for the summarized statistics of the hypergraphs. The details of each dataset are as follows:

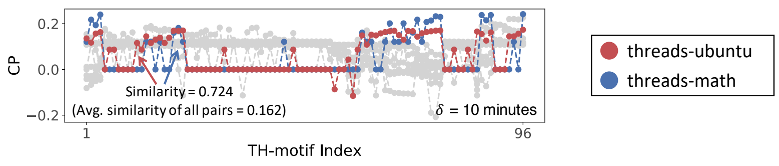
- **email**: Each node is an email account, and each hyperedge is a set of sender and receivers of the email.
- **contact**: Each node is a person, and each hyperedge is a group interaction among people.
- **threads**: Each node is a user, and each hyperedge is a group of users working in a thread.



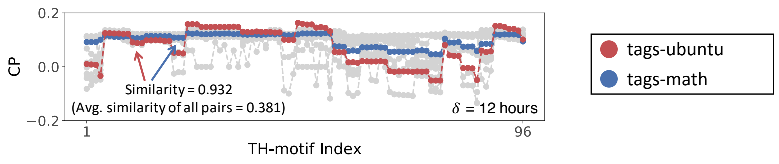
(a) `email` domain



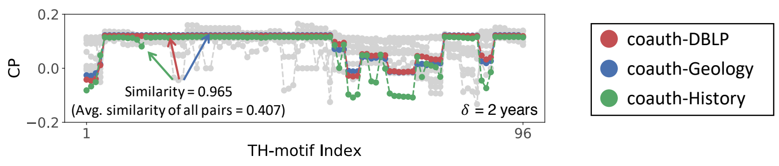
(b) `contact` domain



(c) `threads` domain



(d) `tags` domain



(e) `coauthorship` domain

Fig. 6: Characteristic profiles (CPs) (i.e., normalized significance of each TH-motif) accurately capture the patterns of real-world temporal hypergraphs. The CPs of the temporal hypergraphs from the same domain are similar in terms of the Pearson correlation coefficients, which are the reported numbers, while they are different across domains. Grey lines indicate CPs of the temporal hypergraphs from other domains.

– **tags**: Each node is a tag, and each hyperedge is a set of tags attached to the question.
– **coauthorship**: Each node is an author, and each hyperedge is a set of authors of the publication.

Table 2: Statistics of the 11 real-world hypergraphs from 5 different domains: the number of nodes $|V|$, the number of temporal hyperedges $|\mathscr{E}|$, the number of induced static hyperedges $|E_{\mathscr{E}}|$, and the maximum hyperedge size $\max_{e \in \mathscr{E}} |e|$.

Dataset	$\|V\|$	$\|\mathscr{E}\|$	$\|E_{\mathscr{E}}\|$	$\max_{e\in\mathscr{E}} \|e\|$
`email-Enron`	143	10,885	1,514	37
`email-Eu`	986	235,263	25,148	40
`contact-primary`	242	106,879	12,704	5
`contact-high`	327	172,035	7,818	5
`threads-ubuntu`	90,054	192,947	166,999	14
`threads-math`	153,806	719,792	595,749	21
`tags-ubuntu`	3,021	271,233	147,222	5
`tags-math`	1,627	822,059	170,476	5
`coauth-DBLP`	1,836,596	3,700,681	2,467,389	280
`coauth-Geology`	1,091,979	1,591,166	1,204,704	284
`coauth-History`	503,868	1,813,147	896,062	925

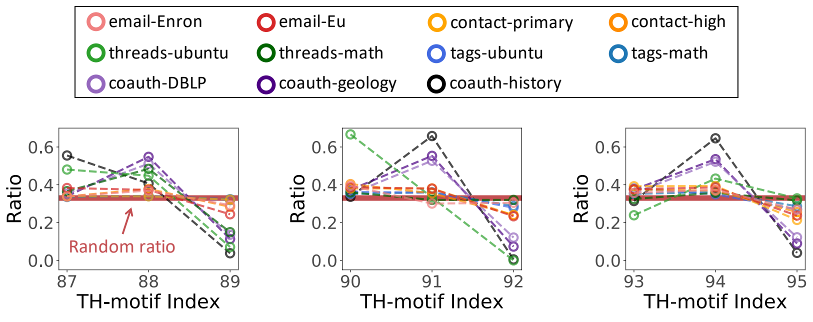
While we assume that timestamps of temporal hyperedges are unique, in some dataset, this may not hold. In such cases, we randomly order the temporal hyperedges whose timestamps are identical.

6.2 Q1. Discoveries

In this subsection, we present several observations that TH-motifs reveal in the 11 real-world hypergraphs. TH-motifs provide a new perspective in analyzing temporal hypergraphs.

Obs 1. Real hypergraphs are not 'random': For an accurate characterization, we compare the number of instances of TH-motifs in real-world temporal hypergraphs against that in randomized ones. To this end, we randomize the real-world temporal hypergraph using HyperCL (Lee et al., 2021), a random hypergraph generator that preserves node degrees and hyperedge sizes. Once the randomized hypergraph is generated, we randomly assign the timestamps of its temporal hyperedges. In Fig. 5, we compare the distribution of the number of instances of each TH-motif in real-world temporal hypergraphs and those in randomized ones. The distributions are clearly different, and the total number of instances is greater in real-world hypergraphs than in random hypergraphs. Specifically, the total number of TH-motifs' instances in real-world hypergraphs are 6.42×, 1.44×, 46.69×, 4.30× of that in randomized hypergraphs in the `email-Eu`, `contact-primary`, `threads-math`, and `tags-ubuntu` datasets, respectively.

