Revisiting Wedge Sampling for Triangle Counting

Ata Turk
Boston University
Boston, Massachusetts
ataturk@bu.edu

Duru Türkoğlu
Loyola University Chicago
Chicago, Illinois
dturkoglu@luc.edu

ABSTRACT

Triangle counts of massive graphs can provide important information regarding the structure of the networks these graphs model. Exact triangle counting can be expensive and thus researchers have proposed a number of approximation approaches. The state-of-the-art triangle count approximation techniques depend on wedge (two-path) sampling. In this paper we offer a mechanism to significantly improve wedge sampling for triangle counting. We shrink the sampling space by eliminating wedges that are less likely to participate in triangles. Experiments over large-scale real-world graphs show that proposed mechanism provides five- to a few hundred-folds sampling space reduction. When compared against the state-of-the-art approaches, it requires as low as ∼100× less sampling to provide the same accuracy, or makes as low as ∼8× less error when used with the same sampling ratio.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; **Approximation algorithms**; • **Information systems** → Social networks.

KEYWORDS

Triangle count estimation; Wedge sampling; Approximation algorithms

ACM Reference Format:
Ata Turk and Duru Türkoğlu. 2019. Revisiting Wedge Sampling for Triangle Counting. In *Proceedings of the 2019 World Wide Web Conference (WWW'19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313534

1 INTRODUCTION

The number of triangles in a graph is an important metric that represents the transitivity of the relations and the degree of clustering of the entities modeled by the graph. This metric has use in applications such as social network analysis [40], gene expression microarray data analysis [18], or text summarization [2].

Currently, practically used exact triangle counting and listing algorithms have a complexity of $O(m^{3/2})$ [30], where m is the number of edges. However, target graphs of interest for triangle counting can have billions of edges and hundreds of billions of triangles, which render exact algorithms costly. Furthermore, for many settings the exact triangle count is not necessary and approximations are sufficient as the graphs are dynamic. Thus, a number of approximation solutions for estimating the triangle count have been proposed [6, 7, 9, 10, 14, 16, 21, 22, 24, 28, 31–33, 36–38].

The state-of-the-art approximation approaches utilize wedge sampling. Wedge-sampling-based solutions have shown to be fast and accurate in estimating triadic measures [32, 38]. These approaches sample wedges from a graph, check whether sampled wedges are "closed" (form a triangle in the original graph) or not, and based on the ratio of closed and open wedges, estimate the triangle count in the original graph. Unfortunately, in large-scale power-law graphs there are too many wedges compared to the number of triangles. The number of wedges in such graphs can easily be in the order of trillions and the chances of finding a "closed" wedge can be as low as one in ten thousand. Estimating the number of triangles available with confidence while sampling from such a large wedge-space requires a significant number of samplings.

In this paper we revisit the wedge-sampling approaches for estimating triangle counts in large graphs. We first show that during sampling it is not necessary to consider all of the wedges in a graph and it is possible to reduce the sampling space significantly. In a preprocessing step, we first define a partial ordering of vertices, and then use that ordering to categorize wedges into three groups. We show that sampling from any one of these groups is sufficient for triangle counting. We use this finding to restrict the sampling-space for triangle counting, which leads to more accurate estimations with far fewer sampled wedges when compared to existing aproaches.

We theoretically prove that our mechanism guarantees at least a three-fold reduction in the sampling space, thus guaranteeing an accuracy no worse than standard wedge sampling, and we show via experimental analysis that we achieve one to two orders of magnitude reduction in practice for real-world graphs. Note that such reduction can pave the way for more accurate analysis and estimations over larger graphs (e.g. in some cases, the number of candidate wedges for sampling decrease so much that running exact algorithms become feasible).

Our contributions can be listed as follows:

- We show that any total order on the vertices of a graph can be used to divide the wedges in a graph into three categories, all of which can be separately and accurately used for triangle count estimation.
- We provide a greedy total ordering heuristic that divides the wedges in a graph such that the number of wedges in a pre-specified category is guaranteed to be at most one third of the total number of wedges.
- We prove that the number of wedges sampled using our greedy heuristic is a 2-approximation, i.e., it is at worst twice as much as that of an optimal ordering for the considered category.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313534

- We provide a wedge-sampling based algorithm that utilizes our categorization for improved triangle count estimation.
- We provide closed form expressions for the Relative Standard Error of our algorithm to allow for comparison against previously proposed algorithms over known graphs.
- We evaluate the triangle counting performance of the proposed algorithm against the state-of-the-art over real-world large-scale graphs. Results indicate that proposed sampling algorithm outperforms the state-of-the-art significantly, requiring between two to hundred-fold less sampling to provide the same estimation accuracy.

In the remainder of the paper, we first provide a brief background and the related work (Section 2), then explain our improved wedge sampling approach (Section 3), and then we evaluate our approach (Section 4) and conclude (Section 5).

2 RELATED WORK

Given an undirected graph $G = (V, E)$, where V is a set of n vertices, and E is a set of m edges, we would like to find out T, the number of triangles (three edge cycles) in G. The triangle counting problem can be considered a special case of the subgraph counting [20] or specific length cycle counting [3] problems.

The most efficient algorithm for exact triangle counting uses the fact that if A is the adjacency matrix of G, then the diagonal elements of A^3 contain (two times) the number of triangles of the corresponding vertex. The complexity of this algorithm is $O\left(m^{2\gamma/(\gamma+1)}\right)$ [3], where m is the number of edges in the graph and γ is the exponent of the used matrix multiplication algorithm. Considering a low known $\gamma < 2.38$ for matrix multiplication [25], this complexity can be listed as $\approx O(m^{1.407})$.

Practically efficient exact triangle counting algorithms generally have a complexity of $O(m^{3/2})$ and we refer [30] for a thorough practical comparison of efficient in-memory sequential counting and listing implementations. As most graphs of interest do not fit in memory, some triangle counting studies focus on reducing the number of disk I/Os performed [12, 19] during counting, while others focus on parallelizing the counting process [4, 11, 39].

In cases where the exact count is not needed and an "accurate enough" approximation to the actual count is sufficient, significantly faster triangle-count approximation approaches can be utilized. These approaches can be further divided into the streaming settings where graph components arrive as a data stream [1, 6, 9, 14, 16, 17, 27, 35], and the static settings where the entire graph is available for analysis [32, 33, 37, 38]. Our focus on this work is on the static scenario, where the sampling approaches randomly sample edges [10, 37] or adjacent edge pairs (wedges) [33, 38] to estimate the number of triangles in G based on the statistics obtained from the sampled graph.

State-of-the-art approximation solutions to triangle counting for static settings are based on wedge sampling [33, 38]. The wedge sampling solution proposed in [33] uniformly samples wedges and multiplies one third of the ratio of closed-wedges among the sampled wedges with the total number of wedges in the graph to approximate the total number of triangles in the graph. The solution proposed in [38] note that in graphs with power-law degree distribution, the triangles are not uniformly distributed across wedges and uniform sampling leads to bias necessitating more sampling. They suggest starting with uniform edge sampling and then converting the sampled edges to wedges via further sampling another edge from the connected edges of the first sampled edge.

In this work we also focus on wedge-sampling based triangle count approximation. Our contribution is in proving that for the problem of triangle counting the space of wedges can be significantly reduced (at least three times provably and one to two orders of magnitude in experimented real-world graphs). This reduction in turn leads to more accurate estimation with less sampling. Our space reduction approach has the additional benefit of eliminating the bias in wedges for power law-graphs. Once the space is reduced, we use the uniform wedge sampling approach described in [33] to estimate the triangle count.

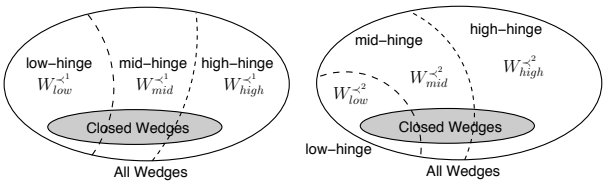
In addition to triangle counting, sampling techniques have been used for estimating counts of other important patterns in graphs such as 4-paths [15], diamonds [5], butterfly patterns in bipartite graphs [29], and cliques of any size [13]. Among these works, 4-path sampling work presented in [15] is of most interest to this work. It utilizes a similar approach to what we propose to first order vertices and use this ordering to reduce the set of 4-paths investigated as an implementation improvement. Unfortunately they only provide empirical analysis of this approach without providing detailed theoretical proofs of its search space reduction implications. In this work we provide a better ordering approach for better search space reduction than their approach, and provide theoretical proofs regarding the quality of the proposed approach. We believe this work sheds further light into why they observe significant improvements in their empirical tests when they use an ordering.

