Fast and Accurate Triangle Counting in Graph Streams using Predictions

Anonymous Author(s)

1 REBUTTAL STAGE

In this document we present some experimental results, not present in the original paper, to address the questions raised by the reviewers. The experimental results are obtained with a new implementation of our algorithm `Tonic` (with different runtimes from the one showed in the paper) that leverages optimized code and data structures. Some of the presented experiments show how such new implementation impacts the running time. The document is organized in the following sections:

- Section 1.1 provides a description of datasets used both in the paper (missing in the original manuscript), and of a new large dataset (up to ≈ 34.6 billion triangles, as in [9]) proposed to better assess the scalability of the methods;
- Section 1.2 shows the results obtained with new implementation for considered datasets;
- Section 1.3 presents the results describing the gain of using a node-based representation rather than edge-based representation for the predictors;
- Section 1.4 describes the experimental results on snapshot sequences using adversarial predictions;
- Section 1.5 shows the overhead times for building the `MinDegPredictor`.

1.1 Datasets description

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

Single graphs:

- *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]);
- *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. In our experiments, we are going to consider each of the 4 single networks independently, and also a large network obtained by merging the 4 single networks, which results in ≈ 41 million nodes, ≈ 1.5 billion edges and ≈ 34.8 billion triangles. The final merged version of this dataset is also used in [9]. *These datasets were not considered in the original version of the manuscript.* The exact computation of the number of triangles on the final merged dataset, required for computing the ground truth, took ≈ 12, 4 hours.

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 RouteViews BGP graph 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 describe the results achieved with our new implementation of `Tonic`. The results are for a subset of the datasets considered in the paper, and also for the new Twitter graphs described in Section 1.1. Due to the time constraint for the rebuttal, our experimental setting is slightly different from the one considered in the original paper: we ran a lower number of independent repetitions for all the algorithms, instead of 50 as in the original paper. In particular, we run 20 repetitions for the subset of datasets in the papers, and 10 for the new Twitter datasets. For the memory budget experiments, we include also the setting of using 0.5% of memory budget (i.e., the x-axis of Figure 1 and Figure 2), which was not considered in the main paper.

Figure 1 shows the accuracy of the considered algorithms for a subset of the datasets reported in our paper. We plot the global relative error vs the percentage of memory budget allowed over the graph edges. The accuracy results are the same as reported in the main paper, so we focus on the estimation times instead. `Tonic` always substantially outperforms `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.

Figure 2 shows the same metrics for Twitter datasets. (We did not manage to obtain results for `Chen` in the time allowed for the rebuttal.) As for accuracy, we note that `Tonic` always outperforms `WRS` with the `ExactOracle`, and almost always outperforms `WRS` with the `MinDegPredictor`, for all the considered memory budgets and datasets. Moreover, for what concerns the running time, `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 analyses of the running time. Note that for small memory budget, `Tonic` significantly outperforms `WRS` in terms of approximation error, and the running time is slightly higher but still reasonable. For larger memory budgets, `Tonic` usually outperforms `WRS` in terms of approximation error (or is at least

comparable to it), while being faster. Therefore, in both scenarios Tonic provides an advantage over WRS. Moreover, note that our exact algorithm takes more than 12 ours, while Tonic running time is much lower, with 10-15x speed-up.

1.3 Overhead of node-based oracles

Starting from our *edge-based* MinDegPredictor representation, we computed *node-based* MinDegPredictor, where the latter contains the highest degree nodes of the whole graph, matching the size of the number of distinct nodes present in the former. Then, in our algorithm Tonic, when we query a node-based MinDegPredictor, we obtain the min-degree of the nodes in the incoming edges if both nodes are present in the predictor, or otherwise we obtain 0 (i.e., the edge is light). Node-based oracles are in someway 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. Note that in our node-based predictor we combine the degrees to obtain the minimum, but many others predictors based on the degree of nodes could be used (e.g., a linear combination with trainable parameters). We leave the exploration of such predictors as future direction.

