Supplementary Material for Revision 1554

1 Notations

We provide the notation table here to improve the presentation and readability of the paper.

Table 1: Notations.

Symbol	Description
Σ	Edge stream
S	Reservoir
k	Reservoir size(budget size)
$G^{(t)}$	Graph formed by edges coming at time t or earlier
(u, v)	Edge formed by vertices u and v
(u, v, w)	Triangle formed by vertices u, v and w
α	Removed probability
r	Computational round
t_{uv}	Coming time of edge (u, v)
r_{uv}	Coming round of edge (u, v)
$p_{uv}^{(s)}$	Probability of edge (u, v) being sampled
$p_{uv}^{(r)}$	Probability of edge (u, v) in the reservoir in round r
P_{uvw}	Probability of discovering triangle (u, v, w)
Δ	Set of real triangles of $G^{(t)}$
τ	Real global triangle count of $G^{(t)}$, $\tau = \lvert\Delta\rvert$
$\widehat{\Delta}$	Set of discovered triangles of $G^{(t)}$
$\widehat{\tau}$	Estimated global triangle count of $G^{(t)}$
Δ_u	Set of real triangles of vertex u in $G^{(t)}$
τ_u	Real global triangle count of vertex u in $G^{(t)}$, $\tau_u = \lvert\Delta_u\rvert$
$\widehat{\Delta}_u$	Set of discovered triangles of vertex u in $G^{(t)}$
$\widehat{\tau}_u$	Estimated global triangle count of vertex u in $G^{(t)}$

2 Proofs of Lemmas

Lemma 3.1: Given an edge (u, v) from the stream in computational round r, the probability of discovering triangle (u, v, w) in GRS is calculated as

$$P_{uvw} = p_{uw}^{(s)} \cdot p_{vw}^{(s)} \cdot (1-\alpha)^{2r-r_{uw}-r_{vw}}. \tag{1}$$

Proof. According to GRS, edge (u, w) is sampled with probability $p_{uw}^{(s)}$ in computational round r_{uw}. Then, in computational round r ($\geq r_{uw}$), the probability of edge (u, w) staying in the reservoir is

$$p_{uw}^{(r)} = p_{uw}^{(s)} \cdot (1-\alpha)^{r-r_{uw}}. \tag{2}$$

Similarly, in computational round r, the probability of edge (v, w) staying in the reservoir is $p_{vw}^{(r)} = p_{vw}^{(s)} \cdot (1-\alpha)^{r-r_{vw}}$. Hence, for the current edge (u, v) in computational round r, the probability of discovering triangle (u, v, w) formed by edge (u, v) and two edges in the reservoir is $P_{uvw} = p_{uw}^{(r)} \cdot p_{vw}^{(r)} = p_{uw}^{(s)} \cdot p_{vw}^{(s)} \cdot (1-\alpha)^{2r-r_{uw}-r_{vw}}$. □

Lemma 3.2: If applying $p_{uv}^{(s)} = (1-\alpha)^{r_{uv}}$, in any computational round r, all the arrived edges stay in the reservoir with probability $(1-\alpha)^r$.

Proof. Given any edge (u, v), suppose that it arrives in round r_{uv} and $p_{uv}^{(s)} = (1-\alpha)^{r_{uv}}$. In round r ($\geq r_{uv}$), the probability of (u, v) in the reservoir is

$$(1-\alpha)^{r_{uv}} \cdot (1-\alpha)^{r-r_{uv}} = (1-\alpha)^{r}.$$

□

Lemma 3.3: Let $p_{uv}^{(s)} = k/t_{uv}$, in any computational round r, t be the number of arrived edges, and $p_{uv}^{(r)}$ be the probability that any arrived edge (u, v) staying in the reservoir. If $\alpha \leq 0.7$,

$$\left| p_{uv}^{(r)} - p^* \right| / p^* \leq 1 - \exp(-2\alpha), \quad p^* = k/t. \tag{3}$$

Proof. Suppose that edge t comes in round R_0 ($R_0 \geq 1$), and Y edges come following t is Y before edge n ($n = t + Y$) coming, and n comes in round R ($R > R_0$). Suppose that after edge t, it needs d_0 edges to fulfill the reservoir. Let the number of edges arrive before n in round R is d. The probability of t staying in the reservoir at time n is $p^* = \frac{k}{n}$ in traditional reservoir sampling, and the probability of t staying in the reservoir at time $Y + t$ in GRS using P-II is $\frac{k}{t} \cdot (1-\alpha)^{R-R_0}$. Firstly, when $R > R_0$, from *Lemma 3.6*, we have

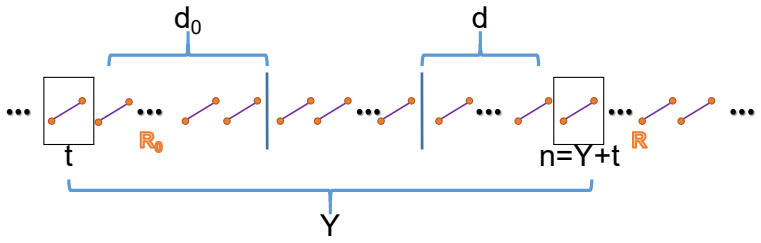


Figure 1: Definition of symbol.

$$\begin{aligned} Y &= d_0 + x_{R_0+1} + x_{R_0+2} \cdots + x_{R-1} + d \\ &= k \cdot (\exp(\alpha) - 1) \cdot (\exp(\alpha \cdot R_0)) \cdot \frac{(\exp(\alpha \cdot (R - R_0 - 1)) - 1}{\exp(\alpha) - 1} + (d + d_0) \\ &= k \cdot (\exp(\alpha \cdot (R-1)) - \exp(\alpha \cdot R_0)) + (d + d_0). \end{aligned} \tag{4}$$

Similarly, we have

$$\begin{aligned} t &= k + x_1 + x_2 \cdots + x_{R_0-1} + x_{R_0} - d_0 \\ &= k \cdot (\exp(\alpha) - 1) \cdot \frac{\exp(\alpha \cdot R_0) - 1}{\exp(\alpha) - 1} + (k - d_0) \\ &= k \cdot (\exp(\alpha \cdot R_0) - 1) + (k - d_0) \\ &= k \cdot \exp(\alpha \cdot R_0) - d_0 \\ &\leq k \cdot \exp(\alpha \cdot R_0). \end{aligned} \tag{5}$$

Combine Equations 4 and 5, we have

$$\begin{aligned} n = Y + t &= k \cdot (\exp(\alpha \cdot (R-1)) - \exp(\alpha \cdot R_0)) + (d + d_0) + k \cdot \exp(\alpha \cdot R_0) - d_0 \\ &= k \cdot \exp(\alpha \cdot (R-1)) + d \\ &\geq k \cdot \exp(\alpha \cdot (R-1)). \end{aligned} \tag{6}$$

At last, combine all the analysis above, we have

$$\begin{aligned}
p_{uv}^{(r)} - p^* &= \frac{k}{n} - \frac{k}{t} \cdot (1-\alpha)^{R-R_0} \\
&\leq \frac{k}{n} - \frac{k}{k \cdot \exp(\alpha \cdot R_0)} \cdot (1-\alpha)^{R-R_0} && (7) \\
&\leq \frac{k}{n} - \exp(-\alpha \cdot R_0) \cdot (\exp(-\alpha(R - R_0 + 1))) && (8) \\
&= \frac{k}{n} - \exp(-\alpha \cdot (R+1)) \\
&= \frac{k}{n} - \exp(-\alpha \cdot (R-1)) \cdot \exp(-2\alpha) \\
&\leq \frac{k}{n} - \frac{k}{n} \cdot \exp(-2\alpha) && (9) \\
&= (1 - \exp(-2\alpha)) \cdot \frac{k}{n}.
\end{aligned}$$

Equation 7 is hold because of Equation 5. Equation 9 is hold because of Equation 6. Equation 8 is hold only when $\alpha < 0.7$, we have

$$\exp(-\alpha(R - R_0 + 1)) \leq (1-\alpha)^{R-R_0}, \tag{10}$$

which we prove as follows. In order to investigate the difference between $\exp(-\alpha(R - R_0 + 1))$ and $(1-\alpha)^{R-R_0}$, we define $Y = R - R_0$ and a function

$$f(Y) = \exp(-\alpha \cdot (Y+1)) - (1-\alpha)^Y. \tag{11}$$

Then, we calculate the derivation of function $f(Y)$

$$f'_Y = -\alpha \cdot \exp(-\alpha \cdot (Y+1)) - \ln(1-\alpha) \cdot (1-\alpha)^Y.$$

Because $Y \geq 1$, $f'_Y < 0$, which means that $f(Y)$ is a decreasing function. When $Y = 1$, we have the maximum of $f(Y)$ is $f_{max} = \exp(-2\alpha) - 1 + \alpha$. Define a function

$$g(\alpha) = \exp(-2\alpha) - 1 + \alpha, \ \alpha \in \left[\frac{1}{k}, 1\right].$$

Then, we calculate the derivation of $g(\alpha)$ function

$$g'(\alpha) = -2\exp(-2\alpha) + 1,$$

which is an increasing function with zero point, so that $g(\alpha)$ is a function that first decreases and then increases when $\alpha = -\frac{1}{2}\ln\frac{1}{2}$, $g_\alpha = g_{min}$, $g_{min} < 0$. Therefore, $g(\alpha)$ has zero points. It is easy to know $g(0) = 0$. Then, using Bisection method, we have $g(0.7) \cdot g(0.8) < 0$. Therefore, $g(\alpha) < 0$ when $0 < \alpha \leq 0.7$. Moreover, $f_{max} < 0$, when $0 < \alpha \leq 0.7$, so that $(\exp(-\alpha \cdot (Y+1))) < (1-\alpha)^Y$. Therefore, we have

$$p_{uv}^{(r)} - p^* \leq (1 - \exp(-2\alpha)) \cdot \frac{k}{n} \Longrightarrow \left| p_{uv}^{(r)} - p^* \right| / p^* \leq 1 - \exp(-2\alpha).$$

□

Lemma 3.4: In algorithm GREAT^I, the expected number of edges arrived in computational round r is $x_r^I = \alpha \cdot k/(1-\alpha)^r$.

Proof. The sampling probability in round r is $p_r = (1-\alpha)^r$, then we have

$$x_r \cdot (1-\alpha)^r = k \cdot \alpha \Longrightarrow x_r^I = \alpha \cdot k/(1-\alpha)^r.$$

□

Lemma 3.5: In algorithm GREAT^I, at the beginning of computational round r, suppose that there are Q free slots in the reservoir where each slot can store one edge. The expected number of edges arrived to put one sampled edge in the i^{th} slot is $y_{r,i}^I = 1/(1-\alpha)^r$.

Proof. The sampling probability in round r is $p_r = (1-\alpha)^r$, then we have

$$y_{r,i} \cdot (1-\alpha)^r = 1 \Longrightarrow y^I_{r,i} = 1/(1-\alpha)^r.$$

□

Lemma 3.6: In algorithm GREAT^{II}, the expected number of edges arrived in computational round r is $x^{II}_r = (\exp(\alpha)-1)\cdot\exp((r-1)\cdot\alpha)\cdot k$.

