THyMe+: Temporal Hypergraph Motifs and Fast Algorithms for Exact Counting **(Supplementary Document)**

Abstract—**This document provides supplementary information to the main paper "THyMe+: Temporal Hypergraph Motifs and Fast Algorithms for Exact Counting." We provide additional experimental results and complexity analysis.**

I. Experimental Results

In this section, we provide results of additional experiments that complement the main paper. We use 11 real-world temporal hypergraphs from 5 different domains.

Real hypergraphs are not 'random': We compare the number of TH-motifs' instances in real-world temporal hypergraphs and that of randomized hypergraphs. Random temporal hypergraphs are obtained by using HyperCL [1], which preserves the degrees of the nodes and the sizes of the hyperedges. Timestamps of hyperedges are randomly assigned. As seen in Figure 1, the distributions of the occurrences of TH-motifs in real-world hypergraphs and randomized hypergraphs are significantly different.

THyMe+ is fast: We evaluate the speed and efficiency of the considered algorithms, DP, THyMe, and THyMe+. As seen in Figure 2, DP and THyMe run out of memory in many datasets and/or for some δ values. THyMe+ is fast and space efficient, performing successfully in all considered settings.

Temporal hyperedges repeat: Duplicated temporal hyperedges tend to appear repeatedly, and the distribution of the numbers of repetitions are heavy-tailed. 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 [2], [3]. As reported in Table I, 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, as confirmed visually in Figure 3.

Temporal hyperedges are temporally local: Future temporal hyperedges are more likely to repeat recent hyperedges than older ones. In Figure 4, we investigate the time intervals of N consecutive duplicated temporal hyperedges, i.e., the average time taken for a hyperedge to be repeated N times. The time intervals within N consecutive temporal hyperedges tend to be shorter in real-world hypergraphs than in randomized ones.

TABLE I: 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**

II. Complexity Analysis

In this section, we analyze the complexity of THyMe+. We assume all sets and maps are implemented using hash tables, and the projected graphs P and Q are weighted graphs where the weight of edge $(e_i, e_j) \in E_P$ or $(\tilde{e}_i, \tilde{e}_j) \in E_Q$ is $|\tilde{e}_i \cap \tilde{e}_j|$.

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, its corresponding TH-motif is determined by 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) $\tilde{e}_i \cap \tilde{e}_j \cap \tilde{e}_k$. 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 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 $O(|\tilde{e}_s|) = O(\min(|\tilde{e}_i|, |\tilde{e}_j|, |\tilde{e}_k|))$

Complexity of updating the projected graph: The bottleneck of updating the projected graph in THyMe and THyMe+ is finding all neighbors of the target hyperedge.

Algorithm 1: 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 2: 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 $N_{\tilde{e}_i}$
```

Specifically, as described in Algorithm 1, given a temporal hyperedge e_i, the set of temporal hyperedges in P that share any number of nodes with e_i is retrieved in THYME. Similarly, as described in Algorithm 2, given an induced hyperedge $\tilde{e}_i$, the set of induced hyperedges in Q that are connected with $\tilde{e}_i$ is retrieved in THYME$^+$. The time complexities of Algorithm 1 and Algorithm 2 are given in Lemma 2 and Lemma 3, respectively.

Lemma 2 (Complexity of Algorithm 1 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 1 is $O(\sum_{e_j \in N_{e_i}} |\tilde{e}_i \cap \tilde{e}_j|)$.

Lemma 3 (Complexity of Algorithm 2 in THYME$^+$). The time complexity of Algorithm 2 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 1 is $O(\sum_{\tilde{e}_j \in N_{\tilde{e}_i}} |\tilde{e}_i \cap \tilde{e}_j|)$.

Complexity of three connected temporal/static hyperedges: Once the projected graph is updated, THYME and THYME$^+$ search for new instances of TH-motifs. Specifically, THYME searches for the set of instances of TH-motifs that contain e_i, whereas THYME$^+$ searches for the set of three connected induced static hyperedges that contain $\tilde{e}_i$. To this end, as described in Algorithm 3 and Algorithm 4, the 1-hop and 2-hop neighbors of the target hyperedge in the projected graphs are explored. The time complexities of Algorithm 3 and Algorithm 4 are given in Lemma 4 and Lemma 5.

Lemma 4 (Complexity of Algorithm 3 in THYME). The time complexity of Algorithm 3 is $O(\sum_{e_j \in N_{e_i}} |N_{e_i} \cup N_{e_j}|)$.

Algorithm 3: 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 4: 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 $S$
```

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.

