Extending Causal Path Analysis in Temporal Network Data for Efficiency

Social Network Analysis for Computer Scientists — Course paper

Filipe Coutinho
f.z.lopes.de.coutinho@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, Netherlands

Senjuti Bala
s.bala@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, Netherlands

ABSTRACT

Graph or network representations are fundamental components of many applications related to data mining and machine learning, especially when dealing with relational data. Popular tools for network analysis, like cluster detection, information ranking, and centrality measures, work under the premise that links indicate direct influence and paths indicate possible indirect influence. On the other hand, timestamped network data containing dynamic social networks, biological sequences, or financial transactions cast doubt on this assumption. The temporal arrangement and chronological ordering of links in these datasets establish the presence of causal pathways connecting nodes. Large-scale datasets may encounter challenges with current approaches to this temporal influence challenge because they frequently require computationally demanding statistics on causal paths.

Our studied paper [3] demonstrates an effective algorithm for counting causal paths in timestamped network data is introduced. In the paper following the suggested method the authors have obtained a more efficient algorithm than a baseline that is built into an OpenSource data analytics package. In our paper we have implemented the algorithm tailored for counting causal paths in different timestamped network data. Our objective was to experiment with the addressed algorithm and overcome the computational challenges associated with existing methods, especially when dealing with large-scale datasets.

KEYWORDS

causal paths, temporal networks, social network analysis, network science

ACM Reference Format:
Filipe Coutinho and Senjuti Bala. 2022. Extending Causal Path Analysis in Temporal Network Data for Efficiency: Social Network Analysis for Computer Scientists — Course paper. In *Proceedings of Social Network Analysis for Computer Scientists Course 2022 (SNACS '22).* ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

Temporal networks refer to a type of network where the connections between nodes change over time. Unlike static networks, they incorporate the notion of time by attaching a timestamp to a connection. In time-varying networks, the edges between nodes can exist or change at different points in time. By analyzing these graphs we can find periodic behaviors, bursts of activity, or other temporal structures that provide insights into the dynamics of the network.

In a static network the paths between nodes are transitive, meaning that if there is a link from node A to node B and another link from node B to node C, there is a direct path from node A to node C. However, in a timestamped network, data records the timing and chronological ordering of interactions which can disrupt the transitivity of paths. This disruption of transitivity makes it challenging to apply traditional network analysis techniques, that may not accurately capture the indirect influence between nodes in timestamped network data.

The paper [3] tackles the task of employing network analysis on timestamped network data. Specifically, it focuses on quantifying the causal paths within the network. Causal Paths refer to the sequence of interactions between node and timestamped network data which determines the existence of influence between them.

Addressing this issue can be computationally challenging for large networks, the paper [3] aims to develop an efficient algorithm to count causal paths in timestamped network data, focusing on the big time-series data in complex networks. The proposed algorithm is designed to work efficiently for different values of the maximum time difference between consecutive links, allowing one to tailor the definition of causal paths to the time scale of data. The algorithm is also designed to support streaming scenarios, enabling real-time analysis of time series data on temporal networks.

This algorithm finds its applications in a variety of fields, including dynamic social networks, biological sequences, and financial transactions. The chronological sequencing of interactions and transactions makes the algorithm a valuable tool for extracting meaningful insights.

In dynamic social networks, the interactions between individuals evolve over time. This algorithm enables the analysis of how information flows through the network based on the sequence of interactions, therefore helping to reveal influential individuals or predict future interactions. In the realm of biological data, elements such as genes or protein interactions, often form a complex network with temporal properties. For example, the activation of one gene can result in a cascade of effects on other genes. Understanding

SNACS '22, Master CS, Fall 2022, Leiden, the Netherlands

the causal pathways is critical in comprehending diseases or developing new treatments. Financial transactions between entities also form a network where the timing of transactions is crucial. By analyzing the causal paths of transactions over time we can detect patterns of transactions that reveal money laundering activities, market manipulation, etc.

In this comprehensive study, we implemented and rigorously tested the PaCo algorithm across two uniquely characterized datasets, each with varying delta and k values, to demonstrate its adaptability and robustness in diverse temporal network scenarios. Our approach was grounded in practical application, involving extensive coding and analysis, to showcase the algorithm's efficacy in processing complex, time-sensitive data. Utilizing the PathPy [1]library as a comparative baseline, we thoroughly examined the algorithm's performance, highlighting its efficiency in different settings. This in-depth exploration into the PaCo algorithm's capabilities in handling temporal networks reveals its potential for wide-ranging applications, from dynamic social network analysis to intricate biological data interpretation.

In our study, we focused on understanding and applying the PaCo algorithm to five distinct datasets, illustrating its practical use in analyzing temporal networks. Through this process, we gained a deeper comprehension of the algorithm's intricacies, particularly in handling time-sensitive, large-scale data. Our work, centered on applying and testing the PaCo algorithm rather than contributing novel findings to the field, showcases our grasp of complex concepts in temporal network analysis and the algorithm's potential in extracting valuable insights from dynamic data environments.

