Paper Code

GREAT: Generalized Reservoir Sampling based Triangle Counting Estimation over Streaming Graphs

Siyue Wu, Dingming Wu, Sinhong Cheuk, Tsz Nam Chan, Kezhong Lu.
College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China.

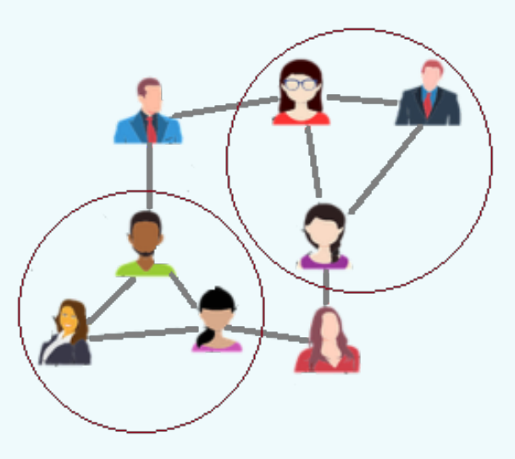
SHENZHEN UNIVERSITY
VLDB 2025 WELCOME TO LONDON

Motivation

Triangle counting estimation on the streaming graphs:

⋆ *Input:* An edge stream with timestamps.

⋆ *Output:* Estimated global triangle count and local triangle counts of each vertex at any timestamp.



⋆ *Restriction:* (i)unbounded edge streams, (ii)limited memory budget, (iii)single-pass processing.

Limitations of existing methods: Exact counting, matrix-based estimations and fixed sampling probability-based estimations: require high space complexity which is related to the number of the coming edges.

$\Longrightarrow$ *Solution:* Fixed memory-based estimations(FM) only require a fixed memory budget k to estimate triangle counts.

Shortcomings of existing FM-based estimations:

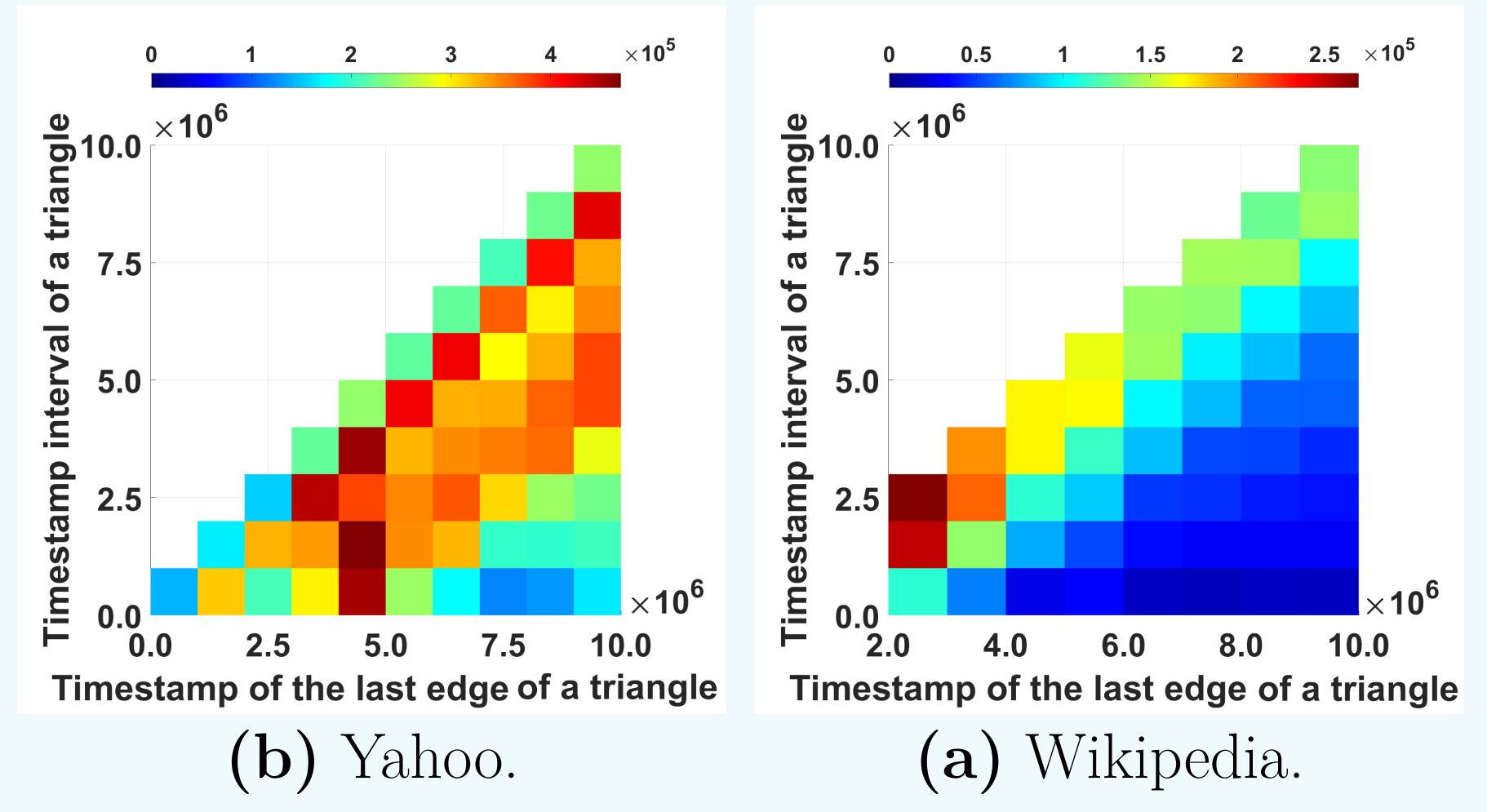
⋆ *Efficiency:* High computational cost because of reservoir sampling (at least $O(k)$ for per edge).

