

Fast and Accurate Triangle Counting in Graph Streams using Predictions

Anonymous Author(s)

ACM Reference Format:

Anonymous Author(s). 2018. Fast and Accurate Triangle Counting in Graph Streams using Predictions. In *Proceedings of KDD (KDD 2024)*. ACM, New York, NY, USA, 7 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 REBUTTAL STAGE

In this document we are going to address questions of reviewers in the rebuttal stage, showing experiments with a fresh new implementation of our algorithm Tonic (different in runtimes from the one showed in the paper) that leverages optimized code and data structures, and how such implementation is going to impact to time of estimations. We divided the document in the following subsections:

- In 1.1 we provide a description of datasets used both in the paper, which was missing in the main manuscript, and of the new big datasets (up to ≈ 34.5 billion triangles, as in [9]) proposed to visualize scalability of the methods;
- In 1.2 we show experiments with new implementation for considered datasets;
- In 1.3 we give insights on the gain of using node-based representation rather than edge-based representation for the predictors;
- In 1.4 we conduct experiments on snapshot sequences using adversarial predictions;
- In 1.5 we show the overhead times for building MinDeg Predictor.

1.1 Datasets description

First of all, we are going to provide a brief description of the datasets used, both in the main paper and in this note. We recall that, from each dataset, we removed self-loops and multiple edges for consistency with past works. Most of the considered datasets have been fetched from [4, 6, 7].

Single graph:

- *Edit EN Wikibooks* contains the edit network of the English Wikipedia, containing users and pages connected by edit events. This dataset is also considered in [2];
- *SOC Youtube Growth* includes a list of all of the user-to-user links in Youtube video-sharing social network;
- *Cit US Patents* [3] represents the citation graph between US patents, where each edge $\{u, v\}$ indicates that patent u cited patent v (used also in [8]);

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

KDD 2024, 2024, Barcelona

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-XXXX-X/18/06...\$15.00

<https://doi.org/XXXXXXX.XXXXXXX>

- *Actors Collaborations* contains actors connected by an edge if they both appeared in a same movie. Thus, each edge is one collaboration between actors;
- *Stackoverflow* represents interactions from the StackExchange site "Stackoverflow". The network is between users, and edges represent three types of interactions: answering a question of another user, commenting on another user's question, and commenting on another user's answer;
- *SNAP LiveJournal* is a friendship network from LiveJournal free on-line community;
- *Twitter* [1, 5] comprises 4 single graphs of the Twitter following/followers network. For our experiments, we are going to consider each single of the 4 networks independently, and the merged overall network which results in ≈ 41 million nodes, ≈ 1.5 billion edges and ≈ 34.5 billion triangles. The final merged version of this dataset is also used in [9].

Snapshot sequence:

- *Oregon* is a sequence of 9 graphs of Autonomous Systems (AS) peering information inferred from Oregon route-views between March 31 2001 and May 26 2001;
- *AS-Caida* contains 122 CAIDA AS graphs, derived from a set of RouteViews BGP table snapshots, from January 2004 to November 2007;
- *as-733* are 733 daily instances which span an interval of 785 days from November 8 1997 to January 2 2000, from the BGP logs.

1.2 Extended Experiments

In the following, we are going to show results achieved with our new Tonic implementation, both on a subset of the datasets considered in the paper, and on the new Twitter graphs described in Subsection 1.1. [We want to enhance the fact that the plots might slightly differ from the ones reported in our main paper because (due to lack of time) we ran a lower number of independent repetitions for all the algorithms (respectively, 20 for the subset of datasets in the papers, and 10 for the new Twitter datasets, instead of 50 as in the main paper). Also, for memory budget experiments, we include also the tick representing the 0.5% of memory budget in the x-axis, not considered in the main paper.]