Proof. In algorithm GREAT^{II}, in round 0, the reservoir is empty and k edges are sampled with probability 1. When the reservoir is full, edges are randomly removed with probability α. There will be $Q(Q = k\cdot\alpha)$ empty slots. In round 1, edges are sampled with probability $\frac{k}{t}$ and t starts at $k+1$. Assume that filling these Q empty slots requires x_1 edges, then we have

$$\frac{k}{k+1} + \frac{k}{k+2}\cdots + \frac{k}{k+x_1} = Q$$
$$\frac{1}{k+1} + \frac{1}{k+2}\cdots + \frac{1}{k+x_1} = \frac{Q}{k} = \alpha.$$

If we have

$$\frac{1}{k+1} + \frac{1}{k+2} + ... + \frac{1}{k+x_1} \approx \int_k^{k+x_1} \frac{1}{u}du, \tag{12}$$

then we can solve

$$\begin{aligned}
&\int_k^{k+x_1} \frac{1}{u}du = \alpha \\
\Longrightarrow &\ln\left(\frac{k+x_1}{k}\right) = \alpha \\
\Longrightarrow &x_1 = k\cdot\exp(\alpha) - k \\
&x_1 = k\cdot(\exp(\alpha)-1).
\end{aligned}$$

The approximation of Equation 12 will be proved later.

Similarly, in round 2, t starts at $k+x_1+1$, and it needs x_2 edges to fulfill these s empty slots, so we have

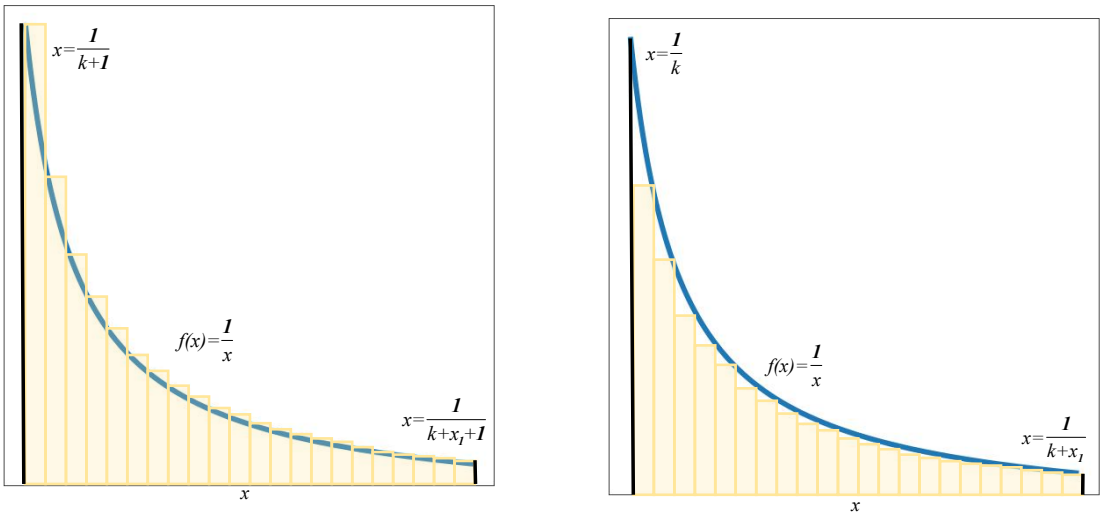
$$\begin{aligned}
&\frac{k}{k+x_1+1} + \frac{k}{k+x_1+2} + ... + \frac{k}{k+x_1+x_2} = s \\
\Longrightarrow &\int_{k+x_1}^{k+x_1+x_2} \frac{1}{u}du = \alpha \\
\Longrightarrow &x_2 = (\exp(\alpha)-1)\cdot(k+x_1).
\end{aligned}$$

Repeat the above process, and after r rounds, we have

$$\begin{aligned}
x_0 &= k \\
x_1 &= (\exp(\alpha)-1)\cdot k \\
x_2 &= (\exp(\alpha)-1)\cdot(k+x_1) \\
x_3 &= (\exp(\alpha)-1)\cdot(k+x_1+x_2) \\
&... \\
x_r &= (\exp(\alpha)-1)\cdot(k+x_1+x_2+...+x_{r-1}).
\end{aligned}$$

Then,

$$\begin{aligned}
x_{r-1} &= (\exp(\alpha)-1)\cdot(k+x_1+x_2+...+x_{r-2}) \\
x_r &= (\exp(\alpha)-1)\cdot(k+x_1+x_2+...+x_{r-2}+x_{r-1}) \\
x_r - x_{r-1} &= (\exp(\alpha)-1)\cdot x_{r-1} \\
x_r &= \exp(\alpha)\cdot x_{r-1}.
\end{aligned}$$



(a) Max. (b) Max.

Figure 2: The approximation of definite integral.

Therefore, $\{x_r^{II}\}_{r=1}$ is a geometric progression. The common ratio is $\exp(\alpha)$ and the first term is $x_1^{II} = k(\exp(\alpha)-1)$, and the general term is

$$x_r^{II} = (\exp(\alpha) - 1) \cdot \exp(\alpha \cdot (r-1)) \cdot k.$$

Then, we give the bound of approximated Equation 12. Because $\frac{1}{k+1} + \frac{1}{k+2} + \ldots + \frac{1}{k+x_1}$ is an approximation of the area of the trapezoid with curved edge, which is constructed by the line of function $f(x) = \frac{1}{x}$, line $x = \frac{1}{k+1}$, line $x = \frac{1}{k+x_1+1}$, and the x-axis. This curved-edge trapezoid can be partitioned into x_1 small rectangles of width 1. The area of the curved-edge trapezoid is approximated by the sum of the rectangular areas and choose the left intersect point of the rectangles and the curved-edge trapezoid as the height, as shown in Figure 2a. Since $f(x) = \frac{1}{x}$ is a decreasing function, the sum of the rectangular areas is greater than the actual area of the curved edge trapezoid. Similarly, $\frac{1}{k+1} + \frac{1}{k+2} + \ldots + \frac{1}{k+1+x_1}$ is also the approximation of the curved-edge trapezium surrounded by the line of function $f(x) = \frac{1}{x}$, line $x = 1$, line $x = \frac{1}{k+x_1}$, and the x-axis. The difference is to choose the right intersect point of the rectangles and the curved-edge trapezoid as the height, as shown in Figure 2b, and the approximate area of the rectangles is less than the actual area of the curved-edge trapezoid. Therefore, we have

$$\int_{k+1}^{k+x_1+1} \frac{1}{u} du \le \frac{1}{k+1} + \frac{1}{k+2} + \ldots + \frac{1}{k+x_1+1} \le \int_{k}^{k+x_1} \frac{1}{u} du.$$

Moreover,

$$\left| \left[\frac{1}{k+1} + \frac{1}{k+2} + \ldots + \frac{1}{k+x_1+1} \right] - \int_{k}^{k+x_1} \frac{1}{u} du \right| \le \left| \int_{k}^{k+x_1} \frac{1}{u} du - \int_{k+1}^{k+x_1+1} \frac{1}{u} du \right| = \ln\left(\frac{(k+1)(k+x_1)}{k(k+x_1+1)} \right).$$

It is the same for round r,

$$\left| \left[\frac{1}{k+\sum_i^{r-1} x_i + 1} + \frac{1}{k+\sum_i^{r-1} x_i + 2} + \ldots + \frac{1}{k+\sum_i^{r} x_i} \right] - \int_{k+\sum_i^{r-1} x_i}^{k+\sum_i^{r} x_i} \frac{1}{u} du \right| < \ln\left(\frac{(k+1)(k+\sum_i^r x_i)}{k(k+\sum_i^r x_i + 1)} \right).$$

Therefore, Equation 12 holds and the lemma is proved.

□

Lemma 3.7: In algorithm GREAT^{II}, at the beginning of computational round r, suppose that there are Q free slots in the reservoir where each slot can store one edge. The expected number of edges arrived to put one sampled

edge in the i^{th} slot is

$$y_{r,i}^{II} = k \cdot \exp(\alpha \cdot (r-1)) \cdot \left(\exp\left(\frac{1}{k}\right) - 1\right) \cdot \exp\left(\frac{1}{k} \cdot (i-1)\right).$$

Proof. The prove is similar with *Lemma 3.6.* In round 0, the reservoir is empty, and k edges are sampled with probability 1. When reservoir is full, edges are randomly removed with probability α. There will be $Q(Q = k \cdot \alpha)$ empty slots. In round r, edges are sampled with probability $\frac{k}{t}$ and t starts at $k + \sum_{j=1}^{r-1} x_j + 1$. Assume that filling the first empty slots requires $y_{r,1}$ edges, then we have

$$\begin{aligned}
&\frac{k}{k + \sum_{j=1}^{r-1} x_j + 1} + \frac{k}{k + \sum_{j=1}^{r-1} x_j + 2} \cdots + \frac{k}{k + \sum_{j=1}^{r-1} x_j + y_{r,1}} = 1 \\
&\frac{1}{k + \sum_{j=1}^{r-1} x_j + 1} + \frac{1}{k + \sum_{j=1}^{r-1} x_j + 2} \cdots + \frac{1}{k + \sum_{j=1}^{r-1} x_j + y_{r,1}} = \frac{1}{k} \\
&\Longrightarrow \int_{k+\sum_{j=1}^{r-1} x_j}^{k+\sum_{j=1}^{r-1} x_j + y_{r,1}} \frac{1}{u} du = \frac{1}{k} \\
&\Longrightarrow \ln\left(\frac{k + \sum_{j=1}^{r-1} x_j + y_{r,1}}{k + \sum_{j=1}^{r-1} x_j}\right) = \frac{1}{k} \\
&\Longrightarrow y_{r,1} = \left(k + \sum_{j=1}^{r-1} x_j\right) \cdot \left(\exp\left(\frac{1}{k}\right) - 1\right).
\end{aligned}$$

Similarly for empty slots 2 to Q, repeat the above process, and reveals that after r rounds, we have

$$\begin{aligned}
y_{r,1} &= \left(k + \sum_{j=1}^{r-1} x_j\right) \cdot \left(\exp\left(\frac{1}{k}\right) - 1\right) \\
y_{r,2} &= \left(k + \sum_{j=1}^{r-1} x_j + y_{r,1}\right) \cdot \left(\exp\left(\frac{1}{k}\right) - 1\right) \\
&\cdots \\
y_{r,Q} &= \left(k + \sum_{j=1}^{r-1} x_j + y_{r,1} + y_{r,2} \cdots + y_{r,Q-2} + y_{r,Q-1}\right) \cdot \left(\exp\left(\frac{1}{k}\right) - 1\right).
\end{aligned}$$

Then, we have

$$\begin{aligned}
y_{r,Q} &= \left(k + \sum_{j=1}^{r-1} x_j + y_{r,1} + y_{r,2} \cdots + y_{r,Q-2} + y_{r,Q-1}\right) \cdot \left(\exp\left(\frac{1}{k}\right) - 1\right) \\
y_{r,Q-1} &= \left(k + \sum_{j=1}^{r-1} x_j + y_{r,1} + y_{r,2} \cdots + y_{r,Q-2}\right) \cdot \left(\exp\left(\frac{1}{k}\right) - 1\right) \\
y_{r,Q} - y_{r,Q-1} &= y_{r,Q-1} \cdot \left(\exp\left(\frac{1}{k}\right) - 1\right) \\
y_{r,Q} &= y_{r,Q-1} \cdot \exp\left(\frac{1}{k}\right).
\end{aligned}$$