⋆ *Accuracy:* Ignore dynamic timestamp intervals of triangles.

Novelty

⋆ *Efficiency:* Propose a new streaming sampling framework *Generalized Reservoir Sampling(GRS)*, then propose *GREAT* algorithms based on GRS improve the efficiency of FM-based estimations.

⋆ *Accuracy:* Time interval of a triangle (x_1, x_2, x_3) means the sum of the time interval between the arrival of the three edges of a triangle $(t_{x_3} - t_{x_2}) + (t_{x_3} - t_{x_1})$. Obtain three new observations of timestamp interval distribution of the triangles over real-world dynamic graph, then propose a more accurate $GREAT^{+}$ extended $GREAT$ based on three observations.



(b) Yahoo. (a) Wikipedia.

Figure 1. Dynamic timestamp intervals distribution.

Observations shown in Figure 1: (i)Non-uniform in most cases. (ii)Not always biased to short intervals. (iii)Varies over time.

Algorithm

Framework of existing FM-based estimations: It can be divided into two parts, how to obtain edge sample set and how to update the triangle counting, shown in Figure 2. There is a reservoir with fixed size k and each edge (u, v) arrived at timestamp t_{uv}.

(C-I) Sample arrived edges. Use Reservoir Sampling (RS) to construct uniform edge sample. Specifically, sample (u, v) with probability $min\{k/t_{uv}, 1\}$. When the reservoir is full, removes one edges randomly.

(C-II) Update triangle counting. Method I: Update the triangle counting whenever an edge arrives, or Method II: update only when sampling succeeds.

$GREAT^{I}$ and $GREAT^{II}$ (improve efficiency):

(C-I) Sample arrived edges. Use Generalized Reservoir Sampling (GRS) to construct edge sample. Specifically, sample (u, v) with probability $p_{uv}^{(s)}$(self-set). When the reservoir is full, removes edges with probability $\alpha > 1/k$(self-set). Investigate two edge sampling probability $p_{uv}^{(s)}$ leading to two specialized algorithms $GREAT^{I}$ and $GREAT^{II}$:

$$p_{uv}^{(s)} = (1-\alpha)^{r_{uv}} \text{ for } GREAT^{I}; \qquad p_{uv}^{(s)} = k/t_{uv} \text{ for } GREAT^{II}.$$

(C-II) Update triangle counting. $GREAT$ updates the triangle counting whenever an edge arrives, which is more accurate.