In Figure 1 we show the accuracy of the considered algorithms for a subset of the datasets considered in our paper. We plot the global relative error vs the percentage of memory budget allowed over the graph edges. As the accuracy has been already taken into consideration within the main paper analyses, here we want to focus on the estimation times: Tonic is always able to substantially outperform WRS (we recall that Chen is not showed here due to impractical implementation that make its times at least 4 times bigger than the others). The dataset statistics (nodes, edges, and triangles) are reported in the subtitle of each subplot. In Figure 2

we depict the same metric for Twitter datasets. We did not manage to obtain results for Chen in time. We notice that Tonic is always and almost always able to outperform WRS with the Exact Oracle and MinDeg Predictor respectively, for all the considered memory budgets and datasets. Moreover, for what concerns the times, Tonic is always faster than WRS excluding very low memory budgets, showing a much milder slope and hence, in practice, able to scale better with respect to worst-case scenarios analyzed in theory.

1.3 Overhead of node-based oracles

Node-based oracles (we studied the combination of min-degree, but many others are possible and we leave that as an open future direction) are in some way encoding edge features in a much more succinct representation. Hence, the resulting oracles' overhead is significantly lower than the one used in edge-based derived oracles. So, starting from our edge-based MinDeg Predictor representation, we computed node-based MinDeg Predictor in such a way that the latter is going to contain the highest degree nodes matching the size of the number of distinct nodes present in the former. Then, in the algorithm, when we query a node-based MinDeg Predictor, we will receive the min-degree of the nodes in the incoming edges (if both nodes are present), or otherwise return 0 (light edge). [\[do we want to emphasize the fact that potentially the two are minimally different? E.g., for a top node degree which is not present in MinDeg edge-based Predictor. In practice, we showed that this is rarely the case\]](#)

In Figure 3 we give some insights about savings of space and times for the Twitter datasets. On the left, we plot the number of entries stored in memory (in log scale) and at the same time we show how the memory budget is varying according to the one set in Figure 2, where the number of "entries" here represent the number of edges stored in memory. Note that the size of Edge-based representation is always $m/10$ entries, since we are retaining the top 10% of heaviest edges. Notice that the space occupied by node-based representation is completely negligible with respect to the memory budget; [\[hence, we could neglect such space to save the oracle when comparing to WRS\]](#). On the right of Figure 3, we show the gain factor when using node-based representation with respect to edge-based ones both from a memory point of view (the ratio between edge and node representation size on the left) and from a time point of view, meant as the time to load the lookup table in memory by our Tonic algorithm.

To be more precise, since in node-based representation each entry is made by two numbers in the lookup table (each row is $u \deg(u)$) instead of three numbers in the one for edge-based representation (each row is $u v f(u, v)$, where f can be a generic function, e.g., heaviness or min degree), the space saving in terms of byte in memory is even bigger. To give some numbers, for Twitter merged (last row of Figure 3) we have edge-based representation size in memory of $\approx 2.6GB$, and oracle's time to be loaded of $\approx 60s$, while for node-based representation we have respectively $\approx 2.2MB$ and $\approx 0.12s$.

1.4 Adversariety of the Predictions

In this subsection, we focus on comparing results when Tonic is provided with bad predictions. More specifically, we build the

most adversarial predictor, that is we set the lightest edges (resp. lowest degree nodes) to be the heaviest (resp. highest degrees), and preserving the sizes of the oracles. Again, we want to emphasize that the node-degree predictor is taking only some dozens of the node-degrees (the highest degrees for the best MinDeg predictor, and the lowest ones for the adversarial MinDeg predictor). In the specific, we are only keeping 67 out of 10670, 82 out of 16301 nodes respectively for Oregon and AS-CAIDA training graph, i.e., the first ones in the sequence. In Figure 4 we report the results containing best and adversarial oracles and predictors. We note that Tonic even with its adversarial predictors and oracles is almost always outperforming Chen using the exact oracle in AS-CAIDA dataset. In any case, Chen with adversarial oracle is performing absolutely poor with respect to other methods. Furthermore, Tonic with adversarial predictors shows comparable to WRS in AS-CAIDA, and for more of the half of the sequence in Oregon. [\[I feel like there might be confusion between adversarial predictor / oracle / edge-based, node-based. We may define a better notation here.\]](#)

