Testing the reliability of temporal path finding algorithms

Social Network Analysis for Computer Scientists — Course paper

Arya Deshmukh
s4182987@vuw.leidenuniv.nl
LIACS, Leiden University
Leiden, The Netherlands

Jesse van Haaster
s4485440@vuw.leidenuniv.nl
LIACS, Leiden University
Hillegom, The Netherlands

ABSTRACT

The analysis of temporal networks, where edges represent interactions between nodes marked with their times of occurrence, plays a crucial role in understanding dynamic processes in real-world systems. However, for very large and dense temporal networks, counting all the causal paths efficiently remains computationally costly as path length and size of the time window increase. This paper discusses the performance evaluation and scalability of the proposed PaCo algorithm for path counting problems using causality on several real-world temporal network datasets: StackOverflow, Ubuntu, EU-email and Reality Mining against a baseline algorithm.

Our experiments demonstrate that PaCo consistently outperforms the baseline in terms of execution time and scalability. While the baseline algorithm has exponential runtime growth with increasing maximum path length and sliding time window (delta) for most datasets, PaCo has a much more gradual runtime increase that can handle larger and more complex networks. However, we notice that in very dense networks, such as StackOverflow, the computational challenge for PaCo arises for path lengths greater than 4 and sliding time windows greater than delta = 900, where the memory and computational demands become too expensive. We provide a comprehensive discussion of these limitations. Overall, this study highlights the effectiveness of PaCo for scalable temporal path analysis while identifying key challenges that arise when processing highly dense and temporally extensive datasets.

ACM Reference Format:
Arya Deshmukh and Jesse van Haaster. 2024. Testing the reliability of temporal path finding algorithms: Social Network Analysis for Computer Scientists — Course paper. In *Proceedings of Social Network Analysis for Computer Scientists Course 2024 (SNACS '24).* ACM, New York, NY, USA, 7 pages.

1 INTRODUCTION

In analyzing networks with time-stamped data, such as social interactions, communications, or transportation flows, it's essential to understand not only who interacts with whom but also when these interactions occur. Traditional network models often assume that links represent direct influence between nodes and that paths represent indirect influence, but they ignore the timing of these interactions. This creates a gap in network analysis, particularly for time-sensitive applications. In reality, for a sequence of interactions to represent influence over time—what we call a causal path—each interaction must occur sequentially and within a limited time frame. Without considering these timing constraints, network analysis may identify paths that do not exist in a meaningful, time-respecting way. For instance, imagine a social network where Person A shares information with Person B, and Person B then shares it with Person C. The information will only spread from A to C if Person B's interaction with C happens after their interaction with A, and ideally within a certain time window. If the sequence or timing is wrong, a causal path cannot be said to exist from A to C. This timing constraint is crucial in various real-world applications, such as tracking the spread of information, mapping infection routes in epidemiology, and determining feasible travel routes in transportation.

To address this, higher-order network models incorporate these time-based dependencies, allowing for more accurate analysis of who can influence whom in networks. However, these models require a detailed enumeration of causal paths, and existing methods for finding these paths are limited in scalability and efficiency. Due to the computational expense, higher-order models cannot yet be applied to many large, time-sensitive datasets in social, biological, and technical systems. Therefore, there is a clear need for an efficient approach to count causal paths, ensuring that meaningful analysis can be conducted in large temporal datasets.

[5]In this paper, we propose a novel algorithm that efficiently counts causal paths in time-stamped network data. Our algorithm operates in linear time with respect to data size and is designed to handle causal paths that occur within a predefined maximum time difference. The approach assumes that time-stamped links are ordered chronologically and builds paths incrementally by checking if each interaction can logically extend an existing path within the allowed time window[3]. This efficient handling of time windows and incremental path-building allows the algorithm to count paths rapidly, even in extensive datasets. Our contributions are as follows:

- We introduce an existing efficient algorithm to count causal paths in time-stamped networks, supporting higher-order network models.
- The algorithm leverages a sliding time window to limit unnecessary computations, achieving scalability in handling large datasets.
- We demonstrate the utility of this algorithm by applying it to diverse datasets and analyzing the performance across various network characteristics, such as density, network size, and time intervals.