Obs 2. TH-motifs distinguish domains: Network motifs have demonstrated their power to distinguish graphs based on their domains. In addition, the count distributions of h-motifs in static hypergraphs are particularly similar within domains but different across domains. To confirm that TH-motifs also possess such distinguishing power, we obtain the characteristic profile (CP) of each hypergraph, a normalized 96-dimensional vector that concatenates the relative significance of each TH-motif, as suggested in (Lee et al., 2020). As seen in Fig. 6, CPs accurately capture patterns of real-world temporal hypergraphs. That is, while CPs of the temporal hypergraphs from the same domain are similar, they are different across



(a) TH-motifs 87-89 (b) TH-motifs 90-92 (c) TH-motifs 93-95

Fig. 7: The numbers of instances of nine pair-inducing TH-motifs depend on the ordering of the temporal hyperedges. The ratio of the occurrences of TH-motifs 89, 92, and 95 are significantly low compared to the other TH-motifs with the same structures.

domains. These results support that TH-motifs play a key role in capturing structural and temporal patterns of real-world temporal hypergraphs.

Obs 3. Orders of hyperedges matter: TH-motifs are asymmetric with respect to the arrival order of the temporal hyperedges, and thus instances that are indistinguishable from static h-motifs can be categorized as different TH-motifs. We are interested in how the orders of the hyperedges affect the occurrences of TH-motifs, and to this end, we statistically investigate nine pair-inducing ones, ranging from TH-motif 87 to 95. TH-motifs in each triple, TH-motifs $87-89$, TH-motifs $90-92$, and TH-motifs $93-95$ share the same structural pattern, but they are distinguished by the orders of the hyperedges. Consider an instance $\langle e_i, e_j, e_k \rangle$ of the pair-inducing TH-motif. The pair-inducing TH-motifs, by definition, consist of a pair of duplicated hyperedges and thus enables three different orderings **O1**: $\tilde{e}_i = \tilde{e}_j \neq \tilde{e}_k$, **O2**: $\tilde{e}_i \neq \tilde{e}_j = \tilde{e}_k$, and **O3**: $\tilde{e}_i \neq \tilde{e}_j \neq \tilde{e}_k$, $\tilde{e}_i = \tilde{e}_k$,. In **O1** and **O2**, duplicated temporal hyperedges occur consecutively, whereas in **O3**, the first and last hyperedges are duplicated. TH-motifs 87, 90, and 93 are **O1**, TH-motifs 88, 91, and 94 are **O2**, and TH-motifs 89, 92, and 95 are **O3**. As seen in Fig. 7, this difference indeed affect the occurrences of the TH-motifs in real-world temporal hypergraphs. The ratio of the TH-motifs whose ordering is **O3** are significantly small, compared to that of **O1** and **O2**. That is, duplicated temporal hyperedges tend to occur in a short time and thus affect the count distributions of TH-motifs.

6.3 Q2. Comparison with Static H-motifs

In this subsection, we demonstrate the usefulness of TH-motifs. We compare TH-motifs and static h-motifs as input features for a hyperedge prediction task.

Obs 4. TH-motifs help predict future hyperedges: To verify the usefulness of temporal h-motifs, we consider the problem of hyperedge prediction, a binary classification problem of predicting whether the given hyperedge is true or not. Given a temporal hypergraph $T = (V, \mathscr{E})$, we generate a set $\mathscr{E}'$ of fake hyperedges, whose number is the same as the true ones (i.e., $|\mathscr{E}| = |\mathscr{E}'|$). Fake hyperedges are generated by using HyperCL (Lee et al., 2021) which preserves the degrees of the nodes and the sizes of the hyperedges. The timestamps of the fake hyperedges are randomly assigned. We sort the entire temporal hyperedges $\mathscr{E} \cup \mathscr{E}'$

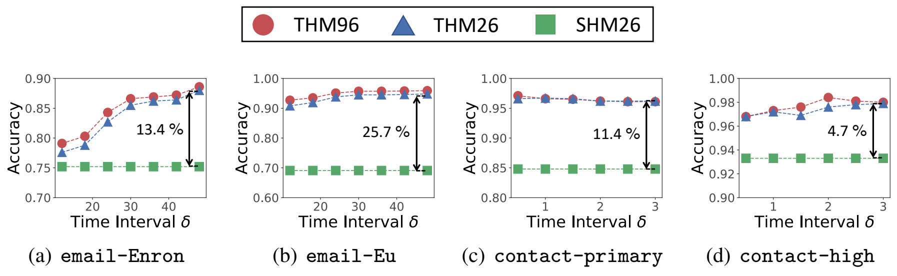


Fig. 8: TH-motifs provide informative features of temporal hyperedges. THM96 and THM26, which use the counts of TH-motifs' instances as features, are more accurate than SHM26, which uses the counts of static h-motifs' instances, in predicting future temporal hyperedges. Results in small datasets where the instances of static h-motifs can be exactly counted are reported.

based on their timestamps and split them into train and test sets in a ratio 8:2. Then we train a logistic regression classifier using the train set with the following three different features of each temporal hyperedge:

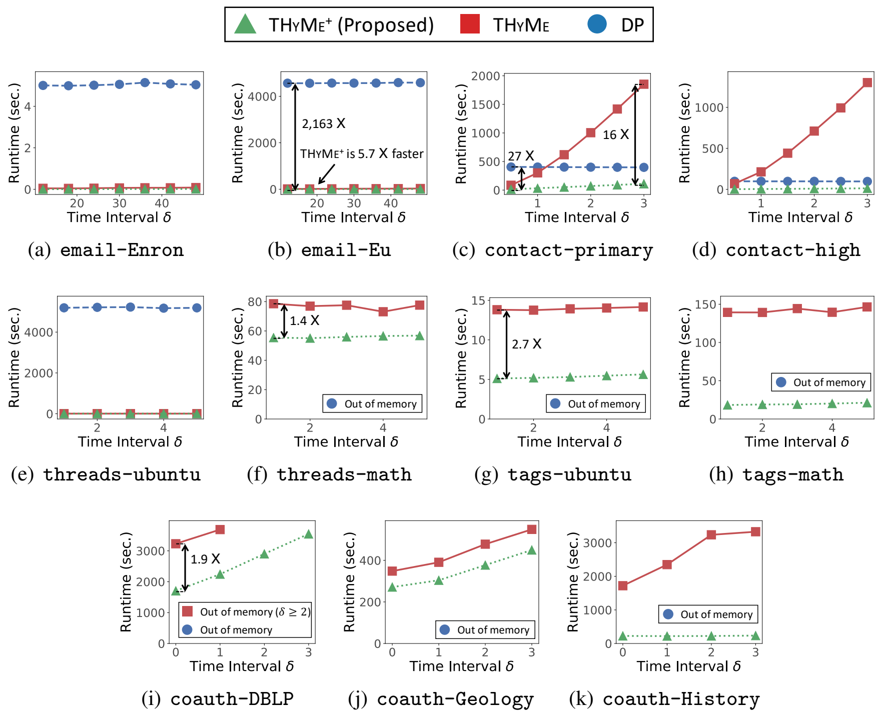
- **THM96** ($\in \mathbb{R}^{96}$)**:** Each dimension represents the number of instances of TH-motifs that contain the hyperedge.
- **THM26** ($\in \mathbb{R}^{26}$)**:** The 26 TH-motifs whose occurrences have the highest variance are selected.
- **SHM26** ($\in \mathbb{R}^{26}$)**:** Each dimension represents the number of instances of static h-motifs that contain the hyperedge. Temporal information is ignored.

