Appendix: Artifact Description/Artifact Evaluation

1 SUMMARY OF THE ARTIFACTS

Our source code encapsulates 3 versions of programs for mining frequent subgraphs in a single graph, namely serial version, parallel version with MNI metric, and parallel version with Fraction-Score metric. In the following sections, we will be mainly describing how to execute and derive results of **parallel version w/ MNI metric**, i.e., **Figure 12** in the paper. The compilation and execution of the other versions are similar and clearly described in our GitHub repository. Regarding the other figures or plots shown in the paper, the results are obtained using the same code with the proper hyper-parameter configurations, hence omitted. Readers can reproduce the results following the descriptions in our paper.

2 HARDWARE DEPENDENCIES

The compilation of our source code should be done on any server with the GCC compiler installed. Here we list the environment where we conducted the experiments.

- Red Hat Enterprise Linux Server, Version 7.9
- Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz

3 SOFTWARE DEPENDENCIES

We compare T-FSM with 6 existing baselines with codebase available online.

- **T-FSM** can be accessed at
  `https://github.com/lyuheng/T-FSM.git`
- **ScaleMine** can be accessed at
  `https://github.com/ehab-abdelhamid/ScaleMine.git`
- **DistGraph** can be accessed at
  `https://github.com/zakimjz/DistGraph.git`
- **Fractal** can be accessed at
  `https://github.com/dccspeed/fractal.git`
- **Pangolin** can be accessed at
  `https://github.com/chenxuhao/GraphMiner/tree/master/src/pangolin`
- **Peregrine** can be accessed at
  `https://github.com/pdclab/peregrine.git`
- **GraMi** can be accessed at
  `https://github.com/ehab-abdelhamid/GraMi.git`

All programs above are compiled with with GCC-10.2.0.

Conference'17, July 2017, Washington, DC, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

4 DATA SOURCES

All datasets we used in the paper can be downloaded at https://drive.google.com/drive/folders/1xxn35FTEKvV6JS7K2zKzx00X-RbkTPw-?usp=sharing
Please download and decompress the folder and rename it as **data**.

5 PROGRAM EXECUTION

We will describe how to compile and execute our T-FSM, as well as the two fastest competitors ScaleMine and DistGraph in this section. For the other systems, please follow the instructions in their respective README.md to run the experiments.

In the following subsections, whenever a command spans beyond one line, we use ↪ to mean a link break.

5.1 T-FSM

(1) Download the source code using
```
git clone https://github.com/lyuheng/T-FSM.git
↪ --depth=1
```
(2) Compile the code using
```
make
```
(3) Run the program on the **patent_citations.lg** dataset with frequency support 28,000 using 32 threads
```
./run -file ../data/patent_citations.lg -freq
↪  28000 -thread 32
```
(4) Run the program on **yeast_fsm** with frequency support 280 and 12 as the maximum number of vertices using 32 threads
```
./run -file ../data/yeast_fsm -freq 280
↪ -maxNodes 12 -thread 32
```

All other datasets can be run in a similar way with the appropriate filename, support frequency and maxNodes options as presented in Figure 12 of our paper.

5.2 ScaleMine

(1) After downloading the source code, please follow the instructions in README.md to compile the code. MPI is needed for compilation, and we used MPICH/3.4.2 in our tests.
(2) Run the program in the single-machine mode on **patent_citations.lg** with support frequency 28,000 using 32 threads
```
mpirun -n 2 ./pfsm -file
↪  ./Datasets/patent_citations.lg -freq
↪  28000 -threads 32
```
(3) Run the program in single-machine mode on **yeast_fsm** with frequency support 280 and 12 as the maximum number of vertices using 32 threads
```
mpirun -n 2 ./pfsm -file ../data/yeast_fsm
↪ -freq 280 -maxNodes 12 -threads 32
```

All other datasets can be run in a similar way with the appropriate filename, support frequency and maxNodes options as presented in Figure 12 of our paper.

5.3 DistGraph

(1) After downloading the source code, please follow the instructions in README.md to compile the code.

(2) Run the program with the single-machine mode on **patent_citations.lg** with support frequency 28, 000 using 32 threads

```
export OMP_NUM_THREADS=32
mpirun -np 1 ./src/parallel/pargraph_hybrid
 ↪ -txt ../data/patent_citations.lg 28000
```

All the other datasets can be run in a similar way with the appropriate filename, and frequency support presented in Figure 12 of our paper. Note that the maxNodes option is not supported in DistGraph.