3 REVISITING WEDGE SAMPLING

In this section we first show that given a total ordering of vertices in a graph, it is possible to categorize the wedges in the graph into three groups such that sampling wedges only from one group is sufficient for accurate triangle count estimation. We then offer a simple greedy mechanism to obtain a total ordering such that the number of wedges in one of the categories is significantly smaller than the total number of wedges in the graph. We prove that via the proposed ordering, the number of wedges in the category we choose is at least three times less than the total number of wedges in the graph. We also prove that proposed ordering leads to at most two times more wedges in the target group compared to an optimal ordering for that group.

3.1 Categorizing Wedges and Using One Category for Triangle Sampling and Counting

Let $\prec$ define a total ordering on the vertices of a graph $G = (V, E)$. Using this total ordering, we can categorize wedges (two-paths) in G into three groups based on the hinge vertex of the wedge as defined below. We say that the *hinge vertex*, or *hinge* for short, of any given wedge w_{ijk} formed using the edges $\{v_i, v_j\}$ and $\{v_j, v_k\}$ is the vertex common to both edges, i.e., v_j. Then, we categorize the wedge w_{ijk} as a *high-hinge*, *mid-hinge*, or *low-hinge* wedge if the hinge vertex v_j is the highest, middle, or the lowest respectively



(a) Categorization due to $\prec^1$ **(b) Categorization due to $\prec^2$**

Figure 1: Orderings $\prec^1$ and $\prec^2$ divide the wedges into low-, mid-, and high-hinge categories. For both orderings, any category can be used for sampling triangles since the number of closed wedges in each category is equal to the number of triangles (closed wedges), i.e., $\overline{W}^{\prec^1}_{low}$=$\overline{W}^{\prec^1}_{mid}$=$\overline{W}^{\prec^1}_{high}$=$\overline{W}^{\prec^2}_{low}$=$\overline{W}^{\prec^2}_{mid}$=$\overline{W}^{\prec^2}_{high}$=$T$. Even though the number of triangles in each category is the same, the ratio of triangles to wedges ($\overline{W}/W$) is different for each category. Using the low-hinge category due to $\prec^2$ would be the best among these six options as that option has the highest triangle-to-wedge ratio.

when compared to the other two vertices of the wedge with respect to the total ordering $\prec$.

To estimate the triangle count of a graph G, it is sufficient to sample wedges only from any one of the wedge categories described above. For a given ordering $\prec$ of V, any triangle in G contains exactly one low-hinge, one mid-hinge, and one high-hinge wedge. To observe this, assume a triangle composed of the edges $\{v_i, v_j\}$, $\{v_i, v_k\}$, and $\{v_j, v_k\}$. Without loss of generality, let $v_i \prec v_j \prec v_k$. Clearly, the wedges w_{jik}, w_{ijk}, and w_{ikj} that have hinge vertices v_i, v_j, and v_k are low-hinge, mid-hinge, and high-hinge wedges respectively.

No matter which category of wedges one samples from, all triangles in G can be found during sampling. Thus, wedge-based sampling algorithms proposed in [33] for triangle count estimation can be applied using only one category of wedges for sampling. Intuitively, the smaller the number of wedges in the target category that will be used for sampling, the higher the improvement achieved over sampling wedges from all categories. This is due to the fact that the number of wedges in a category is equal to the number of triangles plus the number of so-called *open* wedges, which do not form triangles, in that category. This intuition is supported by our theoretical analysis as we show that the sampling error reduces as the ratio of open wedges reduces in a category.

As the number of wedges in each category depend on the ordering, we try to find an ordering that reduces the number of wedges in one category. In our approach, we focus on reducing the number of wedges in the low-hinge category. Note that we could have easily selected the high-hinge category as well since the number of low-hinge wedges in an ordering and the number of high-hinge wedges in the reverse ordering are the same.

For a given total order $\prec$, and any given vertex $v \in V$, let $d^{\prec}_{+}(v)$ be the number of vertices adjacent to v that are ordered after v, i.e., $d^{\prec}_{+}(v) = |\{u \in V \mid \{v,u\} \in E, v \prec u\}|$. Similarly, let $d^{\prec}_{-}(v)$ be the number of vertices adjacent to v that are ordered before v, i.e., $d^{\prec}_{-}(v) = |\{u \in V \mid \{v,u\} \in E, u \prec v\}|$. Clearly, the degree of v is $d(v) = d^{\prec}_{+}(v) + d^{\prec}_{-}(v)$. Given this notation, we define the number of low-hinge wedges in the graph G as $W^{\prec}_{low} = \sum_{v \in V} \binom{d^{\prec}_{+}(v)}{2}$. Similarly, we define the number of high-hinge wedges as $W^{\prec}_{high} = \sum_{v \in V} \binom{d^{\prec}_{-}(v)}{2}$, and the number of mid-hinge wedges as $W^{\prec}_{mid} = \sum_{v \in V} (d^{\prec}_{+}(v) \times d^{\prec}_{-}(v))$. Summing up, the total number of wedges is $W = \sum_{v \in V} \binom{d(v)}{2} = W^{\prec}_{low} + W^{\prec}_{mid} + W^{\prec}_{high}$. Also note that for any total ordering $\prec$, there is exactly one low-hinge, one mid-hinge, and one high-hinge wedge in any triangle. Therefore, the number of *closed* low-hinge wedges ($\overline{W}^{\prec}_{low}$), the number of *closed* mid-hinge wedges ($\overline{W}^{\prec}_{mid}$), and the number of *closed* high-hinge wedges ($\overline{W}^{\prec}_{high}$) are all equal to the number of triangles, i.e., $\overline{W}^{\prec}_{low} = \overline{W}^{\prec}_{mid} = \overline{W}^{\prec}_{high} = T$.

3.2 A Greedy Ordering Heuristic for Small Low-Hinge Wedge Count

As shown in Fig. 1 different orderings lead to different number of wedges in low-, mid-, high-hinge categories. For better estimation via smaller number of samplings, it is ideal to use an ordering that provides the smallest number of wedges in one category. However, ordering itself should not be more expensive than the sampling algorithm. In this section we describe a greedy ordering heuristic that provides a categorization where the number of wedges in the low-hinge category is at most one third of the total number of wedges. Furthermore, we prove that the number of low-hinge wedges produced by our greedy ordering is at most twice as much as the minimum number of low-hinge wedges any ordering can produce.

The intuition of our ordering mechanism stems from the observation that the earlier the order of a vertex, the higher the chances of that vertex to be the hinge of a low-hinge wedge, as a low-hinge wedge can only be formed with vertices ordered after that vertex. Thus, it stands to reason to place vertices that have low degrees early on in the ordering to reduce the number of wedges in the low-hinge category. Intuitively, ordering the vertices in G based on their vertex degrees reduces the number of wedges in the low-hinge category significantly. This is exemplified in Fig. 2, where the total number of low-hinge wedges are an order of magnitude (14× precisely) less than the total number of wedges.

In Fig. 2, we compare the cumulative distribution of wedges, closed wedges, low-hinge wedges, and closed low-hinge wedges for the real world Citeseer citation graph [23] in a log-log graph. Note that the cumulative closed low-hinge wedge figure (shown with lower triangles) saturates very fast and after the vertices of degrees up to 30 are considered, reaches very close to the total number of triangles T, which is its final value. This suggests that most triangles have a low degree vertex (e.g. a vertex with a degree smaller or equal to 30), which is a common pattern observed in real-world power-law graphs. Furthermore, the number of wedges (shown with capital W) starts to deviate from the number of low-hinge wedges significantly when we consider vertices of degree 20 and higher. This confirms our intuition explained in the above paragraph that placing vertices that have low degrees early on in the ordering reduces the number of low-hinge wedges. The number of wedges, as expected, increase quadratically as the degree increases.

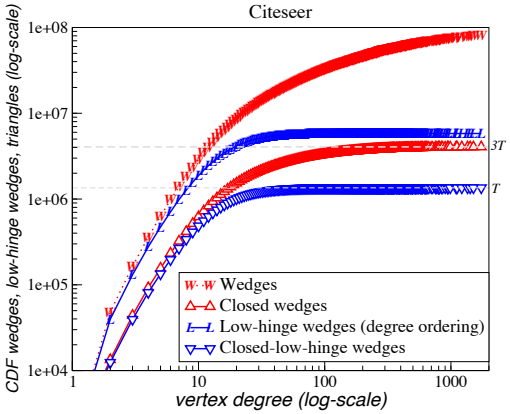


Figure 2: Cumulative distribution function for wedges, closed wedges, low-hinge wedges based on the degree ordering, and closed low-hinge wedges for the Citeseer dataset.