As seen in Fig. 8, THM96 and THM26, which are based on the TH-motifs counts, are more accurate than STM26. While h-motifs only represent structural patterns, TH-motifs incorporate temporal information in addition to them, and thus they are more informative.

6.4 Q3. Performance of Exact Algorithms

We evaluate the speed and efficiency of the proposed algorithms DP, THYME, and THYME$^+$. As seen in Fig. 9, while DP and THYME run out of memory in some datasets or with particular δ values, THYME$^+$ is fast and space efficient enough in all considered settings. Specifically, THYME$^+$ is up to $2,163\times$ faster than DP and $16\times$ faster than THYME. As described in Section 4, THYME$^+$ maintains a small projected graph Q and thus reduces enumeration over the instances in Q. In the next subsection, we provide empirical findings that support the effectiveness of THYME$^+$. Why is THYME$^+$ faster and more space efficient compared to DP and THYME? What properties of real-world temporal hypergraphs make THYME$^+$ efficient? To answer these questions, we examine structural and temporal patterns of temporal hyperedges in real-world temporal hypergraphs and summarize common properties observed as follows.

- **(Obs. 5) Repetitive behavior:** Duplicated temporal hyperedges tend to appear repeatedly, and the distribution of the numbers of repetitions is heavy-tailed.
- **(Obs. 6) Temporal locality:** Future temporal hyperedges are more likely to repeat recent hyperedges than older ones.



(a) `email-Enron` (b) `email-Eu` (c) `contact-primary` (d) `contact-high`

(e) `threads-ubuntu` (f) `threads-math` (g) `tags-ubuntu` (h) `tags-math`

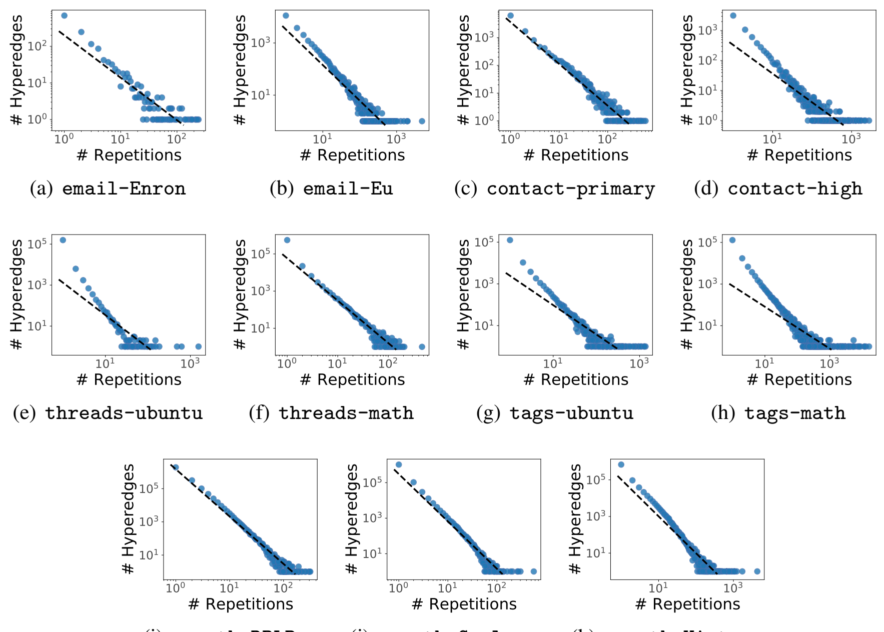
(i) `coauth-DBLP` (j) `coauth-Geology` (k) `coauth-History`

Fig. 9: THyMe$^+$ is faster and more space efficient than DP and THyMe.

Table 3: The log-likelihood ratio when fitting the distributions to each of three heavy-tailed distributions against exponential distributions is positive in all real-world hypergraphs.

Dataset	power-law	truncated power-law	log normal
`email-Enron`	7.930	**8.965**	8.878
`email-Eu`	3.702	**3.865**	3.838
`contact-primary`	3.626	**5.720**	5.453
`contact-high`	22.278	**23.464**	23.340
`threads-ubuntu`	3.353	**3.355**	3.257
`threads-math`	15.003	**16.171**	16.027
`tags-ubuntu`	12.633	**12.714**	12.698
`tags-math`	15.447	15.503	**15.507**
`coauth-DBLP`	25.164	**26.287**	26.157
`coauth-Geology`	18.010	**19.233**	18.965
`coauth-History`	8.148	8.169	**8.183**

Obs. 5. Repetitive behavior: We first investigate the repeating patterns (i.e., duplication) of temporal hyperedges in real-world temporal hypergraphs. As seen in Table 2, the number of induced hyperedges ($|E_{\mathscr{E}}|$) is significantly smaller than that of temporal hyperedges ($|\mathscr{E}|$), implying that temporal hyperedges are frequently repeated. Surprisingly, in the `contact-high` dataset, the number of induced hyperedges is only 4.5% of that of temporal hyperedges, implying that most temporal hyperedges consist of predefined set of nodes. Note that due to