SNACS '24, Master CS, Fall 2024, Leiden, the Netherlands

This work provides a crucial step toward making higher-order network analysis accessible and efficient in time-sensitive applications, paving the way for new insights in fields such as social network analysis, epidemiology, and transportation.

2 RELATED WORK

In recent years, network analysis has increasingly focused on capturing temporal aspects of interactions, as timing often plays a crucial role in determining influence and connectivity. Traditional network models, which aggregate all connections without regard to time, fail to capture this time-sensitive nature of interactions, leading to potentially misleading conclusions about paths and influence. To address the limitations of static models, researchers have explored higher-order network models that account for the sequence and timing of interactions. Lambiotte et al. (2019)[2] provided a foundation for higher-order models, which allow for a more realistic representation of indirect influence by focusing on sequences of interactions that occur in a specific order. Such models have shown promise in capturing influence flows in social networks and understanding information or disease spread, but they also bring computational challenges, particularly as data sizes grow. Several approaches have attempted to find or estimate causal paths in temporal networks. Early efforts focused on heuristic-based or shortest-path methods to approximate influence paths (Kempe et al. 2000)[1]. However, these methods often sacrifice accuracy or completeness, especially in dense or complex datasets. Other work, such as that by Paranjape et al. (2017)[4], focused on counting network motifs to understand temporal structures, but motifs differ from causal paths as they lack strict time-ordering rules between links. Building on these developments, Petrović and Scholtes (2021)[5] introduced the PaCo algorithm, which directly addresses the challenge of counting causal paths in time-stamped data. PaCo provides an efficient approach for finding causal paths within a given time constraint, leveraging a sliding time window to handle large datasets effectively. However, PaCo's application has primarily been demonstrated in a limited set of examples, and there remains a need to examine its performance across varied network characteristics and datasets. Our work builds upon and extends these approaches. We aim to evaluate the effectiveness and scalability of PaCo in diverse time-stamped datasets with varying network densities and structures. By doing so, we provide a comprehensive understanding of how PaCo performs in real-world settings, further demonstrating its potential for enabling higher-order network models in large, time-sensitive datasets. This work also sheds light on the algorithm's practical implications and limitations, contributing to ongoing efforts to make temporal network analysis more feasible and impactful in a range of applications.

3 PRELIMINARIES

To understand the points made in this paper it is important to first explain a few key concepts. First and foremost, the goal of this problem is to evaluate the performance and outputs of the PaCo algorithm on various different algorithm. We will test the algorithm based on the amount of time it takes to find all causal paths, and by comparing the influence of measures such as centrality on the speed of the algorithm. As stated by Petrovic [5], the runtime for PaCo scales linearly when increasing the maximum time difference between nodes, but will scale exponentially when increasing maximum causal path lengths. It is important to know up to what path length PaCo can still efficiently collect all paths, as the exponential increase in runtime can cause the algorithm to become inefficient past a certain path length very quickly. Because the runtime of the algorithm is completely reliant on the used parameters, there is no standard time complexity for the PaCo algorithm. However, the worst case time complexity is as follows: $O\left(N \cdot K^2 \cdot \left[m_\delta \cdot \delta \cdot \lambda_1^{K-2} + \lambda_1^K\right]\right)$ With N being the size of the dataset, K being the maximum causal path length, δ being the maximum time interval between links, $\delta 1$ being the density of the graph and m being the maximum amount of links with the same timestamp. What this time complexity largely boils down to is that K has an exponential impact on the time complexity of the algorithm, meaning it has the largest impact on the time complexity. The datasets used in this paper all contain links in the following form (s,d,t), where s,d are the linked nodes, and t is the timestamp given to the link. A link is part of another link's causal path if and only if: The new link is within the bounds of the maximum allowed path length(e.g. the amount of links already attached to the current link), and the time interval between the two links is smaller than the maximum allowed time difference.