Figure 3 shows the savings of space and times for the Twitter datasets when the node-based MinDegPredictor replaces the edge-based MinDegPredictor. 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 setted in Figure 2, where the number of "entries" here represents 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 edges. Also, 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 in terms of space used. 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 left bar) and from a time point of view, meant as the time to load the lookup table in memory by our Tonic algorithm, on right bar.

To be more precise, since in node-based representation each entry is made by two numbers in the lookup table (each entry is $(u; deg(u))$) instead of three numbers in the one for edge-based representation (each entry 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 Adversarial predictors

In this section, we focus on the results of Tonic when it is provided with an extremely bad predictor in snapshot sequences. 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 degree nodes), and preserving the sizes of the oracles. Again, we want to emphasize that the node-degree predictor in this case is taking only some dozens of the node-degrees (the highest degrees in the graph for the best MinDeg predictor, and the lowest ones for the adversarial MinDegPredictor). In the specific, we are only keeping 67 out of 10k, 82 out of 16k nodes respectively for Oregon and AS-CAIDA training graph, i.e., the first graphs 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 components, is almost always outperforming Chen with the *exact oracle* in AS-CAIDA dataset. In any case, Chen with adversarial oracle is performing absolutely poor with respect to other methods (while Tonic with the adversarial oracle is still competitive). Furthermore, Tonic with adversarial components shows comparable to WRS in AS-CAIDA, and for more of the half of the sequence in Oregon. In summary, the results show that Tonic is robust to poor information of adversarial oracles/predictors, thanks to its combination with waiting room sampling, while is able to take advantage of the information provided when they are useful.

1.5 Time to build predictors/oracles

In the following, we present the results showing the comparison between building OracleExact and MinDegPredictor. We recall that the former requires to solve exactly the problem of counting the number of triangles in the graph, and plus 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 for building oracles and predictors, averaged over 5 independent runs. Due to lack of time for rebuttal, we considered only a subset of the datasets. Note that Tonic is able to outperform or is 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.

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

[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.