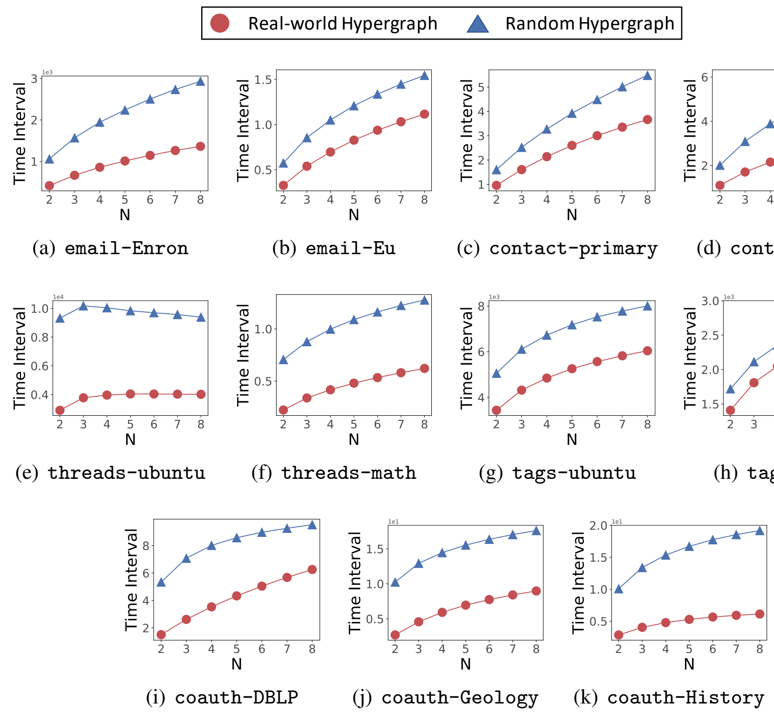


(a) `email-Enron` (b) `email-Eu` (c) `contact-primary` (d) `contact-high`
(e) `threads-ubuntu` (f) `threads-math` (g) `tags-ubuntu` (h) `tags-math`
(i) `coauth-DBLP` (j) `coauth-Geology` (k) `coauth-History`

Fig. 10: Temporal hyperedges in real-world hypergraphs are repetitive, and the number of repetitions follows a near power-law distribution. This claim is supported numerically in Table 3.

the flexibility of hyperedge sizes, a hyperedge can be generated from $O(2^{|V|})$, and thus is extremely unlikely to repeat the exact set of nodes. In addition, we discover that the distributions of hyperedge repetitions in real-world temporal hypergraphs are generally heavy-tailed and close to power-law distributions, as seen in Fig. 10. This claim is supported by the log-likelihood ratios when fitting three representative heavy-tailed distributions (power-law, truncated power-law, and log normal) against the exponential distribution (Clauset et al., 2009; Alstott et al., 2014). As reported in Table 3, all datasets have positive ratios in all heavy-tailed distributions, indicating that the distributions of hyperedge repetitions are heavy-tailed and close to power-law distributions. Among the three heavy-tailed distributions, truncated power-law distribution is the best candidate in most datasets.

Obs. 6. Temporal locality: Now that we have observed the structural behaviors of the temporal hyperedges, we turn our attention to the temporal aspect. The temporal locality of temporal hyperedges is the tendency that recent hyperedges are more likely to be repeated in the near future than the older ones. To show the temporal locality, we investigate the time intervals of the N consecutive identical temporal hyperedges, i.e., the time it takes for a hyperedge to be repeated N times. Fig. 11 shows the average time intervals of all the hyperedges in the real-world hypergraphs and randomly shuffled hypergraphs, where timestamps of the hyperedges are randomly shuffled while preserving the underlying structure. In every dataset, the time intervals within N consecutive hyperedges are shorter in real-world hypergraphs than in randomized ones. That is, future hyperedges are more likely to repeat the recent hyperedges than older ones.



(a) `email-Enron` (b) `email-Eu` (c) `contact-primary` (d) `contact-high`

(e) `threads-ubuntu` (f) `threads-math` (g) `tags-ubuntu` (h) `tags-math`

(i) `coauth-DBLP` (j) `coauth-Geology` (k) `coauth-History`

Fig. 11: Temporal hyperedges in real-world temporal hypergraphs are temporally local. The time interval of N consecutive duplicated temporal hyperedges tends to be shorter in real-world hypergraphs than in randomized ones. The units of time intervals in all datasets from the `email`, `contact`, `threads`, and `tags` domains are hours. The time unit in all datasets from the `coauth` domain is years.

Intuition behind THYME$^+$: How do these properties of real-world temporal hypergraphs provide efficiency to THYME$^+$? Here, we provide some reasons why we expect THYME$^+$ to be faster and more space-efficient than THYME and DP.

- **Connection to Obs. 5:** Each node in the projected graph P used in THYME represents a unique temporal hyperedge, and its size heavily depends on δ. On the other hand, the nodes in the projected graph Q maintained in THYME$^+$ represent induced hyperedges, and several temporal hyperedges can share the same node. Thus, more repetitions of temporal hyperedges provide higher efficiency of THYME$^+$, as observed in real-world temporal hypergraphs.
- **Connection to Obs. 6:** The benefits of the temporal locality of temporal hyperedges are two-fold: (1) The tendency of temporal hyperedges to repeat within a short period of time indicates that duplicated temporal hyperedges are more likely to co-appear in the temporal window in THYME$^+$, which reduces the size of the projected graph Q. (2) If duplicated temporal hyperedges reappear within the temporal window, insertion/deletion of nodes and edges of Q are skipped, which is beneficial in terms of speed.