2 RELATED WORK

2.1 Example: PageRank Citation Ranking: Bringing Order to the Web

A paper [2] titled *PageRank Citation Ranking: Bringing Order to the Web* presents the PageRank algorithm, which uses the web's link structure to rank web pages in an objective manner. Search engines use this 1998-developed idea as the basis for ranking search results. By simulating a random web surfer model, PageRank determines the value of a page based on the number and calibre of links pointing to it. The algorithm effectively ranks pages according to relevance and importance by iteratively calculating PageRank for a sizable amount of pages. This technique enhances search results while providing guidance for web navigation and traffic estimation. This groundbreaking work served as the foundation for Google's search engine and impacted later web-related research, such as the paper's [3] method of studying temporal networks.
The approach that PageRank uses to infer importance from a graph structure is probably the source of inspiration for this paper[3], which applies the same ideas to temporal networks and causal path tracing.

[1]https://github.com/uzhdag/pathpy/tree/master

2.2 Example: node2vec: Scalable Feature Learning for Networks

Node2vec is a feature learning framework for nodes in networks that is introduced in the paper [1] *node2vec: Scalable Feature Learning for Networks*. It defines a flexible node neighbourhood concept and aims to maximise the likelihood of maintaining network neighbourhoods. It seeks to maximize the likelihood of preserving network neighborhoods and defines a flexible node neighborhood concept. The method employs a biased random walk to efficiently explore different neighborhoods, generalizing previous rigid approaches and allowing richer representations. The study shows that node2vec can be broadly applied in complex network analysis by outperforming other methods in classification and link prediction tasks on a variety of networks.
Node2vec's efficient exploration of neighborhoods using biased random walks parallels the way the paper's algorithm [3] needs to explore and count causal paths within temporal networks. We believe that, paper [1] could have inspired the paper [3] by demonstrating the significance of understanding complex structures within a network, which is crucial when identifying causal paths.

2.3 Example: Maps of Random Walks on Complex Networks Reveal Community Structure

An information-theoretic method for revealing community structures in weighted and directed networks is presented in the paper[4] *Maps of random walks on complex networks reveal community structure* by Rosvall and Bergstrom. By breaking a network down into smaller components, this technique maximises the description of information flows while highlighting the regularities and connections within the structure of the network. This method was demonstrated by mapping scientific communication using journal citation patterns, which showed a multi-centric structure within the scientific network and indicated that fields differed in size and integration. Because it offers a framework for comprehending intricate network dynamics and structures—which are necessary for counting causal paths in temporal networks—this research probably had an impact on the studied paper[3].

3 PRELIMINARIES

3.1 Problem Statement:

The paper addresses the problem of efficiently counting causal paths in timestamped network data. In timestamped networks, the chronological ordering and timing of links determine the existence of causal paths between nodes. Existing methods for analyzing time-stamped network data are computationally challenging for big datasets. The paper [3] proposes an efficient algorithm to count causal paths in timestamped network data, considering the maximum time difference (δ) and maximum path length (K) as parameters. The algorithm aims to close the gap in analyzing big-time series data on complex networks by providing a more efficient method for counting causal paths. For our experiment we have utilized the algorithm and as well as the baseline method using

PathPy[2] Library to find results on which way is counting causal paths more efficiently using less time complexity.

We define the following terms for our analysis:

- Set D: A set of directed links (s, d, t), where s and d represent the source and target nodes of the link, and t represents the discrete timestamp of the link occurrence.
- K: The maximum length of a causal path, representing the number of traversed links in the path.
- δ: The maximum time difference between consecutive links on a causal path, limiting the chronological order of the links.

By defining the set D of directed, time-stamped links (s, d, t) where $s, d \in V$ are the source and target nodes, and $t \in \mathbb{N}$ is the discrete-time stamp. A causal path for a maximum time difference δ is a sequence of nodes $n_i \in V$ such that:

(1) $e_i := (n_i, n_{i+1}, t_i) \in D$,
(2) $t_{i+1} > t_i$,
(3) $t_{i+1} - t_i \leq \delta$.

The problem is to efficiently count the number of causal paths, formalized as $C(p)$ for all causal paths p with $\|p\| \leq K$. Formally, we define:

$$C(p) = |\{(e_1, \ldots, e_k) \mid (e_1, \ldots, e_k) \text{ is an instance of } p \in D\}| \quad (1)$$

Where $\|p\|$ denotes the number of traversed links in path p, and $C(p)$ extends the notion of link weights to causal paths of arbitrary length. The equation $C(p)$ is important because it provides a quantitative measure of the number of causal paths that satisfy a specific condition or pattern in timestamped network data.