Therefore, $\{y_{r,i}\}_{i=1}$ is a geometric progression. The common ratio is $\exp\left(\frac{1}{k}\right)$ and first term is $\left(k + \sum_{i=1}^{r-1} x_i\right) \cdot \left(\exp\left(\frac{1}{k}\right) - 1\right)$, and the general term is

$$y_{r,i}^{II} = \left(k + \sum_{j=1}^{r-1} x_j\right) \cdot \left(\exp\left(\frac{1}{k}\right) - 1\right) \cdot \exp\left(\frac{1}{k}(i-1)\right).$$

From *Lemma 3.6* we have

$$\sum_{j=1}^{r-1} x_j = k \cdot (\exp(\alpha) - 1) \cdot \frac{\exp(\alpha \cdot (r-1)) - 1}{\exp(\alpha) - 1} = k \cdot (\exp(\alpha \cdot (r-1)) - 1).$$

Therefore,

$$y_{r,i}^{II} = \left(k + \sum_{j=1}^{r-1} x_j\right) \cdot \left(\exp\left(\frac{1}{k}\right) - 1\right) \cdot \exp\left(\frac{1}{k} \cdot (i-1)\right) = k \cdot \exp(\alpha \cdot (r-1)) \cdot \left(\exp\left(\frac{1}{k}\right) - 1\right) \exp\left(\frac{1}{k}(i-1)\right).$$

□

Lemma 3.8: In algorithm GREAT, Equations 13 and 14 return unbiased estimations for the global and the local triangle counts, i.e.

$$\mathbb{E}(\widehat{\tau}) = \tau \text{ and } \mathbb{E}(\widehat{\tau}_u) = \tau_u, \forall u \in V.$$

The global triangle count is estimated as

$$\widehat{\tau} = \sum_{(u,v,w) \in \widehat{\Delta}} \frac{1}{P_{uvw}}. \tag{13}$$

The local triangle count of vertex u is estimated as

$$\widehat{\tau}_u = \sum_{(u,v,w) \in \widehat{\Delta}_u} \frac{1}{P_{uvw}}. \tag{14}$$

Proof. Given edge stream Σ, at any timestamp t, the edges arrive no later than t are called seen edges and the edges arrive after t are called unseen edges. Given any timestamp t, let Δ be the ground truth set of global triangles formed by seen edges and $\tau = |\Delta|$ is the ground truth global triangle count. Next, we prove that $\mathbb{E}(\widehat{\tau}) = \tau$. Let I_{uvw} be an indicator variable, such that

$$I_{uvw} = \begin{cases} 1 & \text{if triangle } (u,v,w) \text{ is discovered,} \\ 0 & \text{otherwise.} \end{cases}$$

Then, we have that $\mathbb{E}(I_{uvw}) = P_{uvw} \times 1 + (1 - P_{uvw}) \times 0 = P_{uvw}$. The expectation of the estimated global triangle count is

$$\begin{aligned} \mathbb{E}(\widehat{\tau}) &= \mathbb{E}\left(\sum_{(u,v,w) \in \widehat{\Delta}} \frac{1}{P_{uvw}}\right) = \mathbb{E}\left(\sum_{(u,v,w) \in \Delta} \frac{I_{uvw}}{P_{uvw}}\right) \\ &= \sum_{(u,v,w) \in \Delta} \frac{\mathbb{E}(I_{uvw})}{P_{uvw}} = \sum_{(u,v,w) \in \Delta} \frac{P_{uvw}}{P_{uvw}} = \sum_{(u,v,w) \in \Delta} 1 = \tau. \end{aligned}$$

Hence, $\widehat{\tau}$ computes unbiased estimation for the global triangle count.

Similarly, we prove that $\widehat{\tau}_u, \forall u \in V$ computes unbiased estimation for the local triangle count. Given any timestamp t, let Δ_u be the ground truth set of local triangles of vertex u formed by seen edges and $\tau_u = |\Delta_u|$ is the ground truth local triangle count of vertex u.

$$\mathbb{E}(\widehat{\tau}_u) = \mathbb{E}\left(\sum_{(u,v,w) \in \Delta_u} \frac{I_{uvw}}{P_{uvw}}\right) = \sum_{(u,v,w) \in \Delta_u} \frac{\mathbb{E}(I_{uvw})}{P_{uvw}} = \tau_u.$$

□

Lemma 3.9: Let $\tilde{Var}^{TRI}(\widehat{\tau})$, $\tilde{Var}^{I}(\widehat{\tau})$, and $\tilde{Var}^{II}(\widehat{\tau})$ be the simplified variances of algorithms TRIÈST-I, GREAT^{I}, and GREAT^{II}, respectively. We have

$$\tilde{Var}^{TRI}(\widehat{\tau}) > \tilde{Var}^{I}(\widehat{\tau}) > \tilde{Var}^{II}(\widehat{\tau}). \tag{15}$$

Proof. Given an edge e arrived at timestamp t, let r be the computational round of e. For convenience, assume that e is the last edge of the round r, even if it is not, t will only be a very small number less than that of the last edge. Since we perform estimation on unlimited edge streams, this very small number has little effect. We first prove that the simplified variance of GREAT^I is smaller than that of TRIÈST-I. According to *Lemma 3.4*, we have

$$t = \sum_{r=0} x_r^I = k + \sum_{r=1} [\alpha \cdot k/(1-\alpha)^r] = k \cdot (1-\alpha)^{-r^I}. \tag{16}$$

Then, we have

$$\begin{aligned} Var^{TRI}\left(\frac{I_X}{P_X}\right) - Var^I\left(\frac{I_X}{P_X}\right) &= \left[\frac{t_{x_3}(t_{x_3}-1)}{k(k-1)} - 1\right] - \left[(1-\alpha)^{-2r^I_{x_3}} - 1\right] \\ &> \frac{t_{x_3}^2}{k^2} - (1-\alpha)^{-2r^I_{x_3}} \\ &= \frac{\left(k \cdot (1-\alpha)^{-r^I_{x_3}}\right)^2}{k^2} - (1-\alpha)^{-2r^I_{x_3}} = 0. \end{aligned}$$

Therefore, we have $Var^{TRI}\left(\frac{I_X}{P_X}\right) > Var^I\left(\frac{I_X}{P_X}\right)$.

Next, we proceed to prove that the simplified variance of GREAT^{II} is smaller than that of GREAT^I. The same, assume that e is the last edge of the round r. According to *Lemma 3.6*, we have

$$t = \sum_{r=0} x_r^{II} = k + \sum_{r=1} [(\exp(\alpha)-1) \cdot \exp((r-1)\cdot\alpha) \cdot k] = k \cdot \exp(\alpha \cdot r^{II}).$$

According to Equation 17, the computational round r^{II} of edge e arrived at time t in algorithm GREAT^{II} is $r^{II} = \frac{1}{\alpha} \cdot \ln\left(\frac{t}{k}\right)$. According to Equation 16, the computational round r^I of edge e arrived at timestamp t in algorithm GREAT^I is $r^I = -\frac{1}{\ln(1-\alpha)} \cdot \ln\left(\frac{t}{k}\right)$. Then, we have

$$\begin{aligned} Var^{II}\left(\frac{I_X}{P_X}\right) - Var^I\left(\frac{I_X}{P_X}\right) &= [\frac{t_{x_1}t_{x_2}}{k^2} \cdot (1-\alpha)^{-\left(2r^{II}_{x_3} - r^{II}_{x_1} - r^{II}_{x_2}\right)} - 1] - [(1-\alpha)^{-2r^I_{x_3}} - 1] \\ &= \frac{t_{x_1}t_{x_2}}{k^2} \cdot (1-\alpha)^{r^{II}_{x_1}+r^{II}_{x_2}} \cdot (1-\alpha)^{-2r^{II}_{x_3}} - (1-\alpha)^{-2r^I_{x_3}} \\ &= \frac{t_{x_1}t_{x_2}}{k^2} \cdot (1-\alpha)^{\frac{1}{\alpha}\cdot\ln\left(\frac{t_{x_1}\cdot t_{x_2}}{k^2}\right)} \cdot (1-\alpha)^{-\frac{2}{\alpha}\cdot\ln\left(\frac{t_{x_3}}{k}\right)} - (1-\alpha)^{\frac{2}{\ln(1-\alpha)}\cdot\ln\left(\frac{t_{x_3}}{k}\right)} \\ &\approx \frac{t_{x_3}^2}{k^2} \cdot (1-\alpha)^{\frac{1}{\alpha}\cdot\ln\left(\frac{t_{x_3}^2}{k^2}\right)} \cdot (1-\alpha)^{-\frac{2}{\alpha}\cdot\ln\left(\frac{t_{x_3}}{k}\right)} - (1-\alpha)^{\frac{2}{\ln(1-\alpha)}\cdot\ln\left(\frac{t_{x_3}}{k}\right)}. \end{aligned}$$

Equation 17 holds because the edge flow is unlimited, for any given triangle in the edge stream, t_{x_1}, t_{x_2}, and t_{x_3} can be viewed as variables of the same increasing rate and magnitude, and thus can be approximated by each other. Let $s = \frac{t_{x_3}^2}{k^2}$ and we convert the above equation into the following function:

$$h(s) = s^2 \cdot (1-\alpha)^{\frac{1}{\alpha}\cdot\ln(s^2)} \cdot (1-\alpha)^{-\frac{2}{\alpha}\cdot\ln(s)} - (1-\alpha)^{\frac{2}{\ln(1-\alpha)}\cdot\ln(s)}.$$

Easily, $h(s)$ have two zero points: s_1 and s_2. In other words, $h(s) < 0$ when $s > s_2$ or $s < s_1$ and $h(s) > 0$ when $s_1 \le s \le s_2$. Define $\frac{t_{x_3}}{k} = \frac{t_{x_1}}{k} + c_1$, $c_1 \ge 0$. Then, when $t_{x_1} > k \cdot (c_1 + s_2)$[1], we have $\tilde{Var}^I(\widehat{\tau}) > \tilde{Var}^{II}(\widehat{\tau})$.