1.5 Time to build Predictors/Oracles

In the following, we present the results showing the comparison between building OracleExact and MinDeg Predictor. We recall that the former requires to solve exactly the problems of counting the number of triangles in the graph, maintaining a lookup table with the value of the heaviness for each edge e , i.e., the number of triangles adjacent to edge e ; the latter, is built with a fast pass on the stream and stores the degrees of the nodes in the lookup table. In the end, the edge table (resp. node table) is sorted by heaviness (resp. degree) and the top heaviest edges (resp. highest degrees nodes) are retained. In Table 1 we report times and for building oracles and predictors, averaged over 5 independent runs. Due to lack of time, we considered only a subset of the overall presented datasets. Note that, Tonic is able to outperform or being comparable to WRS in runtime even if we add times from Table 1 to algorithm's execution times in Figure 1 for most of the combinations of datasets and memory budgets allowed. [\[here a reviewer may object why we are not merging together times, should we explain that in single graphs this is not our main goal?\]](#)

REFERENCES

- [1] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. 2011. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th international conference on World Wide Web*. 587–596.
- [2] Justin Y Chen, Talya Eden, Piotr Indyk, Honghao Lin, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, Tal Wagner, David P Woodruff, and Michael Zhang. 2022. Triangle and four cycle counting with predictions in graph streams. *arXiv preprint arXiv:2203.09572* (2022).
- [3] Bronwyn H Hall, Adam B Jaffe, and Manuel Trajtenberg. 2001. The NBER patent citation data file: Lessons, insights and methodological tools.
- [4] Jérôme Kunegis. 2013. KONECT – The Koblenz Network Collection. In *Proc. Int. Conf. on World Wide Web Companion*. 1343–1350. <http://dl.acm.org/citation.cfm?id=2488173>
- [5] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a social network or a news media?. In *WWW '10: Proceedings of the 19th international conference on World wide web* (Raleigh, North Carolina, USA). ACM, New York, NY, USA, 591–600. <https://doi.org/10.1145/1772690.1772751>
- [6] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. <http://snap.stanford.edu/data>.
- [7] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. <https://networkrepository.com>