4 APPROACH

In continuation of the challenges that currently exist and also, how the existing methods are computationally challenging for big datasets, the authors have proposed an efficient algorithm to count causal paths in time-stamped network data aiming to close the gap in analyzing big-time series data on complex networks. The authors have compared their method to a baseline algorithm implemented in an open-source data analytics package demonstrating its superior efficiency. The baseline algorithm performs three steps:

- Representing timestamped links as a direct acyclic graph (DAG)
- Computing all causal paths between root and leaf nodes using the DAG
- Counting all shorter paths of length K contained in the longest causal paths

The preliminary experiments to assess the run-time of their algorithm compared different sizes of data, maximum time differences, and maximum length of causal paths. The results showed that the authors' algorithm outperformed the baseline algorithm in terms of run-time efficiency indicating their method being more efficient in counting causal paths.

The proposed algorithm in the studied paper [3] is designed to handle different values of the maximum time difference between consecutive links in timestamped network data. Additionally, the algorithm supports streaming scenarios meaning it can efficiently process data in real-time. The algorithm uses an iterative approach extending causal paths by moving a sliding time window over the data which allows for continuous analysis of timestamped network data as new links are added.

In this paper, to expand on the technical details of our approach, we conducted a comprehensive implementation of the PaCo algorithm using five distinct datasets with inherent temporal characteristics. We developed a Python script to process raw timestamp data and configure the `TimeStampedLinkList` class, which is crucial for the execution of the PaCo algorithm. The script streamlined the conversion of timestamps and prepared the data for subsequent analysis. This involved iterating through the network's links and incrementally building causal paths constrained by the parameters δ and K.

Comparative analysis was performed against a baseline method that involved constructing a graph from time-stamped links and enumerating all possible causal paths. This evaluation served to measure the efficiency of the PaCo algorithm. Our findings underscored the algorithm's capacity for rapid processing of extensive temporal network data, emphasizing its suitability for dynamic network analysis. The effectiveness of our approach, demonstrated through this analysis, affirms the practical applicability of the PaCo algorithm in processing real-world data.

4.1 Purpose of Calculating Causal Paths

In the case of complex networks, causal paths represent how nodes with time-varying topologies influence each other. So calculating the count of causal paths can provide the following:

- The count of causal paths can represent the indirect influence between nodes.
- The count can help identify unusual and unexpected frequencies in a dynamic network.
- The count can be used to rank nodes based on their centrality, providing a measure of their influence.
- It can also signify clusters or communities that have strong indirect connections.

4.2 PaCo: Algorithm Approach for Counting Causal Paths

In the studied paper [3], the authors have approached a way to count all occurrences of all causal paths in a dataset D having a maximum length of the causal paths, K. The algorithm considers three parameters:

- D, data
- δ, a parameter that determines the maximum time difference allowed between consecutive links on a causal path
- K, maximum length of causal paths

The pseudocode of the algorithm is as follows:

[2]https://github.com/uzhdag/pathpy/tree/master

Algorithm 1 Calculate occurrences of all causal paths in data D that are at most K steps long, assuming parameter δ

Require: D, δ, K
$c \leftarrow$ dict()
$W \leftarrow$ list()
for $i \in 1$ to N **do**
  $(s, d, t) \leftarrow D[i]$
  $c_i \leftarrow$ dict()
  $c_i[sd] \leftarrow 1$
  for $j \in 1$ to W.length() **do**
    $(s_j, d_j, t_j, c_j) \leftarrow W[j]$
    if $t_j < t - \delta$ **then**
      W.remove((s_j, d_j, t_j, c_j))
    else
      if $d_j = s \wedge t_j < t$ **then**
        for $p \in c_j$.keys() **do**
          if $\|p\| < K$ **then**
            if $p \oplus d \notin c_i$.keys() **then**
              $c_i[p \oplus d] \leftarrow c_j[p]$
            else
              $c_i[p \oplus d] \leftarrow c_i[p \oplus d] + c_j[p]$
  for $p \in c_i$.keys() **do**
    if $p \notin c$.keys() **then**
      $c[p] \leftarrow c_i[p]$
    else
      $c[p] \leftarrow c[p] + c_i[p]$
  W.append((s, d, t, c_i))
return c

The introduced algorithm is for counting all instances of causal paths of lengths $k \leq K$ for a given maximum path length K and a maximum time difference δ between subsequent links on causal paths. The algorithm iteratively extends causal paths by moving a sliding time window of length δ over the data. Initially, each time-stamped link (n_1, n_2, t) is a causal path $n_1 n_2$ of length one that can be extended by time-stamped links in time window $(t, t + \delta)$. To denote the extension of a causal path, we define a binary operator $\oplus$ on a path $p = n_0 \dots n_k$ and a node n_{k+1} as follows:

$$\overrightarrow{n_0 \dots n_k} \oplus n_{k+1} = \overrightarrow{n_0 \dots n_k n_{k+1}}$$