Therefore, $\tilde{Var}^{TRI}(\widehat{\tau}) > \tilde{Var}^I(\widehat{\tau}) > \tilde{Var}^{II}(\widehat{\tau})$. □

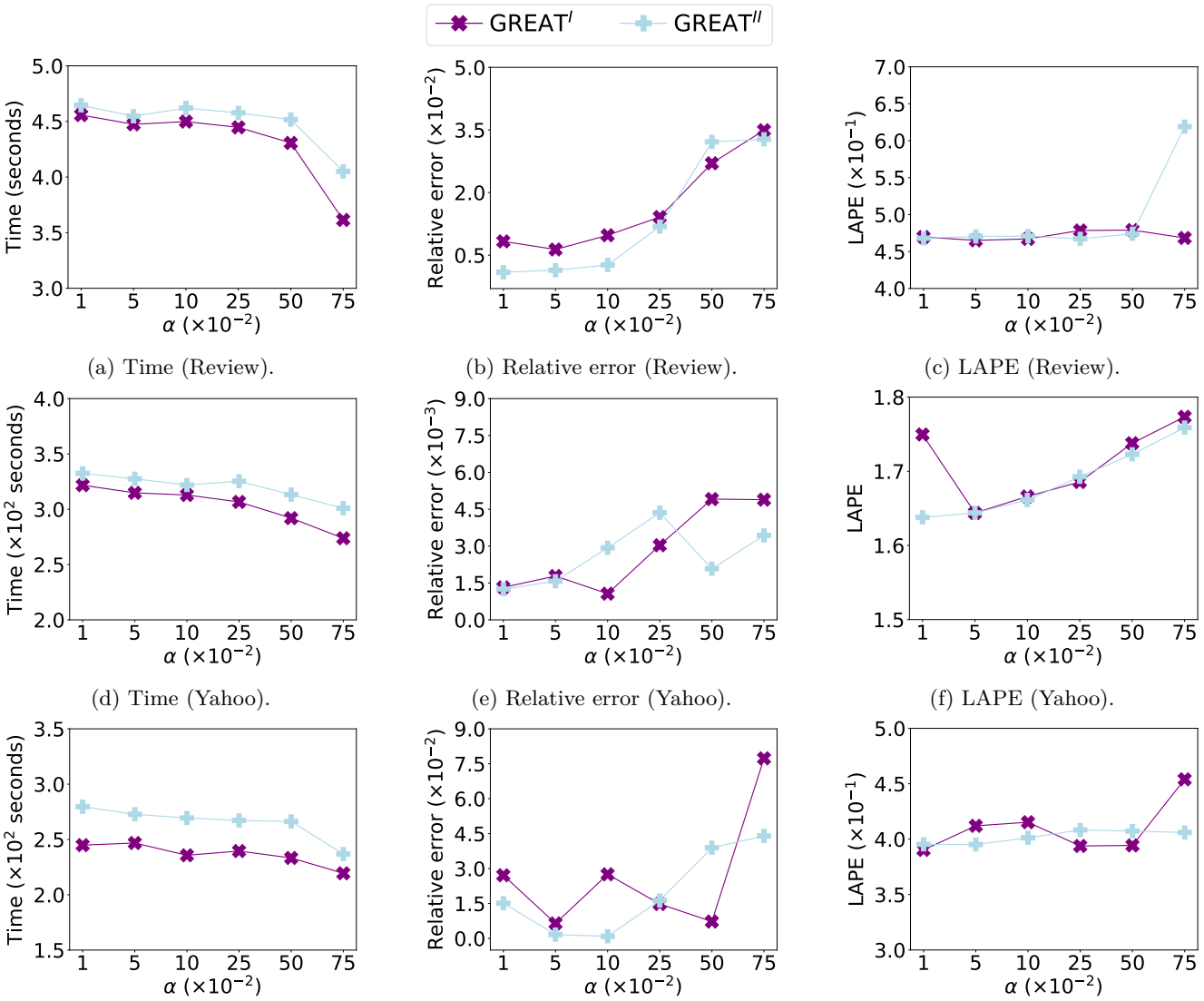
3 Supplementary Experiments

3.1 Parameter Sensitivity

3.1.1 Effect of α in GREAT^I and GREAT^{II}

Figure 3 shows the performance of algorithms GREAT^I and GREAT^{II} when varying α on different datasets. As the probability of edge being removed from the reservoir i.e.,α, increases, we observe that: (i) The elapsed time

[1]This condition necessitates a large dataset, which is characteristic of real-world streaming graphs.



(a) Time (Review). (b) Relative error (Review). (c) LAPE (Review).

(d) Time (Yahoo). (e) Relative error (Yahoo). (f) LAPE (Yahoo).

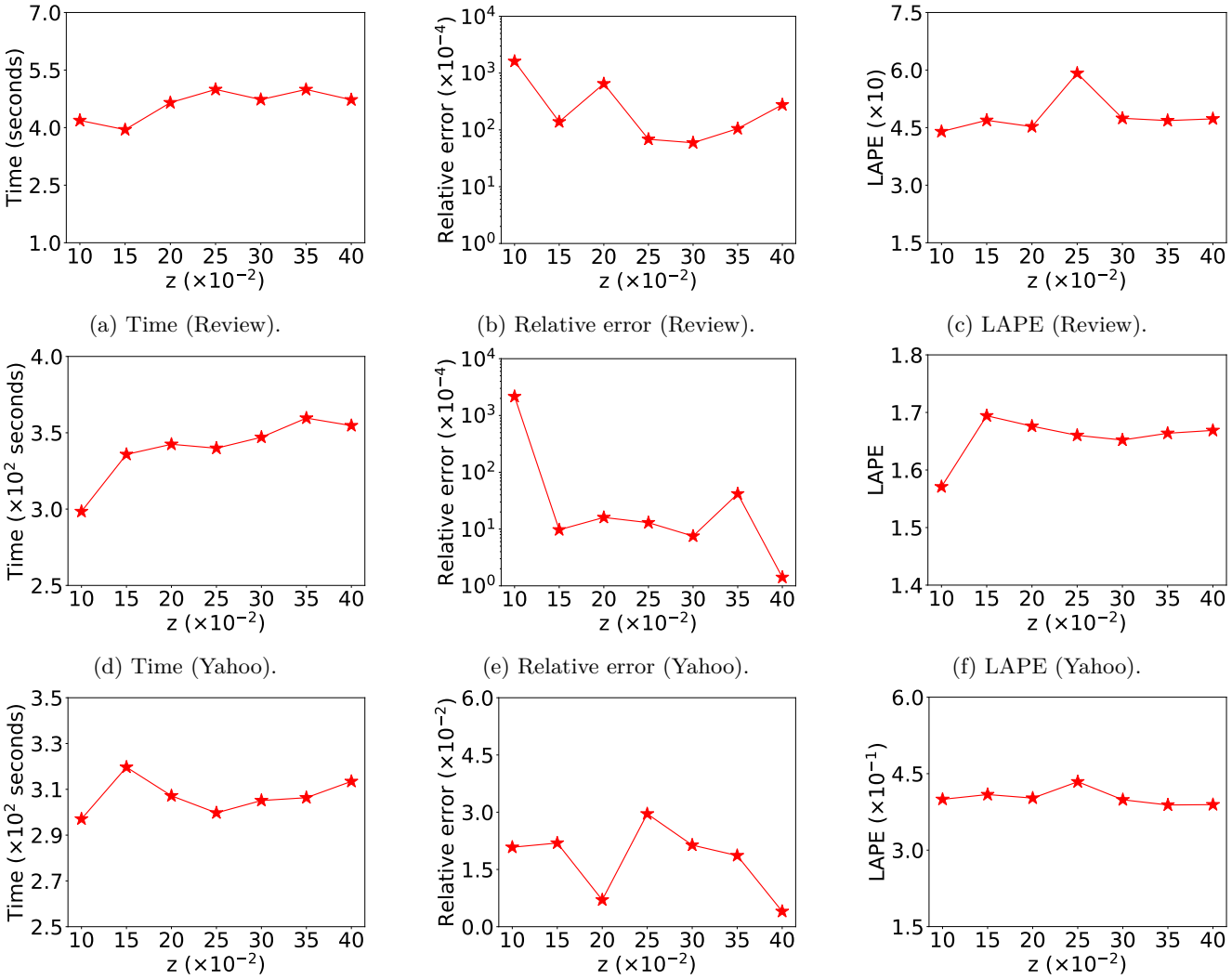
(g) Time (Wikipedia). (h) Relative error (Wikipedia). (i) LAPE (Wikipedia).

Figure 3: Varying α of GREAT^I and GREAT^{II}.

decreases as α increases, which is consistent with the amortized time complexity analysis (see Section 3.4 of the paper). (ii) Both the relative error and the LAPE become worse as α increases. This is because when α is large, the reservoir stores fewer edges so that fewer triangles are discovered. (iii) GREAT^I is slightly more efficient but is slightly less accurate than GREAT^{II} (see Sections 3.4 and 3.5 of the paper) in most cases. This is because the edge sampling probability of GREAT^I becomes significantly less than that of GREAT^{II} as more edges arrive. Then, the reservoir in GREAT^I becomes full slowly (incurring less computational cost) and stores less edges (missing some triangles).

3.1.2 Effect of z in GREAT$^+$

The performance of GREAT$^+$ with different values of z is presented in Figure 4. The larger the value of z, the smaller the value of $\alpha^{(r)}$, leading to more accurate but inefficient results. The main reason is that more edges are stored in the reservoir. Although there is a small vibration of the performance of GREAT$^+$ with different z, in general, we observe that as z increases, the elapsed time increases, and the relative error decreases, which follows our analysis in Section 4 of the paper. For LAPE, it can be seen that as z changes, LAPE vibrates slightly and does not change much, and this is because LAPE is strongly related to the number of discovered vertices, which is



(a) Time (Review). (b) Relative error (Review). (c) LAPE (Review).

(d) Time (Yahoo). (e) Relative error (Yahoo). (f) LAPE (Yahoo).

(g) Time (Wikipedia). (h) Relative error (Wikipedia). (i) LAPE (Wikipedia).

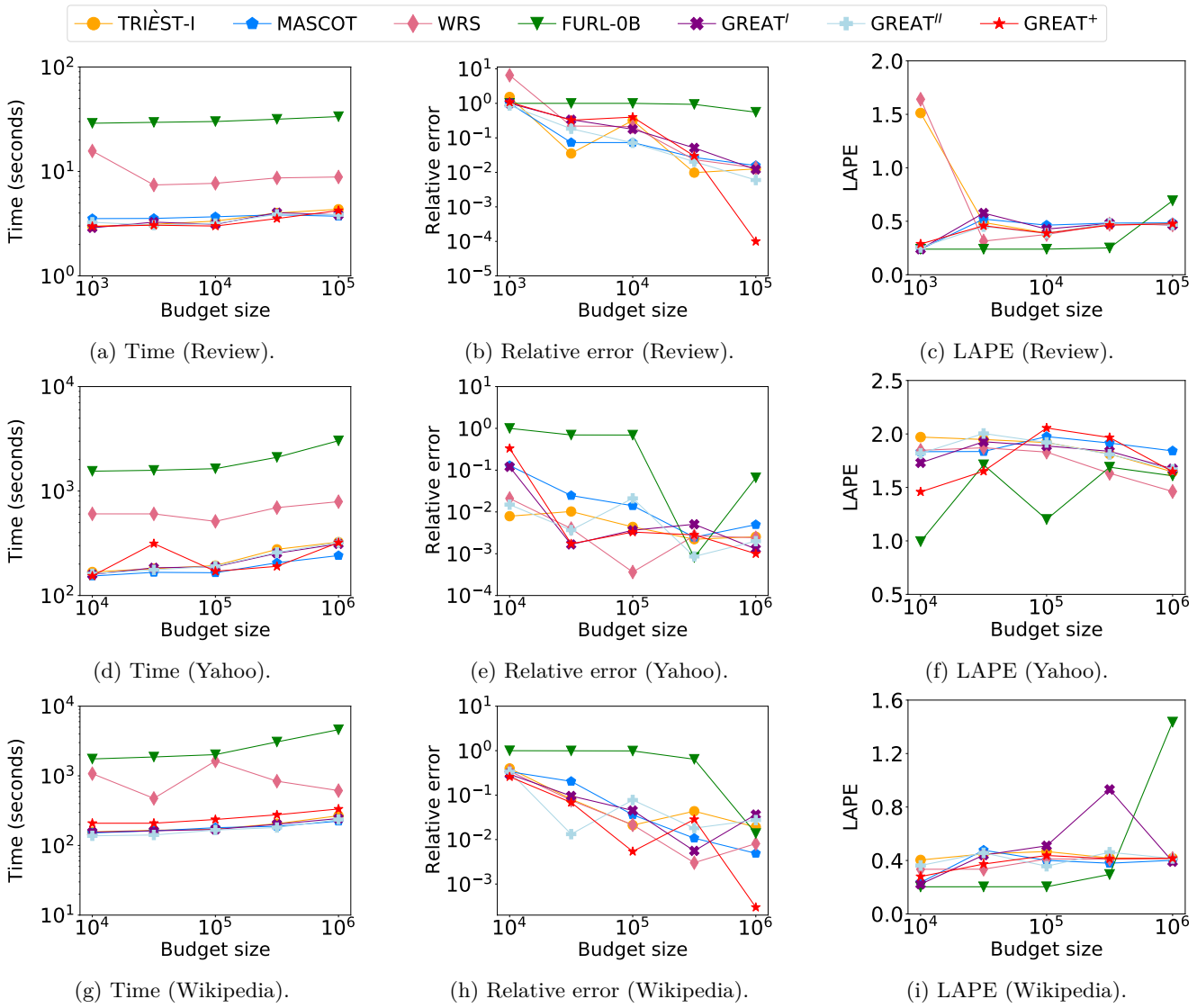
Figure 4: Varying z of GREAT$^+$.

