Reproducibility Report for ACM SIGMOD 2023 Paper: "T-FSM: A Task-Based System for Massively Parallel Frequent Subgraph Pattern Mining from a Big Graph"

CHENHAO MA, School of Data Science, The Chinese University of Hong Kong, Shenzhen, China
YUNHAI WANG, School of Computer Science and Technology, Shandong University, China

The paper proposes an efficient system called T-FSM for parallel mining of frequent subgraph patterns in a big graph. Specifically, T-FSM adopts a novel task-based execution engine design to ensure high concurrency, bounded memory consumption, and effective load balancing. This system also supports a new anti-monotonic frequentness measure called Fraction-Score, which is more accurate than the widely used MNI measure. The experiments use real-world synthetic datasets and compare the SOTA systems, including Scalemine [1], Pangolin [2], Fractal [3], Peregrine [4], and DistGraph [5].

1 INTRODUCTION

This report summarizes the reproducibility evaluation for the paper entitled "T-FSM: A Task-Based System for Massively Parallel Frequent Subgraph Pattern Mining from a Big Graph" by LYUHENG YUAN and DA YAN (Indiana University Bloomington, USA), WENWEN QU (East China Normal University, China), SAUGAT ADHIKARI (Indiana University Bloomington, USA), JALAL KHALIL (The University of Alabama at Birmingham, USA), CHENG LONG (Nanyang Technological University, Singapore), and XIAOLING WANG (East China Normal University, China). The paper appeared in the Proceedings of the 2023 ACM SIGMOD International Conference on Management of Data (SIGMOD'23). The experiments are reproducible and support the key findings of the paper. The experiments provide performance numbers similar or non-contradictory to the paper.

2 SUBMISSION

The repository containing scripts and reproducibility instructions for this paper is available at: https://github.com/lyuheng/T-FSM The data directory contains an example dataset, in lg format. The implementation of algorithms can be found in the serial directory, parallel directory, and fraction-score directory. The readme file contains a short description of how to run algorithms The AE-report file introduces the Artifact Description/Artifact Evaluation in detail.

3 HARDWARE AND SOFTWARE ENVIRONMENT

Table 1. Hardware & Software environment

	Paper	Repro Review
CPU	Intel(R) Xeon(R) Gold 6248R	Intel(R) Xeon(R) Gold 5218R
cores	48	40
GHz	3.0	2.1
RAM	64GB	256GB

3.1 Process

We followed the step-by-step instructions provided in the README file. The processes are listed as follows.

(1) Download the source code using git clone https://github.com/lyuheng/T-FSM.git

(2) Download all datasets from https://drive.google.com/drive/folders/1xxn35FTEKvV6JS7K2zKzx00X-RbkTPw-?usp=sharing
(3) Compile the code using "make" in the serial directory, parallel directory, and fraction-score directory, respectively.
(4) Run the program in serial directory by "./run -file ../data/mico.lg -freq 9480"
(5) Run the program in parallel directory by "./run -file ../data/mico.lg -freq 9480 -thread 4"

3.2 Results

(1) **Fig. 11. Our Implementation v.s. Original GraMi**. This experiment is *partially reproducible*. For the WordNet dataset, we can't get original Grami's results with the output 'Time's up!'. Generally speaking, the trend is consistent with the findings presented in the paper, although there are slight differences in time, we think it is due to the system hardware.

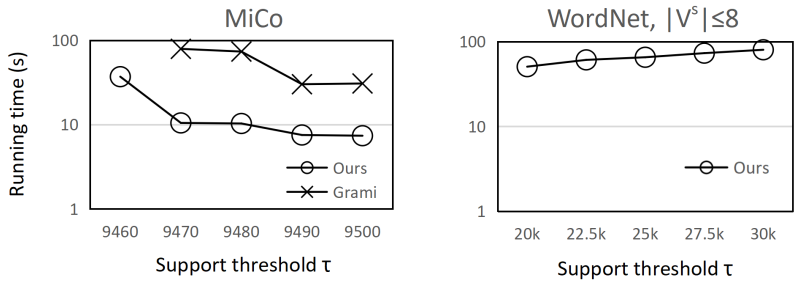

Fig. 1. Our Implementation v.s. Original GraMi (corresponding to original paper Fig. 11)

(2) **Fig. 12. T-FSM v.s. existing FSM systems with 32 Compers**. This experiment is *partially reproducible*. Although the running time does not exactly match the plot in the paper, the results do not violate the main point.