Assuming time-stamped links in the data D are chronologically ordered, our algorithm performs a single pass through the sequence and computes the count C_i of causal path instances ending with link e_i. Each $C_i(p)$ is defined as m, where m counts different instances of causal path p that end with e_i. To calculate the output C, we aggregate path counters C_i by addition. For each path counter C and C':

$$C_{\text{sum}} = C + C' \Leftrightarrow \forall p : C_{\text{sum}}(p) = C(p) + C'(p)$$

The operation $\text{ext}(C, n)$ extends each path in C with node n, obtaining C_{ext}:

$$C_{\text{ext}} = \text{ext}(C, n) \Leftrightarrow \forall p, \|p\| < K : C_{\text{ext}}(p \oplus n) = C(p)$$

Once we have completed a single pass over the time-stamped links, the dictionary c contains counts of all causal path instances with length $k \leq K$ and maximum time difference δ.

4.3 Computational Complexity

The algorithm processes temporal networks by counting causal paths, considering constraints like maximum path length and time difference. It extends paths within a sliding window, updating counts iteratively. The computational complexity, as derived from the algorithm, depends on the number of links in the time window and the maximum number of causal paths. The complexity is influenced by the maximum eigenvalue of the adjacency matrix, affecting runtime with respect to network sparsity and path length constraints. This approach emphasizes scalability and efficiency in analyzing large temporal datasets.

5 DATA

5.1 Reddit Hyperlink Network

The *Reddit Hyperlink Network* used in this study is available on the Stanford Network Analysis Project (SNAP) website [3]. The data is derived from posts on various subreddits and contains hyperlinks to other subreddit's posts. Each dataset entry includes the source post, the target post, a timestamp, and the sentiment of the source community towards the linked post in the target community.

Table 1: Summary Statistics of the Reddit Hyperlink Network

Number of edges	Number of nodes
571927	54075

The authors of this dataset opted to represent the data timestamps as strings, necessitating preprocessing to convert these timestamps into UNIX timestamps.

5.2 CollegeMsg Temporal Network

The *CollegeMsg Temporal Network*[4] utilized in this study comprises private messages exchanged on an online social network at the University of California, Irvine. The dataset captures interactions where users can search the network for others and initiate conversations based on profile information.

In this temporal network, edges (u, v, t) denote user u messaging user v at time t.

Table 2: Summary Statistics of the CollegeMsg Temporal Network

Number of edges	Number of nodes
59835	1899

No preprocessing was necessary for this dataset, as the timestamp data was already in UNIX timestamp format.

5.3 email-Eu-core Temporal Network

The *email-Eu-core Temporal Network* [5] used in this study is derived from email data from a large European research institution. The dataset provides anonymized information about all incoming and

[3] https://snap.stanford.edu/data/soc-RedditHyperlinks.html
[4] https://snap.stanford.edu/data/CollegeMsg.html
[5] https://snap.stanford.edu/data/email-Eu-core-temporal.html

outgoing emails between members of the research institution. The communication is limited to members within the institution, and the dataset does not include messages to or from external entities. Each directed edge (u, v, t) represents the event where person u sent an email to person v at time t. Notably, separate edges are created for each recipient of an email.

Table 3: Summary Statistics of the email-Eu-core Temporal Network

Number of edges	Number of nodes
332334	986

The dataset includes four sub-networks that correspond to communication between members of four different departments at the institution. It is essential to note that node IDs in the sub-networks do not correspond to the same node IDs in the entire network. The static version of this network is equivalent to the largest weakly connected component of the email-Eu-core network, although node IDs may differ.

5.4 Bitcoin Alpha Trust Weighted Signed Network

The *Bitcoin Alpha Trust Weighted Signed Network*[6] represents a who-trusts-whom network of individuals trading Bitcoin on a platform called Bitcoin Alpha. Due to the anonymous nature of Bitcoin users, a trust network is maintained to prevent transactions with fraudulent and risky users. Members of Bitcoin Alpha rate each other on a scale of -10 (total distrust) to +10 (total trust) in steps of 1. This dataset serves as the first explicit weighted signed directed network available for research in Bitcoin trading.

Table 4: Summary Statistics of the Bitcoin Alpha Trust Weighted Signed Network

Number of edges	Number of nodes
24186	3783

This requires no preprocessing as timestamp data is already in UNIX timestamp format.

5.5 Bitcoin OTC Trust Weighted Signed Network

The *Bitcoin OTC Trust Weighted Signed Network*[7] represents a who-trusts-whom network of individuals trading Bitcoin on the Bitcoin OTC platform. Similar to the Bitcoin Alpha dataset, this network addresses the challenge of anonymous users by implementing a reputation system. Members of Bitcoin OTC rate each other on a scale of -10 (total distrust) to +10 (total trust) in steps of 1. This dataset also stands as the first explicit weighted signed directed network available for research in the context of Bitcoin trading.