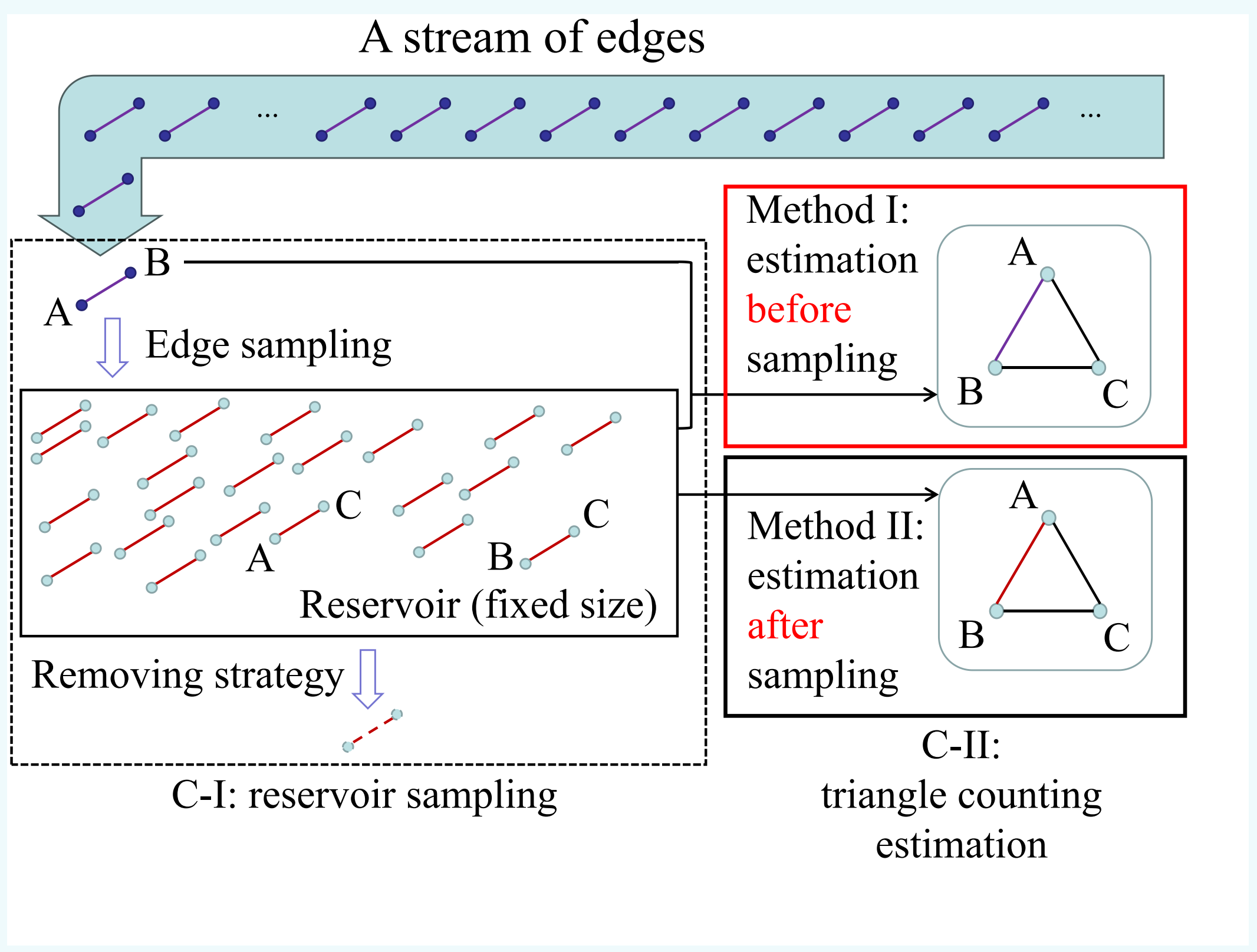

A stream of edges
Edge sampling
Reservoir (fixed size)
Removing strategy
C-I: reservoir sampling
Method I: estimation before sampling
Method II: estimation after sampling
C-II: triangle counting estimation
A
B
C

Figure 2. FM-based framework.

$GREAT^{+}$ (improve accuracy):

⋆ *Idea:* In $GREAT$, the larger α is, the fewer old edges remain in the reservoir, and the more discorded triangles with short time interval.

⋆ *Operation:* Adaptively adjust α to match the time interval distribution over the edge streams. Record the time intervals of the discovered triangles two times when the reservoir is full. If their time interval is short, increase α; otherwise, decrease α.

Theoretical and Experimental Comparison (Efficiency vs. Accuracy)

Theoretical: In Table 1, all methods are unbiased, and the simplified variance can be used to compare the accuracy of different algorithms. Since the edge flow is infinite, the amortized time complexity of each edge is calculated to compare the efficiency. When each edge arrives, perform two steps: triangle counting and reservoir update.