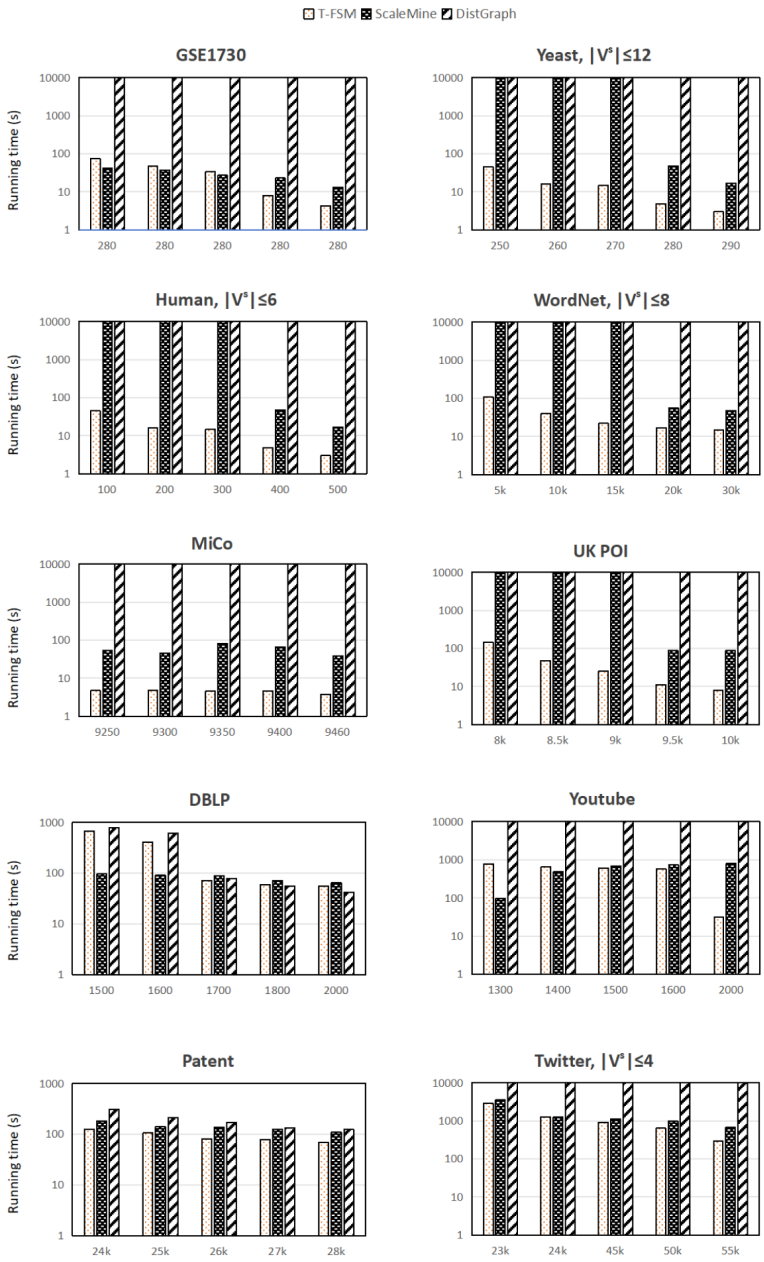

Fig. 2. T-FSM v.s. existing FSM systems with 32 Compers (corresponding to original paper Fig. 12)

- **Fig. 13. CPU Utilization Rates**. This experiment is *partially reproducible*. The CPU utilization of partial methods (T-FSM (paper's method), ScaleMine, and DistGraph) we achieved roughly matches the paper's result.