6.5 Q4. Performance of Approximate Algorithms

We evaluate the speed and accuracy of the proposed algorithms, DTP, THYME-A, THYME-A$^+$, and THYME-A*. To this end, we measure the elapsed time and the relative error, defined as:

$$\frac{\sum_{M=1}^{96} |C[M] - C_{\text{approx}}[M]|}{\sum_{M=1}^{96} C[M]},$$

where $C_{\text{approx}}[M]$ is an approximated count of TH-motif M ($\widetilde{C}$, $\widehat{C}$, or $\overline{C}$). For THYME-A, THYME-A$^+$, and THYME-A*, we fix the sampling time interval to $\triangle = 0.001 \cdot (t_{\max} - t_{\min})$. The number s of samples is set differently across datasets: $s \in \{1,2,3,4,5\}$ for the datasets from the `coauthorship` domain, $s \in \{50,100,150,200,250\}$ for the `threads-ubuntu`, and $s \in \{20,40,60,80,100\}$ for the remaining datasets. For DTP, we perform experiments on the number of shifts $b \in \{1,2,3,4,5\}$, the window size parameter $c = 10$, and the constant $r = \{0.001, 0.01, 0.1, 1.0\}$. Empirically, the performance of DTP is very sensitive to these hyperparameters, and thus we report the results where relative errors are less than 0.8 while being faster than THYME$^+$. As seen in Fig. 12, THYME-A* yields the best trade-offs between speed and accuracy, which is consistent with our theoretical analyses. For example, in `email-Eu`, THYME-A* provides $11.1\times$ lower relative error compared to DTP. Notably, DTP performs poorly in most datasets due to the limitations discussed in Section 5.1.

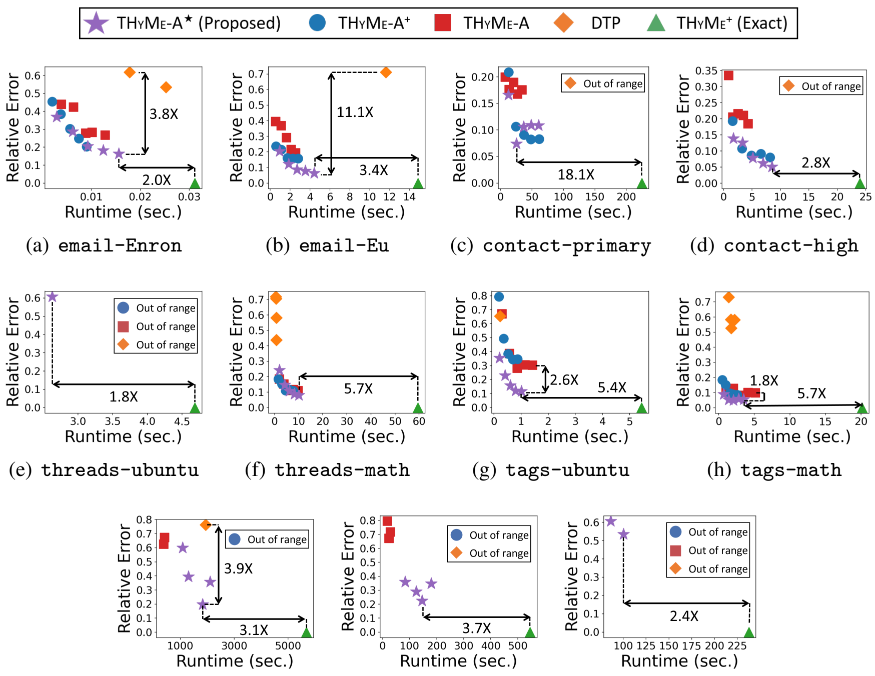
7 Conclusion

In this work, we propose (a) temporal hypergraph motifs (TH-motifs), which are tools for analyzing design principles of time-evolving hypergraphs, (b) THYME$^+$, which is a fast algorithm for exactly counting TH-motifs' instances, and (c) THYME-A*, which is a fast and accurate algorithm for approximately counting TH-motifs' instances. Using them, we investigate 11 real-world hypergraphs from 5 domains. Our contributions are summarized as follows.

- **New concept:** We define 96 temporal hypergraph motifs (TH-motifs) that describe local relational and temporal dynamics in time-evolving hypergraphs.
- **Fast and exact algorithm:** We develop THYME$^+$, a fast and exact algorithm for counting the instances of TH-motifs. It is at most $2,163\times$ faster than the baseline approach.
- **Accurate approximate algorithm:** We develop THYME-A*, an accurate sampling algorithm for approximately counting the instances of TH-motifs. THYME-A* yields up to $11.1\times$ more accurate estimates rapidly, compared to the baseline approach.
- **Empirical discoveries:** TH-motifs reveal interesting structural and temporal patterns in real-world hypergraphs. TH-motifs also provide informative features that are useful in predicting future hyperedges.

Reproducibility: The source code and datasets used in this work are available at `https://github.com/geonlee0325/THyMe`.

Acknowledgements This work was supported by National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2020R1C1C1008296) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-00075, Artificial Intelligence Graduate School Program (KAIST)).



(a) `email-Enron` (b) `email-Eu` (c) `contact-primary` (d) `contact-high`

(e) `threads-ubuntu` (f) `threads-math` (g) `tags-ubuntu` (h) `tags-math`

(i) `coauth-DBLP` (j) `coauth-Geology` (k) `coauth-History`

Fig. 12: THyMe-A* gives the best trade-off between speed and accuracy. It yields up to 11.1× more accurate estimation than DTP, and it is up to 18.1× faster than THyMe$^+$. The results where the relative error is higher than 0.8 or the running time is greater than that of THyMe$^+$ are not shown in the figure. We set δ to 96 hours, 8 hours, 4 hours, 4 hours, and 3 years for the datasets from the `email`, `contact`, `threads`, `tags`, and `coauthorship` domains, respectively.