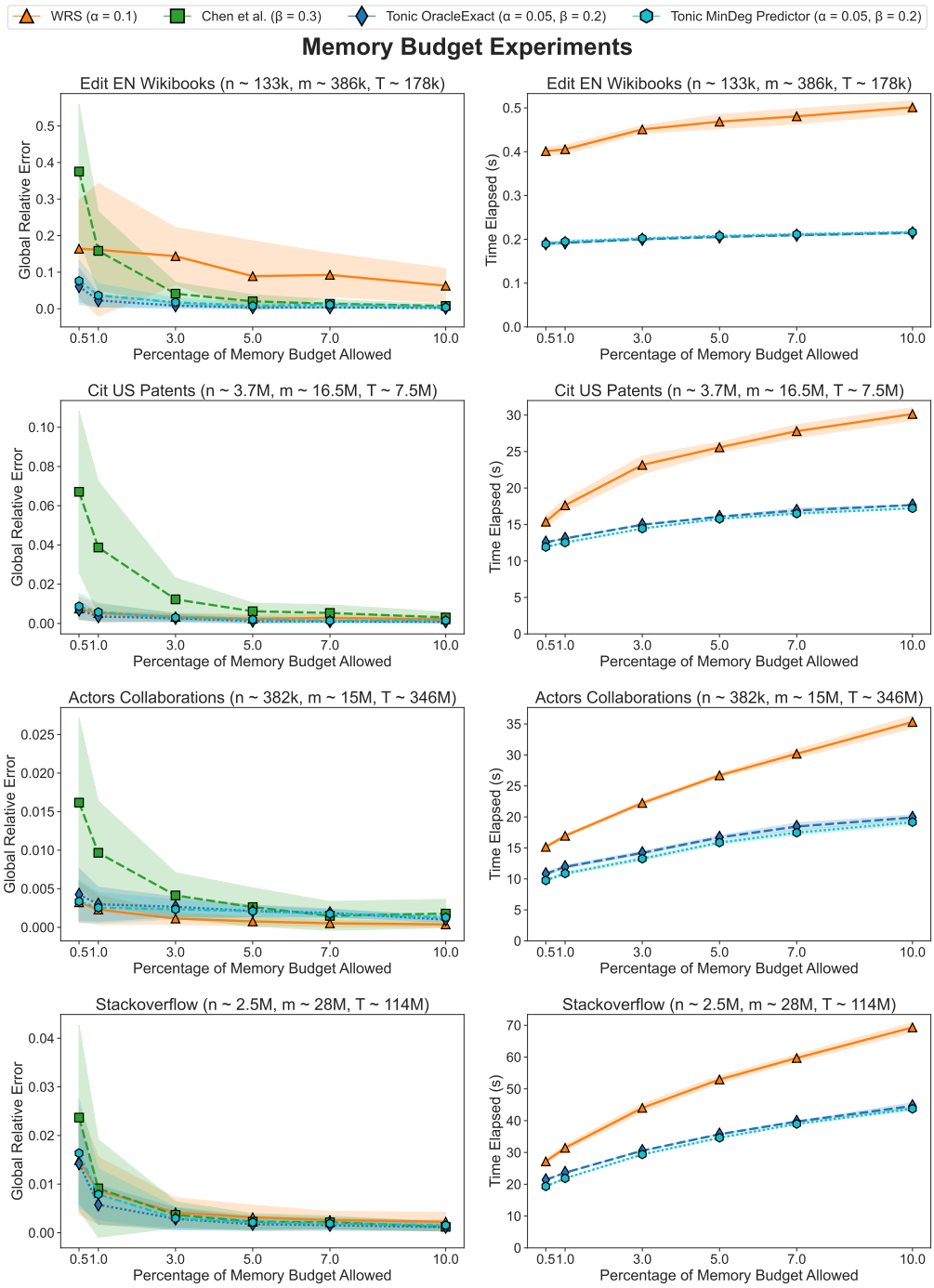

Figure 1: Error (left) and runtime (right) vs memory budget. For each combination of algorithm and parameter (including predictor for `Tonic`), the average and standard deviation over 20 repetitions are shown. The algorithms parameters are as in legend (for `WRS` and `Chen` they are fixed as in the respective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