4 APPROACH

To provide a better understanding of the used algorithm, we will first explain step by step how it goes to work.
The algorithm works as follows:
First, the maximum path length K, and the maximum time interval δ between links is determined. These parameters are important for deciding which nodes can be linked in the same path. It is assumed that the links in the given dataset are chronologically ordered. The algorithm starts at the first link and iteratively moves through the entire set of links. For each link, a new dictionary is initialized. In this dictionary, the length of all the causal paths ending with this link are stored, with maximum length K and maximum time interval δ. If a link is found outside of the time interval, it is removed from the dataset, as it is no longer possible that this link is part of any additional causal paths. After all causal paths for a certain link are found, they are added into the main dictionary. This repeats for every link in the network, and results in a dictionary containing all possible causal paths.
In the original paper, the algorithm is tested on a single temporal network dataset, by measuring the total runtime of the algorithm under different circumstances such as maximum path length, dataset size and maximum time interval, and comparing PaCo's speed to that of a baseline model. In this paper, we will give a more extensive evaluation of the algorithm. We will do this by using various different temporal networks and seeing how well PaCo performs on them. By experimenting with PaCo on different datasets, we hope to find how reliable the algorithm is at efficiently collecting causal path lengths across datasets, and see whether the algorithm is scalable across different types of datasets. We will also, similar to the original paper, try different hyperparameter settings for PaCo to test the influence of the maximum path length and the maximum time interval on different temporal networks.

5 DATA

For our experiment, 4 different datasets will be used. First, the dataset from the original dataset will be used to compare the performance on their hardware to our hardware. The dataset comes from the Reality Mining project and contains a network based on contact between students. We will then use three more datasets: The EU-email dataset, the AskUbuntu dataset and the StackOverflow dataset. These 3 datasets are all directed temporal networks that were downloaded from the Stanford Network Analysis Project (SNAP). Links to all 4 datasets can be found in the appendix. In the table below we list up some of the features of these datasets. We

Dataset	nodes	Edges	graph density	average time interval (seconds)
Reality Mining	96	1,086,404	0.025	18
EU-Email	986	332,334	0.026	209
AskUbuntu	159,316	964,437	2.35×10^{-5}	234
StackOverflow	2,601,977	63,497,050	5.35×10^{-6}	4

document the density of the graph to show how connected each graph is. In the table we can see that the Reality Mining and the email datasets have a similar density, with a substantial amount of the possible links being made. The AskUbuntu and StackOverflow datasets on the other hand are very sparsely linked. Due to the large amount of nodes in these datasets, it makes sense that nodes only have a connection to a small portion of the complete set of nodes. documenting the average time interval between nodes was important to make sure that the time intervals chosen when running the algorithm actually make sense for the current dataset. By measuring the average time interval we can use time intervals that are on the correct scale for the dataset. If this value were to have been kept the same in the algorithm for all datasets, it would give a very unfair comparison of the datasets, as some would have way more potential links than others. For the email dataset for example, it would be unlikely that more than 4 emails would be sent between users within 30 minutes, as response times for emails usually take a lot longer. Therefore it makes sense to increase the maximum time delta to take this into account.

6 EXPERIMENTS

The following experiments were ran on the SSH network of the University of Leiden, which seriously reduced the runtime of the algorithms, as it can make use of the GPU to boost computing power. For each dataset, hyperparameters are used that are deemed fit for the dataset. The performance of PaCo is tested for different maximum path lengths and time intervals. The influence of both hyperparameters on the runtime of the algorithm is then evaluated and compared to the runtime of a baseline algorithm.

In the appendix the graphs with runtime comparisons of the PaCo runtimes against the baseline runtimes of each dataset are provided. In each graph a multitude of different time intervals with different maximum path lengths is documented. For each dataset it will be explained what the results mean and what can be derived from them. It should be noted that the PaCo algorithm was run on more pathlengths than the baseline for most datasets, as the baseline runtimes were very large and became exponential for most datasets.A link to the used notebooks and the datasets can be found in the appendix.

6.1 Reality Mining

First the dataset used in the original paper is tested. In the original paper, it is shown that with a time interval of 30 minutes, the runtime starts rapidly growing exponentially from
pathlengths > 3 onward. However when looking at the results in our case, it can be seen that the model seems to grow logarithmically in time duration for the PaCo algorithm, which is very far from the expectation. Even for a pathlength of up to 10 the model did not start running exponentially, even with time intervals that were larger than 30 minutes. When we look at the amount of paths that are found between pathlengths of 9 compared to 10, it is clear that only few extra paths are found, which could explain the small amount of increase in runtime.