not affected by z obviously. When we choose a larger z even though the relative error becomes better slightly, the elapsed time increases a lot. Therefore, we recommend avoiding very large z.

3.1.3 Effect of memory budget size in all algorithms

Figure 5 shows the performance of our algorithms and the competitors on additional three different datasets when varying the memory budget size. The elapsed time of all the algorithms increase as the memory budget size increases because more edges are maintained in the reservoir, which results in more spending on triangle searching. In general, as the memory budget size increases, the accuracy of most algorithms is improved, which means that LAPE and relative error decrease. This is because the larger the memory budget size is, more edges are stored in the reservoir and more triangles can be discovered.

Moreover, note that GREAT^I, GREAT^{II}, and GREAT$^+$ have the smallest relative errors when using small memory budge size.



(a) Time (Review). (b) Relative error (Review). (c) LAPE (Review).

(d) Time (Yahoo). (e) Relative error (Yahoo). (f) LAPE (Yahoo).

(g) Time (Wikipedia). (h) Relative error (Wikipedia). (i) LAPE (Wikipedia).

Figure 5: Varying memory budget size on four different algorithms.

3.1.4 Effect of λ in GREAT^{+}

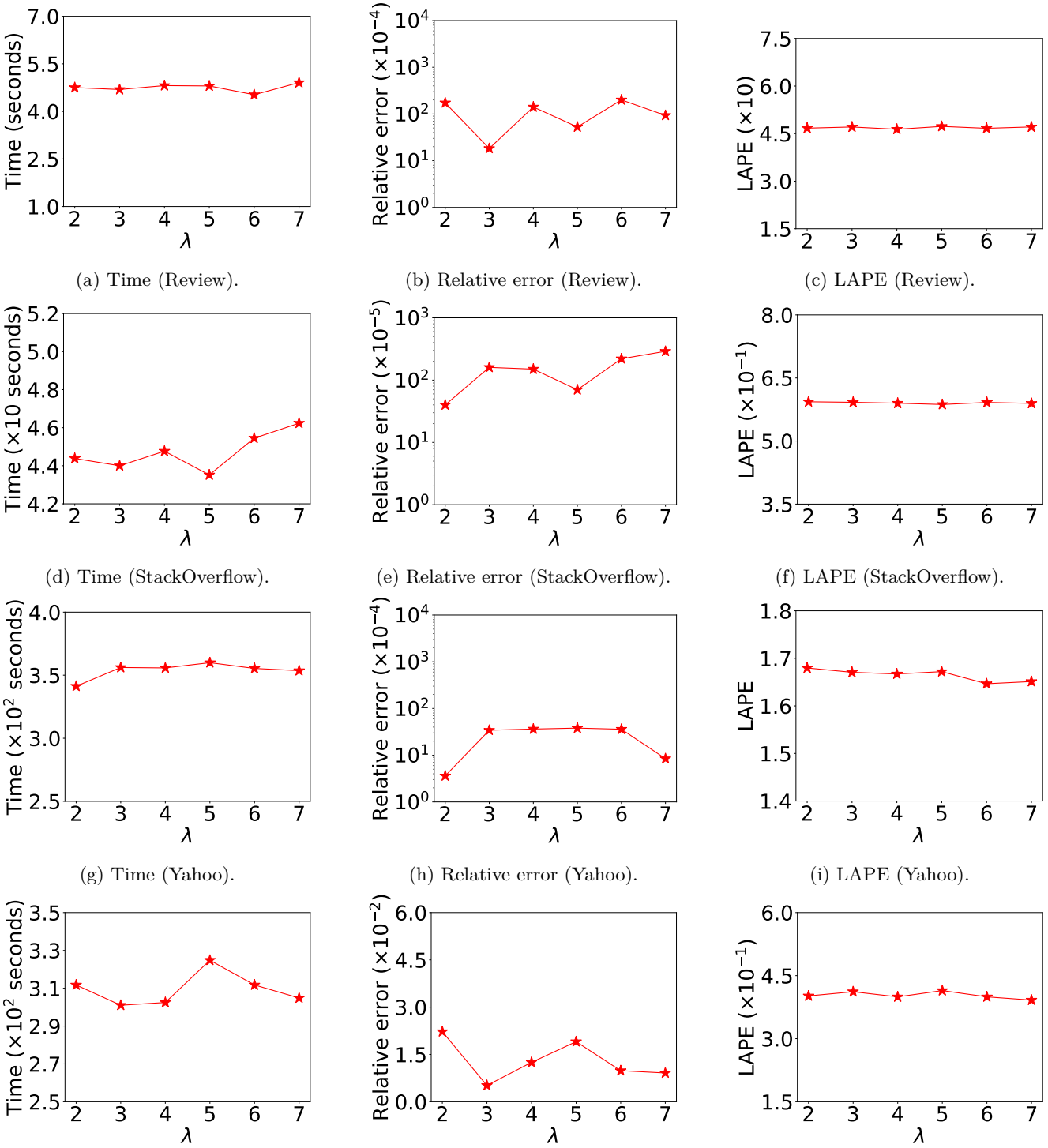
In order to have a preliminary understanding of the edge stream, GREAT^{+} perform GREAT^{II} in the first λ computational rounds with $\alpha^{(r)} = \alpha_0$, $0 \le r \le \lambda$. Next, we will provide the effect of parameters λ and α_0 in the algorithms. Figure 6 shows the performance of algorithm GREAT^{+} when varying λ on four datasets.

We observe that the elapsed time, the relative error and the LAPE are insensitive to λ. The λ is set to avoid the case where $\alpha^{(r)} = 1$ in GREAT^{+} due to r being too small, resulting in y being too small. The larger λ is, the less likely the above case is to occur, but in practice it does not have to be very large, too large λ makes GREAT^{+} degrade to GREAT^{II}. As long as λ is set correctly to avoid the above case, then the accuracy of the algorithm for GREAT^{+} is has little to do with λ. Note that $\alpha^{(r)} = 1$ is allowed, but not due to r being too small.

3.1.5 Effect of α_0 in GREAT^{+}

Parameter α_0 is the initial value of α in algorithm GREAT^{+}. Figure 7 shows the performance of algorithm GREAT^{+} when varying α_0 on four datasets.

In the rounds r whose $\alpha^{(r)} = \alpha_0$, the larger α_0 is, the less elapsed time and the higher the accuracy. A general



(a) Time (Review). (b) Relative error (Review). (c) LAPE (Review).

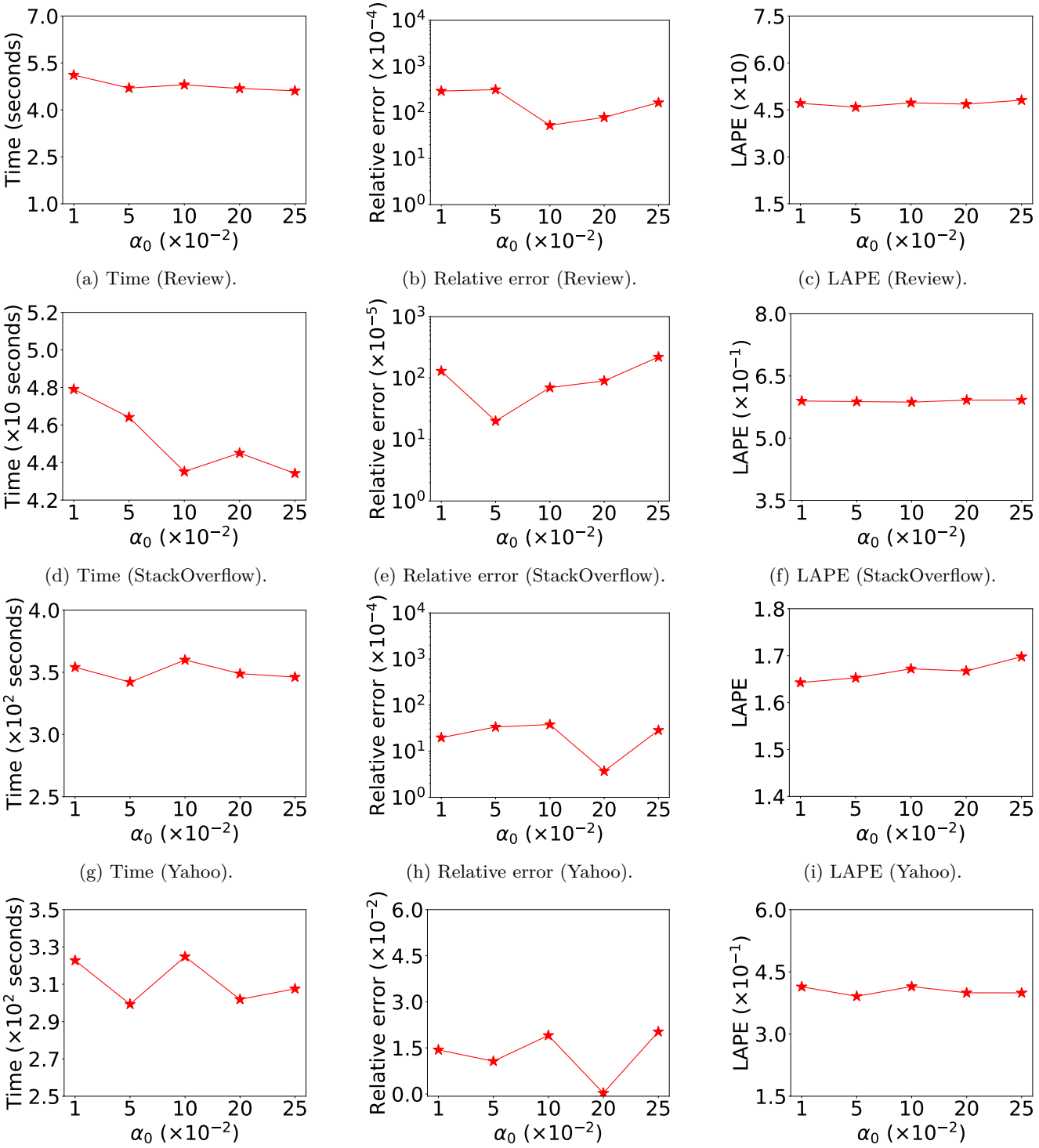
(d) Time (StackOverflow). (e) Relative error (StackOverflow). (f) LAPE (StackOverflow).