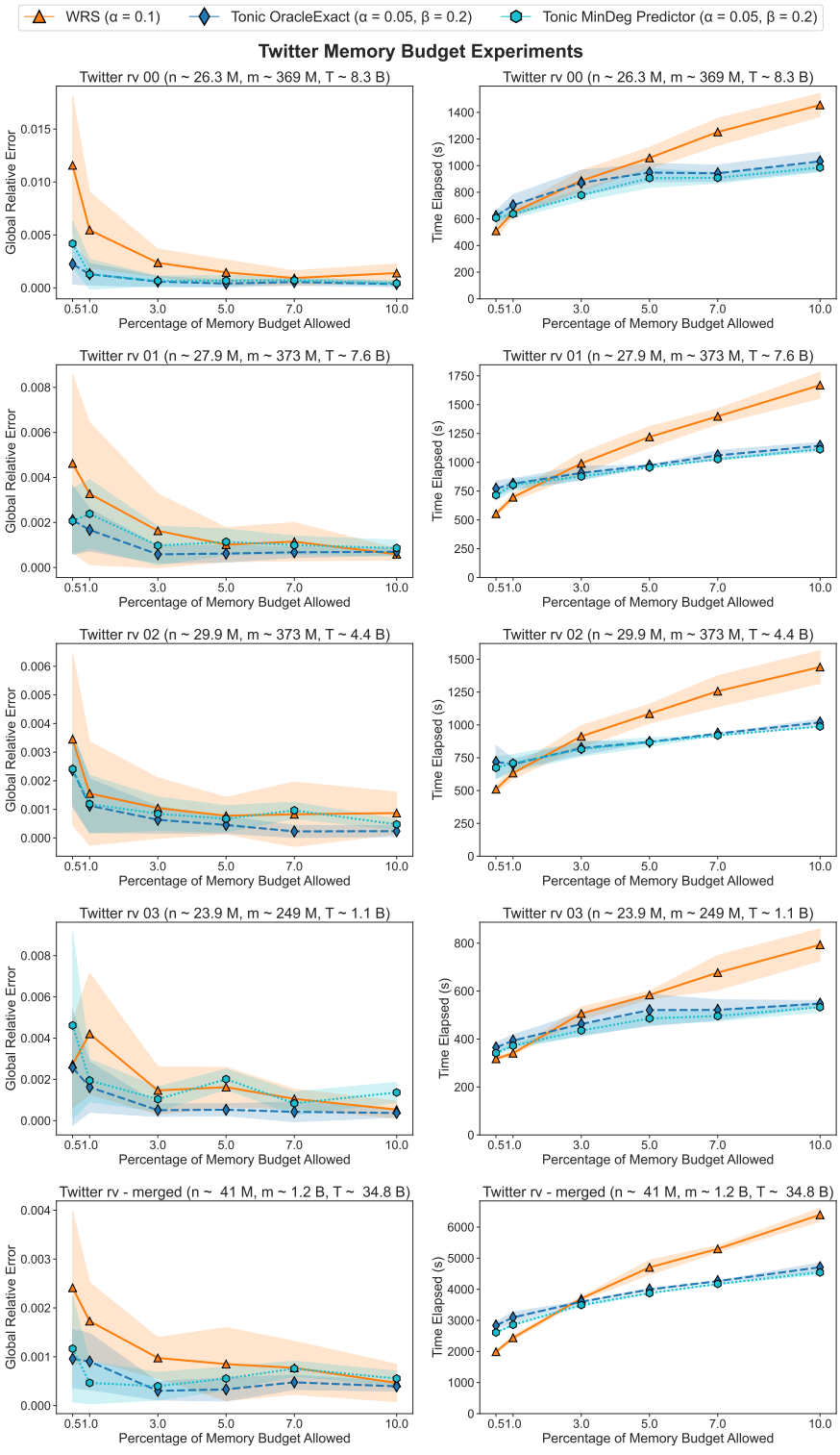

Figure 2: Error (left) and runtime (right) vs memory budget. For each combination of algorithm and parameter (including predictor for `Tonic`), the average and standard deviation over 10 repetitions are shown. The algorithms parameters are as in legend (for `WRS` they are fixed as in the respective publication).