Note the mismatch between the wedge and triangle distributions. As the degree of vertices increase, the number of wedges increase much faster than the number of triangles. Thus a uniform sampling across wedges will be biased towards high-degree vertices. On the other hand, note the similarity between the distributions of the number of low-hinge wedges and the number of closed low-hinge wedges. A uniform low-hinge wedge sampling approach would not have the bias that the uniform wedge-sampling approach has.

The ordering based on the degrees alone can be further improved by taking into account the following observation: the only vertices that matter for a given vertex are those that appear after that given vertex. In this regard, if the vertices are ordered incrementally, placing a vertex into the total order increases the probability of its unordered neighbors becoming the hinge of low-hinge wedges, because their remaining degrees decreases by one. Taking advantage of this observation, we provide a *GreedyOrder* method to describe a total order (Algorithm 1). In *GreedyOrder*, we maintain a min heap priority queue over the degrees of vertices (lines 2-3). In a loop we remove the next smallest degree vertex from the queue (line 5) and reduce the degrees of the all the vertices incident to the removed vertex (lines 7-10). This loop continues till the queue gets empty. Vertices are ordered based on the removal order.

Complexity of *GreedyOrder* is $O(m \log n)$ as all edges are traversed only once by the algorithm and the complexity of the an update (*DecreaseKey*) is $O(\log n)$. Note that the complexity of the simple vertex degree ordering is $O(n \log n)$.

3.2.1 GreedyOrder Reduces Search Space Threefold. Here we prove that for a graph ordered by *GreedyOrder* the number of low-hinge wedges is at most one third of the total number of wedges.

Theorem 1. *Given a graph G and any ordering $\prec_{gr}$ obtained via GreedyOrder(G), the number of low-hinge wedges is at most one third of the total number of wedges, i.e., $W_{low}^{\prec_{gr}} \leq \frac{1}{3}W$.*

Proof. To count W, we will count the total number of wedges in which a given vertex v is the lowest order vertex, and then we will sum these counts for each vertex. Thus, every wedge will be counted exactly once and this sum will be equal to W. When counting, we will also rely on the following invariant that *GreedyOrder* guarantees: any vertex u appearing after another vertex v in the order $\prec_{gr}$ has a remaining degree of at least $d_+^{\prec_{gr}}(v)$ when v is removed from the priority queue. Note that, $d_+^{\prec_{gr}}(v)$ is the degree of v when v is removed from the priority queue in *GreedyOrder*(G).

Let us consider a vertex v when it is removed from the priority queue, i.e., when the order of v is determined. Also, let us consider another vertex u among the $d_+^{\prec_{gr}}(v)$ adjacent vertices of v still in the queue. Using the invariant above, the remaining degree of u is at least $d_+^{\prec_{gr}}(v)$ at the time v is removed from the priority queue. Thus, there are at least $d_+^{\prec_{gr}}(v) - 1$ wedges for which u is the hinge vertex and v is the lowest order vertex. Since there are $d_+^{\prec_{gr}}(v)$ vertices in the priority queue that are adjacent to v, the total number of mid-hinge and high-hinge wedges whose lowest order vertex is v is at least $d_+^{\prec_{gr}}(v) \times (d_+^{\prec_{gr}}(v) - 1)$. Also, recall that $W_{low}^{\prec_{gr}} = \sum_{v \in V} \binom{d_+^{\prec_{gr}}(v)}{2}$. Therefore, we have $W \geq \sum_{v \in V} \binom{d_+^{\prec_{gr}}(v)}{2} + d_+^{\prec_{gr}}(v)(d_+^{\prec_{gr}}(v)-1)$, which implies $W \geq 3W_{low}^{\prec_{gr}}$ proving the theorem. □

Algorithm 1 *GreedyOrder*$(G = (V, E))$

```
1:  L ← ∅                                   ▷ Empty list that will contain the ordering
2:  D ← ComputeVertexDegrees(V, E)          ▷ O(m)
3:  Q ← BuildMinHeap(D)
4:  while Q ≠ ∅ do                          ▷ O(m log n)
5:      x ← VertexID(Q.ExtractMin())
6:      L.Append(v_x)
7:      for all v_y ∈ Adjacency(v_x) do
8:          d_y ← d_y − 1                   ▷ reduce degree of v_x's neighbors
9:          Adjacency(v_y) ← Adjacency(v_y) − {v_x}
10:         Q.DecreaseKey(v_y, d_y)
11: return L                                ▷ return ordered vertex list
```

We note that a three-fold space reduction is very important as in normal wedge sampling the number of closed wedges is three times the number of triangles, i.e. 3T, whereas when sampling low-hinge wedges, the number of closed low-hinge wedges is equal to the number of traingles, i.e. T. If $W < 3W_{low}^{\prec_{gr}}$ was possible, then this would have meant that for some graphs it is more likely to find closed wedges when sampling from W compared to sampling from $W_{low}^{\prec_{gr}}$. With the above proof, we show that sampling from $W_{low}^{\prec_{gr}}$ is always as good as or better than sampling from W. We also note that for certain families of graphs, such as complete graphs, the above theorem is tight, i.e., $W_{low}^{\prec_{gr}} = \frac{1}{3}W$, the number of wedges in the low-hinge category is exactly 1/3 of the total number wedges in the graph.

3.2.2 GreedyOrder is 2−Approximate. Even though Theorem 1 proves that the improvement obtained using Alg. 1 to reduce $W_{low}^{\prec_{gr}}$ is at least threefold, as we will discuss in the evaluation of our algorithms, in practice we observe one to two order of magnitude reduction in the wedge space when we apply this algorithm on various graphs.

Regardless of the reduction ratio, we also prove that our algorithm produces a total ordering of the vertices that is provably very close to the best ordering that minimizes the number of low-hinge wedges. More precisely, we prove that if the best ordering $\prec_{opt}$ has $W_{low}^{\prec_{opt}}$ low-hinge wedges, then the low-hinge wedge count, $W_{low}^{\prec_{gr}}$, of the ordering $\prec_{gr}$ generated by Alg. 1 is at most twice that of the optimal ordering.

To prove our approximation bound we first show that any ordering produced by *GreedyOrder* will satisfy another important invariant. To define this invariant we first expand the ordering notation $\prec$ for a set of vertices as follows. We say that the vertex set X is ordered before vertex set Y if all vertices in X are ordered before all vertices in Y. That is, $X \prec Y$ if $\forall x \in X$ and $\forall y \in Y$, we have $x \prec y$. Using this definition, we show that there exists a paritition of the vertex set V independent of the ordering $\prec_{gr}$ that *GreedyOrder* generates and yet the sets of the partition are ordered with respect to $\prec_{gr}$:

Lemma 1. *Let $V_0, V_1, \ldots, V_k$ be the partition of the set of vertices V where V_i is the smallest subset that ensures that the minimum modified degree in the subgraph obtained by the removal of the vertices $\cup_{j=0}^{i} V_j$ is greater than i. Then, for any total ordering $\prec_{gr}$ generated by GreedyOrder, we have $V_0 \prec_{gr} V_1 \ldots \prec_{gr} V_k$.*

Proof. We prove the following statement on ℓ by induction: Letting $U_\ell = \cup_{j=0}^{\ell} V_j$, for any two vertices $v \in U_\ell$ and $v' \in V \setminus U_\ell$, we have $v \prec v'$ in the ordering provided by *GreedyOrder*. Proving this statement for all $\ell = 0, \ldots, k$ proves our lemma statement.

For the case of $\ell = 0$, the statement holds trivially since all vertices in V_0 have degree 0 and will be processed the earliest by *GreedyOrder* before any other vertex with positive degree.