[6]https://snap.stanford.edu/data/soc-sign-bitcoin-alpha.html
[7]https://snap.stanford.edu/data/soc-sign-bitcoin-otc.html

Table 5: Summary Statistics of the Bitcoin OTC Trust Weighted Signed Network

Number of edges	Number of nodes
35592	5881

6 EXPERIMENTS AND RESULTS

In our experiments, we conducted a comparative analysis between the PaCo algorithm and a baseline algorithm across various datasets. The chosen baseline algorithm is `paths_from_temporal_network_dag` from the `pathpy` library, available at [8].

Due to the runtime cost of the baseline function and to maintain sensible runtime, we chose to conduct the experiments with values of δ represented in seconds only.

6.1 Reddit Hyperlink Network Dataset

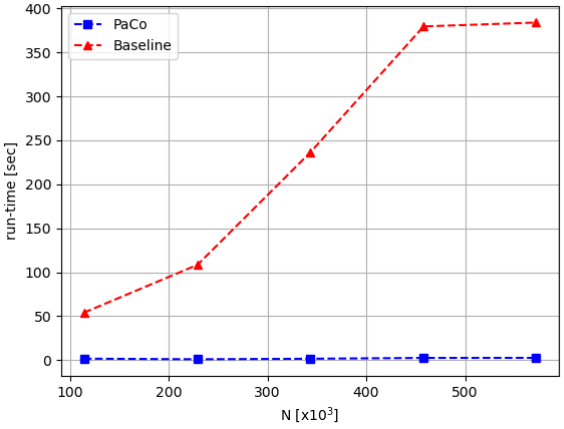

Figure 1: Reddit Hyperlink Network Dataset: Runtime comparison between baseline and PaCo with varying link count, using $\delta = 30$ and $k = 4$.

In Fig. 1, the y-axis illustrates the runtime, while the x-axis denotes the number of links (N, in 10^3). The figure depicts the runtime behavior with fixed parameters of $\delta = 30$ and $K = 4$. Significantly, the baseline runtime experiences a notable increase as the number of edges (N) grows, aligning with expectations. In contrast, the PaCo algorithm consistently maintains a low runtime even with an increasing number of links. These results are in line with the results reported in the studied paper [3].

[8]https://github.com/uzhdag/pathpy/tree/master

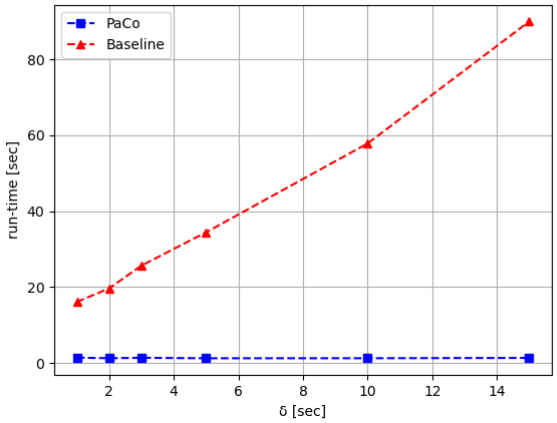

Figure 2: Reddit Hyperlink Network Dataset: Runtime comparison between baseline and PaCo. Varying δ with $K = 3$.

In Fig. 2, we set the value of K to 3 and varied δ. The plot illustrates that, once again, PaCo significantly outperforms the baseline algorithm. While these results are confined to a δ in the scale of seconds, the disparity between the baseline algorithm and PaCo becomes even more pronounced when we extend the value of δ to minutes or hours. In such cases, PaCo completes in minutes, whereas the baseline algorithm takes hours.

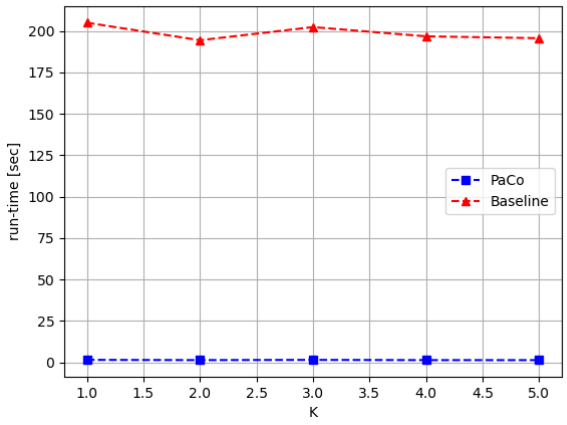

Figure 3: Reddit Hyperlink Network Dataset: Runtime comparison between baseline and PaCo. Varying K with $\delta = 30$.