On top of this difference in results from the original paper, the recorded runtimes are also completely different, with our model running in seconds, where in the original paper the model is reported to calculate it in hours (this is an assumption, it is never explicitly stated what the h for runtime stands for in the original paper).

When looking at the results of the baseline algorithm runs it is clear that this algorithm does run in exponential time. It should be stated that the run for pathlength 4 with time interval 30 minutes (1800 seconds) is an estimation of time duration, as the python kernel continuously kept crashing when the calculation reached 2 minutes. It is therefore possible that it could take even longer to compute. A possible explanation for this issue could be that the resulting graph is too large to handle for the computer. The two points left out for pathlength 5 were also left out due to the kernel crashing. It was tried repeatedly to rerun the model for these parameters, but it kept consistently crashing, indicating it was not possible.

One thing that is clear from these two graphs is that the time it takes to run the algorithm grows linearly with the size of the maximum time interval. This is in line with the original paper, which states that the runtime grows linearly when increasing the time delta. We can see in the graph that all time intervals are separated by an equal amount of time, indicating that the difference in time stems largely from the increase in time interval.

6.2 Email

We now move on to the EU email dataset. This dataset is smaller than the dataset used in the original, containing 332,334 edges compared to the dataset used in the original paper, which has a little over 1 million edges.

PaCo outperforms the baseline across the range of maximum path lengths from 2 to 6. Its' runtime exhibits a more complex, nonlinear relationship with the maximum path length. For shorter path lengths up to around 6, the PaCo runtime increases gradually, but beyond that point, the runtime starts to rise more sharply, especially for the larger delta values of 24,000 and 30,000 seconds. This nonlinear growth pattern in the PaCo algorithm's runtime suggests that it may be showing signs of exponential growth as the maximum path length increases. The steeper increase in runtime at higher path lengths indicates that the algorithm's processing time is growing at a faster rate, potentially in an exponential manner.

Similarly, the baseline algorithm also demonstrates an exponential growth for path lengths higher than 4. The runtime fluctuates somewhat at shorter path lengths, which is odd, as the runtime should not be able to decrease for larger values. It is possible that this has something to do with the server that the algorithm is being run on. There is also some overlap between graphs representing different time deltas.
The second graph, which analyzes runtime against the varying maximum time delta thresholds, provides further insights. The PaCo algorithm maintains its performance advantage over the baseline across all the delta values tested (12,000, 18,000, 24,000, and 30,000 seconds). However, the gap between the two algorithms narrows as the delta value increases.
This suggests that the PaCo algorithm's relative efficiency advantage is more pronounced when the time window for causal connections is smaller (lower delta values). As the delta is increased, allowing for longer time periods between causally linked events, the difference in runtime between PaCo and the baseline diminishes.
This observation, combined with the nonlinear growth pattern seen in the first graph, indicates that the PaCo algorithm may be particularly well-suited for scenarios where the causal relationships in the temporal network are more time-sensitive and occur within tighter time windows. As the time window for causal connections is expanded, the PaCo algorithm's performance advantage becomes less pronounced compared to the baseline.
In summary, the updated performance graphs suggest that the PaCo algorithm is exhibiting signs of exponential growth in runtime as the maximum path length increases, particularly for larger delta values. The PaCo algorithm's relative efficiency advantage is more pronounced when the time window for causal connections is smaller, highlighting its potential suitability for applications where the temporal dynamics are more time-sensitive.