Assuming the statement true for ℓ, we prove the statement for $\ell + 1$. Using the statement for ℓ, for any two vertices $v \in U_\ell$ and $v' \in V \setminus U_{\ell+1} \subset V \setminus U_\ell$, we have $v \prec v'$ in our ordering. Therefore, all we need to prove is that for any two vertices $v \in V_{\ell+1}$ and $v' \in V \setminus U_{\ell+1}$, we have $v \prec v'$ in our ordering. Assume towards a contradiction that we have $v' \prec v$ instead. Then, consider the earliest vertex $y \in V \setminus U_{\ell+1}$ with respect to the ordering $\prec$. From above, we have $y \prec v$. Furthermore, by definition of V_i, we know that $d_+^{\prec_{gr}}(y)$, the modified degree of y, must be at least $\ell + 1$ when y is ordered by *GreedyOrder*. However, this means that every vertex u satisfying $y \prec u$ must have a remaining degree of at least $\ell + 1$ at the time of ordering y. This contradicts the definition of V_ℓ in that it was the smallest such set. This is because there exists a proper subset of V_ℓ whose removal still guarantees that all other vertices have degree larger than ℓ. The contradiction proves the statement and therefore the lemma. □

Using the partitioning from the above lemma, we can now show that the number of low-hinge wedges of any total order $\prec_{gr}$ generated by *GreedyOrder* is at most twice that of any other ordering, and in particular at most twice that of the optimal ordering.

Theorem 2. *$W_{low}^{\prec_{gr}} \leq 2W_{low}^{\prec}$ for any total ordering $\prec_{gr}$ generated by GreedyOrder, and any total ordering $\prec$.*

Proof. Let $V_0, V_1, \ldots, V_k$ be the partitioning of the set of vertices V as defined in Lemma 1. Then, the statement of the lemma can be written as:

$$2W_{low}^{\prec} - W_{low}^{\prec_{gr}} = \sum_{i=0}^{k} \sum_{v \in V_i} 2\binom{d_+^{\prec}(v)}{2} - \binom{d_+^{\prec_{gr}}(v)}{2} \geq 0 \quad (1)$$

To prove this, for $i = 0, \ldots, k$, let $n_i = |V_i|$ and let $\mu_i(\prec)$ be the average low-degree of the vertices in V_i in the ordering $\prec$; i.e., $\mu_i(\prec) = \sum_{v \in V_i} d_+^{\prec}(v)/n_i$. Using the inequality between root-mean square and arithmetic mean, we get:

$$\begin{aligned} \sum_{v \in V_i} 2\binom{d_+^{\prec}(v)}{2} &= \sum_{v \in V_i} (d_+^{\prec}(v))^2 - \sum_{v \in V_i} d_+^{\prec}(v) \\ &\geq n_i \left(\sum_{v \in V_i} d_+^{\prec}(v)/n_i \right)^2 - n_i \mu_i(\prec) \\ \sum_{v \in V_i} 2\binom{d_+^{\prec}(v)}{2} &\geq n_i \mu_i(\prec)(\mu_i(\prec) - 1) \end{aligned} \quad (2)$$

Note that $\mu_i(\prec_{gr}) = \sum_{v \in V_i} d_+^{\prec_{gr}}(v)/n_i$. Also note that in (1), the term $\sum_{v \in V_i} \binom{d_+^{\prec_{gr}}(v)}{2}$ is maximized if we let $d_+^{\prec_{gr}}(v)$ to take the extreme values, the highest value can be i and for simplicity of the proof, we will assume the lowest value to be 1 (a longer proof follows if we allow 0 low-degree vertices). Since the average of $d_+^{\prec_{gr}}(v)$ is $\mu_i(\prec_{gr})$, we can assume a ratio of $i - \mu_i(\prec_{gr})$ vertices with $d_+^{\prec_{gr}}(v) = 1$ and $\mu_i(\prec_{gr}) - 1$ vertices with $d_+^{\prec_{gr}}(v) = i$ in the worst case. Then, we can state:

$$\begin{aligned} \sum_{v \in V_i} \binom{d_+^{\prec_{gr}}(v)}{2} &\leq \frac{n_i}{i-1}(i - \mu_i(\prec_{gr}))\binom{1}{2} \\ &\quad + \frac{n_i}{i-1}(\mu_i(\prec_{gr}) - 1)\binom{i}{2} \\ \sum_{v \in V_i} \binom{d_+^{\prec_{gr}}(v)}{2} &\leq n_i(\mu_i(\prec_{gr}) - 1)\frac{i}{2} \end{aligned} \quad (3)$$

Substituting in (1) the bounds we obtain in (2) and (3), we get $2W_{low}^{\prec} - W_{low}^{\prec_{gr}} \geq \sigma = \sum_{i=1}^{k} \sigma_i$, where σ_i is defined below, and we note that it suffices to show $\sigma \geq 0$ to prove the theorem:

$$\sigma_i = n_i \left(\mu_i(\prec)(\mu_i(\prec) - 1) - (\mu_i(\prec_{gr}) - 1)\frac{i}{2} \right)$$

Letting $\mu_i(\prec) = \mu_i(\prec_{gr}) + \phi_i/n_i$, we rewrite σ_i to obtain:

$$\sigma_i = \phi_i(\mu_i(\prec) + \mu_i(\prec_{gr}) - 1) + n_i(\mu_i(\prec_{gr}) - 1)\left(\mu_i(\prec_{gr}) - \frac{i}{2}\right) \quad (4)$$

To prove that $\sigma \geq 0$, we make the following observations:
(i) $\mu_i(\prec_{gr}) \geq \frac{i}{2}$,
(ii) for any $\ell = 1, \ldots, k$, $\sum_{i=1}^{\ell} \phi_i \leq 0$, and
(iii) $\sum_{i=1}^{k} \phi_i = 0$.

The first observation can be shown by the fact that every vertex $v \in V_i$ has degree at least i in the graph restricted to the vertices $\cup_{j=i}^{k} V_j$, which shows that there are at least $\frac{in_i}{2}$ edges incident to the vertices in V_i, and hence $\mu_i(\prec_{gr}) \geq \frac{i}{2}$. For the second observation, the definition of V_1 maximizes $\sum_{v \in V_1} d_+^{\prec_{gr}}(v)$, and in general, the definition of V_i maximizes $\sum_{v \in \cup_{j=1}^{i} V_j} d_+^{\prec_{gr}}(v)$. Therefore, for

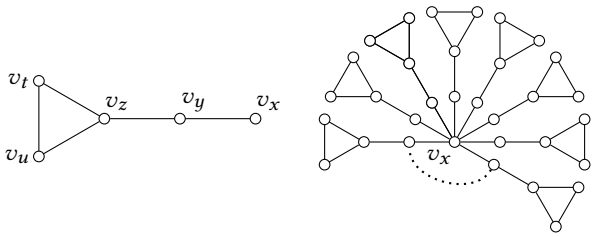


Figure 3: Illustration of proof of Lemma 2.

any ordering $\prec$, and for any $\ell = 1, \ldots, k$, we obtain $\sum_{i=1}^{\ell} \phi_i \leq 0$. Finally, the third observation is correct because each edge in the entire graph is counted exactly once, hence the sum of the modified degrees must be the same regardless of the ordering $\prec$.

First, we simplify the terms σ_j for those indices j with non-negative ϕ_j values. Using the first observation that $\mu_j(\prec_{gr}) \geq \frac{j}{2}$, the property that $\mu_j(\prec) \geq \mu_j(\prec_{gr})$ (directly implied by $\phi_j \geq 0$), and using (4), we can bound σ_j by:

$$\sigma_j \geq \phi_j(j-1)$$

Using the second and the third observations, we know that each reduction in an earlier vertex set V_i must be corrected as an increment in a later vertex set V_j for some $j > i$. Then, we prove $\sigma \geq 0$ by showing that each of the terms σ_i for those indices i with negative ϕ_i values satisfies $\sigma_i \geq i\phi_i$, or equivalently $\hat{\sigma}_i = \sigma_i - i\phi_i \geq 0$. Using the equation (4), we simplify $\hat{\sigma}_i$ to obtain the desired result:

$$\begin{aligned}\hat{\sigma}_i =& (\mu_i(\prec) - i + \mu_i(\prec_{gr}) - 1)\phi_i \\ &+ n_i(\mu_i(\prec_{gr}) - 1)\left(\mu_i(\prec_{gr}) - \frac{i}{2}\right) \\ =& -\frac{i}{2}\phi_i + n_i(\mu_i(\prec) - 1)\left(\mu_i(\prec) - \frac{i}{2}\right)\end{aligned}$$

If $\mu_i(\prec) \geq \frac{i}{2}$, then the above immediately implies $\hat{\sigma}_i \geq 0$ proving the theorem. Otherwise, we have $\mu_i(\prec) < \frac{i}{2}$. Furthermore, since $\mu_i(\prec_{gr}) \geq \frac{i}{2}$, we have $\mu_i(\prec) - \frac{i}{2} \geq \frac{\phi_i}{n_i}$. Still, we can show

$$\hat{\sigma}_i \geq \phi_i \left(\mu_i(\prec) - \frac{i}{2} - 1\right) \geq 0$$

This shows that $\hat{\sigma}_i \geq 0$ in either case, proving the theorem. □

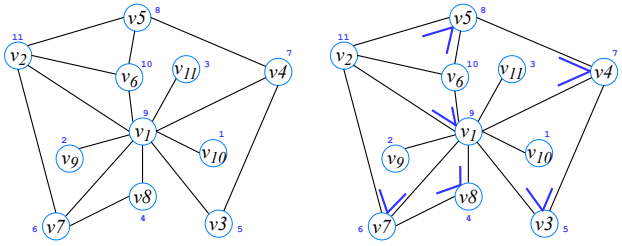
We also show that the 2-approximation of *GreedyOrder* is tight, that is, we can construct examples of graphs and orderings $\prec$ of its vertex set for which $W_{low}^{\prec_{gr}} / W_{low}^{\prec}$ gets arbitrarily close to 2.