Lemma 5 (Complexity of Algorithm 4 in THYME$^+$). The time complexity of Algorithm 4 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.

Complexity of the combinatorial approach in THYME$^+$: In THYME$^+$, we introduce `comb3`, `comb2`, and `comb1` (described in Algorithm 5), which efficiently count the number of triple-inducing, pair-inducing, and single-inducing TH-motifs, respectively, without directly enumerating 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 5 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 5 and line 6 in Algorithm 5 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. By the assumptions above, the size of t_Q is immediately obtainable, and thus line 7 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.

<u>**Overall complexity of THyMe:**</u> 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: (1) insertion into P, (2) removal from P, and (3) enumeration and counting TH-motifs. The time complexity of each step is summarized as follow.

- (1) 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.
- (2) 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.
- (3) 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 4, 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).

From the above analyses, step (3) is the bottleneck of the process, and thus the overall time complexity of THyMe for processing a temporal hyperedge e_i is bounded by Eq. (1).

<u>**Overall complexity of THyMe$^+$:**</u> Based on the analyses above, we sum up the time complexity of THyMe$^+$ in Theorem 2.

Theorem 2 (Complexity of THyMe$^+$)**.** The time complexity

Algorithm 5: Combinatorial TH-motifs' instances counting in THyMe$^+$

```
1 Procedure comb3 ($\tilde{e}_i, \tilde{e}_j, \tilde{e}_k$)
2     $M[h(\tilde{e}_j, \tilde{e}_k, \tilde{e}_i)]$ += $\sum_{t \in t_Q(\tilde{e}_j), t' \in t_Q(\tilde{e}_k)} \mathbb{1}[t < t']$
3     $M[h(\tilde{e}_k, \tilde{e}_j, \tilde{e}_i)]$ += $\sum_{t \in t_Q(\tilde{e}_j), t' \in t_Q(\tilde{e}_k)} \mathbb{1}[t' < t]$
4 Procedure comb2 ($\tilde{e}_i, \tilde{e}_j$)
5     $M[h(\tilde{e}_i, \tilde{e}_j, \tilde{e}_i)]$ += $\sum_{t \in t_Q(\tilde{e}_i) \setminus \{t_i\}, t' \in t_Q(\tilde{e}_j)} \mathbb{1}[t < t']$
6     $M[h(\tilde{e}_j, \tilde{e}_i, \tilde{e}_i)]$ += $\sum_{t \in t_Q(\tilde{e}_i) \setminus \{t_i\}, t' \in t_Q(\tilde{e}_j)} \mathbb{1}[t' < t]$
7     $M[h(\tilde{e}_j, \tilde{e}_j, \tilde{e}_i)]$ += $\binom{|t_Q(\tilde{e}_j)|}{2}$
8 Procedure comb1 ($\tilde{e}_i$)
9     $M[h(\tilde{e}_i, \tilde{e}_i, \tilde{e}_i)]$ += $\binom{|t_Q(\tilde{e}_i) - \{t_i\}|}{2}$
```

for 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: (1) insertion into Q, (2) removal from Q, and (3) enumeration of the three connected nodes in Q that contain $\tilde{e}_i$. The time complexity of each step is summarized as follow.

- (1) 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 3.
- (2) 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.
- (3) 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).

From the above analyses, step (3) is the bottleneck of the process, and thus the overall time complexity of THyMe$^+$ to process a temporal hyperedge e_i is bounded by Eq. (2).

Note that the cardinality of N_{e_i} in Eq. (1) (i.e., the time complexity of THyMe) is greater than or equal to the cardinality of $N_{\tilde{e}_i}$ in Eq. (2) (i.e., the time complexity of THyMe$^+$) for every temporal hyperedge $e_i = (\tilde{e}_i, t_i)$, and this provides an intuition why THyMe$^+$ is empirically faster than THyMe.

References

[1] G. Lee, M. Choe, and K. Shin, "How do hyperedges overlap in real-world hypergraphs?–patterns, measures, and generators," in *WWW*, 2021.

