Faster and Generalized Temporal Triangle Counting, via Degeneracy Ordering

Noujan Pashanasangi
npashana@ucsc.edu
University of California, Santa Cruz
Santa Cruz, California, USA

C. Seshadhri
sesh@ucsc.edu
University of California, Santa Cruz
Santa Cruz, California, USA

ABSTRACT

Triangle counting is a fundamental technique in network analysis, that has received much attention in various input models. The vast majority of triangle counting algorithms are targeted to static graphs. Yet, many real-world graphs are directed and *temporal*, where edges come with timestamps. Temporal triangles yield much more information, since they account for both the graph topology and the timestamps.

Temporal triangle counting has seen a few recent results, but there are varying definitions of temporal triangles. In all cases, temporal triangle patterns enforce constraints on the time interval between edges (in the triangle). We define a general notion $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles that allows for separate time constraints for all pairs of edges.

Our main result is a new algorithm, DOTTT (Degeneracy Oriented Temporal Triangle Totaler), that exactly counts all directed variants of $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles. Using the classic idea of degeneracy ordering with careful combinatorial arguments, we can prove that DOTTT runs in $O(m\kappa \log m)$ time, where m is the number of (temporal) edges and κ is the graph degeneracy (max core number). Up to log factors, this matches the running time of the best static triangle counters. Moreover, this running time is better than existing.

DOTTT has excellent practical behavior and runs twice as fast as existing state-of-the-art temporal triangle counters (and is also more general). For example, DOTTT computes all types of temporal queries in Bitcoin temporal network with half a billion edges in less than an hour on a commodity machine.

CCS CONCEPTS

• **Theory of computation → Graph algorithms analysis**; • **Information systems →** *Data mining*; *Social networks*; • **Mathematics of computing → Graph algorithms**.

KEYWORDS

triangle counting; temporal networks; degeneracy

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467374

ACM Reference Format:
Noujan Pashanasangi and C. Seshadhri. 2021. Faster and Generalized Temporal Triangle Counting, via Degeneracy Ordering. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3447548.3467374

1 INTRODUCTION

Triangle counting is a fundamental problem in network analysis. There has been a rich line of work on counting triangles in graphs [2, 3, 15, 22, 43]. The triangle counts appear in form of different parameters such as *clustering coefficient* [55] and *transitivity ratios* [54]. Triangle counting has many applications such as social networks analysis [36], indexing graph databases [18], community discovery [32], and spam detection [4].

Much of the rich history of triangle counting has focused on static graphs. But many real-world networks, such as communication networks, message networks, and social interaction networks are fundamentally *temporal*. Every edge has an associated timestamp [11, 12, 20]. We can model these attributed networks as *temporal networks* where edges have timestamp. For example, in cryptocurrency transaction networks and email networks, each link is between a sender and a receiver and has a timestamp that could be represented as a directed edge with a timestamp in a temporal network.

Temporal triangle counts provide a far richer set of counts than standard counts. These counts take into account the temporal ordering of edges in a triangle, and potentially impose constraints on the timestamp difference among edges. Temporal triangle and motif counting has applications in graph representation learning [48], expressivity of graph neural networks (GNNs) [7], network classification [47], temporal text network analysis [52], computer networks [50], and brain networks [9]. Recently, there has been significant interest in temporal triangle and motif counting algorithms [6, 7, 25, 27, 35, 46–48, 53].

Counting temporal triangles in (directed) temporal networks introduces new challenges to that of triangle counting in static graphs. The first challenge is actually defining types of temporal triangles (or motifs). In essence, all definitions specify constraints on the time difference between edges of a triangle. For example, Kovanen et al. [20] restrict temporal triangles to cases where the gap between two consecutive edges in the temporal ordering is at most Δ time unit, and the two edges incident to each node are consecutive event of that node. Paranjape-Benson-Leskovec (henceforth PBL) introduce *δ-temporal triangles*, where all edges of the triangle/motif have to occur within δ timesteps [33]. These varying definitions necessitate different algorithms. Our first motivating question is

whether one can design algorithms for a more general notion of temporal triangles.

Secondly, there is a significant gap between the best static triangle counting algorithms and temporal triangle counters. Specifically, the classic and immensely practical triangle counting algorithm of Chiba-Nishizeki runs in time $O(m\kappa)$, where m is the number of edges and κ is the *graph degeneracy* (or max core number). The current state-of-the-art temporal triangle counting algorithm of PBL runs in $O(m\sqrt{\tau})$ time, where τ is the total triangle count (of the underlying static graph). There is a large gap between κ (which is typically in the hundreds and thought of as a constant) and τ (which is superlinear in m).

These twin issues motivate our study. *Can we define a more general notion of temporal triangles, and give an algorithm whose asymptotic running time is closer to that of static triangle counting?*

1.1 Problem Description

The input is a directed temporal graph $T = (V, E)$. Each edge is a tuple of the form (u, v, t) where u and v are vertices in the temporal graph, and t is a timestamp. For notational convenience, we assume all timestamps in a temporal network are unique integers.

We introduce our notion of $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles.

Definition 1.1. Let $e_1 = (u_1, v_1, t_1), e_2 = (u_2, v_2, t_2)$, and $e_3 = (u_3, v_3, t_3)$, be three directed temporal edges where the induced static graph on them is a triangle, and $t_1 < t_2 < t_3$.

(e_1, e_2, e_3) is a $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-*temporal triangle* if $t_2 - t_1 \leq \delta_{1,2}$, $t_3 - t_2 \leq \delta_{2,3}$, and $t_3 - t_1 \leq \delta_{1,3}$.

Thus, we specify timestamp differences between *every* pair of edges. When one also considers the direction of edges, there exist eight different types of temporal triangles as shown in Fig. 2. These types are distinguished by temporal ordering of edges and their direction. Thus, for any choice of $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$, there are eight different types of temporal triangles (one corresponding to each figure in Fig. 2).

We observe that the notion in Definition 1.1 subsumes most existing temporal triangle definitions. Specifically, a $(\delta_{1,3}, \delta_{1,3}, \delta_{1,3})$-temporal triangle becomes a $\delta_{1,3}$-temporal triangles as defined in PBL [33]. Temporal triangles with respect to the temporal motif definition by by Kovanen et al. in [20] consider timestamp differences between consecutive edges in temporal ordering. By our definition, $(2\Delta, \Delta, \Delta)$-temporal triangles capture these types of temporal triangles. Although, the definition in [20] is more restrictive and requires that all edges incident to a node are consecutive events of that node. Most existing temporal triangle counting literature uses these definitions [25, 27, 46, 53].

We describe a simple example to see how Definition 1.1 offers richer temporal information. Let us measure time in hours, so $(2, 1, 1)$-temporal triangle is one where the first and second edge (of the triangle) are at most 1 hour apart, and similarly for the second and third edge. Now consider $(1.5, 1, 1)$-temporal triangles. The time gap between the first and second edge (as well as the second and third) is again 1 hour, but the entire triangle must occur within 1.5 hours. There is a significant difference between these cases, but previous definitions of temporal triangles would not distinguish these.

We note that more general temporal motifs, beyond triangles, have been defined. Yet, to the best of our knowledge, most fast algorithms that scale to millions of edges have been designed for triangles. Paranjape et al. specialized algorithm for 3-edge triangle motifs (temporal triangles) is up to 56x faster than their general motif counting algorithm [33]. Our focus was on scalable algorithms, and hence, on triangle counting. We believe that generalizing Definition 1.1 (and our DOTTT algorithm) for general motifs would be compelling future work.

1.2 Main Contributions

Our main result is the *Degeneracy Oriented Temporal Triangle Totaler* algorithm, DOTTT that counts $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles as defined in Definition 1.1. The running time is only a logarithmic overhead over static triangle counting. We detail our contributions below.

Theoretically bridging gap between temporal and static triangle counting: Our main theorem is the following.

Theorem 1.2. *Given $\delta_{1,3}, \delta_{1,2}$ and $\delta_{2,3}$, the* DOTTT *algorithm exactly counts each of the eight types of $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles (Fig. 2) in a temporal graph in $O(m\kappa \log m)$ time. (Here, m is the total number of temporal edges, and κ is the degeneracy of the underlying static graph.)*

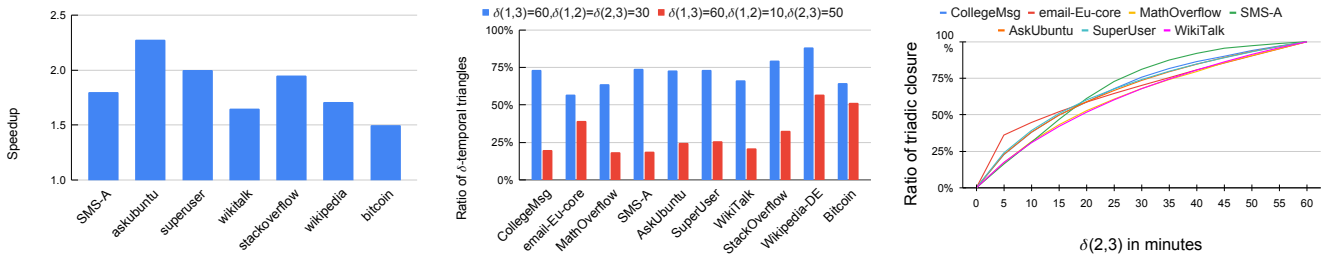
Observe that, up to a logarithmic factor, our theoretical running time for temporal triangle counting matches the $O(m\kappa)$ bound for static triangle counting. As mentioned earlier, the previous best bound was $O(m\tau^{1/2})$. We stress that there is no dependence on the time intervals $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$.