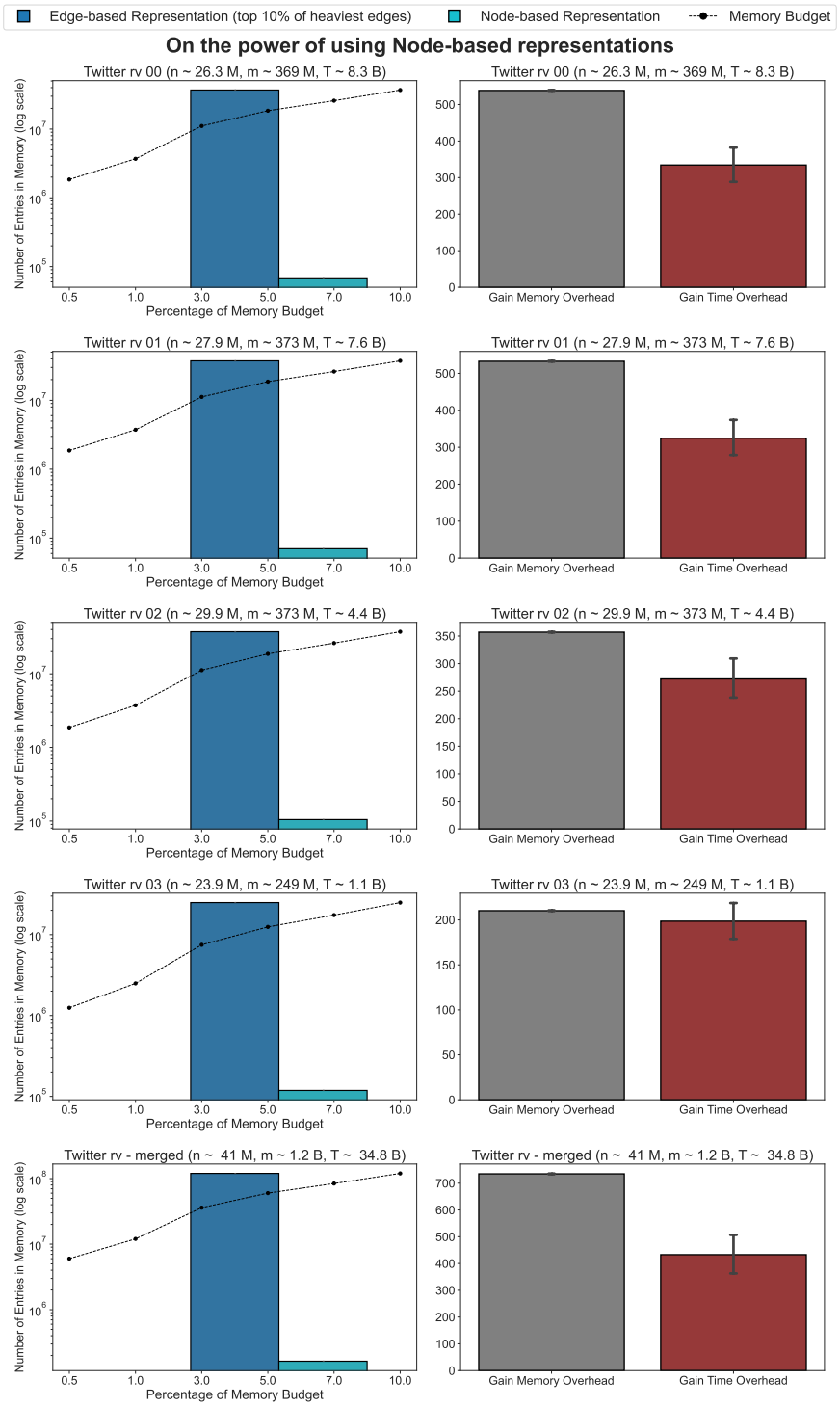

Figure 3: Number of Entries Stored in Memory (left) and Gain in Memory and Time (right) when using edge-based and node-based representations. The data have been collected from the same settings as in Figure 2.