References

Alstott J, Bullmore E, Plenz D (2014) powerlaw: a python package for analysis of heavy-tailed distributions. PloS one 9(1):e85777

Amburg I, Veldt N, Benson A (2020) Clustering in graphs and hypergraphs with categorical edge labels. In: WWW

Arenas A, Fernandez A, Fortunato S, Gomez S (2008) Motif-based communities in complex networks. Journal of Physics A: Math Theor 41(22):224001

Benson AR, Gleich DF, Leskovec J (2016) Higher-order organization of complex networks. Science 353(6295):163–166

Benson AR, Abebe R, Schaub MT, Jadbabaie A, Kleinberg J (2018a) Simplicial closure and higher-order link prediction. Proceedings of the National Academy of Sciences 115(48):E11221–E11230

Benson AR, Kumar R, Tomkins A (2018b) Sequences of sets. In: KDD

Borgatti SP, Everett MG (1997) Network analysis of 2-mode data. Social networks 19(3):243–269

Chodrow PS (2020) Configuration models of random hypergraphs. Journal of Complex Networks 8(3):cnaa018

Choe M, Yoo J, Lee G, Baek W, Kang U, Shin K (2022) Midas: Representative sampling from real-world hypergraphs. In: WWW

Choo H, Shin K (2022) On the persistence of higher-order interactions in real-world hypergraphs. In: SDM

Clauset A, Shalizi CR, Newman ME (2009) Power-law distributions in empirical data. SIAM review 51(4):661–703

Do MT, Yoon Se, Hooi B, Shin K (2020) Structural patterns and generative models of real-world hypergraphs. In: KDD

Feng Y, You H, Zhang Z, Ji R, Gao Y (2019) Hypergraph neural networks. In: AAAI

Gurukar S, Ranu S, Ravindran B (2015) Commit: A scalable approach to mining communication motifs from dynamic networks. In: SIGMOD

Hwang T, Tian Z, Kuangy R, Kocher JP (2008) Learning on weighted hypergraphs to integrate protein interactions and gene expressions for cancer outcome prediction. In: ICDM

Karypis G, Aggarwal R, Kumar V, Shekhar S (1999) Multilevel hypergraph partitioning: Applications in vlsi domain. TLVLSI 7(1):69–79

Kim S, Choe M, Yoo J, Shin K (2022) Reciprocity in directed hypergraphs: Measures, findings, and generators. In: ICDM

Ko J, Kook Y, Shin K (2022) Growth patterns and models of real-world hypergraphs. Knowledge and Information Systems 64(11):2883–2920

Kook Y, Ko J, Shin K (2020) Evolution of real-world hypergraphs: Patterns and models without oracles. In: ICDM

Kovanen L, Karsai M, Kaski K, Kertész J, Saramäki J (2011) Temporal motifs in time-dependent networks. Journal of Statistical Mechanics: Theory and Experiment 2011(11):P11005

Lee G, Shin K (2021) Thyme+: Temporal hypergraph motifs and fast algorithms for exact counting. In: ICDM

Lee G, Ko J, Shin K (2020) Hypergraph motifs: concepts, algorithms, and discoveries. PVLDB 13:2256–2269

Lee G, Choe M, Shin K (2021) How do hyperedges overlap in real-world hypergraphs?– patterns, measures, and generators. In: WWW

Lee G, Choe M, Shin K (2022a) Hashnwalk: Hash and random walk based anomaly detection in hyperedge streams. In: IJCAI

Lee G, Yoo J, Shin K (2022b) Mining of real-world hypergraphs: Patterns, tools, and generators. In: CIKM

Lee JB, Rossi RA, Kong X, Kim S, Koh E, Rao A (2019) Graph convolutional networks with motif-based attention. In: CIKM

Li P, Milenkovic O (2017) Inhomogoenous hypergraph clustering with applications. In: NIPS

Li PZ, Huang L, Wang CD, Lai JH (2019) Edmot: An edge enhancement approach for motif-aware community detection. In: KDD

Li Y, Lou Z, Shi Y, Han J (2018) Temporal motifs in heterogeneous information networks. In: MLG Workshop

Liu P, Benson AR, Charikar M (2019) Sampling methods for counting temporal motifs. In: WSDM

Milo R, Shen-Orr S, Itzkovitz S, Kashtan N, Chklovskii D, Alon U (2002) Network motifs: simple building blocks of complex networks. Science 298(5594):824–827

Milo R, Itzkovitz S, Kashtan N, Levitt R, Shen-Orr S, Ayzenshtat I, Sheffer M, Alon U (2004) Superfamilies of evolved and designed networks. Science 303(5663):1538–1542
Paranjape A, Benson AR, Leskovec J (2017) Motifs in temporal networks. In: WSDM
Redmond U, Cunningham P (2013) Temporal subgraph isomorphism. In: ASONAM
Rossi RA, Ahmed NK, Koh E (2018a) Higher-order network representation learning. In: WWW Companion
Rossi RA, Zhou R, Ahmed NK (2018b) Deep inductive graph representation learning. IEEE TKDE 32(3):438–452
Rossi RA, Ahmed NK, Carranza A, Arbour D, Rao A, Kim S, Koh E (2020a) Heterogeneous graphlets. ACM TKDD 15(1):1–43
Rossi RA, Ahmed NK, Koh E, Kim S, Rao A, Abbasi-Yadkori Y (2020b) A structural graph representation learning framework. In: WSDM
Shen-Orr SS, Milo R, Mangan S, Alon U (2002) Network motifs in the transcriptional regulation network of escherichia coli. Nature Genetics 31(1):64–68
Tsourakakis CE, Pachocki J, Mitzenmacher M (2017) Scalable motif-aware graph clustering. In: WWW
Yadati N, Nimishakavi M, Yadav P, Nitin V, Louis A, Talukdar P (2018) Hypergcn: A new method of training graph convolutional networks on hypergraphs. arXiv preprint arXiv:180902589
Yang D, Qu B, Yang J, Cudre-Mauroux P (2019) Revisiting user mobility and social relationships in lbsns: A hypergraph embedding approach. In: WWW
Yin H, Benson AR, Leskovec J, Gleich DF (2017) Local higher-order graph clustering. In: KDD
Yoon Se, Song H, Shin K, Yi Y (2020) How much and when do we need higher-order information in hypergraphs? a case study on hyperedge prediction. In: WWW
Yu J, Tao D, Wang M (2012) Adaptive hypergraph learning and its application in image classification. TIP 21(7):3262–3272
Yu Y, Lu Z, Liu J, Zhao G, Wen Jr (2019) Rum: Network representation learning using motifs. In: ICDE
Zhao H, Xu X, Song Y, Lee DL, Chen Z, Gao H (2018) Ranking users in social networks with higher-order structures. In: AAAI