The idea of degeneracy orientations is tailored to static graphs, and one of our contributions is to show it can help for temporal triangle counting. A key insight in DOTTT is to process (underlying) static edges in the exact order of the Chiba-Nishizeki algorithm, but carefully consider neighboring edges to capture all temporal triangles. By a non-trivial combinatorial analysis, we can prove that number of times that a temporal edge is processed is upper bounded by κ. We need additional data structure tricks to get the counts efficiently, leading to an extra logarithmic factor.

Excellent practical behavior of DOTTT**:** DOTTT consistently determines temporal triangle counts in less than ten minutes for datasets with tens of millions of edges. We only use a single commodity machine with 64GB memory, without any parallelization. We directly compare DOTTT with the state-of-the-art PBL algorithm. Our algorithm is consistently faster, and as illustrated in Fig. 1a we typically get a factor 1.5 speedup for larger graphs. (We note that DOTTT can count a more general class of temporal triangles.)

We note that for the largest dataset in our experiments, Bitcoin (515.5M edges), DOTTT only uses 64GB memory and runs in less than an hour, while existing methods ran out of memory (details in §6).

Richer triadic information from $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles: We demonstrate how $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles can give a richer network analysis method. Consider Fig. 1b. For a collection of temporal datasets, we generate the counts of (1 hr, 30 min, 30 min)-temporal triangle counts, as well as those for (1 hr, 10 min, 50 min)-temporal triangles. We plot these numbers as a ratio of (1 hr, 1 hr, 1 hr)-temporal triangles. Across the datasets, the ratios



(a) Speedup **(b) Expressivity** **(c) Triadic closure over time**

Figure 1: (a): The Speedup of DOTTT for counting $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles over the PBL algorithm for counting $\delta_{1,3}$-temporal triangles. (b): We fix $\delta_{1,3}$ to 1 hr. Blue bars show the ratio of (1 hr, 30 mins, 30 mins)-temporal triangles to (1 hr, 1 hr, 1 hr)-temporal triangle. The red bars illustrate the ratio for the case of (1 hr, 10 mins, 50 mins)-temporal triangles and is more restrictive. (c): We fix $\delta_{1,3} = 2$ hrs and $\delta_{1,2} = 1$ hr. At t we plot the ratio of (2 hrs, 1 hr, t)-temporal triangles to (2 hrs, 1 hr, 1hr)-temporal triangles.

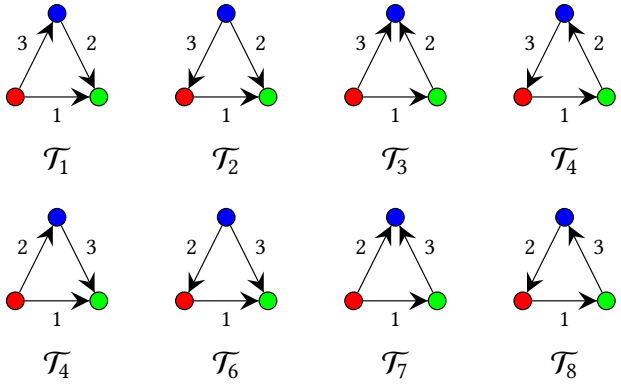


Figure 2: All possible temporal triangle types. The start point of the first edge (in temporal ordering of edges) is shown in red and the end point in green.

are at most 75%. The red bars are typically at most 25%, showing the extra power of Definition 1.1 in distinguishing temporal triangles.

We note here that for each dataset, DOTTT has the same running time for obtaining the counts for (1 hr, 30 min, 30 min)-temporal triangle and (1 hr, 10 min, 50 min)-temporal triangles, as it has no dependency on the time intervals $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$.

An interesting study is presented in Fig. 1c. The transitivity and clustering coefficients are fundamental quantities of study in network science. In temporal graphs, in addition to these measure, the time it takes for a wedge (2-path) to close could also be of importance. (Zingnani et al. proposed the triadic closure delay metric that capture the time delay between when a triadic closure is first possible, and when they occur [58].) In Fig. 1c, we fix $\delta_{1,3}$ = 2 hrs and $\delta_{1,2}$ = 1 hr. We then vary $\delta_{2,3}$ from zero to 60 minutes, and plot the ratio of $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles to (2hrs, 1hr, 1hr)-temporal triangles. We can see the trends in triadic closure with respect to the time for the third edge. We observe that, by and large, half the triangles are formed within 20 minutes of the first two edges appearing. And by 30 minutes, almost 75% of these triangles are formed. These are examples of triadic analyses enabled by DOTTT.

1.3 Main challenges

In a temporal graph, the number of temporal edges is typically two to three times the number of underlying static edges. Since most triangle counting algorithms are based on some form of wedge enumerations, this leads to a significant increase in the number of edges. One method used for temporal triangle counting is to simply prune the temporal edges based on the time period [27, 45]. But such algorithms have a dependency on the time period and are inefficient for large time periods.

Another significant challenge is the multiplicity of an individual edge can be extremely large. The same edge often occurs many hundreds to thousands of times in a temporal network (in the BitCoin network, there is an edge appearing 447K times Tab. 2). These edges create significant bottlenecks for enumeration methods. It is not clear how efficient methods on the underlying static graphs (which ignores multiplicities) can help with this problem. Triangle counting often works by finding a wedge (2-path) and checking for the third edge. With multiple temporal edges between the same pair of vertices, this method requires many edge lookups. Paranjape et al. used a clever idea to process edges on a pair of vertices $O(\tau^{1/2})$ times. The challenge is to bound it by the degeneracy of the graph.

The time constraints expressed by $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles create additional challenges. A clever wedge enumeration exploiting the degeneracy may produce wedges containing the first and second edges of the triangle, the first and third, or the second and third. This makes the lookup (or counting) of possible "matches" for the remaining edge challenging, since it appears we need to look at all multiple edges. On the other hand, if we enumerated wedges that only involved the first and second edge, we cannot benefit for the efficiencies of degeneracy-based methods. Some of these problems can be circumvented for (δ, δ, δ)-temporal triangles, but the general case is challenging.

Overall, we can state the main challenge as follows. Fast triangle counting methods (such as degeneracy based methods) necessarily ignore time constraints while generating wedges, making it hard to look for the "closing" edge. On the other hand, a method that exploits the timestamps by (say) pruning cannot get the efficiency

gains of degeneracy based methods. One of the insights of DOTTT is a resolution of this tension.

2 RELATED WORK

There is rich history of work on triangle counting in static graphs. Various algorithm for triangle and motif counting in attributed graphs have also been proposed [13, 30, 37, 41, 42, 56]. Here we only focus on temporal networks and refer the reader to [2] and the tutorial [44] for a more detailed list of related work.

Graph orientation, in particular degeneracy ordering, is a classic idea in counting triangles and motifs in static graphs, pioneered by Chiba-Nizhizeki [8]. Recently, there has been a number of triangle counting and motif counting algorithms inspired by these techniques [10, 16, 17, 31, 34, 38]. The main benefit of degeneracy ordering is that the out-degree of each vertex becomes small when we orient the static graph based on this ordering.

Kovanen et al. called two temporal edges ΔT-adjacent if they share a vertex and the difference of their timestamps are at most ΔT [20]. In their definition of temporal motifs, temporal edges must represent consecutive events for a node. Redmond et al. gave an algorithm for counting δ-temporal motifs but their algorithm does not take the temporal ordering of edges into account [39], and only counts motifs where incoming edges occur before outgoing edges. Gurukar et al. present a heuristic for counting temporal motifs [14].

More related to our work, Paranjape-Benson-Leskovec defined the δ-temporal motifs where all edges occur inside a time period δ and also the temporal ordering of edges are taken into account [33]. They gave a general algorithm for counting k-node ℓ-edge motifs in temporal networks. The main idea behind their algorithm is a moving time window of size δ over the sequence of all temporal edges for each static motif matching the underlying static motif of the temporal motif of interest. For temporal triangles, their algorithm runs in $O(\tau m)$ time where τ is the number of triangles in the underlying static graph of the input temporal graph T, as it might enumerate temporal edges on static edges with high multiplicity $O(\tau)$ time. They also presented a specialized, more efficient algorithm for counting 3-edge temporal triangles that runs in time $O(\tau^{1/2}m)$. We call their algorithm *PBL* and use it as our baseline.

Mackey et al. presented a backtracking algorithm for counting δ-temporal motifs that maps edges of the motif to the edges of the host graph one by one in temporal (chronological) ordering. For each edge, it only searches through edges that occur in the correct temporal ordering and respect the time gap restriction. [27]. Unlike the PBL algortihm and ours, this algorithm could be inefficient for large values of δ as its runtime depends on the value of δ.

Liu et al. [26] introduced a comparative survey of temporal motif models. Boekhout et al. gave an algorithm for counting δ-temporal multi-layer temporal motifs [6]. Li et al., developed an algorithm for counting temporal motifs in heterogeneous information networks [24]. Petrovic et al. gave an algorithm for counting causal paths in time series data on networks [35].

There has also been recent progress on approximating the counts of temporal motifs and triangles [46, 53]. Particularly , Liu et al. presented a sampling framework for approximating the counts of $\delta_{1,3}$-temporal motifs [25].