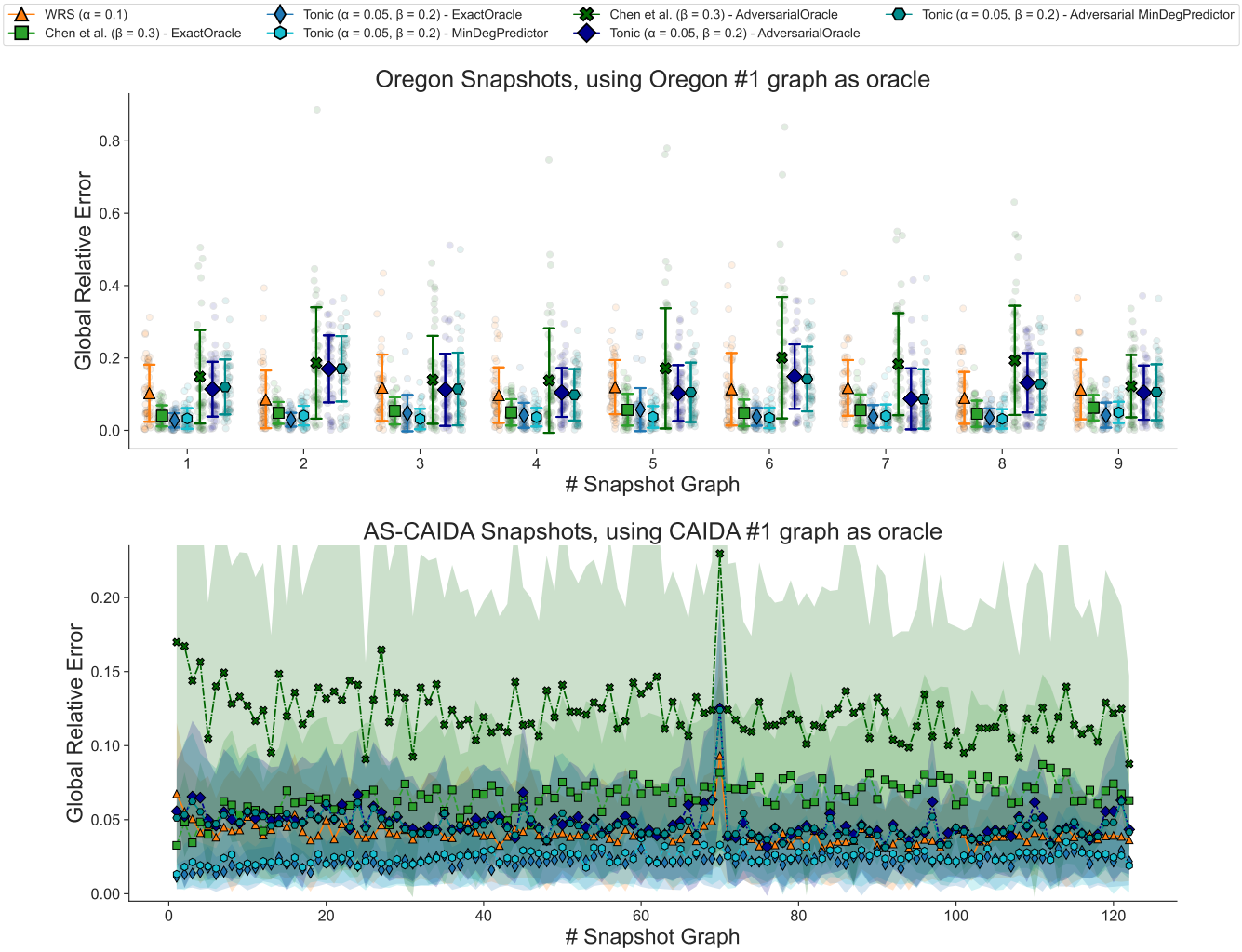

Figure 4: Error with snapshot networks with sequence of graph streams. In all cases the predictors are trained only on the first graph stream of the sequence (with results not shown on such graph stream). For each combination of algorithm and parameter (including predictor for `Tonic`), the average and 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 ± 3.157	7.785 ± 0.353
Actors Collaborations	$382k$	$15M$	$346.8M$	158.223 ± 7.688	5.687 ± 0.428
Stackoverflow	$2.5M$	$28.1M$	$114.2M$	792.415 ± 28.596	11.271 ± 0.123
Twitter 00	$23.6M$	$369M$	$8.3B$	$12.8 \times 10^3 \pm 582.497$	173.917 ± 2.3
Twitter - merged	$41M$	$1.5B$	$34.6B$	$58.4 \times 10^3 \pm 690.076$	547.720 ± 4.625

Table 1: Time to build `OracleExact` vs Time to build `MinDeg Predictor`. The results include also the times for sorting the lookup tables. For each dataset, average and standard deviation over 5 independent repetitions are reported