A Details of DYNAMIC PROGRAMMING (DP)

In this subsection, we provide details of the DYNAMIC PROGRAMMING (DP), a straightforward extension of temporal network motif counting (Paranjape et al., 2017). As shown in Algorithm 6, it utilizes a dynamic programming approach to reduce the redundant enumeration of every instance of TH-motifs of the input temporal hypergraph T. The procedure `count` (lines 9-19) counts the instances of TH-motifs that induce a set of ℓ connected static hyperedges. That is, given a set $s = \{\tilde{e}_1, \dots, \tilde{e}_\ell\}$ of ℓ connected static hyperedges, `count` first constructs a time-sorted sequence $e(s)$ of temporal hyperedges whose nodes is one of s (line 10). It also introduces a map C that maintains the counts of ordered hyperedges of length at most ℓ. Then `count` scans through the temporal hyperedges in $e(s)$ and tracks the subsequences that occur within the temporal window that spans temporal hyperedges within δ time units. As the temporal window slides through the temporal hyperedges $e(s)$, the count of the sequences are computed based on the subsequences counted in C. Refer to (Paranjape et al., 2017) for more intuition behind this dynamic programming formulation.

B Details of DISCRETE TIME PARTITIONING (DTP)

In this subsection, we provide details of the DISCRETE TIME PARTITIONING (DTP), a straightforward extension of sampling algorithm for approximate temporal network motif counting (Liu et al., 2019). As shown

Algorithm 6: DP: Preliminary Algorithm for Exact Counting of TH-motifs

```
Input  : (1) temporal hypergraph: $T = (V, \mathscr{E})$
         (2) time interval $\delta$
Output: # of each temporal h-motif $t$'s instances: $M[t]$
$S \leftarrow$ set of instances of static h-motifs in $G_T$
for each instance $\{\tilde{e}_i, \tilde{e}_j, \tilde{e}_k\} \in S$ do
    count($\{\tilde{e}_i, \tilde{e}_j, \tilde{e}_k\}$)
for each pair of overlapping hyperedges $\{\tilde{e}_i, \tilde{e}_j\} \in \wedge_{\mathscr{E}}$ do
    count($\{\tilde{e}_i, \tilde{e}_j\}$)
for each hyperedge $\tilde{e}_i \in E_{\mathscr{E}}$ do
    count($\{\tilde{e}_i\}$)
return $M$
Procedure count($s = \{\tilde{e}_1, \dots, \tilde{e}_\ell\}$)
    $e(s) \leftarrow$ sorted($I(\tilde{e}_1) \cup \cdots \cup I(\tilde{e}_\ell)$)
    $w_s \leftarrow 1$
    $C \leftarrow$ map initialized to 0
    for each temporal hyperedge $e_i = (\tilde{e}_i, t_i) \in e(s)$ do
        while $t_{w_s} + \delta < t_i$ do
            decrement($\tilde{e}_{w_s}$)
            $w_s \leftarrow w_s + 1$
        increment($\tilde{e}_i$)
    for each $\langle e_i, e_j, e_k \rangle \in$ permutations($\{\tilde{e}_i, \tilde{e}_j, \tilde{e}_k\}$) do
        $M[h(\tilde{e}_i, \tilde{e}_j, \tilde{e}_k)]$ += $C$[concat($\tilde{e}_i, \tilde{e}_j, \tilde{e}_k$)]
Procedure increment($\tilde{e}$)
    for each prefix in $C$.keys.reverse of length $< \ell$ do
        $C$[concat(prefix, $\tilde{e}$)] += $C$[prefix]
    $C[\tilde{e}] \leftarrow C[\tilde{e}] + 1$
Procedure decrement($\tilde{e}$)
    $C[\tilde{e}] \leftarrow C[\tilde{e}] - 1$
    for each suffix in $C$.keys of length $< \ell - 1$ do
        $C$[concat($\tilde{e}$, suffix)] -= $C$[suffix]
```

in Algorithm 7, it begins with randomly drawing a *shift* s from $\{-c\delta + 1, \cdots, 0\}$ for the predefined input integer $c > 0$ that controls the size of the sampling windows (line 3). Then, based on the selected shift s, the time is discretely partitioned into $c\delta$-sized intervals:

$$\mathscr{I}_s = \{[s + (j-1)c\delta, s + j \cdot c\delta - 1], j = 1, 2, ...\}.$$

The probability of an instance $\langle e_i, e_j, e_k \rangle$ of TH-motif to be completely contained within an interval $\mathscr{I}_s$ is:

$$1 - \frac{\triangle(\langle e_i, e_j, e_k \rangle)}{c\delta}.$$

Then, for each interval $I \in \mathscr{I}_s$, DTP constructs a sub-temporal hypergraph that consists of temporal hyperedges within I (line 6). From the partial temporal hypergraph, it uses THYME to exhaustively discover the instances of TH-motifs. It associates a weighted count of the number of instances of TH-motifs by incrementing the counts by the inverse of the probability to be completely contained in the interval I. By adding up the weighted counts of all intervals, it yields an unbiased estimation of the number of instances of each TH-motif.