6.3 AskUbuntu

We now move on to the AskUbuntu dataset, this dataset contains a similar amount of edges to the Reality Mining dataset, but contains a lot more nodes. This dataset is also a lot less connected than the Reality Mining dataset, as can be seen in the table, the graph density is significantly smaller than the Reality Mining network. The dataset contains answers to questions, comments to questions, and comments to answers. Because the amount of users in the dataset is very large, and every user is only active in a small community, the graph density for this dataset is very low.
In the graph for PaCo's performance we can see that the runtime increases exponentially for this dataset. However due to the efficiency of the PaCo algorithm, the amount of time it takes to finish calculations is still very fast even for maximum path lengths up to 10. The efficiency of the algorithm is especially visible when we compare it to the runtime of the baseline algorithm.
The baseline algorithm for this dataset returns results containing a lot of abnormalities. First, the algorithm seems to run in linear time, the increase in runtime is steady with no major increases at larger path lengths. Second, our results show an anomaly for the time delta of 18,000 (5 hours) where running this delta takes much longer than running for the time delta of 24,000. This is odd, as theoretically, a larger time delta should always lead to the model finding at least the same amount, if not more paths than a lower delta. Lastly it should be stated that the model was not able to run a time delta of 24000 with a path length of 6, as the kernel could not handle this.
The most important take-away is that the runtime of the baseline is significantly larger than the runtime of PaCo, showing the effectiveness of PaCo in finding temporal paths efficiently. However due to the runtime of PaCo increasing exponentially, from a certain path length the runtimes of both models will likely converge to approximately the same runtime.
What is interesting to see is that the amount of time saved between the two algorithms for the AskUbuntu dataset is a lot bigger than the time saved on the Reality Mining dataset. This shows that the effectiveness of the PaCo algorithm is heavily reliant on the type of dataset that is used, and depends on the amount of interaction between users.

6.4 StackOverflow

The StackOverflow dataset is a very massive dataset, containing 2,6 million nodes and 63,5 million edges based on StackOverflow answers to questions, comments to answers, and comments on questions. In this sense the dataset is very similar to the AskUbuntu dataset. The biggest difference lies in the size of the dataset, and the average time interval between subsequent edges.
For the StackOverflow dataset, due to the sheer size of the dataset, it was only possible to use the PaCo algorithm on this dataset, as the code would immediately crash anytime we attempted running it on the baseline model. This alone already shows the usefullness of PaCo, as PaCo was able to find the temporal paths in the data, regardless of the size of the dataset. When we look at the graph for the PaCo algorithm, the runtime seems to increase in linear time. One thing that is visible is that the graph stops increasing in runtime for the lowest time delta, but is still increasing for the time delta of 900. This has likely got to do with the fact that it is unlikely that more than 6 users answered one another in a timespan of just 5 minutes. Compared to runtimes of other datasets it is clear that the large size of the dataset does have a big impact on the runtime of PaCo. However in spite of this, the algorithm still returns its' results in a reasonable amount of time.

7 DISCUSSION

In our report we test the efficiency of the PaCo algorithm across different datasets, however a few things should be noted about our work.

7.1 Script

First, the original authors of the PaCo algorithm did not provide a script for their code, therefore the code used in this project is based on the pseudocode they explain in their paper. Because of this, it is possible that the code we created works slightly differently from the code used by the authors. The resulting found temporal paths however should not be affected by this difference, as the key

features for the algorithm explained in both their paper and pseudocode, have also all been applied in our version of the code.

7.2 Hyperparameters

Secondly, in our code we apply varying values for the time interval based on the average time interval between consecutive nodes. However, the datapoints being close together does not necessarily mean that connected nodes are always close together. For the StackOverflow dataset for example, the maximum time delta was kept relatively low, as increasing this value too much would lead to very high run times for the algorithm.

7.3 Kernel Crash

Kernel crashes observed during the execution of temporal path-counting algorithms, particularly on larger datasets like StackOverflow, Ubuntu, and Reality Mining, highlight the inherent computational challenges of these tasks. These datasets are characterized by either their substantial size or their high density, which exacerbate the computational burden as the maximum path length and sliding time window (delta) increase. The baseline algorithm, designed with less efficient mechanisms, struggles with the exponential growth in the number of temporal paths as network complexity rises. This inefficiency becomes particularly apparent in dense networks, where each node participates in numerous interactions over short intervals, leading to a combinatorial explosion in candidate paths.

Even with its optimizations, PaCo encounters similar limitations on the StackOverflow dataset when the maximum path length exceeds 4 or when the sliding time window grows significantly. High-density temporal networks, like StackOverflow, generate an overwhelming number of candidate paths within relatively short time intervals. This results in a surge in memory usage and computational overhead, eventually leading to kernel crashes. For instance, with a sliding time window of 900, the algorithm must account for a vast range of interactions, causing an exponential increase in the number of potential paths that need to be processed. This overwhelms system memory and computational resources, despite PaCo's more efficient design compared to the baseline algorithm.