In Fig. 3, we maintain a fixed value of δ at 30 while analyzing the runtime with varying values of the maximum path length (K). These results differ notably from those in [3], where PaCo consistently outperforms the baseline algorithm. However, the baseline exhibits an almost flat line in the plot. Upon closer analysis, we observed that, for the window of size δ, the maximum path length is 2. Consequently, varying for every $K > 2$ yields similar results and approximately the same runtime.

6.2 CollegeMsg Dataset

Following a similar experimental setup as described in the previous subsection, we present the results of our analysis on the CollegeMsg Dataset. Remarkably, the trends observed in this dataset closely mirror those observed in the first dataset. Figures 4, 5, and 6 showcase constant outperformance by PaCo and are analogous to the corresponding figures in the Reddit Hyperlink Network Dataset.

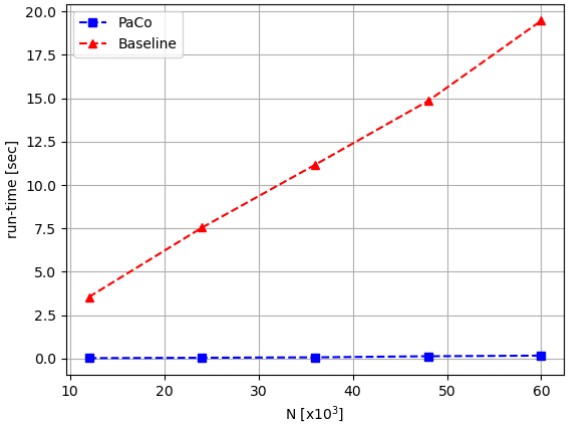

Figure 4: CollegeMsg Dataset: Runtime comparison between baseline and PaCo with varying link count, using $\delta = 30$ and $k = 4$.

Similar to our last experiment with reddit's dataset, here in Fig 4, we observe PaCo's runtime remaining low and flat, while the baseline's runtime increases linearly with the number of links, highlighting PaCo's scalability and efficiency in processing larger datasets.

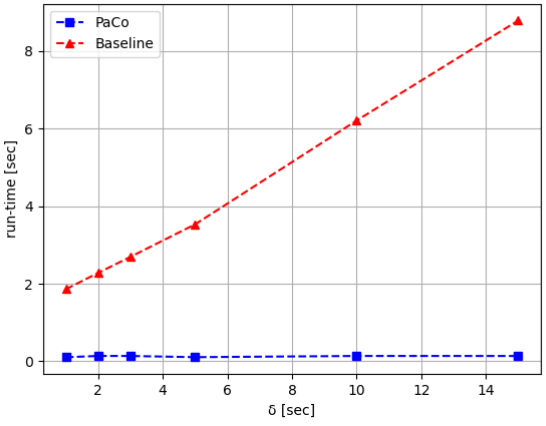

Figure 5: CollegeMsg Dataset: Runtime comparison between baseline and PaCo. Varying δ with $K = 3$.

In Fig 5, we can see that, PaCo's runtime remains consistently low, whereas the baseline's runtime grows with δ, reinforcing PaCo's effectiveness in handling temporal data with varying time intervals.

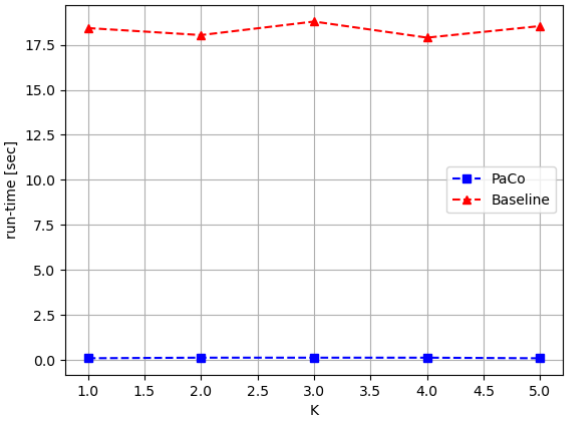

Figure 6: CollegeMsg Dataset: Runtime comparison between baseline and PaCo. Varying K with $\delta = 30$.

Here in Fig 6 as well we can see that, PaCo's runtime advantage is evident as it remains relatively flat across different values of K, suggesting that its efficiency is less affected by path length than the baseline algorithm.