3 PRELIMINARIES

The input graph is a directed temporal graph that we denote by $T(V, E)$. Let $|V| = n$ and $|E| = m$. Temporal graph T is presented as a collection of m directed temporal edges $e = (u, v, t)$ where $u, v \in V$, and t is the timestamp for edge e where $t \in \mathbb{R}$. We use $t(e)$ to denote the timestamp of a temporal edge e. Note that there could be multiple temporal edges on the same pair of nodes. We assume that all the timestamps in T are unique. This assumption leads to the clean definition of different types of temporal triangles (Fig. 2), but is not a necessity of our algorithm. To be more specific, our algorithm also works for temporal graph including temporal edges with equal timestamp.

We denote the underlying undirected static graph of T as $G = (V, E_s)$ and put $|E_s| = m_s$. Two vertices in the static graph G are connected if there is at least one temporal edge between them. Formally, $E_s = \{\{u, v\} \mid \exists t : (u, v, t) \in E \vee (v, u, t) \in E\}$. For $v_1, v_2 \in V$, let $\sigma((v_1, v_2))$ denote the temporal multiplicity, that is the number of temporal edges on $\{v_1, v_2\}$ directed from v_1 to v_2.

As shown in Fig. 2, there are eight different types of temporal triangles. The time restrictions $\delta_{1,3}$, $\delta_{1,2}$, and $\delta_{2,3}$ is not involved in definition of these types and could be applied to each of them. Note that these different types account for all possible ordering of temporal edges in the triangle in addition to their directions.

In a directed graph G, we use $N^+(v) : \{(v, u) \in E(G)\}$ to denote the out-neighborhood and $N^-(v) : \{(u, v) \in E(G)\})$ to denote the in-neighborhood of a vertex v. For a vertex $v \in V(G)$, we define out-degree as $d^+(v) = |N^+(v)|$ and in-degree as $d^-(v) = |N^-(v)|$.

Vertex ordering is a central idea in triangle counting and motif analysis in general [5, 8, 16, 31, 34, 38, 49]. Let G be an undirected static simple graph. Given any ordering $\prec$ of $V(G)$, we can construct a DAG $G_\prec$ by orienting each edge $\{u, v\} \in E(G)$ from u to v iff $u \prec v$. Next we define *degeneracy* and *degeneracy ordering* formally.

Definition 3.1. The degeneracy of a graph G, denoted by κ, is the smallest integer k such that there exists an ordering $\prec$ of $V(G)$ where $d^+_v(G_\prec) \leq k$ for each $v \in V(G)$.

Definition 3.2. Degeneracy ordering of an undirected simple static graph G is obtained by removing a vertex with minimum degree repeatedly. The order of the removal of vertices is the degeneracy ordering of G.

There is an algorithm for finding the degeneracy ordering of a graph G in $O(|E(G)|)$ time [28]. Let $\prec$ denote the degeneracy ordering.

4 MAIN IDEAS

Our algorithm first enumerates static triangles in G, the underlying static graph of the input temporal graph T. Let $\{u, v, w\}$ be a static triangle. We consider all possible temporal orderings as shown in Fig. 3, and all possible orientations as shown in Fig. 4, for a temporal triangle corresponding to $\{u, v, w\}$. A temporal ordering and a temporal orientation together determine the type of the temporal triangle. For example π_1 and ρ_8 correspond to $\mathcal{T}_1$. Tab. 1 lists all possible pairs of temporal ordering and orientations and their corresponding type of temporal triangle as a function ψ.

We store the input temporal edges of the input temporal graph T in a data structure in the CSR format. Thus, we can assume that

we have constant time access to temporal edges on each pair of vertices for each direction in the order of increasing timestamps. Let π denote the temporal ordering, and ρ denote the orientation for which we want to count the temporal triangles. In this section, from here we only consider temporal edges that follow the orientation ρ.

Assume that the timestamps of two of the edges of a temporal triangle corresponding to the static triangle $\{u, v, w\}$ is given. WLOG, assume that these temporal edges correspond to $\{u, v\}$ and $\{u, w\}$. We can use a binary search to find the number of temporal edges on the pair $\{v, w\}$ that are compatible with the two given temporal edges. Note that compatibility of timestamps is determined by the timestamp of edges and the temporal ordering π. Thus, all we need is to enumerate all possible pairs of temporal edges on $\{u, v\}$ and $\{u, w\}$. This could be an expensive enumeration if both these static edges have high multiplicity of temporal edges.

We show that we can obtain the counts of temporal triangles on $\{u, v, w\}$ without enumeration of all possible pairs of temporal edges on $\{u, v\}$ and $\{u, w\}$. Let $e_1, \ldots, e_{\sigma(\rho(\{u,v\}))}$ denote the sequence of temporal edges in the order of increasing timestamp on $\{u, v\}$, and $e'_1, \ldots, e'_{\sigma(\rho(\{u,w\}))}$ denote that of $\{u, w\}$. We enumerate each of these sequences of temporal edges separately, and for each edge we store cumulative counts of compatible temporal edge on $\{v, w\}$. In other words, for each edge e in these two sequence, we store the counts of edges e_3 on $\{v, w\}$ that are compatible with e or any other temporal edge in the same sequence with a smaller timestamp.

Then we enumerate the temporal edges on $\{u, v\}$, and for each edge e_i we use binary search to find the sequence of temporal edges $e'_j, \ldots, e'_k$, in increasing order of timestamp, on $\{v, w\}$ that are compatible with e_i. We use the cumulative counts of compatible edges on $\{v, w\}$ that we stored for e_i, e'_k, and e'_j to compute the counts of all temporal triangles on $\{u, v, w\}$ that include e_i.

Although we avoid the enumeration of pairs of temporal edges on $\{u, v\}$ and $\{u, w\}$, our algorithm could still be inefficient. The reason is that static edges $\{u, v\}$ could have high multiplicity of temporal edges and also participate in a large number of static triangles. Here is where we use the power of vertex ordering and graph orientation techniques.

DOTTT enumerates static triangles in $G_\prec$, and when processing a static triangle $\{u, v, w\}$ where u comes first in the degeneracy ordering $\prec$ of G, it only enumerates temporal edges on $\{u, v\}$ and $\{u, w\}$. Thus, each temporal edge on a pair $\{x, y\}$ where $x \prec y$, is processed only for static triangles where the third vertex is in the out-neighborhood of x. But we know that the out-degree of each vertex is bounded by κ in $G_\prec$. Therefore, each such temporal edge on $\{x, y\}$ is processed $O(\kappa)$ times.

5 OUR MAIN ALGORITHM

In this section we describe our algorithm for getting $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles counts. Let $T = (V, E)$ be the input directed temporal graph given as a list of temporal edges sorted by timestamps. Although not necessary for our algorithm, assuming that edges are given in increasing order of timestamp is common in temporal networks as the edges are recorded in their order of occurrence [33].

We first extract the static graph $G(V, E_s)$ from T. Then, we obtain the degeneracy ordering of G, denoted by $\prec$ using the algorithm by Matula and Beck [28], and orient the edges of G with respect to $\prec$ to get the DAG $G_\prec$. We start by enumerating static triangles in $G_\prec$. This can be done in $O(m_s\kappa)$ where κ is the degeneracy of G [8, 38].

Note that all triangles in $G_\prec$ are acyclic as $G_\prec$ is a DAG, so each triangle in G correspond to an acyclic triangle in $G_\prec$. In order to enumerate all triangles in G, we enumerate all directed edges in $G_\prec$, and for each directed edge (u, v) we enumerate $N^+(u)$. For each vertex $w \in N^+(u)$, we check whether $\{u, v, w\}$ is a triangle by checking the existence of an edge between v and w.

We call vertex u in a static triangle $\{u, v, w\}$ the *source vertex* if $u \prec v$ and $u \prec w$. Let $\{u, v, w\}$ be the triangle being processed while enumerating triangles in $G_\prec$. WLOG, assume u is the source vertex

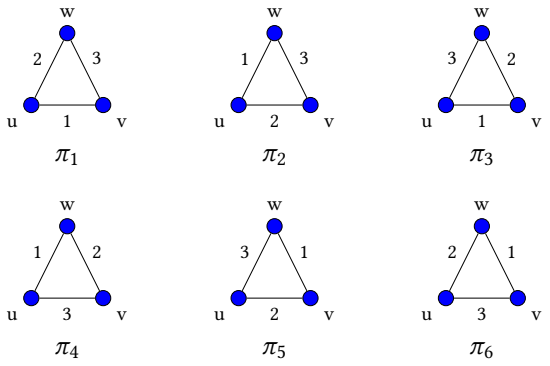


Figure 3: All possible ordering of temporal edges of a temporal triangle corresponding to a static triangle $\{u, v, w\}$.

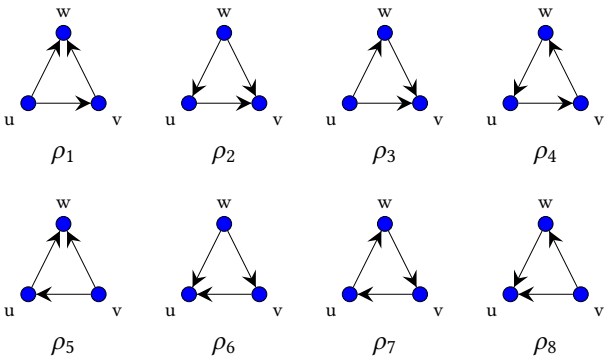


Figure 4: All possible orientations of temporal edges of a temporal triangle corresponding to a static triangle $\{u, v, w\}$.