[2] A. Clauset, C. R. Shalizi, and M. E. Newman, "Power-law distributions in empirical data," *SIAM review*, vol. 51, no. 4, pp. 661–703, 2009.

[3] J. Alstott, E. Bullmore, and D. Plenz, "powerlaw: a python package for analysis of heavy-tailed distributions," *PloS one*, vol. 9, no. 1, p. e85777, 2014.

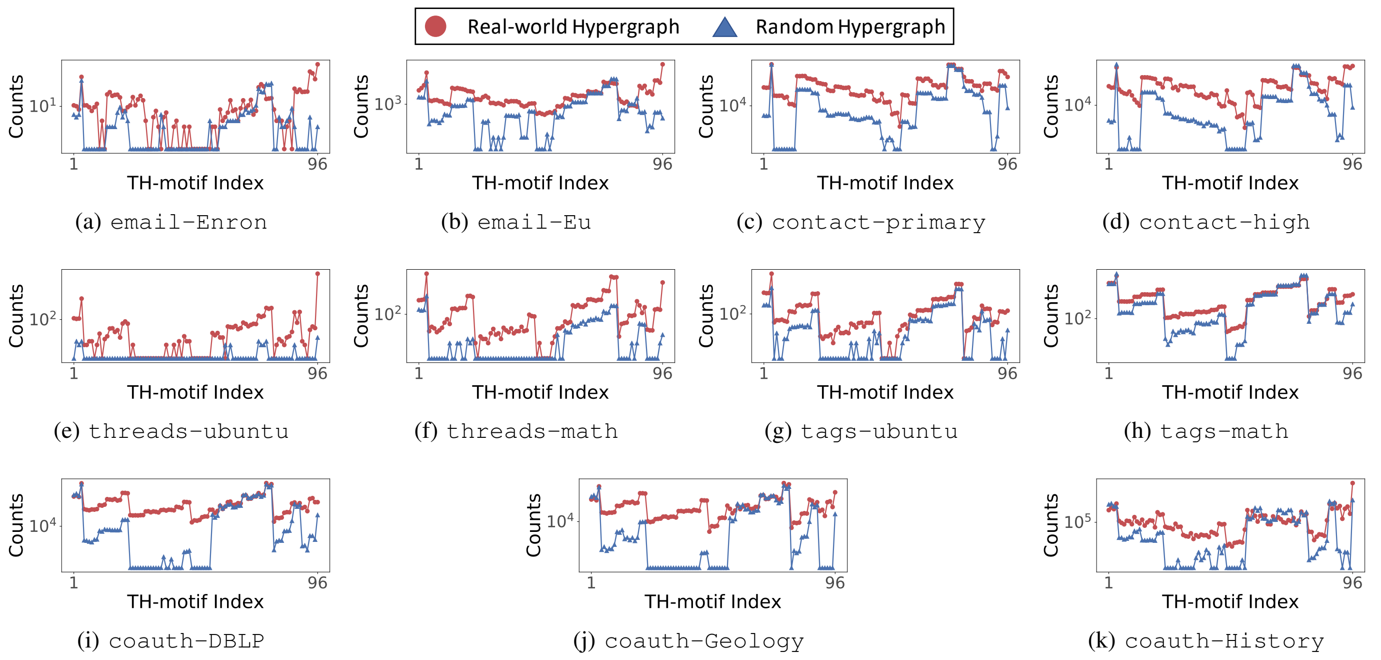


Fig. 1: The distribution of the numbers of TH-motifs' instances in real-world temporal hypergraphs and that in randomized temporal hypergraphs are significantly different. We set δ to 1 hour in all datasets from the email, contact, threads, and tags domains. For all datasets from the coauth domain, we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

