WRS: Waiting Room Sampling for Accurate Triangle Counting in Real Graph Streams

May-24-2017

Kijung Shin (kijungs@cs.cmu.edu)

1   General Information

- Version: 1.0

- Date: May-24-2017

- Authors: Kijung Shin (kijungs@cs.cmu.edu)

2   Introduction

WRS (Waiting Room Sampling) is a single-pass streaming algorithm for global and local triangle counting in real graph streams. **WRS** exploits a temporal dependency pattern in real dynamic graph streams. **WRS** has the following properties:

- *fast and any time*: WRS scales linearly with the number of edges in the input graph stream, and gives estimates at any time while the input graph grows

- *effective*: estimation error in WRS is up to 47% smaller than those in state-of-the-art methods

- *theoretically sound*: WRS gives unbiased estimates with small variance under the temporal locality.

Detailed information about the method is explained in the following paper

- Kijung Shin, "*WRS: Waiting Room Sampling for Accurate Triangle Counting in Real Graph Streams*", IEEE International Conference on Data Mining (ICDM) 2017, New Orleans, USA

3   Installation

- This package requires that java 1.7 or greater be installed in the system and set in PATH.

- For compilation (optional), type *./compile.sh*

- For packaging (optional), type *./package.sh*

- For demo (optional), type *make*

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 integers separated by a tab. Extra information in each line is ignored. Additionally, we assume the followings:

- No duplicate edge. For example, both edge (1,2) and edge (2,1) cannot be in the input file at the same time.

- Edges are ordered in the increasing order of their creation time.

example_graph.txt is an example of the input file.

5   Output Files Format

Two output files are created.

- *global_count.out*: this file has the estimated number of global triangles.

- *Local_counts.out*: 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, separated by a tab.

output directory contains the examples of the output files.

6   Running WRS

6.1 How to Run

./run.sh input_path output_path k alpha

6.2 Parameters

- *input_path*: path of the input file. See 4 for the detailed format of the input file

- *output_path*: path of the directory for output files. See 5 for the detailed format of the output files

- *k*: maximum number of sampled edges (an integer greater than or equal to 2)

- *alpha:* the relative size of the waiting room (a real number in [0,1))

7   APIs for WRS

7.1 Package: *wrs*

7.2 Class: *WRS*

7.3 Methods:

- public *WRS* (int *k, double alpha, int random_seed*)

  - create a *WRS* object

  - *k*: maximum number of sampled edges (an integer greater than or equal to 2)

  - *alpha*: the relative size of the waiting room (a real number in [0,1))

  - *random_seed*: an integer

- public void *processEdge* (int src, int dst)

  - process an edge

  - *src*: id of the source node

  - *dst*: id of the destination node

- public double *getGlobalTriangle* ()

  - return the estimated number of global triangles

- public it.unimi.dsi.fastutil.ints.Int2DoubleMap *getLocalTriangle* ()

  - return the estimated number of local triangles of each node

  - *return*: a map whose keys are node ids and values the estimated number of local triangle counts of the corresponding node.

7.4 Example Code: see *Example.java* for an example code using *WRS.*