(g) Time (Yahoo). (h) Relative error (Yahoo). (i) LAPE (Yahoo).

(j) Time (Wikipedia). (k) Relative error (Wikipedia). (l) LAPE (Wikipedia).

Figure 6: Varying λ in GREAT$^+$.

edge stream has $\alpha^{(r)} > \alpha_0$, at $r > \lambda$, so α_0 acts on very few rounds, which means that it's possible that the effect of α_0 is not significant. We observe that the LAPE is insensitive to α_0 on Review, StackOverflow and Wikipedia, while the relative error and the elapsed time is insensitive to α_0 on Yahoo and Wikipedia. The other cases slightly follow the rule that the larger the α_0, the better the efficiency and the worse the accuracy. But α_0 in GREAT$^+$ is significantly less pronounced than α in GREAT^I and GREAT^{II}.



(a) Time (Review). (b) Relative error (Review). (c) LAPE (Review).

(d) Time (StackOverflow). (e) Relative error (StackOverflow). (f) LAPE (StackOverflow).

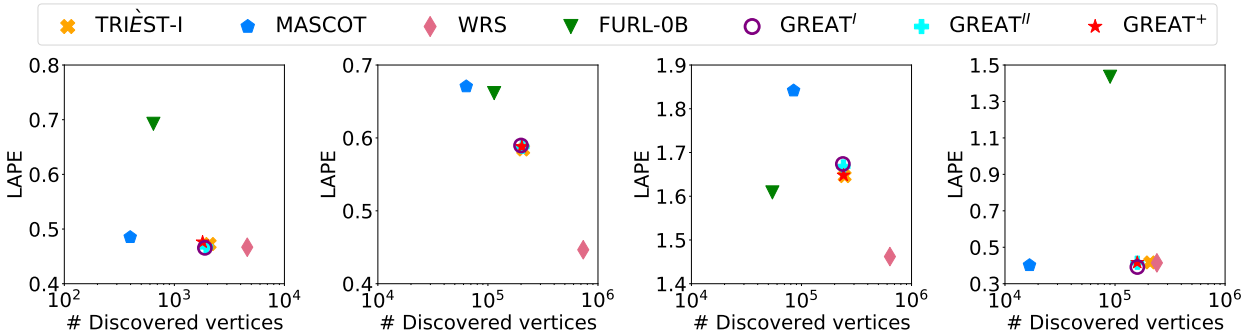
(g) Time (Yahoo). (h) Relative error (Yahoo). (i) LAPE (Yahoo).

(j) Time (Wikipedia). (k) Relative error (Wikipedia). (l) LAPE (Wikipedia).

Figure 7: Varying α_0 in GREAT$^+$.

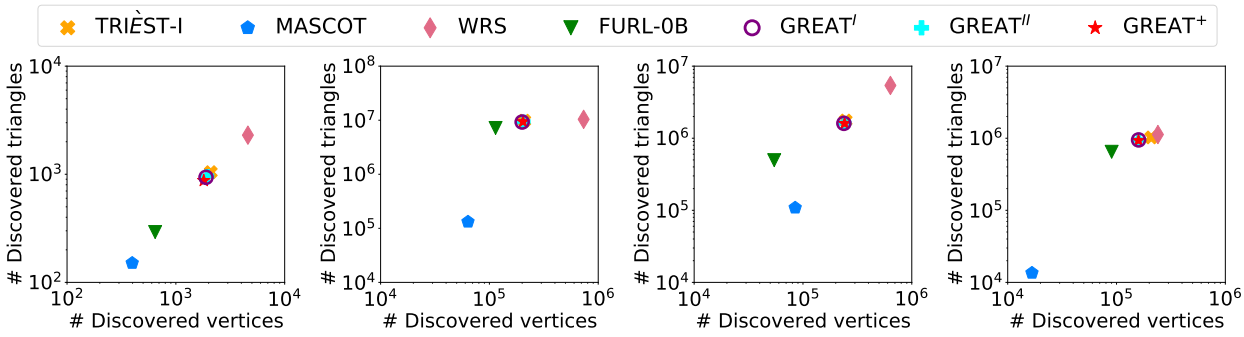
3.2 Discussion of LAPE

Figure 10 shows the relationship between time and LAPE of different algorithms. Figure 8 represents the relation between LAPE and the number of discovered vertices, and it shows that removing the anomaly of FRUL-0B, whose performance is determined by the default hash function, the LAPE is negatively correlated with the number of discovered vertices. It makes sense that when the number of discovered vertices is higher, the more vertices can be



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

Figure 8: LAPE vs. the number of discovered vertices.



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

Figure 9: the number of discovered vertices vs. the number of discovered triangles.

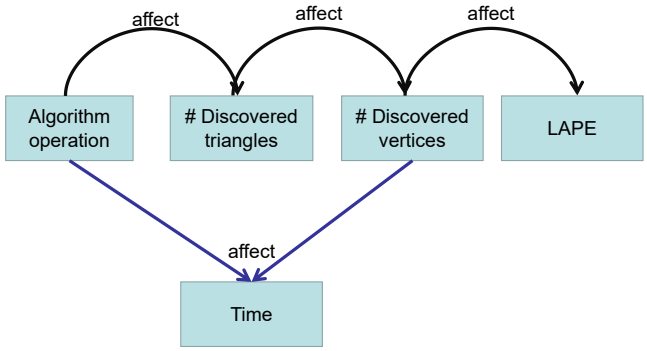
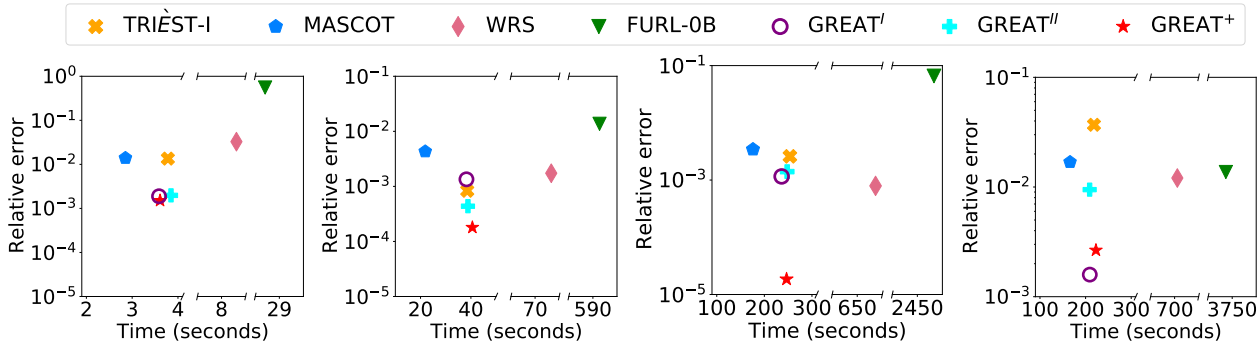


Figure 10: The effect of LAPE.

computed its local triangles, the lower the LAPE value is.

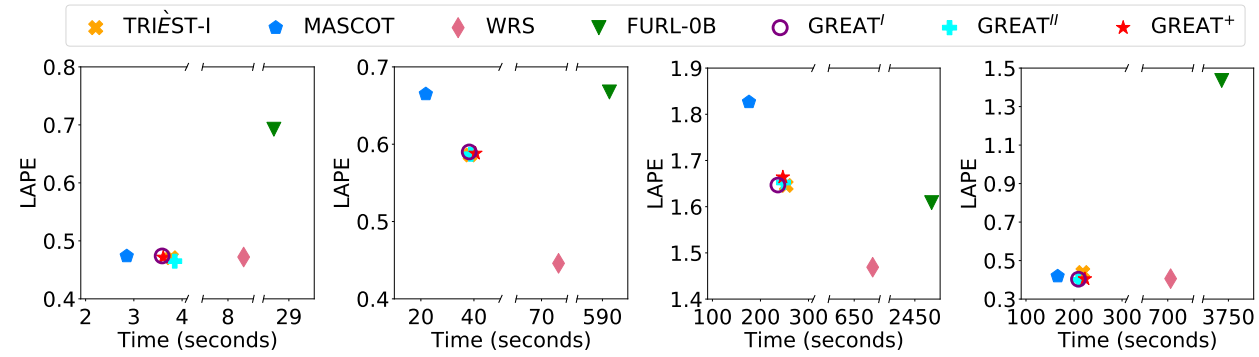
It is intuitive to think that the reason of the difference in the number of discovered vertices between different algorithms is the difference in the number of discovered triangles. Figure 9 demonstrates the relationship between the number of discovered triangles and the number of discovered vertexes, and it shows that the higher the number of discovered triangle, the higher the number of discovered vertices. MASCOT discovers less triangles because it maintains less edges in the reservoir, and WRS discovers most triangles because the waiting room is certainly capable of discovering all triangles with short time interval. GREAT^{I}, GREAT^{II}, GREAT^{+}, and TRIÈST-I discover similar numbers of triangles, and therefore similar numbers of discovered vertices, and therefore LAPE is close. Thus, it is the number of discovered triangles that really affects LAPE.

Moreover, in most cases, the lower LAPE is, the higher computational time is. This is because the higher the cost of maintaining the counting of local triangles, and naturally the efficiency decreases and accuracy increases. In summary, LAPE is indirectly related to the computational time, while it is directly related to the number of discovered vertices, which is affected by the number of discovered triangles.



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

Figure 11: Relative error vs. computational time (on the new machine).



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

Figure 12: LAPE vs. computational time (on the new machine).

3.3 Experiments on Another Machine

To corroborate that the performance of all algorithms is related to the set budget k instead of the machine memory, we executed the experiments on another machine. We add the experiments on a machine with 2 Intel Xeon Platinum 8260 CPUs (48 cores, 2.4 GHz per core) and 512 GB of DDR4 ECC memory and with Ubuntu 20.04 LTS. As Figures 11 and 12 show, the results are consistent with those presented in the previous version.