Table 1: Conversion from temporal ordering and orientation to temporal triangle type.

ψ	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6	ρ_7	ρ_8
π_1	$\mathcal{T}_7$	$\mathcal{T}_6$	$\mathcal{T}_5$	$\mathcal{T}_8$	$\mathcal{T}_3$	$\mathcal{T}_2$	$\mathcal{T}_4$	$\mathcal{T}_1$
π_2	$\mathcal{T}_5$	$\mathcal{T}_3$	$\mathcal{T}_7$	$\mathcal{T}_4$	$\mathcal{T}_6$	$\mathcal{T}_1$	$\mathcal{T}_8$	$\mathcal{T}_2$
π_3	$\mathcal{T}_3$	$\mathcal{T}_2$	$\mathcal{T}_1$	$\mathcal{T}_4$	$\mathcal{T}_7$	$\mathcal{T}_6$	$\mathcal{T}_8$	$\mathcal{T}_5$
π_4	$\mathcal{T}_1$	$\mathcal{T}_7$	$\mathcal{T}_3$	$\mathcal{T}_8$	$\mathcal{T}_2$	$\mathcal{T}_5$	$\mathcal{T}_4$	$\mathcal{T}_6$
π_5	$\mathcal{T}_6$	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_8$	$\mathcal{T}_5$	$\mathcal{T}_3$	$\mathcal{T}_4$	$\mathcal{T}_7$
π_6	$\mathcal{T}_2$	$\mathcal{T}_5$	$\mathcal{T}_6$	$\mathcal{T}_4$	$\mathcal{T}_1$	$\mathcal{T}_7$	$\mathcal{T}_8$	$\mathcal{T}_3$

in $\{u, v, w\}$. Thus, the number of times we visit $\{u, v\}$ or $\{u, w\}$ in a static triangle are limited by $d^+_{G_\prec}(u)$ that is bounded by κ. But the number of times we visit the static edge $\{v, w\}$ is not bounded by κ, so we want to avoid enumerating temporal edges on $\{v, w\}$. Next, we show how to count the number of $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles corresponding to the static triangle $\{u, v, w\}$.

We define the temporal ordering of a temporal triangle corresponding to a static triangle $\{u, v, w\}$ as a mapping $\pi : \{1, 2, 3\} \to \{\{u, v\}, \{u, w\}, \{v, w\}\}$. There are six different possible temporal orderings as shown in Fig. 3.

We define the orientation of a temporal triangle corresponding to a static triangle $\{u, v, w\}$ as a mapping ρ from each pair of vertices of $\{u, v, w\}$ to one of the two possible ordered pairs of the same pair of vertices. For example, $\rho_1(\{u, v\}) = (u, v)$ for ρ_1 in Fig. 4. The orientation of a temporal triangle simply determines the direction of its temporal edges. Each such temporal edge can take two possible directions, so there are eight types of orientation such a temporal triangle can take as shown in Fig. 4. Note that orientation of a temporal triangle is independent of its temporal ordering.

It is easy to see that the temporal ordering and orientation determine the type of the temporal triangle. But different combinations of temporal orderings and orientation could result in the same type. The temporal triangle type for all possible pairs of temporal ordering and orientation are shown in Tab. 1.

For a temporal ordering π and for $i \in \{1, 2, 3\}$, we use $S_i(\pi, \rho)$ to denote the sequence of temporal edges between the pair of vertices $\pi(i)$ that have the direction $\rho(\pi(i))$, in sorted order of timestamp. When π and ρ are clear from the context, we use S_i instead of $S_i(\pi, \rho)$. We assume that we have access to S_1, S_2, and S_3 in constant time. Let σ_i denote the length of S_i. We use $S_i[\ell]$ to denote the ℓ-th edge in the sequence S_i, and $S_i[\ell : \ell']$ to denote the consecutive subsequence of S_i ranging from $S_i[\ell]$ to $S_i[\ell']$.

For a sequence S of temporal edges in increasing order of timestamp and timestamps t and t' where $t \le t'$, let $\mathrm{EC}([t, t'], S)$ denote the number of edges in S with a timestamp in the time window $[t, t']$. For given $\delta_{1,3}, \delta_{1,2}$, and $\delta_{2,3}$, let $\mathrm{TTC}(\{u, v, w\}, \pi, \rho)$ denote the number of $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles corresponding to the static triangle $\{u, v, w\}$, temporal ordering π, and orientation ρ.

Lemma 5.1. *For a static triangle $\{u, v, w\}$, a temporal ordering π, and an orientation ρ,*

$$\mathrm{TTC}(\{u, v, w\}, \pi, \rho) = \sum_{e_2 \in S_2} \sum_{\substack{e_1 \in S_1 \\ t(e_1) \in [t(e_2) - \delta_{1,2}, t(e_2))}} \mathrm{EC}([t(e_2), \min(t(e_2) + \delta_{2,3}, t(e_1) + \delta_{1,3})], S_3)$$

Proof. If temporal edge $e_1 \in S_1$ is in a $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangle with edge $e_2 \in S_2$, then $t(e_1) \in [t(e_2) - \delta_{1,2}, t(e_2))$. Fix a pair of temporal edges (e_1, e_2) in $S_1 \times S_2 = \{(e_1, e_2) \mid e_1 \in S_1 \wedge e_2 \in S_2\}$ where $t(e_2) \in [t(e_1), t(e_1) + \delta_{1,2}]$. A temporal edge $e_3 \in S_3$ composes a $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangle with e_1 and e_2 iff $t(e_3) \in [t(e_2), \min(t(e_2) + \delta_{2,3}, t(e_1) + \delta_{1,3})]$. □

For a triangle $\{u, v, w\}$ where u is the source vertex, we divide all six possible temporal orderings into three categories based on the place of $\{v, w\}$ in them. Recall that we want to avoid enumerating temporal edges on $\{v, w\}$. In π_1 and π_2, $\{v, w\}$ is assigned to the third place. $\{v, w\}$ is assigned to the second place in π_3 and π_4, and finally to the first place in temporal ordering π_5 and π_6.

Temporal orderings π_1 and π_2: Using Lemma 5.1, one can compute $\mathrm{TTC}(\{u, v, w\}, \pi, \rho)$ by enumerating pairs of temporal edges in $S_1 \times S_2 = \{(e_1, e_2) \mid e_1 \in S_1 \wedge e_2 \in S_2\}$. For each pair we compute $\mathrm{EC}([t(e_2), \min(t(e_2) + \delta_{2,3}, t(e_1) + \delta_{1,3}), S_3)$ using binary search. To get the final counts we sum $\mathrm{EC}([t(e_2), \min(t(e_2) + \delta_{2,3}, t(e_1) + \delta_{1,3}), S_3)$ over all pairs $(e_1, e_2) \in S_1 \times S_2$. But enumerating $S_1 \times S_2$ could be expensive and this process overall runs in time $O(\sigma_1 \sigma_2 \log(\sigma_3))$. Next, we show that we can compute the same count by enumerating edges in S_1 and S_2 separately and storing cumulative counts of compatible edges on S_3 for each edge.

For $i, j \in \{1, 2, 3\}$ where $i \ne j$, and $\ell, \ell' \in \{1, \ldots, \sigma_i\}$ where $\ell \le \ell'$ we use $\mathrm{CEC}_{+\delta_{1,3}}(S_i[\ell : \ell'], S_j)$ to denote the cumulative count of edges in S_j with a timestamp in $[t(e), t(e) + \delta_{1,3}]$ for edges e in the sequence $S_i[\ell : \ell']$. Formally

$$\mathrm{CEC}_{+\delta_{1,3}}(S_i[\ell : \ell'], S_j) = \sum_{\ell \le r \le \ell'} \mathrm{EC}([t(S_i[r]), t(S_i[r]) + \delta_{1,3}], S_j).$$

Cumulative counts CEC_{∞}, $\mathrm{CEC}_{-\delta_{1,3}}$, and $\mathrm{CEC}_{-\infty}$, are defined the same way with time intervals $[t(e), \infty)$, $[t(e) - \delta_{1,3}, t(e)]$, and $(-\infty, t(e)]$, respectively. $\mathrm{CEC}_{-\delta_{1,2}}$, $\mathrm{CEC}_{+\delta_{1,2}}$, $\mathrm{CEC}_{-\delta_{2,3}}$, and $\mathrm{CEC}_{+\delta_{2,3}}$ are defined similarly. Note that we can compute $\mathrm{CEC}(S_i[1 : \ell], S_j)$ for each $\ell \in \{1, \ldots, \sigma_i\}$ with one pass over S_i, and once we have these counts, we can get the cumulative counts $\mathrm{CEC}(S_i[\ell' : \ell''], S_j)$, for each consecutive subsequence $S_i[\ell' : \ell'']$ of S_i as follows.

$$\begin{aligned}\mathrm{CEC}_{+\delta_{1,3}}(S_i[\ell' : \ell''], S_j) = {} & \mathrm{CEC}_{+\delta_{1,3}}(S_i[1 : \ell''], S_j) \\ & - \mathrm{CEC}_{+\delta_{1,3}}(S_i[1 : \ell' - 1], S_j).\end{aligned}$$

where $\mathrm{CEC}_{+\delta_{1,3}}(S_i[1 : 0], S_j) = 0$.