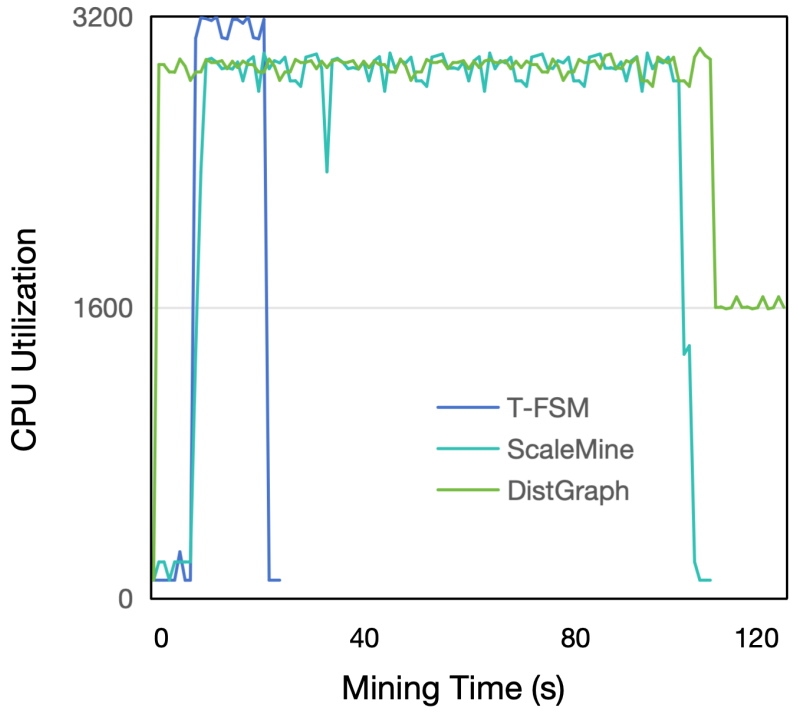

Fig. 3. CPU Utilization Rates (corresponding to original paper Fig. 13)

- **Fig. 14. Ablation Study**. This experiment is *partially reproducible*. The trend is mostly consistent with the findings presented in the paper. However, the blue line in the left picture and the purple line in the right picture have some differences with the paper's results.

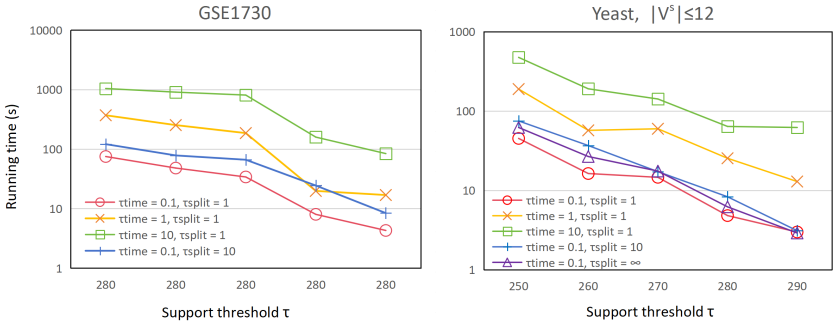

Fig. 4. Ablation Study (corresponding to original paper Fig. 14)

(3) **Table 3. Number of Results**. The results are *fully reproducible*.
(4) **Fig. 15. Scalability**. This experiment is *reproducible*. The trend is consistent with the findings presented in the paper.