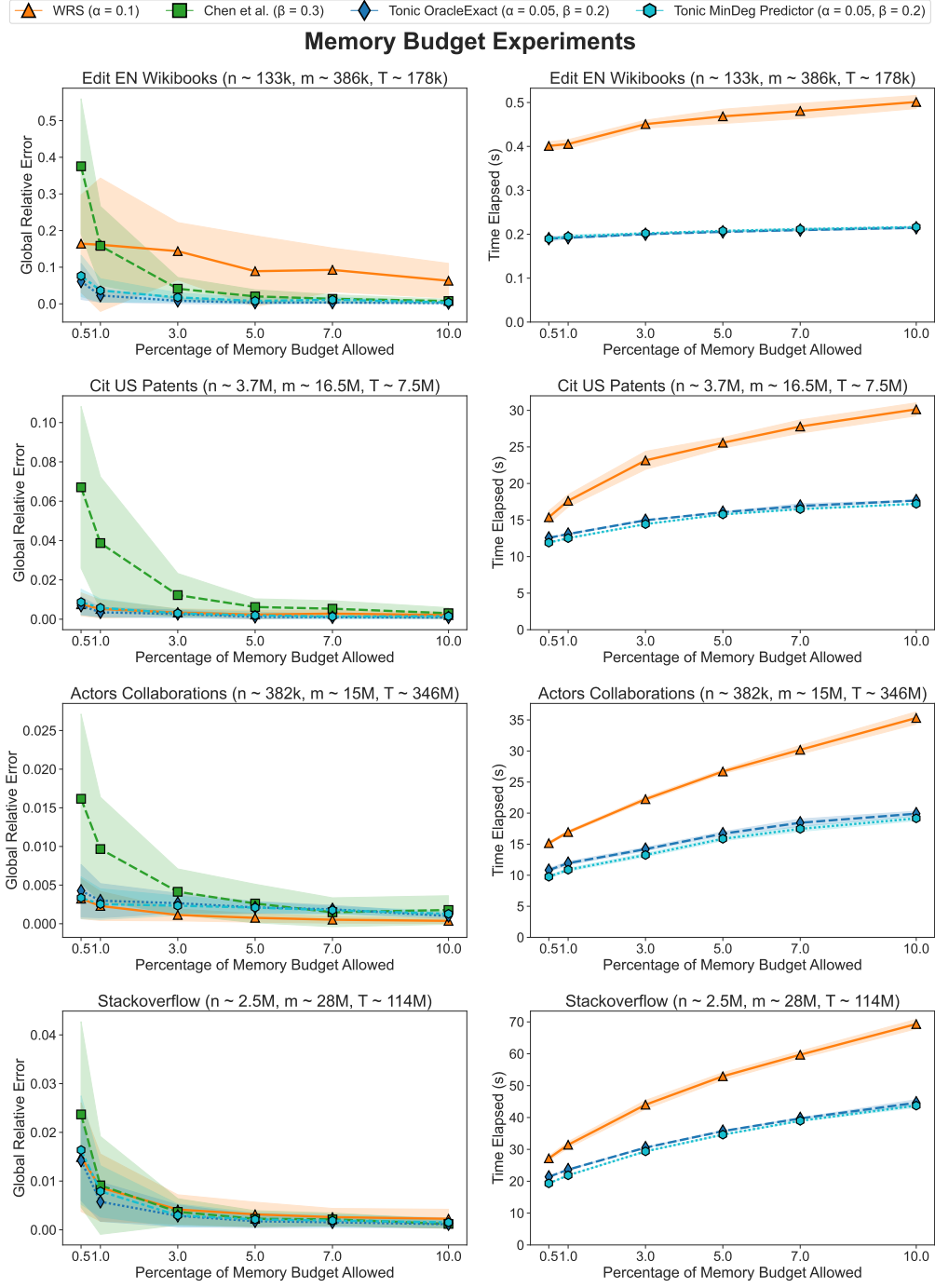


Figure 1: Error (left) and runtime (right) vs memory budget. For each combination of algorithm and parameter (including predictor for Tonic), the average and standard deviation over 20 repetitions are shown. The algorithms parameters are as in legend (for WRS and Chen et al. [2] they are fixed as in the respective publications).