We first enumerate edges in S_1. For each edge $e_1 \in S_1$ we compute $\mathrm{CEC}_{+\delta_{1,3}}(S_1[1 : \ell], S_3)$ and $\mathrm{CEC}_{\infty}(S_1[1 : \ell], S_3)$ for each $\ell \in \{1, \ldots, \sigma_1\}$ and store them for e_1. Next, we enumerate edges in S_2 and compute $\mathrm{CEC}_{\infty}(S_2[1 : \ell], S_3)$ for each $\ell \in \{1, \ldots, \sigma_2\}$.

Fix an edge $e_2 \in S_2$. Let ℓ_f and ℓ_ℓ be the indices of the first and last edges in S_1 with a timestamp in $[t(e_2) - \delta_{1,2}, t(e_2)]$. Also let $\ell_{\delta_{2,3}}$ be the index of the last edge in S_1 with a timestamp at most $t(e_2) - \delta_{1,3} + \delta_{2,3}$. We can find ℓ_f, $\ell_{\delta_{2,3}}$, and ℓ_ℓ using a binary search on S_1. Note that $\delta_{1,3} \le \delta_{1,2} + \delta_{2,3}$, thus $\ell_f \le \ell_{\delta_{2,3}} \le \ell_\ell$.

First consider the temporal edges $S_1[i]$ where $\ell_f \le i \le \ell_{\delta_{2,3}}$. For any such edge $t(S_1[i]) + \delta_{1,3} \le t(e_2) + \delta_{2,3}$, so the timestamp of compatible edges in S_3 lie in the interval $[t(e_2), t(S_1[i]) + \delta_{1,3}]$. Having stored the cumulative counts described above, we can compute the number of pairs of temporal edges $(e_1, e_3) \in S_1[\ell_f : \ell_{\delta_{2,3}}] \times S_3$ that compose a $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangle on $\{u, v, w\}$ with e_2, complying with π_1 and ρ, as follows.

$$\begin{aligned}&\sum_{\ell \le i \le \ell_{\delta_{2,3}}} \mathrm{EC}([t(e_2), t(S_1(i)) + \delta_{1,3}], S_3) = \\ &\quad \mathrm{CEC}_{+\delta_{1,3}}(S_1[\ell_f : \ell_{\delta_{2,3}}], S_3) - \mathrm{CEC}_{\infty}(S_1[\ell_f : \ell_{\delta_{2,3}}], S_3) \\ &\quad + (\ell_{\delta_{2,3}} - \ell_f + 1) \cdot \mathrm{EC}([t(e_2), \infty), S_3)\end{aligned}$$

Now, we count the number of temporal edges in S_3 that compose a triangle with e_2 and $S_1[i]$, where $\ell_{\delta_{2,3}} < i \le \ell_\ell$. For a temporal edge $S_1[i]$ where $\ell_{\delta_{2,3}} < i \le \ell_\ell$, we have $t(S_1[i]) + \delta_{1,3} > t(e_2) + \delta_{2,3}$. Thus, there are $\mathrm{EC}([t(e_2), t(e_2) + \delta_{2,3}], S_3)$ edges on S_3 that compose a triangle with e_2 and $S_1[i]$. So the final count of pairs $(e_1, e_3) \in S_1 \times S_3$

that are in a temporal triangle with e_2 corresponding to the static triangle $\{u, v, w\}$ can be computed as follows.

$$\sum_{\substack{e_1 \in S_1, t(e_1) \in \\ [t(e_2)-\delta_{1,2}, t(e_2)]}} \mathrm{EC}([t(e_2), \min(t(e_2)+\delta_{2,3}, t(e_1)+\delta_{1,3})], S_3)$$
$$= \sum_{\ell_f \le i \le \ell_{\delta_{2,3}}} \mathrm{EC}([t(e_2), t(S_1(i))+\delta_{1,3}], S_3)$$
$$+(\ell_\ell - \ell_{\delta_{2,3}}) \cdot \mathrm{EC}([t(e_2), t(e_2)+\delta_{2,3}], S_3)$$

By Lemma 5.1, to get $\mathrm{TTC}(\{u, v, w\}, \pi, \rho)$, we only need to sum these counts over edges in S_2. Let $\langle u, v, w\rangle$ denote a static triangle where $u \prec v \prec w$. Alg. 1 formalizes the procedure described above for computing $\mathrm{TTC}(\langle u, v, w\rangle, \pi, \rho)$ where π is either π_1 or π_2.

Algorithm 1 Counting $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles corresponding to a static triangle and temporal orientation π_1 or π_2

1: **procedure** TTC-VW3($\delta_{1,3}$, $\delta_{1,2}$, $\delta_{2,3}$,$\langle u, v, w\rangle$, π, ρ)
▷ $\pi(\{v, w\}) = 3$
2: Enumerate S_1 and compute $\mathrm{CEC}_{+\delta_{1,3}}$ and CEC_{∞} on S_3.
3: count = 0
4: **for** $i = 1, \ldots, \sigma_2$ **do**
5: Let ℓ_f = LOWERBOUND($t(S_2[i]) - \delta_{1,2}, S_1$)
6: Let $\ell_{\delta_{2,3}}$ = UPPERBOUND($t(S_2[i]) - \delta_{1,3} + \delta_{2,3}, S_1$)
7: Let ℓ_ℓ = UPPERBOUND($t(S_2[i]), S_1$)
▷ Edges in $S_1[\ell_f : \ell_{\delta_{2,3}}]$
8: count += $\mathrm{CEC}_{+\delta_{1,3}}(S_1[\ell_f : \ell_{\delta_{2,3}}], S_3)$
9: count −= $\mathrm{CEC}_{\infty}(S_1[\ell_f : \ell_{\delta_{2,3}}], S_3)$
10: count += $(\ell_{\delta_{2,3}} - \ell_f + 1) \cdot \mathrm{EC}([t(S_2[i]), \infty), S_3)$
▷ Edges in $S_1[\ell_{\delta_{2,3}} + 1 : \ell_\ell]$
11: count += $(\ell_\ell - \ell_{\delta_{2,3}}) \cdot \mathrm{EC}([t(S_2[i]), t(S_2[i]) + \delta_{2,3}], S_3)$
12: **return** count

The algorithms for the category of π_3 and π_4 and the category of π_5 and π_6 are similar to Alg. 1, so we defer these to Appendix A. Here, we suffice to say that similar to Alg. 1, in algorithms for the remaining temporal orderings, for each static triangle $\{u, v, w\}$ where u is the source vertex, we only enumerate temporal edges on $\{u, v\}$ and $\{u, w\}$. And for each temporal edge we perform a constant number of binary searches on temporal edges on the other two static edges of the static triangle $\{u, v, w\}$.

5.1 Getting the Counts for All Temporal Triangle Types

Now that we can count $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles for each combination of temporal ordering and orientation, it only remains to get the counts for each temporal triangle type (Fig. 2). Let $\psi(\pi, \rho)$ denote the triangle type for π and ρ. Alg. 2 gets the counts for all eight types. Now, we can finally prove Theorem 1.2.

Proof of Theorem 1.2. Extracting the static graph G from T can be done in $O(m)$ time. We simply enumerate all temporal edges

Algorithm 2 Counting $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles for each temporal triangle types

1: **procedure** COUNT-TEMPORAL-TRIANGLES(T, $\delta_{1,3}$, $\delta_{1,2}$, $\delta_{2,3}$)
2: Extract the static graph G of T.
3: Find the degeneracy ordering $\prec$ of G.
4: Derive $G_\prec$ by orienting G with respect to $\prec$.
5: Initialize Counts to 0 for $\mathcal{T}_1, \ldots, \mathcal{T}_8$.
6: **for all** Static triangles $\{u, v, w\}$ **do**
▷ WLOG let $u \prec v \prec w$
7: **for all** Temporal ordering π and orientation ρ **do**
8: Counts($\psi(\pi, \rho)$) += TTC($\langle u, v, w\rangle, \pi, \rho$) ▷ Tab. 1
▷ Using Alg. 1, Alg. 3, and Alg. 4

of T and for each temporal edge $e = (v_1, v_2, t)$, we add an static edge between $\{v_1, v_2\}$ in G if they are not connected already. The degeneracy ordering of G could be obtained in $O(m_s)$ time [28], and $G_\prec$ could also be derived in time $O(m_s)$. For enumerating static triangles we first enumerate each edge in $G_\prec$. For each edge (u, v), we enumerate $N^+(u)$ which takes $O(\kappa)$ as $d^+_{G_\prec}(u) \le \kappa$. We can lookup if there is an edge between v and w in constant time. Thus, enumerating triangles take $O(m_s \cdot \kappa)$ time overall.

Note that for each static triangle we only enumerate temporal edges on static edges incident to the source vertex. So for the static triangle $\langle u, v, w\rangle$, we only enumerate temporal edges on the pairs $\{u, v\}$ and $\{u, w\}$. While processing a temporal edge during enumeration of temporal edges on $\{u, v\}$ or $\{u, w\}$, we either perform a constant time operation, or spend $O(\log(\sigma_{\max}))$ time for a constant number of binary searches over the temporal edges of the other two static edges in the static triangle $\langle u, v, w\rangle$. Thus,

$$T(\mathcal{A}) = O(m_s \cdot \kappa + \sum_{\langle u,v,w\rangle} (\sigma(u, v) + \sigma(u, w)) \log(\sigma_{\max}))$$

where $\mathcal{A}$ denotes Alg. 2, and $T(\mathcal{A})$ denotes the worst case time complexity of $\mathcal{A}$.