To speed up the estimation, DTP incorporates importance sampling to pick only a subset of intervals and combines the counts from them. Let I_j be the j-th interval. Specifically, it samples an interval $I_j \in \mathscr{I}_s$ independently with the predefined probability z_j (line 5). Here, DTP uses the ratio of the number of temporal hyperedges that arrived within the interval, which can be easily obtained from the input temporal hypergraph. Formally, the ratio z_j is

$$z_j = r \cdot \frac{|\{e = (\tilde{e}, t) \in \mathscr{E} : t \in I_j\}|}{|\mathscr{E}|}$$

Algorithm 7: DTP: Preliminary Algorithm for Approximate Counting of TH-motifs

```
Input  : (1) temporal hypergraph: $T = (V, \mathscr{E})$
         (2) time interval $\delta$
         (3) sampling probabilities $q$
         (4) number of shifts $b$
         (5) window size parameter $c$
Output: approximated # of each temporal h-motif $M$'s instances: $\ddot{C}[M]$
1  $\ddot{C} \leftarrow$ map initialized to zero
2  for each shift $k = 1, \cdots, b$ do
3      $s \leftarrow$ random integer from $\{-c\delta + 1, \cdots, 0\}$
4      for each interval index $j = 1, \cdots, 1 + \lceil \frac{t_{max}}{c\delta} \rceil$ do
5          if $Uniform(0,1) \leq z_j$ then
6              $T_j \leftarrow \{e = (\tilde{e}, t) \in \mathscr{E} : t \in [s + (j-1) \cdot c\delta, s + j \cdot c\delta - 1]\}$
7              $m_j \leftarrow$ set of instances discovered in $T_j$ using THyMe
8              for each instance $\langle e_i, e_j, e_k \rangle \in m_j$ do
9                  $\ddot{C}[h(\tilde{e}_i, \tilde{e}_j, \tilde{e}_k)]$ += $\frac{1}{\left(1 - \frac{\triangle(\langle e_i, e_j, e_k \rangle)}{c\delta}\right) \cdot z_j}$
10 for each TH-motif $M$ do
11     $\ddot{C}[M] \leftarrow \frac{1}{b} \ddot{C}[M]$
12 return $\tilde{M}$
```

Algorithm 8: Finding the neighbors N_{e_i} of a temporal hyperedge e_i in the projected graph P in THyMe.

```
Input  : (1) projected graph $P = (V_P, E_P)$
         (2) target temporal hyperedge $e_i = (\tilde{e}_i, t_i)$
Output: set of neighbors $N_{e_i}$
1  $N_{e_i} \leftarrow \varnothing$
2  for each node $v \in \tilde{e}_i$ do
3      for each temporal hyperedge $e_j \in V_P$ where $v \in \tilde{e}_j$ do
4          $N_{e_i} \leftarrow N_{e_i} \cup \{e_j\}$
5  return $N_{e_i}$
```

Algorithm 9: Finding the neighbors $N_{\tilde{e}_i}$ of an induced hyperedge $\tilde{e}_i$ in the projected graph Q in THyMe$^+$.

```
Input  : (1) projected graph $Q = (V_Q, E_Q, t_Q)$
         (2) target induced hyperedge $\tilde{e}_i$
Output: set of neighbors $N_{\tilde{e}_i}$
1  $N_{\tilde{e}_i} \leftarrow \varnothing$
2  for each node $v \in \tilde{e}_i$ do
3      for each induced hyperedge $\tilde{e}_j \in V_Q$ where $v \in \tilde{e}_j$ do
4          $N_{\tilde{e}_i} \leftarrow N_{\tilde{e}_i} \cup \{\tilde{e}_j\}$
5  return $N_{\tilde{e}_i}$
```

where r is a constant that controls the magnitude of the probability. Then, the number of instances counted in the interval is additionally weighted by $\frac{1}{z_j}$ (line 9). These computations are repeated b times to reduce the variance of the estimation. Refer to (Liu et al., 2019) for details of the original algorithm.

Algorithm 10: Finding the set of three connected temporal hyperedges in P that contain e_i in THYME.

```
Input  : (1) projected graph $P = (V_P, E_P)$
         (2) target temporal hyperedge $e_i$
Output: set of three connected temporal hyperedges $S$
1 $S \leftarrow \varnothing$
2 for each temporal hyperedge $e_j \in N_{e_i}$ do
3     for each temporal hyperedge $e_k \in (N_{e_i} \cup N_{e_j} \setminus \{e_i, e_j\})$ do
4         if $e_k \notin N_{e_i}$ or $j < k$ then
5             if $t_j < t_k$ then
6                 $S \leftarrow S \cup \{\langle e_j, e_k, e_i \rangle\}$
7             else
8                 $S \leftarrow S \cup \{\langle e_k, e_j, e_i \rangle\}$
9 return $S$
```

Algorithm 11: Finding the set of three connected static hyperedges in Q that contain $\tilde{e}_i$ in THYME$^+$.

```
Input  : (1) projected graph $Q = (V_Q, E_Q)$
         (2) target static hyperedge $\tilde{e}_i$
Output: set of three connected static hyperedges $S$
1 $S \leftarrow \varnothing$
2 for each static hyperedge $\tilde{e}_j \in N_{\tilde{e}_i}$ do
3     for each static hyperedge $\tilde{e}_k \in (N_{\tilde{e}_i} \cup N_{\tilde{e}_j} \setminus \{\tilde{e}_i, \tilde{e}_j\})$ do
4         if $\tilde{e}_k \notin N_{\tilde{e}_i}$ or $j < k$ then
5             $S \leftarrow S \cup \{\{\tilde{e}_k, \tilde{e}_j, \tilde{e}_k\}\}$
6 return $S$
```

C Sub-algorithms of THYME and THYME$^+$

In this subsection, we provide pseudocode of four sub-algorithms of THYME and THYME$^+$. These algorithms are used for analyzing the time complexity of THYME and THYME$^+$ in Section 4.

Author Biographies

Geon Lee is a Ph.D. student at the Kim Jaechul Graduate School of AI at KAIST. He received his B.S. degree in Computer Science and Engineering from Sungkyunkwan University in 2019. His research interests include graph mining and its applications. Especially, his studies of hypergraphs have appeared in major data mining venues, including VLDB, WWW, and ICDM.

Kijung Shin is an Ewon Endowed Assistant Professor in the Kim Jaechul Graduate School of AI and the School of Electrical Engineering at KAIST. He received his Ph.D. in Computer Science from Carnegie Mellon University in 2019 and his B.S. in Computer Science and Engineering from Seoul National University in 2015. He has published over 50 referred articles, and his work has appeared in top-tier data mining conferences and journals, including KDD, WWW, ICDM, and ICDE. His research involves designing scalable data mining algorithms, with emphasis on graphs, hypergraphs, and tensors, and applying those algorithms to real-world problems.