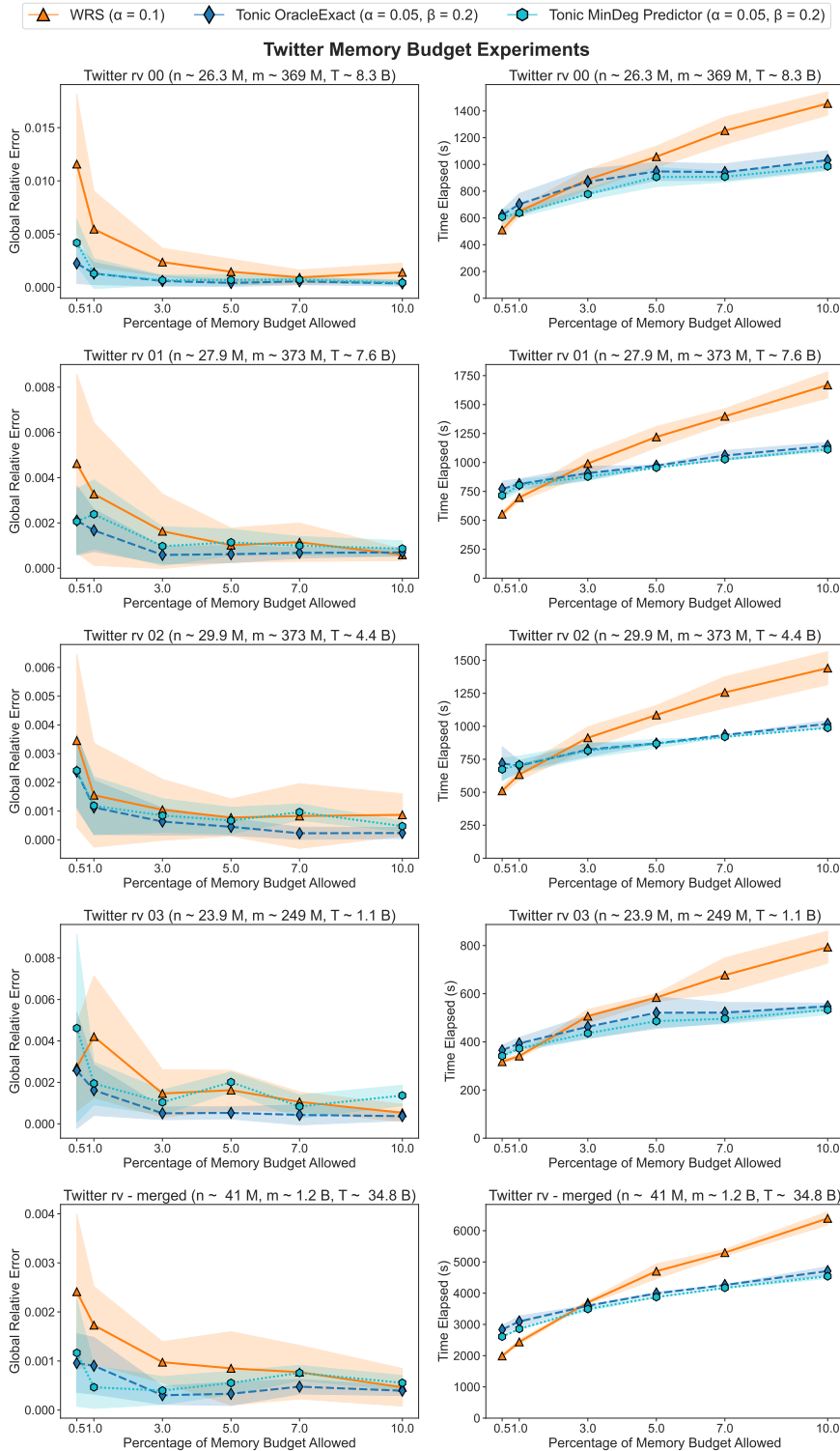


Figure 2: Error (left) and runtime (right) vs memory budget. For each combination of algorithm and parameter (including predictor for Tonic), the average and standard deviation over 10 repetitions are shown. The algorithms parameters are as in legend (for WRS they are fixed as in the respective publication).