For each vertex $u \in V$, $d^+_{G_\prec}(u) \le \kappa$, so each edge (u, v) in $G_\prec$, is a part of at most κ static triangles where u is the source vertex. Therefore, the temporal edges on each edge $\{u, v\}$ in G are enumerated at most $O(\kappa)$ times. Thus,

$$T(\mathcal{A}) = O(m_s \cdot \kappa + \sum_{\{u,v\} \in E_s} (\sigma(u, v) + \sigma(v, u)) \cdot \kappa \log(\sigma_{\max})).$$

Hence,

$$T(\mathcal{A}) = O(m\kappa \log(\sigma_{\max})).$$

□

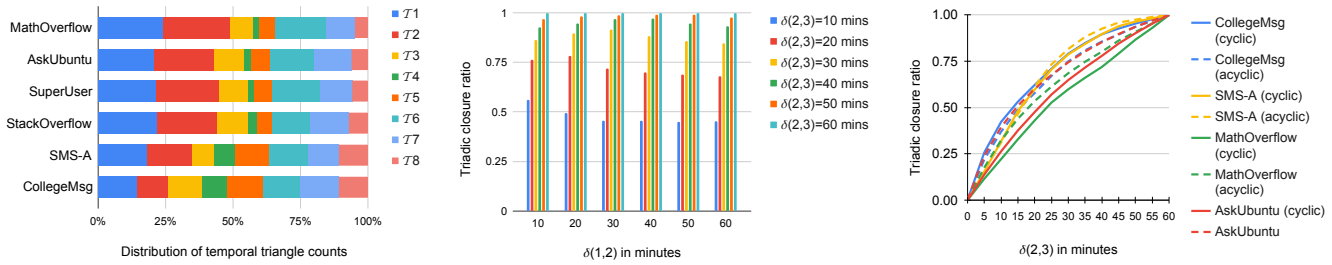
6 EXPERIMENTAL EVALUATIONS

We implemented our algorithm in C++ and used a commodity machine from AWS EC2: R5d.2xlarge to run our experiments. This EC2 instance has Intel(R) Xeon(R) Platinum 8175M CPU @ 2.50GHz and 64GB memory. On this AWS machine, PBL runs out of memory for the Bitcoin graph, so we used one with more than 256GB memory for this case. The implementation of DOTTT is available at [1].

We performed our experiments on a collection of temporal graphs from SNAP [23], KONECT[21], and the Bitcoin transaction dataset from [19], consisting of all transactions up to Feb 9,

Table 2: Descriptions of the datasets and runtime of DOTTT and PBL .

dataset	#vertices	#edges	#static edges	#static triangles	degeneracy	max multiplicity	time span (years)	DOTTT runtime	PBL runtime
CollegeMsg	1.9K	59.8K	13.8K	14.3K	20	98	0.51	0.09	0.07
email-Eu-core	986	332K	16.1K	105K	34	2.8K	2.2	2.31	3.37
MathOverflow	24.7K	390K	188K	1.4M	78	225	6.46	3.17	3.6
SMS-A	44.1K	545K	52.22K	10K	9	5.3K	0.92	0.45	0.81
AskUbuntu	157K	727K	456K	680K	48	154	7.09	2.23	5.08
SuperUser	192K	1.11M	715K	1.54M	61	78	7.59	4.41	8.84
WikiTalk	1.09M	6.11M	2.79M	8.12M	124	1.1K	6.21	34	56
StackOverflow	2.58M	47.9M	28.18M	114.2M	198	549	7.60	347	678
Wikipedia-DE	2.17M	86.21M	39.71M	169.9M	265	347	10.18	576	987
Bitcoin	59.61M	515.5M	366.4M	706.2M	604	447K	5.98	2923	4374



(a) Temporal triangle count distribution **(b) Effect of $\delta_{1,2}$ and $\delta_{2,3}$ on triadic closure** **(c) Triadic closure in cyclic and acyclic cases**

Figure 5: (a):The distribution of (1 hr, 1 hr, 1 hr)-temporal triangle counts over all eight temporal triangle types as shown in Fig. 2. (b):We fix $\delta_{1,3}$ to 2 hrs. We vary $\delta_{1,2}$ from 0 to 60 minutes and plot the ratio of (2 hrs, $\delta_{1,2}, \delta_{2,3}$)-temporal triangles to (2hrs, $\delta_{1,2}$, 1hr)-temporal triangles for $\delta_{2,3}$ ranging from 0 to 60 minutes. (c) We plot the ratio of (2 hrs, 1 hr, $\delta_{2,3}$)-temporal triangles to (2hrs, 1hr, 1hr)-temporal triangles for $\delta_{2,3}$ ranging from 0 to 60 minutes, for cyclic and acyclic triangles.

2018. The timestamp of each transaction is the creation time of the block on the blockchain that contains it[40].

Running time: All the running times are shown in Tab. 2. We ran all experiments on a single thread. In most instances, DOTTT takes a few seconds to run. For graphs with tens of millions of temporal edges, DOTTT runs in less than ten minutes. Even for the Bitcoin graph with 515M edges, DOTTT takes less than an hour.

Running time independent of time periods: The running time of both DOTTT and PBL algorithms are independent of the time periods. DOTTT has the same running time for time restrictions ranging from 0 to the time span of the input dataset. For comparison with *PBL*, we set $\delta_{1,3} = \delta_{1,2} = \delta_{2,3} = 1$ hr.

Comparison with PBL : We compare our algorithm with the PBL algorithm that counts $\delta_{1,3}$-temporal triangles, as it is the closest to our work. We typically get a 1.5x-2x speedup over PBL for large graphs (more than 0.5M edges) as shown in Fig. 1a. Note that DOTTT computes $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangle counts while PBL only gets the counts of $\delta_{1,3}$-temporal triangles.

Distribution of counts over types of triangles: The distribution of (1 hr, 1 hr, 1 hr)-temporal triangle counts for our datasets are shown in Fig. 5a. As we expected [29, 33, 51, 57], networks from similar domains have similar distributions. It is easy to see in Fig. 5a, that all the stack exchange networks have similar distributions. The same holds for the message networks CollegeMsg and SMS-A.

We observe that cyclic temporal triangles, $\mathcal{T}_4$ and $\mathcal{T}_8$, have a larger share in temporal triangle counts in messaging networks than in stack exchange networks.

Triadic closures in temporal networks: In static triangles, the transitivity measures the ratio of number of static triangles to the number of all wedges. In temporal graphs, in addition to transitivity, the time it takes for a wedge to appear and close is of importance [58]. In Fig. 5b, we study the effect of the time it takes for a wedge to appear from an edge, on the time it takes to close for CollegeMsg graph. We fix $\delta_{1,3} = 2$ hrs. For $\delta_{1,2}$ ranging from zero to 60 minutes (10 minute steps), we vary $\delta_{2,3}$ from zero to 60 minutes and plot the ratio of $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles over (2hrs, $\delta_{1,2}$, 1hr)-temporal triangles. We observe that the set of ratios for all values of $\delta_{2,3}$ are almost identical for different values of $\delta_{1,2}$. For instance, for all values of $\delta_{1,2}$, roughly half the triangles are formed in 10-20 minutes. This implies that once a wedge is formed, the time it took to appear does not affect the time it takes to close.

As another demonstration of DOTTT, for $\delta_{1,3} = 2$ hrs and $\delta_{1,2} = 1$ hrs, we plot the ratio of $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles to (2hrs, 1hr, 1hr)-temporal triangles, this time separately for cyclic and acyclic temporal triangles in Fig. 5c. We observe that for stack exchange networks, acyclic temporal triangles tend to take a shorter time to close from the moment their second edge appears than cyclic temporal triangles. As we see in Fig. 5c, this is not the case for message networks.

ACKNOWLEDGMENTS

The authors are supported by NSF DMS-2023495, NSF CCF-1740850, NSF CCF-1813165, CCF-1909790, and ARO Award W911NF1910294. Noujan Pashanasangi is supported by Jack Baskin and Peggy Downes-Baskin Fellowship.

REFERENCES

[1] DOTTT. https://github.com/nojanp/temporal-triangle-counting, 2021.

[2] Al Hasan, M., and Dave, V. S. Triangle counting in large networks: a review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8*, 2 (2018).

[3] Alon, N., Yuster, R., and Zwick, U. Finding and counting given length cycles. *Algorithmica 17*, 3 (1997), 209–223.

[4] Becchetti, L., Boldi, P., Castillo, C., and Gionis, A. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (2008), pp. 16–24.

[5] Bera, S. K., Pashanasangi, N., and Seshadhri, C. Linear time subgraph counting, graph degeneracy, and the chasm at size six. In *Proc. 11th Conference on Innovations in Theoretical Computer Science* (2020), Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[6] Boekhout, H. D., Kosters, W. A., and Takes, F. W. Efficiently counting complex multilayer temporal motifs in large-scale networks. *Computational Social Networks 6*, 1 (2019), 1–34.

[7] Bouritsas, G., Frasca, F., Zafeiriou, S., and Bronstein, M. M. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint arXiv:2006.09252* (2020).

[8] Chiba, N., and Nishizeki, T. Arboricity and subgraph listing algorithms. *SIAM Journal on computing 14*, 1 (1985), 210–223.