The fundamental challenge lies in the sheer scale of temporal path enumeration in dense networks. As delta increases, the algorithm must evaluate a broader range of interactions within the specified time window, which dramatically amplifies the computational demands. In highly dense datasets, the impact is even more pronounced, as a larger delta aggregates an immense number of interactions, compounding the complexity. The combinatorial nature of these interactions results in a bottleneck, pushing the system beyond its computational and memory capacities.

While PaCo demonstrates significant improvements in efficiency over the baseline algorithm, its performance is still constrained by these intrinsic computational demands. These limitations underscore a trade-off in temporal network analysis: increasing the sliding time window provides a more comprehensive view of temporal dynamics but also introduces prohibitive computational costs.

7.4 Differences in Performance

One of the biggest points of discussion stems from the differences in runtime compared to the original dataset. The authors of paper clearly show the runtime of PaCo increasing exponentially for the Reality Mining dataset, however when we run the same model with the same hyperparameters, we do not experience the same behaviour, as our model stagnates in runtime for increasingly larger path lengths. On top of that, the runtimes we experience are way faster than the runtimes shown in the original paper, both for the baseline and the PaCo algorithm. This leaves it up for debate what caused our runtimes to be that much faster. It is possible that the code we wrote to function as PaCo functions slightly differently from the algorithm used for the original paper, as we had to base our algorithm on the provided pseudocode. It could also be that the runtimes in the original paper were not documented correctly. In any case, further validation of the algorithm is needed to verify what the actual runtime of the algorithm is.

8 CONCLUSION

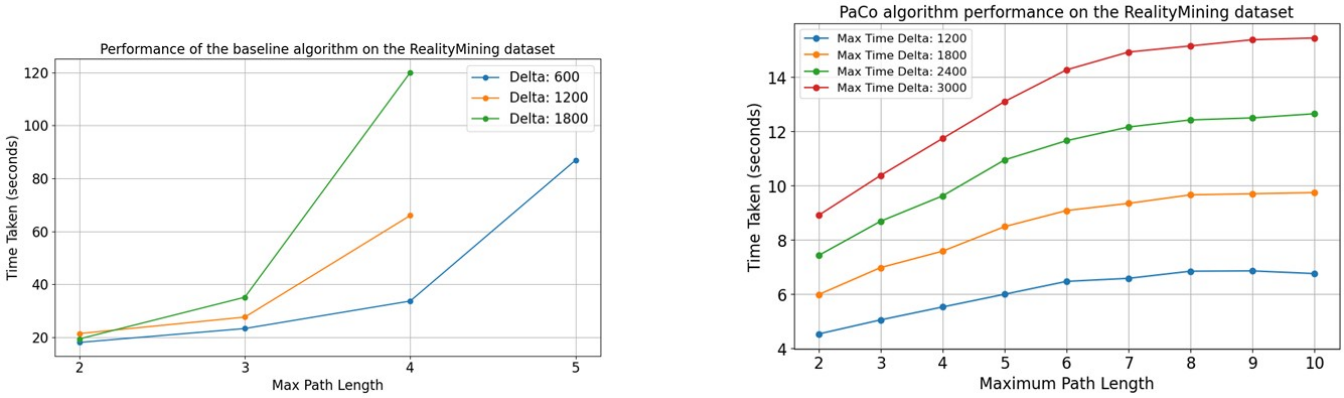
In our work we provide an extensive oversight of the efficiency of the PaCo algorithm. The algorithm was found to consistently outperform a baseline algorithm for finding temporal paths across various different datasets. Even for large values of maximum path length and large datasets the algorithm held up well. In the future, this algorithm could be used in large temporal datasets to model the complex temporal relations in a dataset, and prove to be useful in enabling these datasets to use meaningful network measures.
It did become clear that the performance of the dataset is dependent on the type of dataset, as its efficiency fluctuated for datasets with different densities and size. Due to the large difference in performance that was measured between the original paper and our experiment, further research should be done to verify the validity of this algorithm.