Lemma 2. *For any given $\varepsilon > 0$, there exists a graph and an ordering $\prec$ of its vertices where $W_{low}^{\prec_{gr}} > (2-\varepsilon) W_{low}^{\prec}$.*

Proof. Consider a gadget of five vertices as shown in Figure 3. The gadget includes a triangle v_u, v_t, v_z and a two path v_z–v_y–v_x. Also, assume that the degree of v_x is high enough that our ordering algorithm chooses first the vertex v_y followed by v_z, v_t, v_u. For these four vertices our low wedge count is increased by two. However, the ordering $\prec$ where the vertices of this gadget are ordered v_u, v_t, v_z, and v_y would have resulted in a wedge count of one. To magnify the impact of such savings and prove Lemma 2, we consider the graph with at least $c > 1/\varepsilon$ copies of this gadget such that the vertex v_x is shared among the gadgets as shown in Figure 3. In this example graph, *GreedyOrder* can produce two low-hinge wedges for all gadgets but the last one, for which it will produce a single wedge. Therefore, we get $W_{low}^{\prec_{gr}} = 2c - 1$ in the worst case. Clearly the ordering $\prec$ would produce one wedge per gadget resulting with $W_{low}^{\prec} = c$, proving $W_{low}^{\prec_{gr}} > (2-\varepsilon) W_{low}^{\prec}$ as desired. □

Algorithm 2 *LowHingeWedgeSample*$(G = (V, E), \prec_{gr}, k)$

1: $S \leftarrow 0$
2: **for** $v \in V$ **do**
3: $\quad p_v(\prec_{gr}) \leftarrow \binom{d_+^{\prec_{gr}}(v)}{2} / W_{low}^{\prec_{gr}}$
4: **for** $i = 1, \ldots, k$ **do**
5: $\quad$ Select v_i from V w.r.t. probability distribution $\left\{p_{v_i}(\prec_{gr})\right\}$
6: $\quad H_{v_i} \leftarrow \{\, x \mid x \in adj(v_i) \wedge v_i \prec_{gr} x \,\}$
7: $\quad$ Select u, w uniformly at random from H_{v_i}
8: $\quad$ **if** $(u, w) \in E$ **then**
9: $\quad\quad S \leftarrow S + 1$
10: **return** $S/k \times W_{low}^{\prec_{gr}}$ ▷ return prediction



(a) Example graph with 11 vertices, 16 edges, 56 wedges, and 5 triangles. Vertex ordering (shown with blue indexes) is performed by *GreedyOrder*.

(b) Low-hinge wedges based on the ordering in (a) are marked with blue wedges. There are 6 low-hinge wedges 5 of which are closed.

Figure 4: Illustration of low-hinge wedges on an example graph.

3.3 Triangle Count Estimation via Sampling Low-Hinge Wedges

In this section we describe how we estimate the number of triangles in a graph by taking advantage of the properties of the ordering $\prec_{gr}$. Proposed approach is presented in Alg. 2. As seen in the algorithm, we first assign every vertex v a probability $p(v)$ equal to $\binom{d_+^{\prec_{gr}}(v)}{2} / W_{low}^{\prec_{gr}}$ (lines 2–3). Then, we sample k random vertices from the vertex set V with repetitions according to the assigned probability distribution (line 5). For each sampled vertex v_i, we sample a wedge out of the $\binom{d_+^{\prec_{gr}}(v_i)}{2}$ wedges of v_i (lines 6-7). If the sampled wedge is closed, i.e., the third edge is in the original graph, then we increment a sum S by one (lines 8-9). Finally, we output our triangle count prediction by multiplying S/k (the ratio of discovered closed wedges to the sampled wedges) with $W_{low}^{\prec_{gr}}$, the number of low-hinge wedges (line 10).

Figure 4(a) shows a sample graph with 11 vertices, 16 edges, 56 wedges, and 5 triangles[1]. A vertex ordering using a run of the *GreedyOrder* is indicated with a blue index number placed next to each vertex. Figure 4(b) indicates the low-hinge wedges of G based on the ordering in Figure 4(a). Out of the 6 low-hinge wedges, 5 of them are closed. The wedge space reduction is ~ 9× (from 56 to 6). As an example, if we run *LowHingeWedgeSample* on the sample graph with $k = 3$, we'll find 2 or 3 closed wedges, which will lead to a prediction of 2/3*6=4 or 3/3*6=6 triangles.

3.4 Expected Value, Variance, and RSE of Low-Hinge Wedge Sampling

In this section we analyze and provide closed form expressions for the expected value, the variance, and the relative standard error of our algorithm to be able to reason about its performance on graphs whose properties are known.

Considering every wedge selection step using $i = 1$ to k, and numbering every triangle in the graph using $j = 1$ to T, we introduce the following indicator variables for our analysis: let w_{ij} be the indicator variable whose value is 1 if and only if the low-hinge wedge for the j^{th} triangle is chosen at the i^{th} wedge selection step; and 0 otherwise[2]. Using these indicator variables, the output of our algorithm, S, can be expressed as:

$$S = \sum_{i=1}^{k} \sum_{j=1}^{T} w_{ij}.$$

We express the output of our algorithm in terms of the ratio of triangles T in the graph to the number of considerable wedges $W_{low}^{\prec gr}$, i.e., $R^{gr} = T/W_{low}^{\prec gr}$. Note that intuitively, as the ratio R^{gr} gets larger, the accuracy of our algorithm should increase and the number of samples we need to pick to achieve a given error rate in estimation should decrease. We can express $\mathbb{E}(S)$, the expected value of S, and $var(S)$, the variance of S, in terms of R^{gr} and in terms of k, the number of samples:

Lemma 3. *The expected value of S is kR^{gr}.*

Proof. We know that $\mathbb{E}(w_{ij}) = 1/W_{low}^{\prec gr}$, therefore, we get

$$\mathbb{E}(S) = \sum_{i=1}^{k} \sum_{j=1}^{T} \mathbb{E}(w_{ij}) = \sum_{i=1}^{k} \sum_{j=1}^{T} 1/W_{low}^{\prec gr} = kT/W_{low}^{\prec gr} = kR^{gr}$$

as desired, proving the lemma. □

Lemma 4. *The variance of S is $kR^{gr}(1 - R^{gr})$.*

Proof. For brevity, in the following analysis we will use the notation Λ to indicate $W_{low}^{\prec gr}$. We have $var(w_{ij}) = \frac{1}{\Lambda} - \frac{1}{\Lambda^2}$. Also, for any $i \neq i'$, we have $covar(w_{ij}, w_{i'j'}) = 0$ since the two selection processes at the two steps i and i' are independent of each other. Finally, for any $j \neq j'$, we have $covar(w_{ij}, w_{ij'}) = 0 - \frac{1}{\Lambda^2}$ since only one wedge is selected at each selection step. Therefore, we have:

$$\begin{aligned}
var(S) &= \sum_{i=1}^{k} \sum_{j=1}^{T} var(w_{ij}) + \sum_{i=1}^{k} \sum_{j \neq j'} covar(w_{ij}, w_{ij'}) \\
&= \sum_{i=1}^{k} \sum_{j=1}^{T} \left(\frac{1}{\Lambda} - \frac{1}{\Lambda^2}\right) + \sum_{i=1}^{k} \sum_{j \neq j'} \left(-\frac{1}{\Lambda^2}\right) \\
&= kT\frac{1}{\Lambda} - kT\frac{1}{\Lambda^2} - kT(T-1)\frac{1}{\Lambda^2} \\
&= k\frac{T}{\Lambda}\left(1 - \frac{1}{\Lambda} - \frac{T-1}{\Lambda}\right) \\
&= kR^{gr}\left(1 - \frac{T}{\Lambda}\right) \\
&= kR^{gr}\,(1 - R^{gr})
\end{aligned}$$

proving the variance stated in the lemma. □

We compare our method to previous work using the Relative Standard Error (RSE) values of all of the methods and base our arguments on the RSE values. Applying the formula for the relative standard error $RSE(S) = \sqrt{var(S)}/\mathbb{E}(S)$, we obtain:

$$RSE(S) = \frac{\sqrt{kR^{gr}(1 - R^{gr})}}{kR^{gr}} = \sqrt{\frac{1 - R^{gr}}{kR^{gr}}} \tag{5}$$

4 EVALUATION

In this section we provide an experimental analysis of the proposed methodology.