We have implemented the same methodology for three more datasets. The experimental results show that in scenarios with varying link counts, time intervals δ, and path counts K, PaCo algorithm maintains a comparatively low and stable runtime, which does not increase significantly with the size or complexity of the dataset.
The experimental results of the rest of the datasets are as follows:

6.3 email-Eu-core Temporal Network

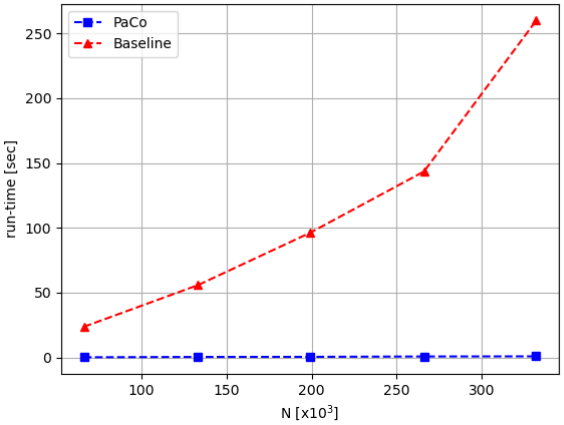

Figure 7: email-Eu-core Dataset: Runtime comparison between baseline and PaCo with varying link count, using $\delta = 30$ and $k = 4$.

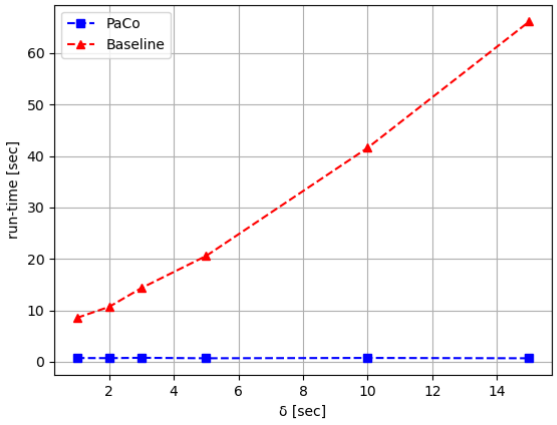

Figure 8: email-Eu-core Dataset: Runtime comparison between baseline and PaCo. Varying δ with $K = 3$.

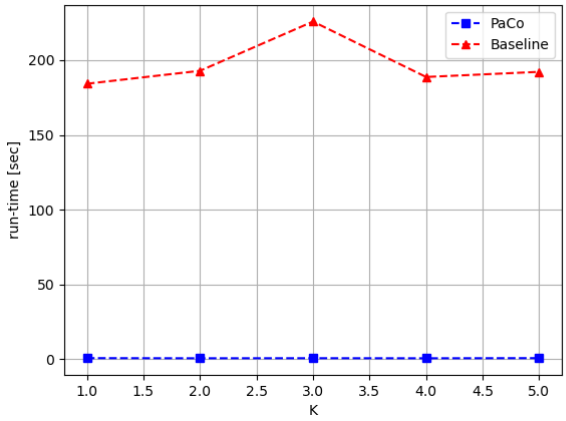

Figure 9: email-Eu-core Dataset: Runtime comparison between baseline and PaCo. Varying K with $\delta = 30$.

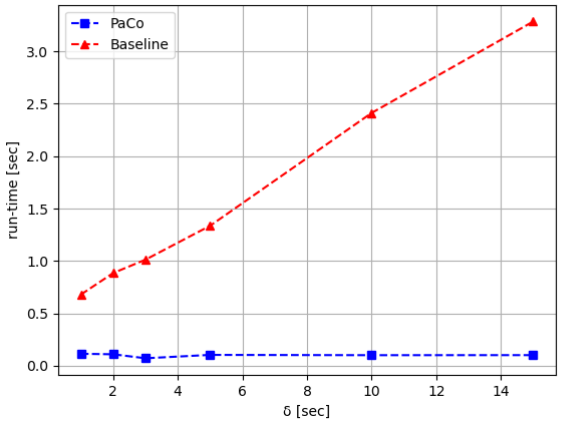

Figure 11: Bitcoin Alpha Dataset: Runtime comparison between baseline and PaCo. Varying δ with $K = 3$.

6.4 Bitcoin Alpha Trust Weighted Signed Network

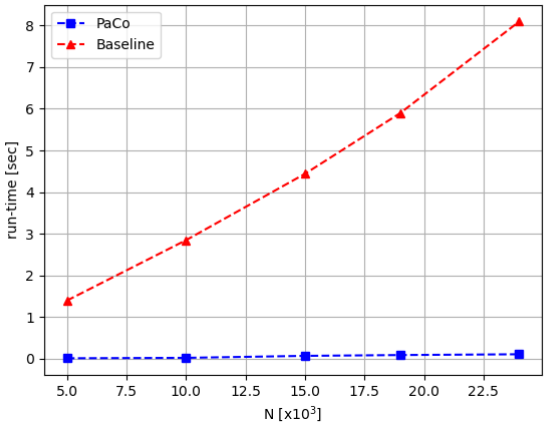

Figure 10: Bitcoin Alpha Dataset: Runtime comparison between baseline and PaCo with varying link count, using $\delta = 30$ and $k = 4$.