REFERENCES

[1] David Kempe, Jon Kleinberg, and Eva Tardos. 2000. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 137–146.
[2] Renaud Lambiotte, Martin Rosvall, and Ingo Scholtes. 2019. Higher-order models in network science. *The European Physical Journal Special Topics* 227, 10-11 (2019), 1237–1249.
[3] N. Masuda and R. Lambiotte. 2016. *A Guide to Temporal Networks.* CRC Press. https://www.maths.ox.ac.uk/node/26371
[4] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. 2017. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 601–610.
[5] Luka V. Petrović and Ingo Scholtes. 2021. PaCo: Fast Counting of Causal Paths in Temporal Network Data. In *Proceedings of the Web Conference 2021 (WWW '21 Companion)*. ACM / IW3C2, 521–526. https://doi.org/10.1145/3442442.3452050

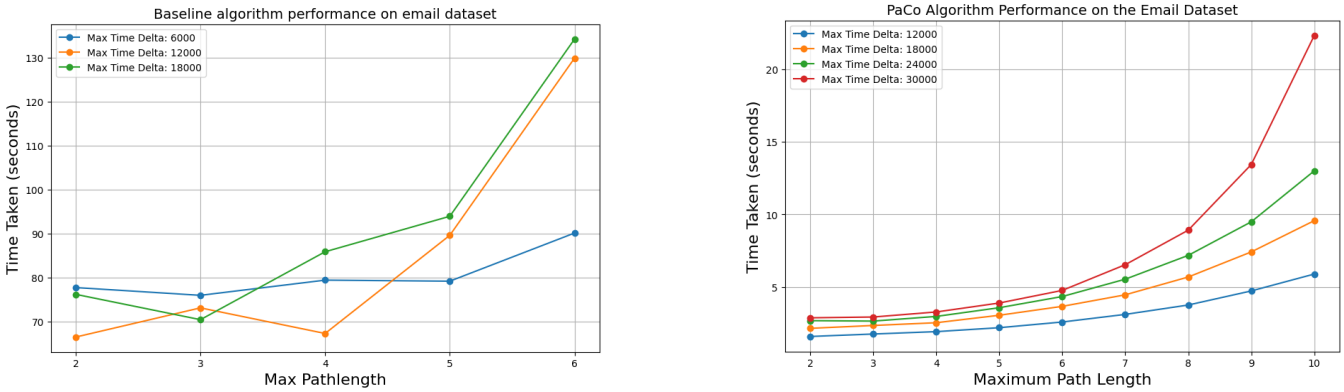
9 APPENDIX

- Reality Mining Dataset
- StackOverflow dataset
- EU-email dataset
- AskUbuntu dataset
- GitHub Repository



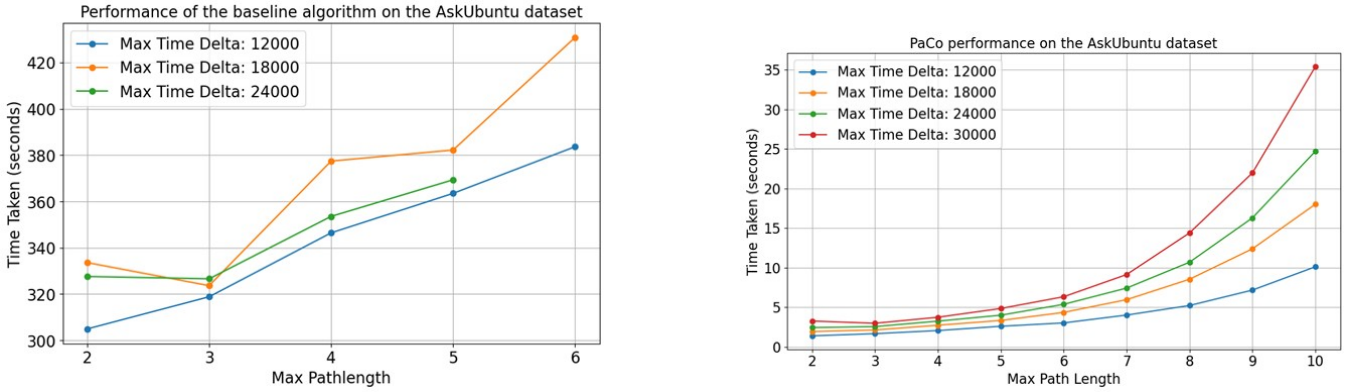
(a) baseline **(b) PaCo**

Figure 1: Performance of PaCo and the baseline algorithm on the Reality Mining dataset