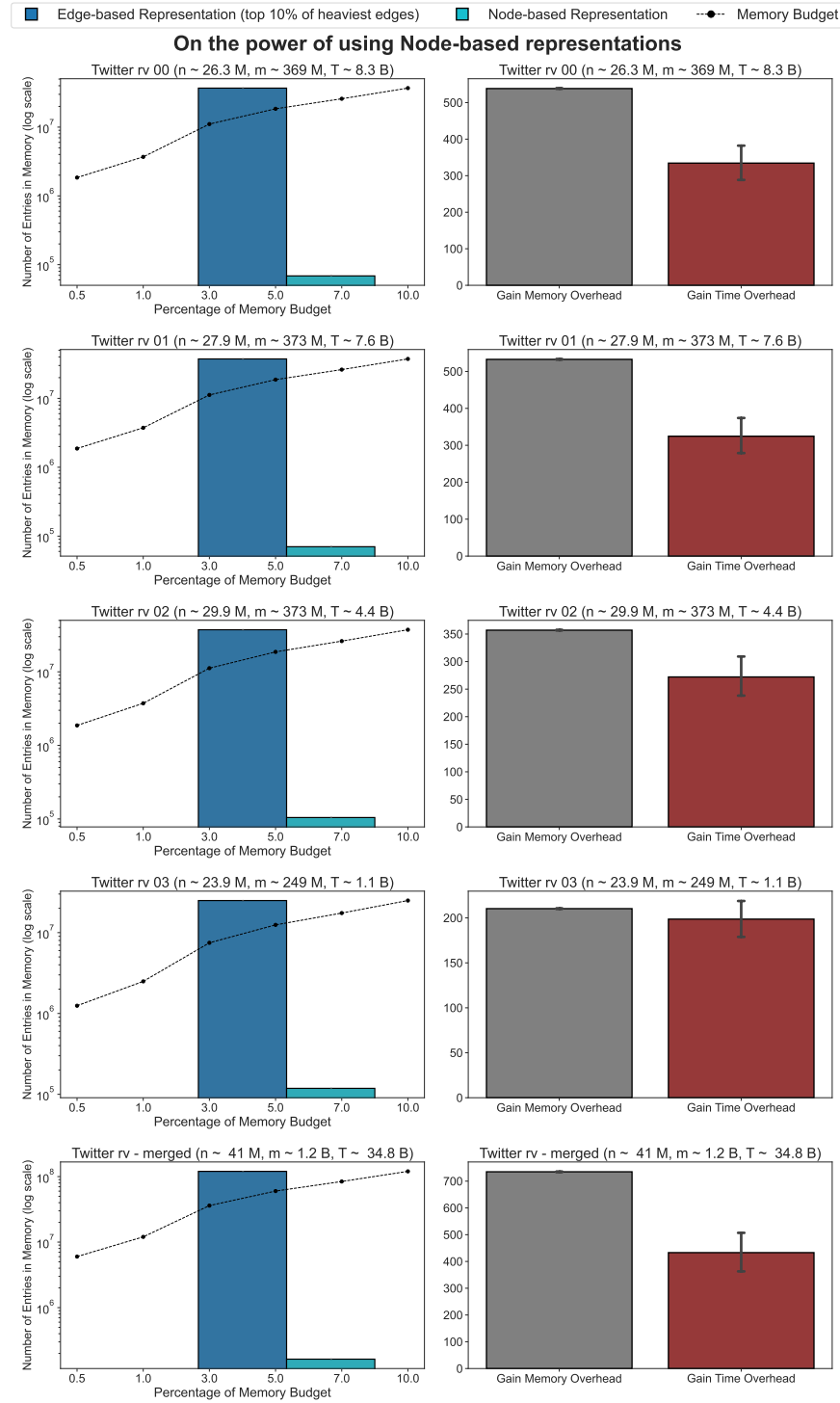


Figure 3: Number of Entries Stored in Memory (left) and Gain in Memory and Time (right) when using edge-based and node-based representations. The data have been collected from the same settings as in Figure 2.