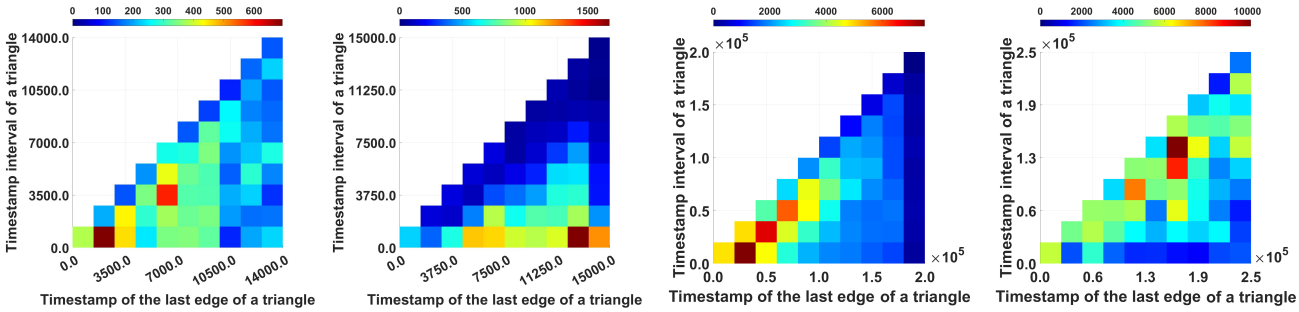
3.4 Experiments on Supplementary Datasets

Table 2: Statistics of new Datasets.

Dataset	**#Vertices**	**#Edges**	**#Triangles**
CollegeMsg [15]	1,899	59,835	14,318
Bitcoin [12]	3,783	24,186	22,146
Facebook [20, 13]	46,952	876,993	122,852
Wiki-edits [13]	35,868	540,090	220,783

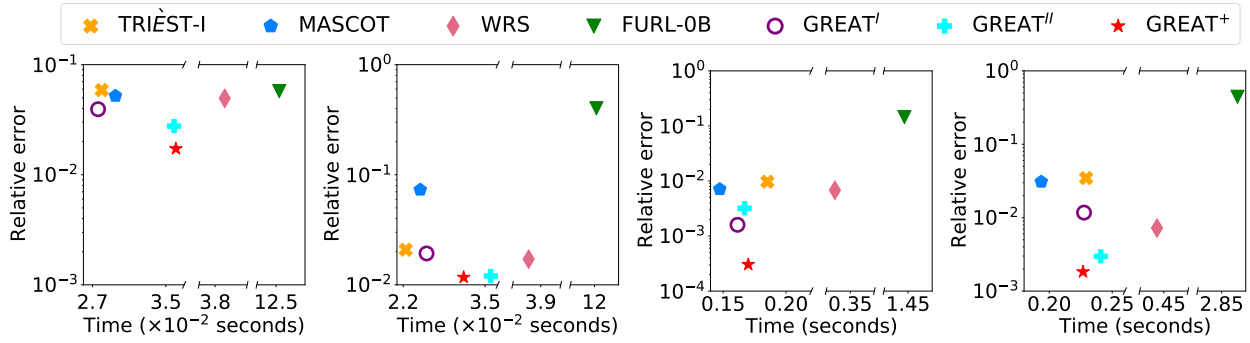
To further explore the proposed algorithms, we additionally implement the algorithms in four new smaller real-world graph streams. The CollegeMsg dataset represents the online social network of the University of California, Irvine, where each edge corresponds to a message sent between two user vertices. The Bitcoin dataset is a network of users who transact with Bitcoin on a platform called Bitcoin Alpha. Dataset Facebook [20, 13] is a network which represents a user leaving a post on another user's wall. Dataset Wiki-edits is a network which represents the edits of web pages by users on the Hungarian Wikibooks [13].

The number of edges and vertices of these four new algorithms is shown in Table 2. Figure 13 shows the time interval distribution changing with time of four new datasets. CollegeMsg and Facebook form distributions of



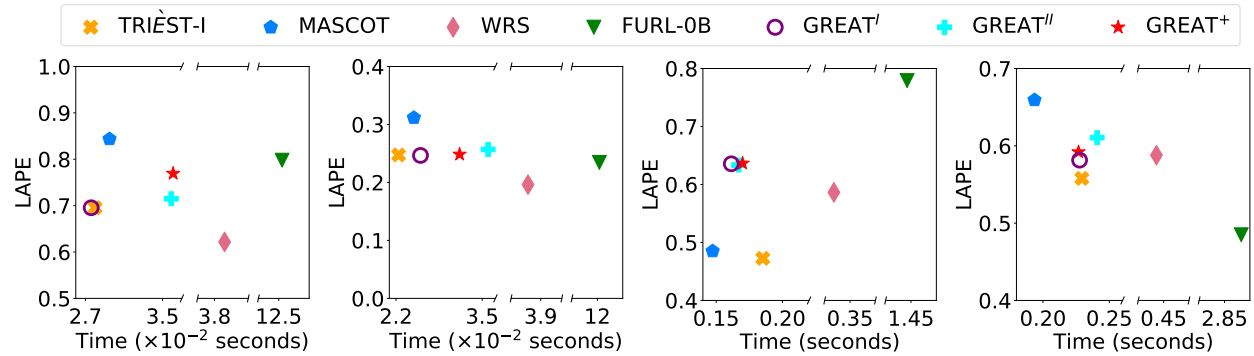
(a) CollegeMsg. (b) Bitcoin. (c) Facebook. (d) Wiki-edits.

Figure 13: Dynamic timestamp intervals of triangles (in new datasets).



(a) CollegeMsg. (b) Bitcoin. (c) Facebook. (d) Wiki-edits.

Figure 14: Relative error vs. computational time (on new datasets).



(a) CollegeMsg. (b) Bitcoin. (c) Facebook. (d) Wiki-edits.

Figure 15: LAPE vs. computational time (on new datasets).

triangles interval which slightly skew towards short time intervals at each timestamp, while Bitcoin and Wiki-edits form distributions of triangles interval which slightly skew towards long time intervals at each timestamp.

The results on these four new datasets are consistent with those in our the paper. As shown in Figure 14, GREAT^{+} outperforms the other competitors in terms of relative error. On the CollegeMsg and Wiki-edits datasets, GREAT^{I} and GREAT^{II} do not show significant superiority in terms of accuracy but follow GREAT^{+} in terms of relative error. For LAPE, WRS performs best on CollegeMsg and Bitcoin, while our proposed methods perform comparably to WRS but with greater efficiency. FURL-0B performs best on Facebook, and TRIÈST-I performs best on Wiki-edits. However, these methods show poor performance in terms of relative error or time, while our approach performs well across all metrics.

3.5 Case Study

In this section, We perform anomaly detection on StackOverflow, which doesn't have labels, so we only focus on structural anomalies. StackOverflow represents a question-and-answer website for computer programmers, where each vertex represents a user, and interactions between users generate edges.

3.5.1 Case I: Discovering anomalous vertices.

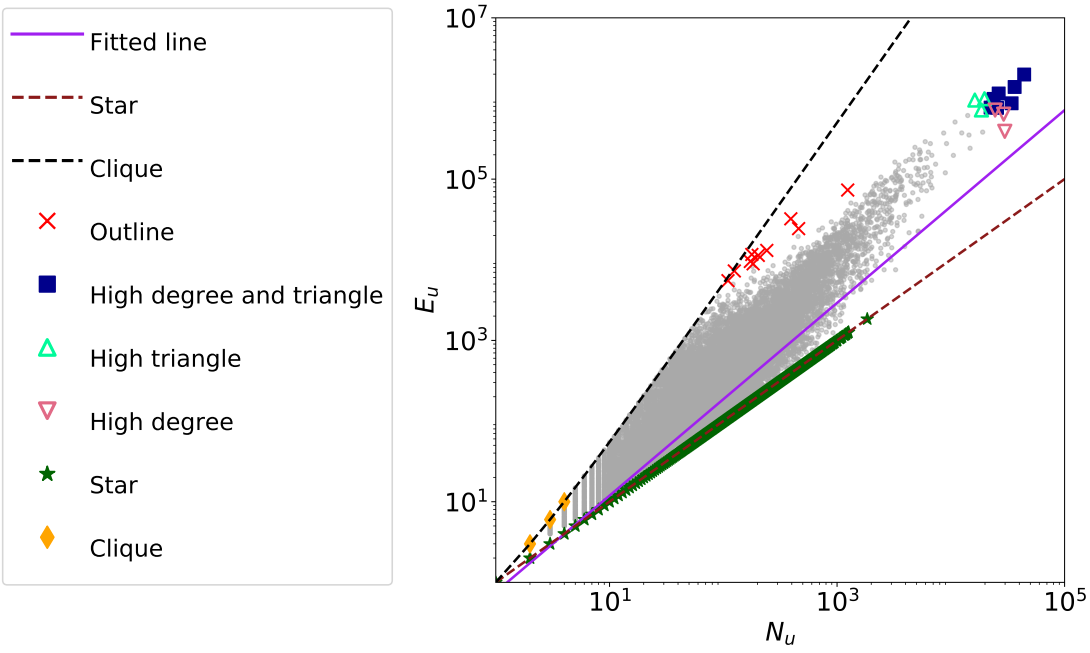


Figure 16: Anomalous vertices.

Local triangle counting plays an important role in determining the characteristics of the vertex which is one of the main features in anomaly detection [2, 14]. Anomalous behavior may imply violations such as transaction fraud [5], network hacking [16], spam [3], etc. In addition to revealing violations or dangerous behaviors, anomaly detection is also useful for discovering rare events as well as critical elements [8, 11].

Following the case study in MASCOT [14], we adopt the Oddball strategy [1] to determine anomalous vertices based on their degrees and local triangle counts. Oddball use the structure of the egonet to determine anomalous vertices. An egonet of vertex u is an induced subgraph of u and its neighboring vertices, and N_u is the number of neighbors of u and E_u is the number of edges in the egonet. Therefore, N_u is also the degree of u. Let τ_u be the local triangle counts of u, so $E_u = N_u + \tau_u$. Specifically, we apply GREAT$^+$ to estimate the local triangle counts and maintain the degree values of all vertices.

The following 6 types of vertices are considered as anomalous, including vertices with high Outline values, vertices with high degree values, vertices with high local triangle counts, vertices with both high degree and local triangle counts, vertices with Star egonets and vertices with Clique egonets. Figure 16 shows these anomalous vertices represented with different markers, and we explain the details bellow.

First, we select the top 10 vertices with the highest Outline values, denoted by ×.

$$Outline(u) = \frac{\max\left\{E_u, C \times N_u^\beta\right\}}{\min\left\{E_u, C \times N_u^\beta\right\}} \times \log\left(\left|E_u - C \times N_u^\beta\right| + 1\right), \quad 1 \le \beta \le 2$$

where N_u is the degree of vertex u, $E_u = N_u + \tau_u$ where τ_u is the local triangle count of vertex u, and C and β are parameter. It is proved that the relationship between the degree of a vertex and its local triangle count should obey the power law rule [11], and it is represented as

$$E_u \propto N_u^\beta, 1 \le \beta \le 2,$$

where N_u be the degree of vertex $u \in V$, and E_u is $E_u = N_u + \tau_u$, where τ_u be the local triangle counts of u. The Oddball use the Outline of the vertex u to measure the distance of vertex u from the power law rule, which means that how far the degree and the local triangle count of u is from the fitting line. The Outline value of vertex u measures how far the degree and the local triangle count of u is from the power law fitting line, denoted by —.