[9] Dimitriadis, S. I., Laskaris, N. A., Tsirka, V., Vourkas, M., Micheloyannis, S., and Fotopoulos, S. Tracking brain dynamics via time-dependent network analysis. *Journal of neuroscience methods 193*, 1 (2010), 145–155.

[10] Eden, T., Levi, A., Ron, D., and Seshadhri, C. Approximately counting triangles in sublinear time. *SIAM Journal on Computing 46*, 5 (2017), 1603–1646.

[11] Farajtabar, M., Gomez-Rodriguez, M., Wang, Y., Li, S., Zha, H., and Song, L. Coevolve: A joint point process model for information diffusion and network co-evolution. In *Companion Proceedings of the The Web Conference 2018* (2018).

[12] Gaumont, N., Magnien, C., and Latapy, M. Finding remarkably dense sequences of contacts in link streams. *Social Network Analysis and Mining 6*, 1 (2016), 1–14.

[13] Gu, S., Johnson, J., Faisal, F. E., and Milenković, T. From homogeneous to heterogeneous network alignment via colored graphlets. *Scientific reports 8*, 1 (2018), 1–16.

[14] Gurukar, S., Ranu, S., and Ravindran, B. Commit: A scalable approach to mining communication motifs from dynamic networks. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (2015).

[15] Itai, A., and Rodeh, M. Finding a minimum circuit in a graph. *SIAM Journal on Computing 7*, 4 (1978), 413–423.

[16] Jain, S., and Seshadhri, C. A fast and provable method for estimating clique counts using turán's theorem. In *Proc. 26th Proceedings, International World Wide Web Conference (WWW)* (2017), International World Wide Web Conferences Steering Committee, pp. 441–449.

[17] Jha, M., Seshadhri, C., and Pinar, A. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *Proc. 24th Proceedings, International World Wide Web Conference (WWW)* (2015), International World Wide Web Conferences Steering Committee, pp. 495–505.

[18] Khan, A., Li, N., Yan, X., Guan, Z., Chakraborty, S., and Tao, S. Neighborhood based fast graph search in large networks. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data* (2011).

[19] Kondor, D., Csabai, I., Szüle, J., Pósfai, M., and Vattay, G. Inferring the interplay between network structure and market effects in bitcoin. *New Journal of Physics 16*, 12 (2014), 125003.

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

[21] Kunegis, J. KONECT – The Koblenz Network Collection. In *Proc. Int. Conf. on World Wide Web Companion* (2013), pp. 1343–1350.

[22] Latapy, M. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theoretical computer science 407*, 1-3 (2008), 458–473.

[23] Leskovec, J., and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

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

[25] Liu, P., Benson, A., and Charikar, M. A sampling framework for counting temporal motifs. *arXiv preprint arXiv:1810.00980* (2018).

[26] Liu, P., Guarrasi, V., and Sariyuce, A. E. Temporal network motifs: Models, limitations, evaluation. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[27] Mackey, P., Porterfield, K., Fitzhenry, E., Choudhury, S., and Chin, G. A chronological edge-driven approach to temporal subgraph isomorphism. In *2018 IEEE International Conference on Big Data (Big Data)* (2018), IEEE, pp. 3972–3979.

[28] Matula, D. W., and Beck, L. L. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM 30*, 3 (1983), 417–427.

[29] Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., Sheffer, M., and Alon, U. Superfamilies of evolved and designed networks. *Science 303*, 5663 (2004), 1538–1542.

[30] Mongioví, M., Micale, G., Ferro, A., Giugno, R., Pulvirenti, A., and Shasha, D. glabtrie: A data structure for motif discovery with constraints. In *Graph Data Management*. Springer, 2018, pp. 71–95.

[31] Ortmann, M., and Brandes, U. Efficient orbit-aware triad and quad census in directed and undirected graphs. *Applied network science 2*, 1 (2017).

[32] Palla, G., Derényi, I., Farkas, I., and Vicsek, T. Uncovering the overlapping community structure of complex networks in nature and society. *nature 435*, 7043 (2005), 814–818.

[33] Paranjape, A., Benson, A. R., and Leskovec, J. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (2017), pp. 601–610.

[34] Pashanasangi, N., and Seshadhri, C. Efficiently counting vertex orbits of all 5-vertex subgraphs, by evoke. In *Proc. 13th International Conference on Web Search and Data Mining (WSDM)* (2020), pp. 447–455.

[35] Petrovic, L. V., and Scholtes, I. Counting causal paths in big times series data on networks. *arXiv preprint arXiv:1905.11287* (2019).

[36] Pfeiffer, J. J., La Fond, T., Moreno, S., and Neville, J. Fast generation of large scale social networks while incorporating transitive closures. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing* (2012), IEEE, pp. 154–165.

[37] Pfeiffer III, J. J., Moreno, S., La Fond, T., Neville, J., and Gallagher, B. Attributed graph models: Modeling network structure with correlated attributes. In *Proceedings of the 23rd international conference on World wide web* (2014).

[38] Pinar, A., Seshadhri, C., and Vishal, V. Escape: Efficiently counting all 5-vertex subgraphs. In *Proceedings, International World Wide Web Conference (WWW)* (2017), International World Wide Web Conferences Steering Committee.

[39] Redmond, U., and Cunningham, P. Temporal subgraph isomorphism. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)* (2013), IEEE, pp. 1451–1452.

[40] Reid, F., and Harrigan, M. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*. Springer, 2013, pp. 197–223.

[41] Ribeiro, P., and Silva, F. Discovering colored network motifs. In *Complex Networks V*. Springer, 2014, pp. 107–118.

[42] Rossi, R. A., Ahmed, N. K., Carranza, A., Arbour, D., Rao, A., Kim, S., and Koh, E. Heterogeneous network motifs. *arXiv preprint arXiv:1901.10026* (2019).

[43] Seshadhri, C., Pinar, A., and Kolda, T. G. Triadic measures on graphs: The power of wedge sampling. In *Proceedings of the 2013 SIAM international conference on data mining* (2013), SIAM, pp. 10–18.

[44] Seshadhri, C., and Tirthapura, S. Scalable subgraph counting: The methods behind the madness: Www 2019 tutorial. In *Proceedings of the Web Conference (WWW)* (2019), vol. 2, p. 75.

[45] Sun, X., Tan, Y., Wu, Q., Chen, B., and Shen, C. Tm-miner: Tfs-based algorithm for mining temporal motifs in large temporal network. *IEEE Access 7* (2019).

[46] Sun, X., Tan, Y., Wu, Q., Wang, J., and Shen, C. New algorithms for counting temporal graph pattern. *Symmetry 11*, 10 (2019), 1188.

[47] Tu, K., Li, J., Towsley, D., Braines, D., and Turner, L. D. Network classification in temporal networks using motifs. *arXiv preprint arXiv:1807.03733* (2018).

[48] Tu, K., Li, J., Towsley, D., Braines, D., and Turner, L. D. gl2vec: Learning feature representation using graphlets for directed networks. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (2019), pp. 216–221.

[49] Turk, A., and Turkoglu, D. Revisiting wedge sampling for triangle counting. In *Proceedings, International World Wide Web Conference (WWW)* (2019), ACM.

[50] Valverde, S., and Solé, R. V. Network motifs in computational graphs: A case study in software architecture. *Physical Review E 72*, 2 (2005), 026107.

[51] Vazquez, A., Dobrin, R., Sergi, D., Eckmann, J.-P., Oltvai, Z. N., and Barabási, A.-L. The topological relationship between the large-scale attributes and local interaction patterns of complex networks. *Proceedings of the National Academy of Sciences 101*, 52 (2004), 17940–17945.

[52] Vega, D., and Magnani, M. Foundations of temporal text networks. *Applied network science 3*, 1 (2018), 1–26.

[53] Wang, J., Wang, Y., Jiang, W., Li, Y., and Tan, K.-L. Efficient sampling algorithms for approximate temporal motif counting. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (2020).

[54] Wasserman, S., Faust, K., et al. *Social network analysis: Methods and applications*. Cambridge university press, 1994.

[55] Watts, D. J., and Strogatz, S. H. Collective dynamics of 'small-world'networks. *nature 393*, 6684 (1998), 440–442.

[56] Wernicke, S., and Rasche, F. Fanmod: a tool for fast network motif detection. *Bioinformatics 22*, 9 (2006), 1152–1153.

[57] Yaveroğlu, Ö. N., Malod-Dognin, N., Davis, D., Levnajic, Z., Janjic, V., Karapandza, R., Stojmirovic, A., and Pržulj, N. Revealing the hidden language of complex networks. *Scientific reports 4*, 1 (2014), 1–9.

[58] Zignani, M., Gaito, S., Rossi, G. P., Zhao, X., Zheng, H., and Zhao, B. Link and triadic closure delay: Temporal metrics for social network dynamics. In *Proceedings of the International AAAI Conference on Web and Social Media* (2014).

A MISSING ALGORITHMS FROM SECTION 5

Here we will provide the algorithms for counting $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles for temporal orderings π_3, π_4, π_5, and π_6.

