Tri-Fly: Distributed Estimation of Global and Local Triangle Counts in Graph Streams

Nov-10-2017

Kijung Shin (kijungs@cs.cmu.edu)

1 General Information

- Version: 1.0
- Date: Nov-10-2017
- Author(s): Jinoh Oh and Kijung Shin
- Main Contact: Kijung Shin (kijungs@cs.cmu.edu)

2 Introduction

Tri-Fly is a distributed streaming algorithm for global and local triangle counting in graph streams. **Tri-Fly** has the following advantages compared to baselines:

- *Accurate*: Tri-Fly produces up to 4.5X smaller estimation error
- *Fast*: Tri-Fly runs in linear time up to 8.8X faster
- *Theoretically sound*: Tri-Fly gives unbiased estimates with variances inversely proportional to the number of machines

Detailed information about the algorithm is explained in the following paper:

- Kijung Shin, Mohammad Hammoud, Euiwoong Lee, Jinoh Oh, and Christos Faloutsos, *"Tri-Fly: Distributed Estimation of Global and Local Triangle Counts in Graph Streams"*, 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD) 2018, Melbourne, Australia. (To Appear)

3 Installation

- This package requires that *c++ 0x* (or higher) be installed in the system and set in PATH.
- This package requires that *MPICH 3.1* (or higher) be installed in the system and set in PATH.
- For compilation (optional), type *'make'*

- For demo (optional), type '*make demo'*

4 Input File Format

The input file lists edges in a graph. Each line corresponds to an edge and consists of the source node id and the destination node id, which are separated by a tab. Additionally, we assume the followings:

- No duplicate edges. For example, both edge (1,2) and edge (2,1) cannot be in the input file at the same time.
- Node ids are integers in range [0, #nodes -1]

example_graph.txt is an example of the input file.

5 Output Files Format

Two output files are created for each trial.

- *global(trial#).txt*: this file has the estimated number of global triangles.
- *local(trial#).txt*: this file lists the estimated number of local triangles of each node. Each line consists of the node id and the number of its local triangle count, which are separated by a tab.

example_output directory contains the examples of the output files.

6 How to Run

6.1 Simulating Tri-Fly in a Single Machine

```
mpirun -n [#processes] ./bin/mpi --trial [#trials] --budget [budget] [input_graph] [output_directory]
```

- *#processes*: number of processes that will be used to run the algorithm. One of the processes runs the master and the aggregator. This parameter should be an integer greater than or equal to two.
- *#trials*: number of times that Tri-Fly will be executed. This parameter should be an integer greater than or equal to one.
- *budget*: maximum number of edges that can be stored in each worker. This parameter should be an integer greater than two.
- *input_graph*: the path of an input graph file. See 4 for the detailed format of the input file.
- *output_directory:* the path of the directory where output files will be stored. See 5 for the detailed formats of the output files.

6.2 Running Tri-Fly in a Distributed Setting

```
mpiexec -n [#processes] –f [machinefile] ./bin/mpi --trial [#trials] --budget [budget] [input_graph] [output_directory]
```

- *#processes*: number of processes that will be used to run the algorithm. One of the processes runs the master and the aggregator. This parameter should be an integer greater than or equal to two.
- *machinefile*: the path of a machinefile. The host listed first runs the master and the aggregator; and the remaining hosts run workers. See 10 of mpich manual for the details of a machinefile.
- *#trials*: number of times that Tri-Fly will be executed. This parameter should be an integer greater than or equal to one.
- *budget*: maximum number of edges that can be stored in each worker. This parameter should be an integer greater than two.
- *input_graph*: the path of an input graph file in the host machine of the master. See 4 for the detailed format of the input file.
- *output_directory:* the path of the directory where output files will be stored. The directory is located in the host machine of the master. See 5 for the detailed formats of the output files.