Then, find the vertices with large Outlof. Outline is a simple and easy to compute metric that not only helps in detecting anomalous vertices but also provides a way to rank the vertex based on outlier score. However, the Oddball suggests that a vertex far from most of vertices but still follows the power law rule is also an anomalous vertex, and in their observations such a vertex usually has a high degree value or and high local triangle counts. Moreover, the Oddball suggests to use the $Outlof(u)$ in IOF [4] to measure the distance between vertex u and all other vertices. We also highlight the vertices with the highest Outlof values as anomaly vertices, and they happen to be the vertices with the largest degree or estimated local triangle count. The top 10 vertices with the highest degree values are denoted by ▽ and ■. The users represented by these vertices interact frequently with other users. These users may be enthusiasts or practitioners of the domain. The top 10 vertices with the highest estimated local triangle counts are denoted by △ and ■. These vertices represent the users whose neighbors have frequent interactions with each other. It is highly likely that the users and its neighbors are in a community, which means that they may belong to the same interest group or professional organization. The vertices with both top 10 estimated degree values and top 10 estimated local triangle counts are denoted by ■. The users represented by these vertices interact frequently with other users and their neighbors also have active communications, which means that they may be the superstars or pioneers in the whole website community.

At last, find the vertices with Star or Clique structure. In most real-world networks, it is normal to have some connectivity between the neighbors of a vertex, "a friend of a friend is usually a friend". Therefore, either extreme of too dense(Clique) or too sparse(Star) connectivity between the neighbors of a vertex is suspicious. If the degree

(a) Star. (b) Clique.

Figure 17: Anomaly structure.

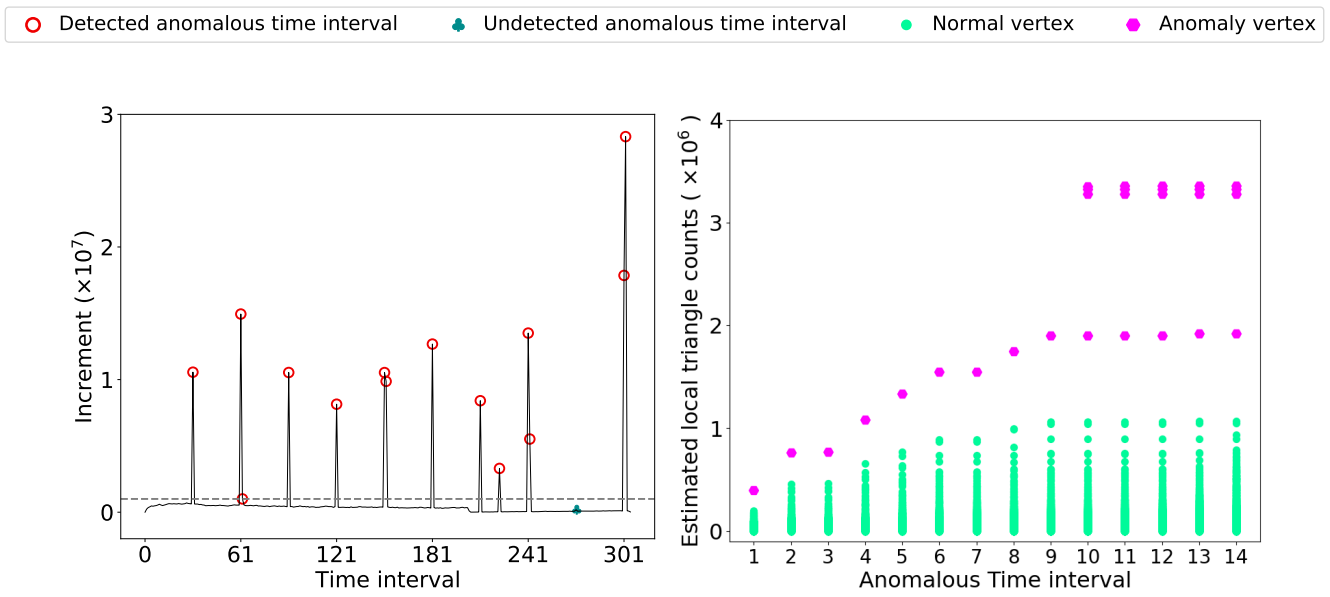
of a vertex is not 0 but its local triangle counts is 0, the structure of its egonet is called Star. The vertices with Star structures are denoted by ⋆. Theses vertices represent the users who interact with users who do not know each other. It is possible that their neighbors are from different areas and these users are "viral users" or "online water army" [6], who purposefully interacts with users in order to drive a trend or bring awareness to certain topics. Therefore, these user need to pay attention to. The vertices with Clique structures (if the triangle density is 1) are denoted by ♦. The structure of a egonet is called Clique, if triangle density of the vertex is 1. The triangle density is

$$density(u) = \frac{\tau_u}{\binom{N_u}{2}},$$

where τ_u is the local triangle counts of vertex u and N_u is the degree value, i.e., the egonet of vertex u is a complete graph. Theses vertices represent the users who are in a community, and they may be leaders or hot followers in the community. They may join the "follower-boosting" service [10, 17] and interact with fake users frequently to maliciously increase the popularity. These fake users will control be managed together to provide this service , so they will be strong connections. We plot the dotted line - - -in Figure 16 to represent the Star and - - - in Figure 16 to represent the Clique, and they are the two most extreme cases of triangle density of egonet.

3.5.2 Case II: Detecting anomalous time intervals.

A sudden change in the number of triangles often signals an anomaly in the dynamic graph, therefore, detecting the number of triangles in graph streams is an important task for observing and managing the dynamic graph [9, 18]. The spike of the increment of triangle counts in a graph stream implies that anomalous dense structures are



(a) Detection of anomalous time intervals by the increment of estimated global triangle counts.

(b) Detection of anomalous vertices in anomalous time intervals.

Figure 18: Detection of anomalous time intervals and vertices.

present in the graph, which in social networks means the invasion of a large number of 'Online Water Army' [6]. In trading networks, The spike of the increment of triangle counts may mean that frauds are occurring [7]. In email networks, the spike of the increment of triangle counts may mean the rapid spreading of viral emails [19].

Following ThinkSpot [18], we injected anomalous communities of size 100-500 every 3,000,000 edges in the StackOverflow graph stream. GREAT$^+$ is applied to estimate global triangle counts every 100,000 edges to see whether they can be detected. If the increment between the estimated triangle counts of two consecutive time intervals is larger than the threshold 10^6, the latter time interval is identified as anomalous time interval. Figure 18a shows that our algorithm successfully detects 14 out of 15 anomalous time intervals. ◯ denotes the anomalous communities being detected, and ♣ is the anomalous communities without being undetected.

For each detected anomalous time interval, output the estimated local triangle counts of all vertices. The vertices with estimated local triangle counts away from others in each anomalous time interval are considered anomalous and are highly likely to cause the spike of the increment. In Figure 18b, the marker ⬢ denotes the anomaly vertices in the each anomalous time interval, and all these discovered anomalous vertices may from the anomalous community.

References

[1] Akoglu, L., McGlohon, M., and Faloutsos, C. oddball: Spotting anomalies in weighted graphs. In *Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part II* (2010), M. J. Zaki, J. X. Yu, B. Ravindran, and V. Pudi, Eds., vol. 6119 of *Lecture Notes in Computer Science*, Springer, pp. 410–421.

[2] Becchetti, L., Boldi, P., Castillo, C., and Gionis, A. Efficient algorithms for large-scale local triangle counting. *ACM Trans. Knowl. Discov. Data 4*, 3 (2010), 13:1–13:28.

[3] Boykin, P. O., and Roychowdhury, V. P. Leveraging social networks to fight spam. *Computer 38*, 4 (2005), 61–68.

[4] Breunig, M. M., Kriegel, H., Ng, R. T., and Sander, J. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA* (2000), W. Chen, J. F. Naughton, and P. A. Bernstein, Eds., ACM, pp. 93–104.

[5] Chau, D. H., Pandit, S., and Faloutsos, C. Detecting fraudulent personalities in networks of online auctioneers. In *Knowledge Discovery in Databases: PKDD 2006, 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Berlin, Germany, September 18-22, 2006, Proceedings* (2006), J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds., vol. 4213 of *Lecture Notes in Computer Science*, Springer, pp. 103–114.

[6] Chen, C., Wu, K., Srinivasan, V., and Zhang, X. Battling the internet water army: Detection of hidden paid posters. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (2013), pp. 116–120.

[7] Coppolino, L., D'Antonio, S., Formicola, V., and Romano, L. Applying extensions of evidence theory to detect frauds in financial infrastructures. *Int. J. Distributed Sens. Networks 11* (2015), 980629:1–980629:16.

[8] Dasu, T., and Johnson, T. *Exploratory Data Mining and Data Cleaning*. John Wiley, 2003.

[9] Han, G., and Sethu, H. Edge sample and discard: A new algorithm for counting triangles in large dynamic graphs. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, Sydney, Australia, July 31 - August 03, 2017* (2017), J. Diesner, E. Ferrari, and G. Xu, Eds., ACM, pp. 44–49.

[10] Hooi, B., Shin, K., Song, H. A., Beutel, A., Shah, N., and Faloutsos, C. Graph-based fraud detection in the face of camouflage. *ACM Trans. Knowl. Discov. Data 11*, 4 (2017), 44:1–44:26.

[11] Kang, U., Meeder, B., Papalexakis, E. E., and Faloutsos, C. Heigen: Spectral analysis for billion-scale graphs. *IEEE Trans. Knowl. Data Eng. 26*, 2 (2014), 350–362.

[12] Kumar, S., Hooi, B., Makhija, D., Kumar, M., Faloutsos, C., and Subrahmanian, V. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (2018), ACM, pp. 333–341.

[13] Kunegis, J. KONECT: the koblenz network collection. In *WWW* (2013), pp. 1343–1350.

[14] Lim, Y., Jung, M., and Kang, U. Memory-efficient and accurate sampling for counting local triangles in graph streams: From simple to multigraphs. *ACM Trans. Knowl. Discov. Data 12*, 1 (2018), 4:1–4:28.

[15] Panzarasa, P., Opsahl, T., and Carley, K. M. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *J. Assoc. Inf. Sci. Technol. 60*, 5 (2009), 911–932.

[16] Sequeira, K., and Zaki, M. J. ADMIT: anomaly-based data mining for intrusions. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada* (2002), ACM, pp. 386–395.

[17] Shin, K., Eliassi-Rad, T., and Faloutsos, C. Patterns and anomalies in k-cores of real-world graphs with applications. *Knowl. Inf. Syst. 54*, 3 (2018), 677–710.

[18] Shin, K., Oh, S., Kim, J., Hooi, B., and Faloutsos, C. Fast, accurate and provable triangle counting in fully dynamic graph streams. *ACM Trans. Knowl. Discov. Data 14*, 2 (2020), 12:1–12:39.

[19] Stolfo, S. J., Hershkop, S., Hu, C.-W., Li, W.-J., Nimeskern, O., and Wang, K. Behavior-based modeling and its application to email analysis. *ACM Transactions on Internet Technology (TOIT) 6*, 2 (2006), 187–221.

[20] Viswanath, B., Mislove, A., Cha, M., and Gummadi, P. K. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM Workshop on Online Social Networks, WOSN 2009, Barcelona, Spain, August 17, 2009* (2009), J. Crowcroft and B. Krishnamurthy, Eds., ACM, pp. 37–42.