4.1 Experimental Setup

We compare the performance of our low-hinge-based wedge sampling algorithm (LWS) with the state-of-the-art wedge sampling [33] (WS) and edge-based wedge sampling [38] (EWS) approaches using large-scale graphs modeling real-world networks. As a comparison metric (and to verify our theoretical analysis presented in Section 3.4), we observe the relative standard error (RSE) values of different algorithms. We also compare these values with the theoretical analysis we provided in Section 3.4. In the figures, in addition to the RSE estimation we provide for LHS, we draw the estimations for WS and EWS using the formulations for the expected RSE values of these algorithms presented in [38]. It is important to note that the formulation for WS (shown below) has a striking similarity to the formulation in Eq. 5, allowing us to compare WS and LHS using the clustering coefficients (C) and the R values directly:

$$RSE(WS) = \sqrt{\frac{1 - C}{kC}} \tag{6}$$

In order to provide comparable results, as was done in [10, 38], we report the corresponding RSE values of the three algorithms for a given sampling probability. We vary the sampling probability until one of the algorithms achieves RSE ≤ 0.01. Considering the randomized nature of the sampling based approximation algorithms, we calculate the experimental RSE values observed with k different runs of the algorithms using the formula:

$$RSE = \frac{\sqrt{\frac{1}{k}\sum_{i=1}^{k}(\Delta_i - \mu)}}{T},$$

[1] We used the same sample graph presented in [38].

[2] We would like to thank the anonymous reviewer who pointed out the error in our initial indicator variable defition.

Table 1: Datasets used in experiments and their features.

Dataset	n	m	Description
Ego-Facebook [26]	4K	88K	Online social netw. Facebook
Enron-email [23]	36K	183K	Email comm. netw. in Enron
Brightkite [26]	58K	214K	Online social netw. Brightkite
Dblp-coauthor [26]	317K	1049K	Co-authorship netw. in DBLP
Amazon [26]	334K	925K	Co-purchasing from Amazon
Web-NotreDame [23]	325K	1090K	Web graph of Notre Dame
Citeseer [23]	384K	1736K	Citation in Citeseer
Dogster [23]	426K	8543K	OSN from dogster.com website
Web-Google [23]	875K	4322K	Web graph from Google
YouTube [26]	1134K	2987K	Online social netw. in Youtube
DBLP [23]	1314K	5362K	Co-authorship netw. in DBLP
As-skitter [26]	1696K	11095K	Internet connections Skitter
Flicker [23]	2302K	22838K	Online social netw. in Flicker
Orkut [23]	3072K	117185K	Online social netw. in Orkut
LiveJournal [26]	3997K	34681K	OSN in LiveJournal
Orkut2 [8]	11514K	327036K	Online social netw. in Orkut
Web-Arabic [8]	22743K	553903K	Web graph Arabian countries
MicrosoftAG [34]	46742K	528463K	Citation in Microsoft Academic
Twitter [23]	41652K	1202513K	Online social netw. in Twitter
Friendster [23]	65608K	1806067K	OSN from website Friendster

where Δ_i is the estimate obtained in the i-th run and μ is the mean of all of the k runs, i.e., $\mu = \frac{1}{k}\sum_{i=1}^{k}\Delta_i$. In all of our experiments, we use $k = 1000$.

We conducted the experiments on a server with 16 CPUs and 64 GB of RAM. The implementation was done on C++. We experimented over 20 real-world datasets with varying sizes that were made available in [10]. Largest of the datasets contain more than 40 million vertices and 30 billion triangles.

Features of the datasets (the number of vertices (n), edges (m), and description) used in the experiments are listed in Table 1. Other interesting features of the datasets such as the number of triangles (T), the number of wedges (W), and the clustering coefficients (C), as well as the number of low-hinge wedges ($W_{low}^{\prec_{gr}}$) and the R values for the greedy (R^{gr}) and degree ordering (R^{dg}) schemes, which can be used to estimate the RSE values using Eq. 5 are presented in Table 2.

4.2 Empirical Analysis

As seen in Table 2, the number of low-hinge wedges $W_{low}^{\prec_{gr}}$ are in general one to two orders of magnitude lower than the total number of wedges W for all graphs, indicating a significantly reduced sampling space for LHS. Also we can see in the table that for all datasets both R^{gr} and R^{dg} are significantly larger than C indicating that our approach has a much higher chance of hitting a triangle than in WS. Furthermore, for all datasets, R^{gr} is larger than R^{dg} showing that *GreedyOrder* reduces the sampling space more than a degree ordering solution would do. We observe from Equations (5) and (6) that the higher (and closer to 1) the R or C value is, the lower the RSE is expected to be. Given that R^{gr} is larger than C and R^{dg} for all datasets, LHS using the greedy ordering is expected to lead to lower RSE compared to other approaches for all datasets.

More strikingly, in Table 2 we observe that the value of R^{gr} is very close to one for many datasets (e.g. Web-NotreDame, Web-Arabic), which indicate that our approach shrank the sampling space so much that the number of low-hinge wedges is almost equal to the number of triangles and almost all samplings lead to closed wedges. Simply, by sampling very few low-hinge wedges (e.g., approximately 34 low-hinge wedges to get an RSE of 0.05 for Web-NotreDame or for Web-Arabic) and by mutliplying the number of closed wedge ratio with the number of low-hinge wedges, we can accurately estimate the total number of triangles (e.g., Web-NotreDame and Web-Arabic has more than 8 million and 36 billion triangles respectively). Note here that, irrespective of the utilized approach, to offer a correct sampling, the sampling space cannot be reduced to a size less than the total number of triangles, which shows how effective our reduction is.

Figure 5, provides an empirical comparison of the relative standard errors of WS, EWS and LHS over the 20 datasets listed in Table 1. The figure also depicts the RSE estimations indicated by [38] for WS and EWS, and by Equation 5 for LHS. In the figure the x-axis indicates different sampling ratios experimented with (in terms of number of sampled wedges) and the y-axis indicates the corresponding RSE values obtained by the three algorithms. Both the x- and y-axis is presented in log-scale and the x-axis range is selected such that the RSE values for LHS will range between 0.20 and 0.01.

First we note that for all datasets depicted in Figure 5, the empirical RSE observations we make for LHS perfectly match the theoretical bounds we compute using Equation 5. This proves the validity of our theoretical analysis. We also observe that, as expected, the datapoints in Figure 5 for WS, EWS and LHS all exhibit parallel trends in the log-log graph irrespective of the sampling ratio. In other words, all lines in the figure have the same slope with different constants defining their difference. This is expected as WS and LHS practically run the same algorithm albeit on different sampling spaces and all of WS, EWS, and LHS depend linearly on the sampling ratio, therefore, the ratio of their RSE values remain the same across different sampling probabilities.

In all of the datasets LHS performs significantly better than the other algorithms and offers highly confidant estimations with very small number of samplings. We note that since Figure 5 is a log-log figure, the constant difference between the lines in fact indicate a multiplicative difference. i.e. for all datasets, LHS offers two to eight times lower RSE at the same sampling ratio and this multiplicative distance remains fixed per dataset at all sampling ratios.