⋆ *Efficiency:* Existing FM-based methods require maintaining a size k sample, so the amortized time complexity of per edge is at least $O(k)$. Our algorithms use a sample size smaller than k, so it is more efficient. Sampling probability of $GREAT^{I}$ is smaller than $GREAT^{II}$, so $GREAT^{I}$ is more difficult to sample successfully than $GREAT^{II}$, leading to more efficiency.

⋆ *Accuracy:* As the number of arrived edges increases, the simplified variance of different algorithms satisfy below inequality.

$$\tilde{Var}^{FRUL-0B} > \tilde{Var}^{TRIÈST-I} > \tilde{Var}^{GREAT^{I}} > \tilde{Var}^{GREAT^{II}} > \tilde{Var}^{GREAT^{+}}.$$

Case 1 and Case 2 of WRS have smaller variance, but Case 3 has the same simplified variance as that of TRIÈST-I, which is worse than our algorithms.

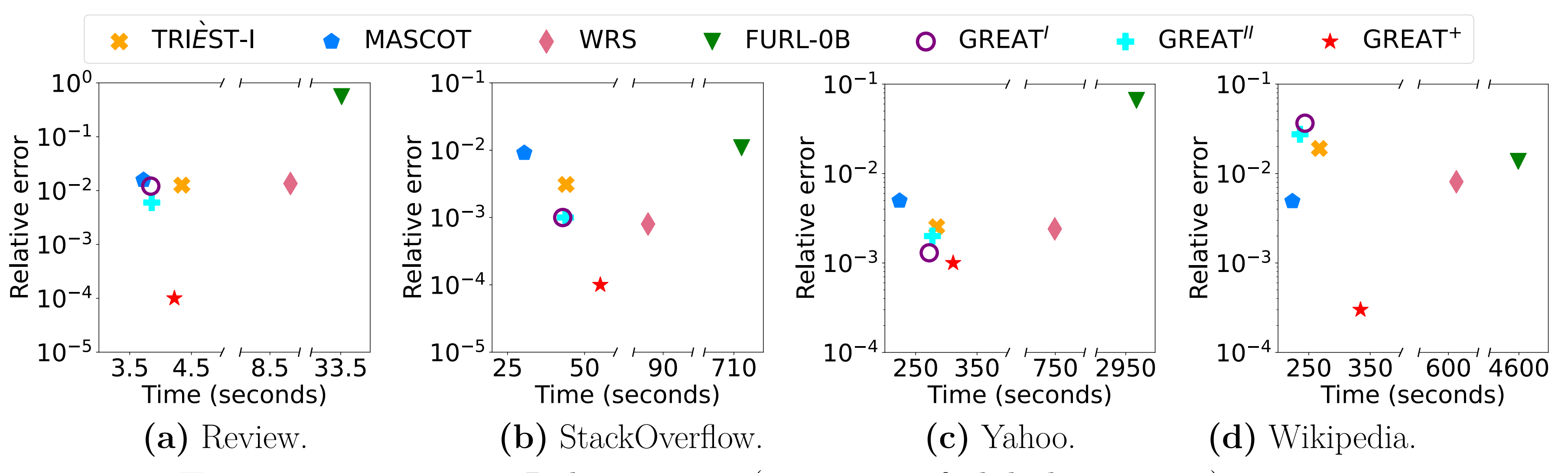
Experimental: We test all competitors on real-world datasets with $10^4 - 10^7$ vertices. Figure 3 and Figure 4 compares our algorithms with the competitors according to the accuracy (relative error for global counting and LAPE for local counting) and the efficiency (elapsed time). In Table 1, all algorithms are ranked according to their accuracy and efficiency. The Experimental result is same as theoretical analysis.

⋆ *Efficiency:* MASCOT is the fastest but it requires unlimited memory. $GREAT^{I}$ and $GREAT^{II}$ are the fastest among FM-based algorithms. $GREAT^{+}$ is slower because of the adaptive strategy.

⋆ *Accuracy:* The accuracy of global counting in $GREAT^{+}$ is the best, while $GREAT^{II}$ and $GREAT^{I}$ is second and third place, respectively. As the number of discovered vertices increases, the accuracy of local triangle counting tends to decrease, therefore, $GREAT^{I}$, $GREAT^{II}$, $GREAT^{+}$, and TRIÈST-I exhibit similar LAPEs. WRS has the best LAPE, but leads to a decrease in efficiency.