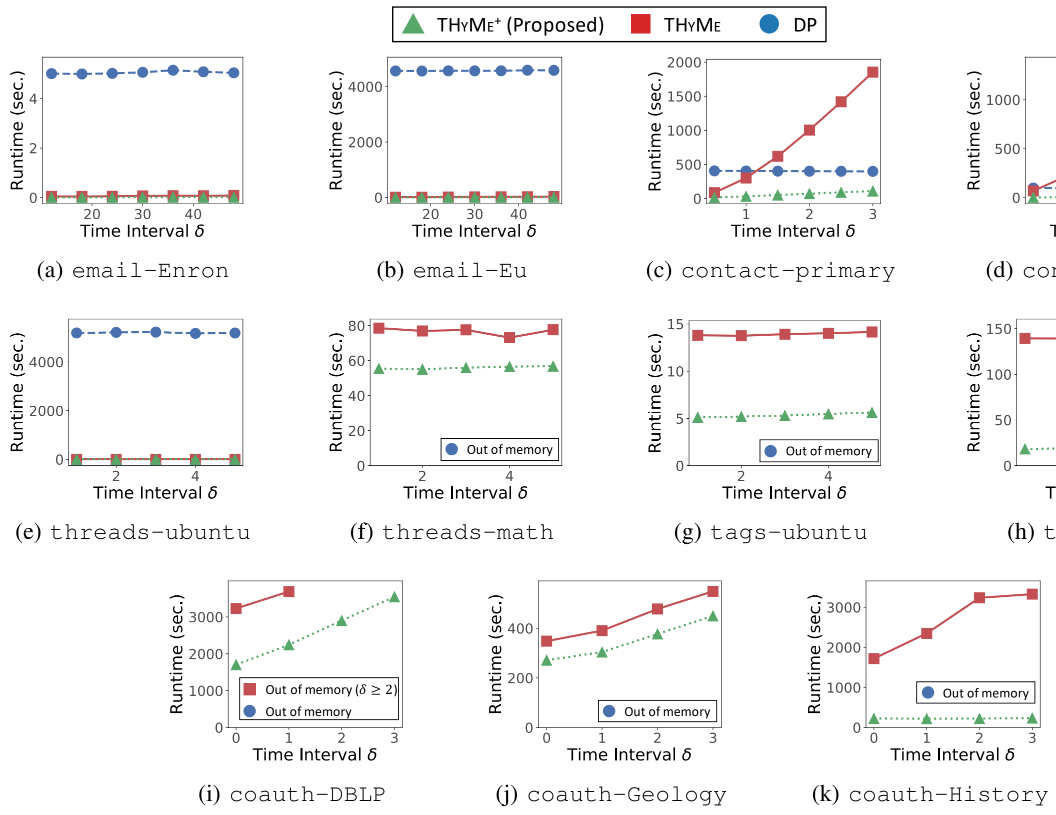
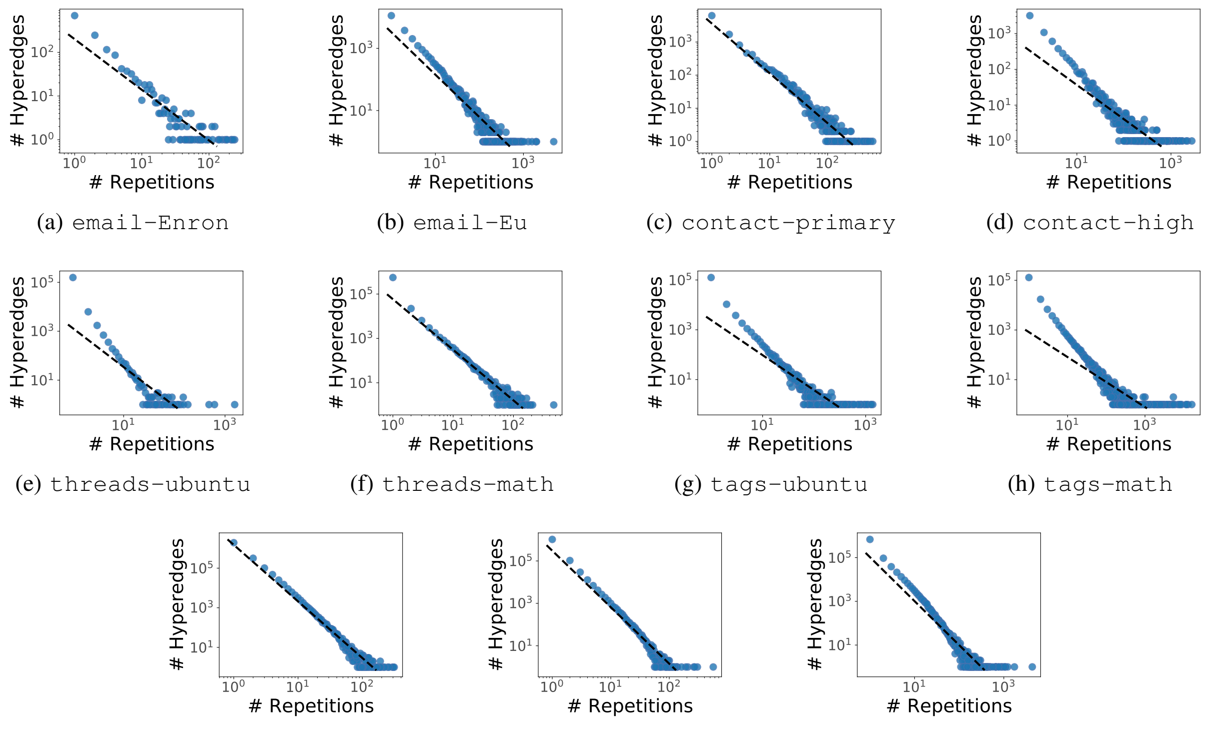


