Appendix

A. Proofs

Proof of Claim III.7 Consider any pair of edges (x, u, t_x) and (v, w, t_w) where x, w are arbitrary vertices, $t_x \in [t-\delta, t]$ and $t_w \in [t, t+\delta]$. Then, these edges form a $\langle +, -, <, > \rangle$ δ-centered 3-path with e at the center. The number of edges (x, u, t_x) where $t_x \in [t-\delta, t]$ is precisely the indegree $d_u^-[t-\delta, t]$. Similarly, the number of edges (v, w, t_w) where $t_w \in [t, t+\delta]$ is the outdegree $d_v^+[t, t+\delta]$. The product of these degrees gives the total number of desired 3-paths, which is $d_u^-[t-\delta, t] \cdot d_v^+[t, t+\delta]$.

Proof of Claim III.8 We first give the running time of PREPROCESS. Observe that it performs four binary searches for each edge (two each in the out-neighbors and in-neighbors). The total running time is $O(m \log m)$. m is the number of edges in G.

For SAMPLE, the first step is to sample from the distribution given by $\{p_{e,\delta}\}$ values. This can be done using a binary search, which takes $O(\log m)$ time. After that, it performs two binary searches and two random number generations. So the total running time is $O(\log m)$.

Proof of Lemma III.8.1 Consider a $\langle -, +, <, > \rangle$ δ-centered 3-path (e_1, e_2, e_3). Let $e_2 = (u, v, t)$. Then e_1 is an in-edge of u with timestamp in $[t-\delta, t]$. Also, e_3 is an out-edge of v with timestamp in $[t, t+\delta]$.

The probability of sampling e_2 is $w_{e_2,\delta}/W_\delta$. Conditioned on this sample, the probability of sampling e_1 is precisely $1/|\Lambda_u^-[t-\delta, t]|$, which is $1/d_u^-[t-\delta, t]$. Similarly, the probability of sampling e_3 is $1/d_v^+[t, t+\delta]$. Multiplying all of these, we get the probability of sampling the entire 3-path (e_1, e_2, e_3). Applying Claim III.7, the probability is

$$\begin{aligned} & \frac{w_{e_2,\delta}}{W_\delta} \cdot \frac{1}{d_u^-[t-\delta, t]} \cdot \frac{1}{d_v^+[t, t+\delta]} \\ = \quad & \frac{d_u^-[t-\delta, t] \cdot d_v^+[t, t+\delta]}{W_\delta} \cdot \frac{1}{d_u^-[t-\delta, t] \cdot d_v^+[t, t+\delta]} = \frac{1}{W_\delta} \end{aligned}$$

Hence, the probability of sampling (e_1, e_2, e_3) is $1/W_\delta$, which corresponds to the uniform distribution. (By definition, the total number of $\langle -, +, <, > \rangle$ δ-centered 3-paths is W_δ.)

B. Experiement Setup

Datasets We evaluate the temporal motif counts across a spectrum of datasets, encompassing medium to large-scale graphs. The link to those public datasets is listed below:

- wiki-talk (WT): https://snap.stanford.edu/data/wiki-talk-temporal.html
- stackoverflow (SO) https://snap.stanford.edu/data/sx-stackoverflow.html
- bitcoin (BI) https://www.cs.cornell.edu/~arb/data/temporal-bitcoin/
- reddit-reply (RE) https://www.cs.cornell.edu/~arb/data/temporal-reddit-reply/

Exact Count Baseline We use BT [25] as an exact temporal motif count baseline, which does a backtracking search on chronologically-sorted temporal edges. The original C++ implementation is single-threaded and runs for more than a week in many cases. We implemented a multi-threaded version of BT with OpenMP using dynamically scheduled work-stealing threads without using costly synchronization/atomic primitives.

Approximate Baselines PRESTO [52] is a sampling algorithm that runs an exact motif count algorithm on sampled intervals to get estimated results. is similar to IS [37], but does not require partitioning all edges into non-overlapping windows. Instead, it leverages uniform sampling. It provides two variants, *PRESTO-A* and *PRESTO-E*. We use their open-source implementation from [59]. We run *PRESTO-A* and *PRESTO-E* with a scaling factor of sampling window size $c = 1.25$. The number of samples configured such that its runtime is around $10\times$ slower than TEACUPS.

TEACUPS Setup We implement TEACUPS in C++ and run the experiments on an AMD EPYC 7742 64-Core CPU with 64MB L2 cache, 256MB L3 cache, and 1.5TB DRAM memory. For multi-threaded code, we use the OpenMP library.

For the detailed number of samples per graph, motif and δ in our experiments, please refer to https://anonymous.4open.science/r/TEACUPS_ICDM24/reproduce.py.