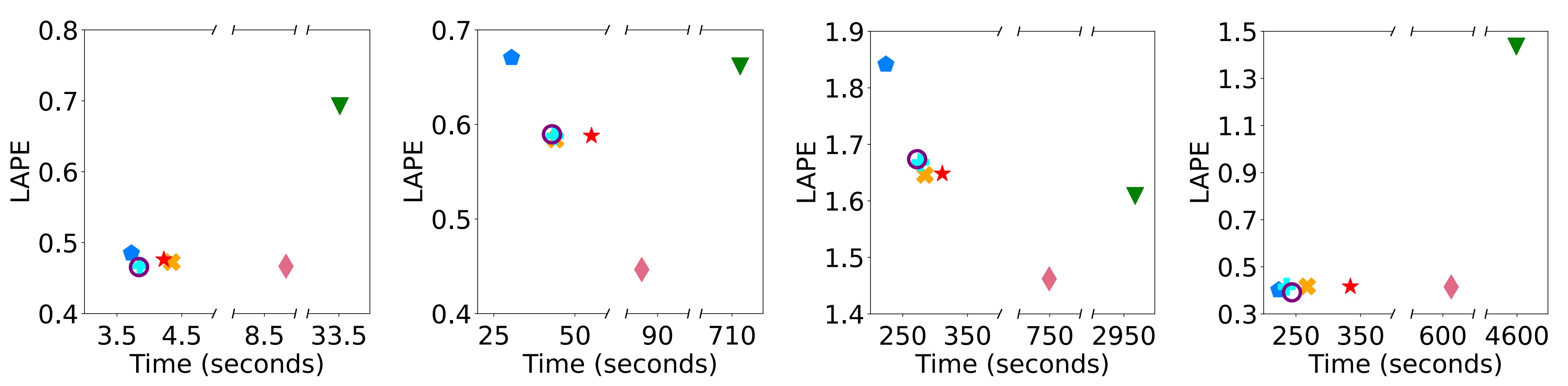
Table 1. Theoretical and experimental comparison.

Algorithm	Theoretical			Experimental ranking		
	Accuracy	Efficiency		Accuracy		Efficiency
		Count	Update	Global	Local	
$GREAT^{I}$	$(1-\alpha)^{-2r_{x_3}} - 1$	$O\left(\left(1-\frac{\alpha}{2}\right)\cdot k - \frac{1}{2}\right)$	$O\left(p_{uv}^{(s)}\right)$	3	2	1
$GREAT^{II}$	$\frac{t_{x_1}t_{x_2}}{k^2}\cdot(1-\alpha)^{-(2r_{x_3}-r_{x_1}-r_{x_2})} - 1$	$O\left(k - \mathbb{C}_1\right)$	$O\left(p_{uv}^{(s)}\right)$	2	2	2
$GREAT^{+}$	$\leq \frac{t_{x_1}t_{x_2}}{k^2}\cdot\frac{1}{z} - 1$	$O\left(k - \mathbb{C}_2\right)$	$O\left(p_{uv}^{(s)}\right)$	1	2	4
TRIÈST-I	$\frac{t_{x_3}(t_{x_3}-1)}{k(k-1)} - 1$	$O(k)$	$O\left(p_{uv}^{(s)}\right)$	5	2	3
WRS	$\begin{cases} 0 & \text{(Case 1)} \\ \frac{\tilde{t}_{x_3}}{(1-\gamma)k} - 1 & \text{(Case 2)} \\ \frac{\tilde{t}_{x_3}(\tilde{t}_{x_3}-1)}{(1-\gamma)k((1-\gamma)k-1)} - 1 & \text{(Case 3)} \end{cases}$	$O(k)$	$O(1)$	4	1	5
FURL-0B	$\frac{t_{x_3}(t_{x_3}-1)(t_{x_3}-2)}{k(k-1)(k-2)} - 1$	$O(k)$	$O\left(p_{uv}^{(s)}\cdot k\cdot \log k\right)$	6	6	6



(a) Review. (b) StackOverflow. (c) Yahoo. (d) Wikipedia.

Figure 3. $\alpha = 0.1$, Relative error(accuracy of global counting) vs. time.



(a) Review. (b) StackOverflow. (c) Yahoo. (d) Wikipedia.

Figure 4. $\alpha = 0.1$, LAPE(accuracy of local counting) vs. time.