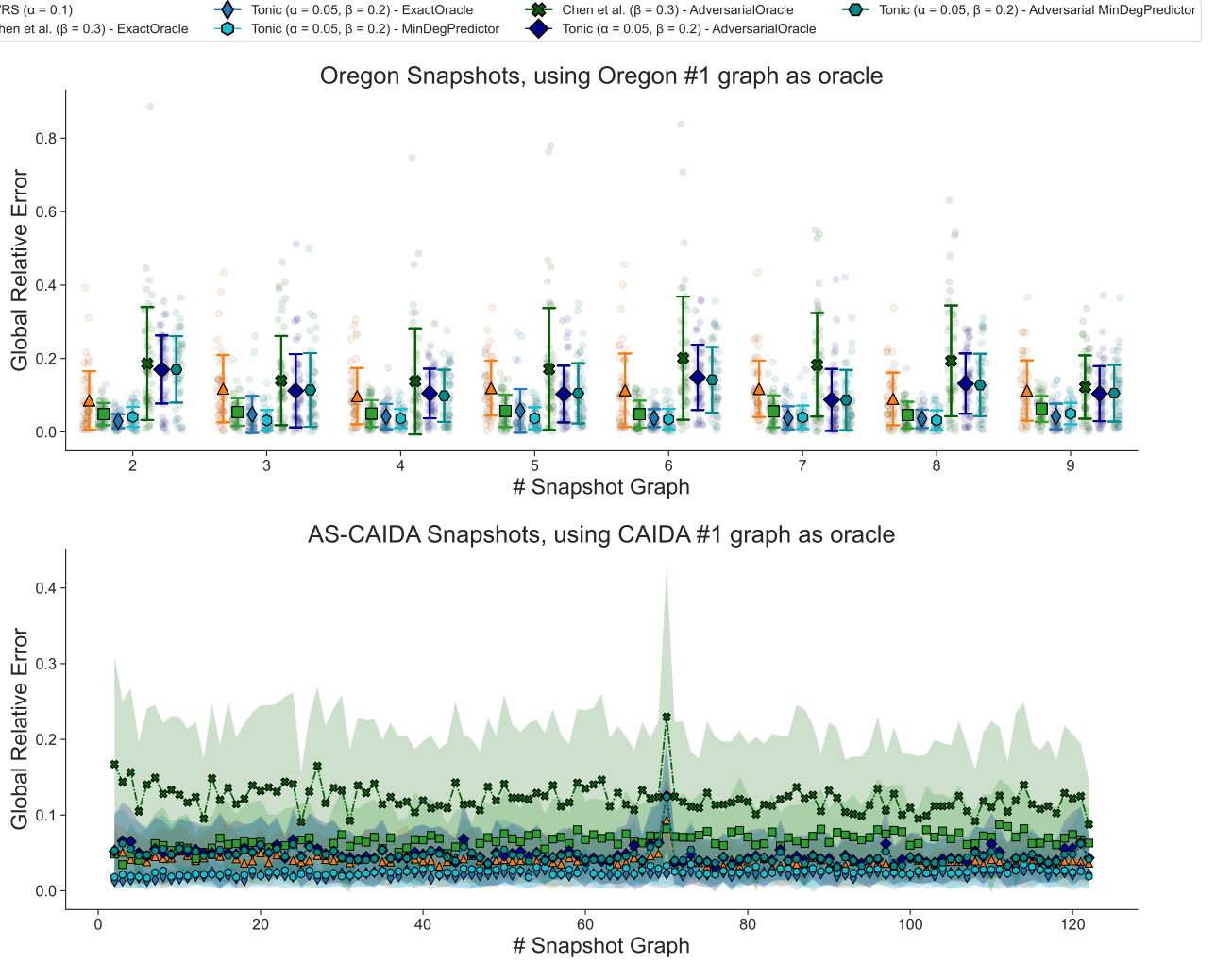


Figure 4: Error with snapshot networks with sequence of graph streams. In all cases the predictors are trained only on the first graph stream of the sequence (with results not shown on such graph stream). For each combination of algorithm and parameter (including predictor for Tonic), the average and standard deviation over 50 repetitions are shown. The algorithms parameters are as in legend (for WRS and Chen they are fixed as in the respective publications; for Tonic they are as chosen in the main paper).

Name	n	m	T	$t_{OracleExact}(s)$	$t_{MinDegPredictor}(s)$
Cit US Patents	3.7M	16.5M	7.5M	44.610 \pm 3.157	7.785 \pm 0.353
Actors Collaborations	382k	15M	346.8M	158.223 \pm 7.688	5.687 \pm 0.428
Stackoverflow	2.5M	28.1M	114.2M	792.415 \pm 28.596	11.271 \pm 0.123
Twitter 00	23.6M	369M	8.3B	TBD	TBD
Twitter - merged	41M	1.5B	34.6B	TBD	TBD

Table 1: Time to build OracleExact vs Time to build MinDeg Predictor. For each dataset, average and standard deviation over 5 independent repetitions are reported

- [8] Kijung Shin, Sejoon Oh, Jisu Kim, Bryan Hooi, and Christos Faloutsos. 2020. Fast, accurate and provable triangle counting in fully dynamic graph streams. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14, 2 (2020), 1–39.

- [9] Lorenzo De Stefani, Alessandro Epasto, Matteo Riondato, and Eli Upfal. 2017. Triest: Counting local and global triangles in fully dynamic streams with fixed memory size. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 11, 4 (2017), 1–50.