Note that the parallel nature of the figures will continue even if more samplings are done and the RSE gets lower. For any sampling ratio LHS will provide more accurate results, and for any desired RSE value LHS will be able to achieve that value with lower number of samplings. For example, in the As-skitter dataset, to achieve an RSE of around 0.11, LHS needs to sample just 165 wedges whereas EWS needs to sample 9476 wedges (a 57× difference). This difference is more profound in Web-Arabic where LHS needs 113× less sampling. LHS offers the highest RSE improvement ($\sim$ 8×) at the same sampling ratio over the best of WS and EWS at the Web-Arabic and Web-NotreDame datasets. This is expected as on these datasets, LHS achieves close to one R^{gr}, which indicate that LHS performs close to optimal wedge-space reduction.

5 CONCLUSION

We proposed a methodology to significantly improve the performance of wedge-sampling based triangle counting approaches by reducing the size of sampling space. We achieved this by ordering

Table 2: Number of low-hinge wedges and R values with greedy and degree orderings.

Dataset	T	W	$W_{low}^{\prec gr}$	$W/W_{low}^{\prec gr}$	$C=\frac{3T}{W}$	$R^{gr}=T/W_{low}^{\prec gr}$	$R^{dg}=T/W_{low}^{\prec dg}$
Ego-Facebook [26]	1.6M	9.3M	1.9M	4.9	0.5192	0.8552	0.8385
Enron-email [23]	0.7M	25.6M	1.4M	18.3	0.0853	0.5264	0.4982
Brightkite [26]	0.5M	13.4M	0.9M	14.9	0.1106	0.5195	0.4862
Dblp-coauthor [26]	2.2M	21.8M	2.7M	8.1	0.3064	0.8145	0.7830
Amazon [26]	0.7M	9.8M	1.0M	9.8	0.2052	0.6414	0.5859
Web-NotreDame [23]	8.9M	304.9M	9.7M	31.4	0.0877	0.9220	0.9114
Citeseer [23]	1.4M	81.7M	5.1M	16.0	0.0496	0.2665	0.2342
Dogster [23]	83.5M	17,554.4M	266.4M	65.9	0.0143	0.3134	0.2923
Web-Google [23]	13.4M	727.4M	15.9M	45.7	0.0552	0.8427	0.8085
YouTube [26]	3.1M	1,474.5M	13.9M	106.1	0.0062	0.2194	0.1947
DBLP [23]	12.2M	214.7M	18.7M	11.5	0.1703	0.6503	0.6231
As-skitter [26]	28.8M	16,021.7M	87.1M	184.0	0.0054	0.3302	0.3044
Flicker [23]	837.6M	23,342.9M	2,030.7M	11.5	0.1076	0.4125	0.3978
Orkut [23]	627.6M	45,625.5M	3,787.8M	12.0	0.0413	0.1657	0.1553
LiveJournal [26]	177.8M	4,255.8M	422.6M	10.1	0.1253	0.4208	0.3954
Orkut2 [8]	223.1M	2,543,767.1M	35,698.9M	71.3	0.0003	0.0062	0.0054
Web-Arabic [8]	36,895.4M	3,531,929.7M	40,075.6M	88.1	0.0313	0.9206	0.9135
MicrosoftAG [34]	578.2M	115,067.1M	5,724.7M	20.1	0.0151	0.1010	0.0919
Twitter [23]	34,824.9M	123,435,589.6M	136,394.5M	905.0	0.0008	0.2553	0.2314
Friendster [23]	4,173.7M	720,655.0M	76,910.8M	9.4	0.0174	0.0543	0.0507

the vertices of the graph, using this ordering to categorize wedges, only using low-hinge wedges for sampling. If degrees of vertices are available, we showed that a simple degree ordering can significantly reduce the size of sampling space. We proved that the reduction achieved by a slightly more complex ordering mechanism is at least three-fold in theory, is a 2-approximation to the best achievable reduction for low-hinge wedges, and provides one to two orders of reduction in practice. Through rigorous analysis over large-scale real-world graphs, we show that our approach offers highly confident estimations always outperforming the state-of-the-art approaches two to eight times in terms of accuracy or up to 100× in terms of sampling size for the same level of accuracy.

REFERENCES

[1] Nesreen K Ahmed, Nick Duffield, Theodore L Willke, and Ryan A Rossi. 2017. On sampling from massive graph streams. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1430–1441.

[2] Yazan Alaya AL-Khassawneh, Naomie Salim, and Obasa Adekunle Isiaka. 2014. Extractive Text Summarisation using Graph Triangle Counting Approach: Proposed Method. In *1 st International Conference of Recent Trends in Information and Communication Technologies in Universiti Teknologi Malaysia, Johor, Malaysia*. 300–311.

[3] N. Alon, R. Yuster, and U. Zwick. 1997. Finding and counting given length cycles. *Algorithmica* 17, 3 (01 Mar 1997), 209–223.

[4] A. Azad, A. BuluÃğ, and J. Gilbert. 2015. Parallel Triangle Counting and Enumeration Using Matrix Algebra. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. 804–811.

[5] Grey Ballard, Tamara G Kolda, Ali Pinar, and C Seshadhri. 2015. Diamond sampling for approximate maximum all-pairs dot-product (MAD) search. In *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 11–20.

[6] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. 2002. Reductions in Streaming Algorithms, with an Application to Counting Triangles in Graphs. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '02)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 623–632. http://dl.acm.org/citation.cfm?id=545381.545464

[7] Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. 2008. Efficient Semi-streaming Algorithms for Local Triangle Counting in Massive Graphs. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*. ACM, New York, NY, USA, 16–24. https://doi.org/10.1145/1401890.1401898

[8] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. 2011. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th international conference on World wide web*. ACM, 587–596.

[9] Luciana S. Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. 2006. Counting Triangles in Data Streams. In *Proceedings of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '06)*. ACM, New York, NY, USA, 253–262. https://doi.org/10.1145/1142351.1142388

[10] Roohollah Etemadi, Jianguo Lu, and Yung H. Tsin. 2016. Efficient Estimation of Triangles in Very Large Graphs. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. ACM, New York, NY, USA, 1251–1260. https://doi.org/10.1145/2983323.2983849

[11] Oded Green, Pavan Yalamanchili, and Lluís-Miquel Munguía. 2014. Fast Triangle Counting on the GPU. In *Proceedings of the 4th Workshop on Irregular Applications: Architectures and Algorithms (IA3 '14)*. 1–8.

[12] Xiaocheng Hu, Yufei Tao, and Chin-Wan Chung. 2014. I/O-Efficient Algorithms on Triangle Listing and Counting. *ACM Trans. Database Syst.* 39, 4 (2014), 27:1–27:30.

[13] Shweta Jain and C Seshadhri. 2017. A Fast and Provable Method for Estimating Clique Counts Using Turán's Theorem. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 441–449.

[14] Madhav Jha, C. Seshadhri, and Ali Pinar. 2013. A Space Efficient Streaming Algorithm for Triangle Counting Using the Birthday Paradox. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. ACM, New York, NY, USA, 589–597. https://doi.org/10.1145/2487575.2487678