(a) baseline **(b) PaCo**

Figure 2: Performance of PaCo and the baseline algorithm on the EU-email dataset



(a) Baseline **(b) PaCo**

Figure 3: Performance of PaCo and the baseline algorithm on the Ubuntu dataset

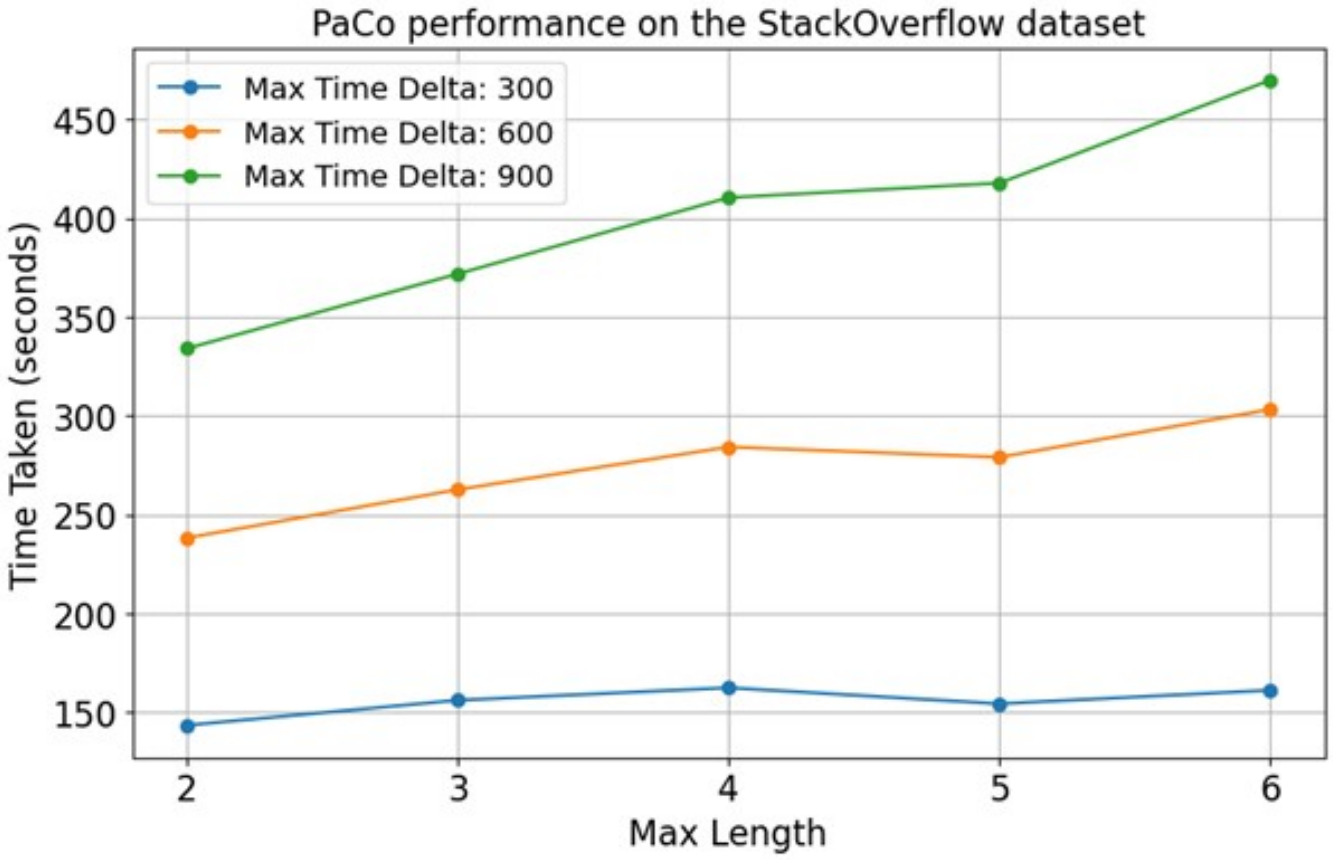


Figure 4: Performance of PaCo on StackOverflow dataset