Temporal orderings π_3 and π_4: Consider an orientation ρ, and a static triangle on vertices $\{u, v, w\}$ enumerated in $G_{\prec}$, where u is the source vertex. The category of temporal orderings π_3 and π_4 is more intricate because $\pi_3(\{v, w\}) = \pi_4(\{v, w\}) = 2$. Recall that we want to avoid enumerating temporal edges on $\{v, w\}$, so we do not enumerate edges on S_2 as in the case of π_1 and π_2. Instead, we enumerate edges on S_1, and compute the counts of edges on S_2 that form a temporal triangle with compatible edge in S_3.

We start by enumerating edges on S_1. Consider an edge $e_1 \in S_1$. Let ℓ_f and ℓ_ℓ denote the indices of first and last edge in S_3 with a timestamp in $[t(e_1), t(e_1) + \delta_{1,3}]$. Also, let $\ell_{\delta_{1,2}}$ denote the index of the first edge in S_3 with a timestamp greater than $t(e_1) + \delta_{1,2}$, and $\ell_{\delta_{2,3}}$ be the index of the first edge in S_3 that has a timestamp greater than $t(e_1) + \delta_{2,3}$. Note that $\ell_{\delta_{1,2}}$ and $\ell_{\delta_{2,3}}$ divide $S_3[\ell : \ell_\ell]$ into three consecutive subsequences. We show how to count temporal triangles that involve temporal edges in each of these three subsequences.

For a temporal edge $S_3[i]$ where $\ell_f \le i < \min(\ell_{\delta_{1,2}}, \ell_{\delta_{2,3}})$, each edge $e_2 \in S_2$ where $t(e_2) \in [t(e_1), t(S_3[i])]$ form a $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangle with e_1 and $S_3[i]$. To obtain the counts of these edges in S_2, it suffices to store $\text{CEC}_{-\infty}$ on S_2 for each edge $e_3 \in S_3$.

Now, consider the temporal edges $S_3[i]$ where $\max(\ell_{\delta_{1,2}}, \ell_{\delta_{2,3}}) < i \le \ell_\ell$. The timestamp of these edges are in time window $[t(e_1) + \max(\delta_{1,2}, \delta_{2,3}), t(e_1) + \delta_{1,3}]$, and together with temporal edge e_1 form a $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangle with each temporal edge $e_2 \in S_2$ where $t(e_2) \in [t(S_3[i]) - \delta_{2,3}, t(e_1) + \delta_{1,2}])$. To count the number of such temporal edges in S_2 we only need to store $\text{CEC}_{-\delta_{2,3}}$ and $\text{CEC}_{-\infty}$ on S_2 for each temporal edge e_3 in S_3.

Finally, consider a temporal edge $S_3[i]$ where $\min(\ell_{\delta_{1,2}}, \ell_{\delta_{2,3}}) \le i \le \max(\ell_{\delta_1}, \ell_{\delta_{2,3}})$. The number of compatible edges in S_2 depend on how $\delta_{1,2}$ compares to $\delta_{2,3}$. There are two cases: $(a) : \delta_{1,2} < \delta_{2,3}$ and $(b) : \delta_{2,3} < \delta_{1,2}$. In case (a) edges in S_2 have to have a timestamp in $[t(e_1), t(e_1) + \delta_{1,2}]$ to form a triangle with e_1 and $S_3[i]$, and in case (b) their timestamps should be in the time window $[t(S_3[i]) - \delta_{2,3}, t(S_3[i])]$. Alg. 3 give the step by step procedure for counting temporal triangles for temporal orderings π_3 and π_4.

Temporal orderings π_5 and π_6: This case is similar to the case of temporal ordering π_1 and π_2. The difference is that while enumerating edges in S_2, we will first find the compatible edges in S_3 instead of S_1, and then count edges in S_1 that complete a temporal triangle. Consider a static triangle $\{u, v, w\}$ and orientation ρ. Fix a temporal edge e_2 in S_2. Let ℓ_f and ℓ_ℓ denote the indices of the first and last temporal edges in S_3 with a timestamp in the time period $[t(e_2), t(e_2) + \delta_{2,3}]$. Let $\ell_{\delta_{1,2}}$ be the first temporal edge in S_3 such that $t(S_3(\ell_{\delta_{1,2}})) > t(e_2) + \delta_{1,3} - \delta_{1,2}$. Since $\delta_{1,3} \le \delta_{1,2} + \delta_{2,3}$, we have $\ell_f \le \ell_{\delta_{2,3}} \le \ell_\ell$. Consider a temporal edge $S_3[i]$ where $\ell_f \le i < \ell_{\delta_{1,2}}$. The number of edges in S_1 that form a triangle with e_2 and $S_3[i]$ is $\text{EC}(S_1, [t(e_2) - \delta_{1,2}, t(e_2)])$. For each temporal edge $S_3[i]$ where $\ell_{\delta_{1,2}} \le i \le \ell_\ell$, there are $\text{EC}(S_1, [t(S_3[i]) - \delta_{1,3}, t(e_2)])$ edges that complete a temporal triangle. This is the same as in the case of π_1 and π_2 with different time windows. We need to get $\text{CEC}_{-\infty}$ and $\text{CEC}_{-\delta_{1,3}}$ on S_1 for each edge $e_3 \in S_3$.

Algorithm 3 Counting $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles corresponding to a static triangle and temporal orientation π_3 or π_4

procedure CTT-vw2($\delta_{1,3}$, $\delta_{1,2}$, $\delta_{2,3}$,$\langle u, v, w\rangle$, π, ρ)
▷ $\pi(\{v, w\}) = 2$
  count = 0
  Enumerate S_3 and compute $\text{CEC}_{-\infty}$ and $\text{CEC}_{-\delta_{2,3}}$ on S_2
  for $i = 1, \ldots, \sigma_1$ **do**
    Let ℓ_f = lowerBound($t(S_1[i]), S_3$)
    Let $\ell_{\delta_{1,2}}$ = upperBound($t(S_1[i]) + \delta_{1,2}, S_3$)
    Let $\ell_{\delta_{2,3}}$ = lowerBound($t(S_1[i]) + \delta_{2,3}, S_3$)
    Let ℓ_ℓ = upperBound($t(S_1[i]) + \delta_{1,3}, S_3$)
    Let $\ell_{\min} = \min(\ell_{\delta_{1,2}}, \ell_{\delta_{2,3}})$
    Let $\ell_{\max} = \max(\ell_{\delta_{1,2}}, \ell_{\delta_{2,3}})$
  ▷ Edges in $S_3[\ell_f : \ell_{\min}]$
    count += $\text{CEC}_{-\infty}(S_3[\ell_f : \ell_{\min}], S_2)$
    count −= $(\ell_{\min} - \ell_f + 1) \cdot \text{EC}((-\infty, t(S_1[i])], S_2)$
  ▷ Edges in $S_3[\ell_{\min} + 1 : \ell_{\max} - 1]$
    if $\delta_{1,2} \le \delta_{2,3}$ **then**
      count += $(\ell_{\delta_{2,3}} - \ell_{\delta_{1,2}})$
        $\cdot \text{EC}([t(S_1[i]), t(S_1[i]) + \delta_{1,2}], S_2)$
    else if $\delta_{2,3} \le \delta_{1,2}$ **then**
      count += $\text{CEC}_{-\delta_{2,3}}(S_3[\ell_{\delta_{2,3}} : \ell_{\delta_{1,2}}], S_2)$
  ▷ Edges in $S_3[\ell_{\max} : \ell_\ell]$
    count += $\text{CEC}_{-\delta_{2,3}}(S_3[\ell_{\max} : \ell_\ell], S_2)$
    count −= $\text{CEC}_{-\infty}(S_3[\ell_{\max} : \ell_\ell], S_2)$
    count += $(\ell_\ell - \ell_{\max}) \cdot \text{EC}((-\infty, t(S_1[i]) + \delta_{1,2}], S_2)$
  return count

Algorithm 4 Counting $(\delta_{1,3}, \delta_{1,2}, \delta_{2,3})$-temporal triangles corresponding to a static triangle and temporal orientation π_5 or π_6

procedure TTC-vw1($\delta_{1,3}$, $\delta_{1,2}$, $\delta_{2,3}$,$\langle u, v, w\rangle$, π, ρ)
▷ $\pi(\{v, w\}) = 1$
  count = 0
  Enumerate S_3 and compute $\text{CEC}_{-\delta_{1,3}}$ and $\text{CEC}_{-\infty}$ on S_1
  for $i = 1, \ldots, \sigma_2$ **do**
    Let ℓ_f = lowerBound($t(S_2[i]), S_3$)
    Let $\ell_{\delta_{1,2}}$ = lowerBound($t(S_2[i]) + \delta_{1,3} - \delta_{1,2}, S_3$)
    Let ℓ_ℓ = upperBound($t(S_2[i]) + \delta_{2,3}, S_3$)
  ▷ Edges in $S_3[\ell_{\delta_{1,2}} : \ell_\ell]$
    count += $\text{CEC}_{-\delta_{1,3}}(S_3[\ell_{\delta_{1,2}} : \ell_\ell], S_1)$
    count −= $\text{CEC}_{-\infty}(S_3[\ell_{\delta_{1,2}} : \ell_\ell], S_1)$
    count += $(\ell_\ell - \ell_{\delta_{1,2}} + 1) \cdot \text{EC}((-\infty, t(S_2[i])], S_1)$
  ▷ Edges in $S_3[\ell_f : \ell_{\delta_{1,2}} - 1]$
    count += $(\ell_{\delta_{1,2}} - \ell_f) \cdot \text{EC}([t(S_2[i]) - \delta_{1,2}, t(S_2[i])], S_1)$
  return count