Fig. 2: THYME$^+$ is faster and more space efficient than DP and THYME.

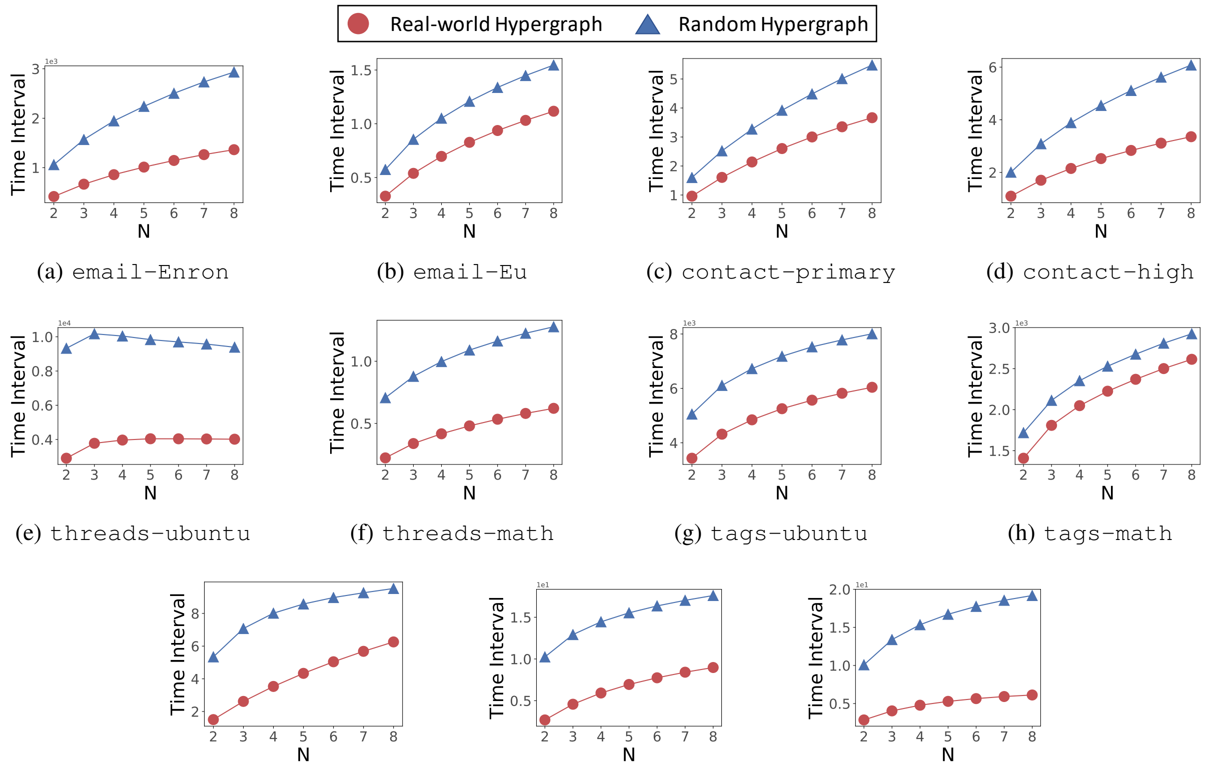


(a) `email-Enron` (b) `email-Eu` (c) `contact-primary` (d) `contact-high`

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

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

Fig. 3: 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 I.



(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. 4: 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.