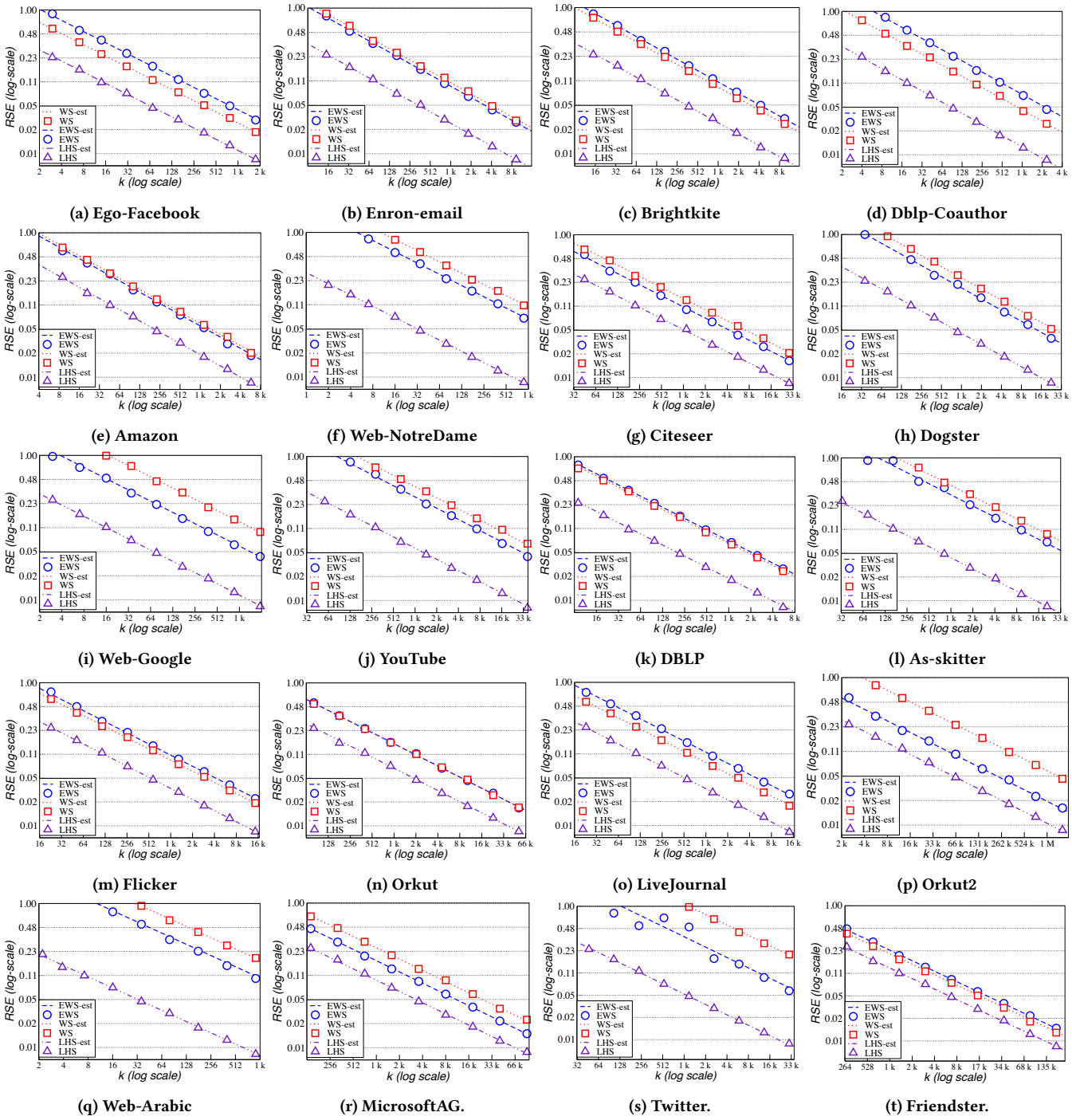
[15] Madhav Jha, C Seshadhri, and Ali Pinar. 2015. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 495–505.

[16] Hossein Jowhari and Mohammad Ghodsi. 2005. New Streaming Algorithms for Counting Triangles in Graphs. In *Proceedings of the 11th Annual International Conference on Computing and Combinatorics (COCOON'05)*. Springer-Verlag, Berlin, Heidelberg, 710–716. https://doi.org/10.1007/11533719_72

[17] John Kallaugher and Eric Price. 2017. A hybrid sampling scheme for triangle counting. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 1778–1797.

[18] Gabriela Kalna and Desmond J. Higham. 2007. A Clustering Coefficient for Weighted Networks, with Application to Gene Expression Data. *AI Commun.* 20, 4 (Dec. 2007), 263–271. http://dl.acm.org/citation.cfm?id=1365534.1365536

[19] Hyeonji Kim, Juneyoung Lee, Sourav S. Bhowmick, Wook-Shin Han, JeongHoon Lee, Seongyun Ko, and Moath H.A. Jarrah. 2016. DUALSIM: Parallel Subgraph Enumeration in a Massive Graph on a Single Machine. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD'16)*. 1231–1245.



(a) Ego-Facebook (b) Enron-email (c) Brightkite (d) Dblp-Coauthor

(e) Amazon (f) Web-NotreDame (g) Citeseer (h) Dogster

(i) Web-Google (j) YouTube (k) DBLP (l) As-skitter

(m) Flicker (n) Orkut (o) LiveJournal (p) Orkut2

(q) Web-Arabic (r) MicrosoftAG. (s) Twitter. (t) Friendster.

Figure 5: *RSE* vs k for the datasets.

[20] T. Kloks, D. Kratsch, and H. Müller. 2000. Finding and counting small induced subgraphs efficiently. *Inform. Process. Lett.* 74, 3 (2000), 115 – 121.

[21] Tamara G Kolda, Ali Pinar, Todd Plantenga, C Seshadhri, and Christine Task. 2014. Counting triangles in massive graphs with MapReduce. *SIAM Journal on Scientific Computing* 36, 5 (2014), S48–S77.

[22] Mihail N. Kolountzakis, Gary L. Miller, Richard Peng, and Charalampos E. Tsourakakis. 2010. *Efficient Triangle Counting in Large Graphs via Degree-Based Vertex Partitioning.* Springer Berlin Heidelberg, Berlin, Heidelberg, 15–24. https://doi.org/10.1007/978-3-642-18009-5_3

[23] Jérôme Kunegis. 2013. Konect: the koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web.* ACM, 1343–1350.

[24] Matthieu Latapy. 2008. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theoretical Computer Science* 407, 1-3 (2008), 458–473.

[25] François Le Gall. 2014. Powers of Tensors and Fast Matrix Multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC '14)*. ACM, New York, NY, USA, 296–303. https://doi.org/10.1145/2608628.2608664

[26] Jure Leskovec and Andrej Krevl. 2015. SNAP Datasets: Stanford Large Network Dataset Collection.

[27] Yongsub Lim and U Kang. 2015. Mascot: Memory-efficient and accurate sampling for counting local triangles in graph streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 685–694.

[28] Rasmus Pagh and Charalampos E. Tsourakakis. 2012. Colorful triangle counting and a MapReduce implementation. *Inform. Process. Lett.* 112, 7 (2012), 277 – 281. https://doi.org/10.1016/j.ipl.2011.12.007

[29] Seyed-Vahid Sanei-Mehri, Ahmet Erdem Sariyuce, and Srikanta Tirthapura. 2018. Butterfly Counting in Bipartite Networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2150–2159.

[30] Thomas Schank and Dorothea Wagner. 2005. Finding, Counting and Listing All Triangles in Large Graphs, an Experimental Study. In *Proceedings of the 4th International Conference on Experimental and Efficient Algorithms (WEA'05)*. 606–609.

[31] Thomas Schank and Dorothea Wagner. 2005. Finding, Counting and Listing All Triangles in Large Graphs, an Experimental Study. In *Experimental and Efficient Algorithms: 4th International Workshop, WEA*, Sotiris E. Nikoletseas (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 606–609.

[32] Comandur Seshadhri, Ali Pinar, and Tamara G Kolda. 2013. Triadic measures on graphs: The power of wedge sampling. In *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 10–18.

[33] C. Seshadhri, Ali Pinar, and Tamara G. Kolda. 2014. Wedge sampling for computing clustering coefficients and triangle counts on large graphsâĂă. *Statistical Analysis and Data Mining* 7, 4 (2014), 294–307. https://doi.org/10.1002/sam.11224

[34] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15 Companion)*. 243–246.

[35] Lorenzo De Stefani, Alessandro Epasto, Matteo Riondato, and Eli Upfal. 2017. Triest: Counting local and global triangles in fully dynamic streams with fixed memory size. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 11, 4 (2017), 43.

[36] Siddharth Suri and Sergei Vassilvitskii. 2011. Counting Triangles and the Curse of the Last Reducer. In *Proceedings of the 20th International Conference on World Wide Web (WWW '11)*. ACM, New York, NY, USA, 607–614. https://doi.org/10.1145/1963405.1963491

[37] Charalampos E. Tsourakakis, U. Kang, Gary L. Miller, and Christos Faloutsos. 2009. DOULION: Counting Triangles in Massive Graphs with a Coin. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*. ACM, New York, NY, USA, 837–846. https://doi.org/10.1145/1557019.1557111

[38] Duru Türkoglu and Ata Turk. 2017. Edge-Based Wedge Sampling to Estimate Triangle Counts in Very Large Graphs. In *2017 IEEE International Conference on Data Mining, ICDM 2017*. 455–464.

[39] Leyuan Wang, Yangzihao Wang, Carl Yang, and John D. Owens. 2016. A Comparative Study on Exact Triangle Counting Algorithms on the GPU. In *Proceedings of the ACM Workshop on High Performance Graph Processing (HPGP '16)*.

[40] Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Y Zhao, and Yafei Dai. 2014. Uncovering social network sybils in the wild. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 8, 1 (2014), 2.