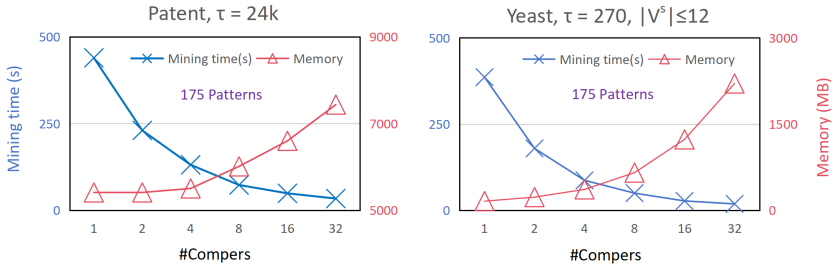

Fig. 5. Scalability (corresponding to original paper Fig. 15)

(5) **Fig. 18. Results with Fraction-Score as Support and Fig. 19. Results with MNI as Support**. These two experiments are *fully reproducible*. The trend is consistent with the findings presented in the paper.

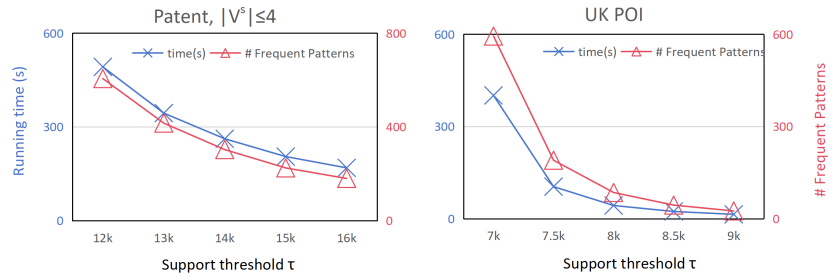

Fig. 6. Results with Fraction-Score as Support (corresponding to original paper Fig. 18)

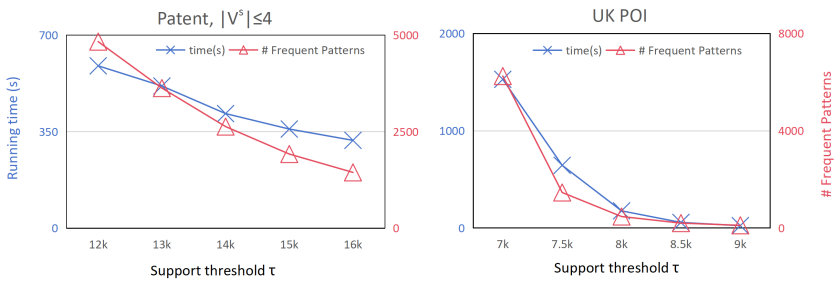

Fig. 7. Results with MNI as Support (corresponding to original paper Fig. 19)

4 SUMMARY

Overall most paper claims have been reproduced with minimal effort. This deemed results reproducible.

REFERENCES

[1] Ehab Abdelhamid, Ibrahim Abdelaziz, Panos Kalnis, Zuhair Khayyat, and Fuad T. Jamour. 2016. Scalemine: scalable parallel frequent subgraph mining in a single large graph. In *SC, 2016*. IEEE Computer Society, 716–727.

[2] Xuhao Chen, Roshan Dathathri, Gurbinder Gill, and Keshav Pingali. 2020. Pangolin: An Efficient and Flexible Graph Mining System on CPU and GPU. *Proc. VLDB Endow.* 13, 8 (2020), 1190–1205.

[3] Vinícius Vitor dos Santos Dias, Carlos H. C. Teixeira, Dorgival O. Guedes, Wagner Meira Jr., and Srinivasan Parthasarathy. 2019. Fractal: A General-Purpose Graph Pattern Mining System. In *SIGMOD, 2019*. ACM, 1357–1374.

[4] Kasra Jamshidi, Rakesh Mahadasa, and Keval Vora. 2020. Peregrine: a pattern-aware graph mining system. In *EuroSys, 2020*. ACM, 13:1–13:16.

[5] Nilothpal Talukder and Mohammed J. Zaki. 2016. A distributed approach for graph mining in massive networks. *Data Min. Knowl. Discov.* 30, 5 (2016), 1024–1052.