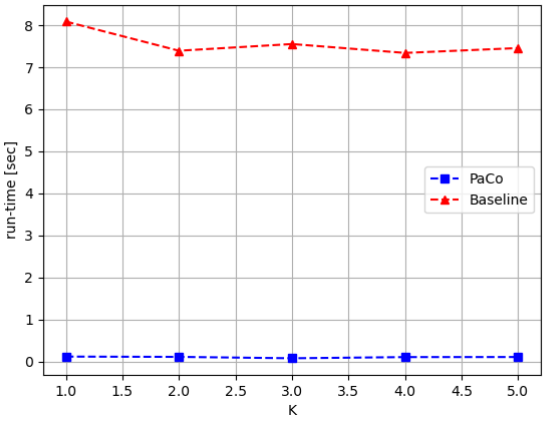

Figure 12: Bitcoin Alpha Dataset: Runtime comparison between baseline and PaCo. Varying K with $\delta = 30$.

6.5 Bitcoin OTC Trust Weighted Signed Network

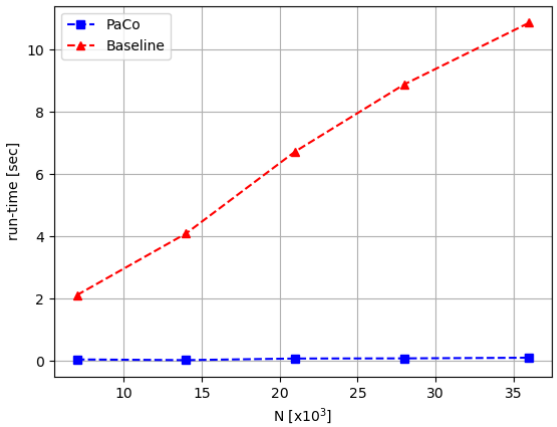

Figure 13: Bitcoin OTC Dataset: Runtime comparison between baseline and PaCo with varying link count, using $\delta = 30$ and $k = 4$.

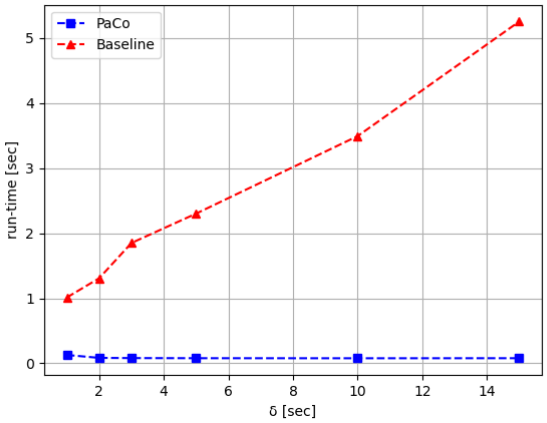

Figure 14: Bitcoin OTC Dataset: Runtime comparison between baseline and PaCo. Varying δ with $K = 3$.

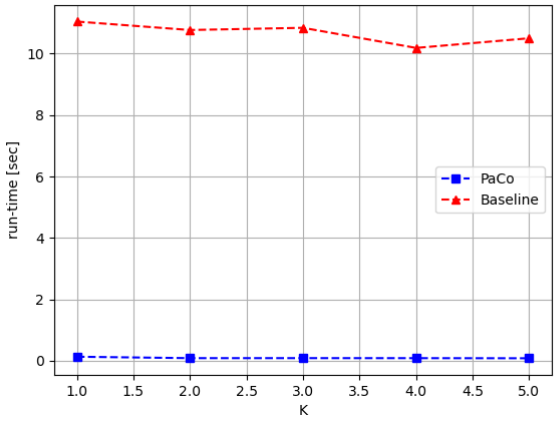

Figure 15: Bitcoin OTC Dataset: Runtime comparison between baseline and PaCo. Varying K with $\delta = 30$.

From the experimental results we can conclude that this uniform efficiency across different parameters indicates the algorithm's scalability and robustness, suggesting it is well-suited for analyzing large-scale temporal networks.

7 CONCLUSION

In this study, we successfully applied the paper's [3] algorithm to several temporal network datasets, showcasing its robustness and efficiency. Our tests validated that the algorithm outperformed the baseline approach, especially when processing large amounts of temporal network data. This work has emphasized the algorithm's potential in various applications and deepened our understanding of its practicality. As the paper [3] has described, the parallelization method can be further explored for even higher efficiency in future studies, and extending the algorithm's usefulness by modifying it for real-time streaming data is a promising direction.

REFERENCES

[1] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[2] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. (1999). http://ilpubs.stanford.edu:8090/422/.

[3] Luka V Petrović and Ingo Scholtes. 2021. Paco: Fast counting of causal paths in temporal network data. In *Companion Proceedings of the Web Conference 2021*. 521–526.

[4] Martin Rosvall and Carl T Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences* 105, 4 (2008), 1118–1123.