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

Dongjin Lee (dongjin.lee@kaist.ac.kr) and Kijung Shin (kijungs@kaist.ac.kr)

1 General Information

- Version: 2.0
- Date: Aug 18, 2020
- Authors: Dongjin Lee (dongjin.lee@kaist.ac.kr) and Kijung Shin (kijungs@kaist.ac.kr)

2 Introduction

WRS (Waiting Room Sampling) is a single-pass streaming algorithm for global and local triangle counting in (fully dynamic) 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
- Dongjin Lee, Kijung Shin, and Christos Faloutsos, "*Temporal Locality-Aware Sampling for Accurate Triangle Counting in Real Graph Streams*", The VLDB Journal, 2020

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 for WRS$_{\text{INS}}$ and WRS$_{\text{DEL}}$

4-1. Insertion-only Graph Stream

The input file lists edges in a graph in the order that they arrive. Each line corresponds to an edge and consists of a source-node id, a destination-node id, and a timestamp, which are separated by a tab. *example_graph.txt* is an example of the input file.

4-2. Fully Dynamic Graph Stream

The input file lists the edges in an *undirected* and *unweighted* graph in the order that they arrive. Each line corresponds to an edge addition or deletion. Each line consists of a source-node id, a destination-node id, and an indicator (1 for addition and -1 for deletion), which are integers separated by a tab. See *example_graph_dynamic.txt* for an example input file.

In both cases, we assume that there are no parallel edges. For example, both edge (1,2) and edge (2,1) cannot be in the input file at the same time. For a fully dynamic graph stream, if an edge has been added and has not been deleted yet, the same edge cannot be added.

5 Output Files Format for WRS$_{\text{INS}}$ and WRS$_{\text{DEL}}$

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_ins and output_del directory contains the examples of the output files.

6 Running WRS$_{\text{INS}}$

6.1 How to Run

```
./run_ins.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$_{\text{INS}}$

7.1 Package: *wrs*

7.2 Class: *WRSIns*

7.3 Methods:

- public *WRSIns* (int *k, double alpha, int random_seed*)
  - create a *WRSIns* 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 *ExampleIns.java* for an example code using *WRSIns.*

8 Running WRS$_{\text{DEL}}$

8.1 How to Run

```
./run_del.sh input_path output_path k alpha
```

8.2 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))

9 APIs for WRS$_{\mathrm{DEL}}$

9.1 Package: *wrs*

9.2 Class: *WRSDel*

9.3 Methods:

- public *WRSDel* (int *k, double alpha, int random_seed*)
  - create a *WRSDel* 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, boolean add)
  - process an edge
  - *src*: id of the source node
  - *dst*: id of the destination node
  - *add:* indicator (1 for addition and -1 for deletion)
- 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.

9.4 Example Code: see *ExampleDel.java* for an example code using *WRSDel.*