Z-Shadow: An Efficient Method for Estimating Bicliques in Massive Graphs Using Füredi's Theorem

Bole Chang
Fujian Agriculture And Forestry University
Fuzhou, China
1221193017@fafu.edu.cn

Linxin Xie
Fuzhou University
Fuzhou, China
225420010@fzu.edu.cn

Wei Li
Fujian Agriculture And Forestry University
Fuzhou, China
liwei@fafu.edu.cn

Meng Qin
Department of CSE, HKUST
Hong Kong SAR
mengqin_az@foxmail.com

Jianfeng Hou
Fuzhou University
Fuzhou, China
jfhou@fzu.edu.cn

ABSTRACT

In real-world scenarios, bipartite graphs are commonly used to reveal relationships between objects. In this sense, biclique counting has become a fundamental issue in bipartite network analysis and has drawn a lot of attention recently. However, a fundamental barrier is the exponential blow-up in the search space of large bicliques and the exponential growth of the number of bicliques in large dense graphs. In this paper, we design a **Z-Shadow** randomized algorithm for large biclique estimation, based on Füredi's Theorem and a more reasonable and provable sampling threshold. We also present an unbiased online sampling method to take full advantage of shadows and minimize memory storage.

The numerical experiments are conducted on both real databases and artificial networks. The results show that our method achieves less than 0.5% error and up to 100X and 500x speedup compared to the state-of-the-art estimate and exact algorithm, respectively, which indicates that **Z-Shadow** is suitable for large-scale bipartite graphs, and it is memory-saving, efficient, and accurate.

1 INTRODUCTION

In real-world scenarios, bipartite graphs are frequently used to model relationships between distinct groups of objects, such as actor-movie networks, author-paper collaborations, and consumer-product connections [19]. The widespread applicability of bipartite graphs underscores their critical role in network analysis.

A (s, t)-biclique, denoted as $K_{s,t}$, is a complete bipartite subgraph comprising two sets of disjoint vertices with s vertices in one set and t vertices in the other, where every vertex in one set is connected to every vertex in the other. In Figure 1, for instance, the subgraph induced by the vertex sets $\{u_2, u_3, u_5, v_2, v_3, v_4\}$ forms a $K_{3,3}$. In this paper, we focus on the fundamental problem of estimating the number of bicliques in network analysis.

1.1 Motivation

In real-world graphs, here are many networks such as user-content interaction networks, protein-compound networks, and User-Product networks, which should be modeled as large and dense bipartite graphs. On the other hand, large and dense networks mean massive search space, and existing methods are often limited by rapid growth in computational complexity, which makes lots of basic graph algorithms inefficient and unsuitable for massive real-world graphs.

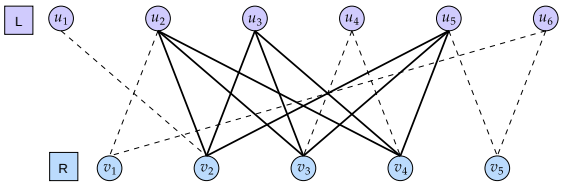


Figure 1: $K_{3,3}$ in a bipartite graph $G = (L, R, E)$

Densely connected subgraph searching over bipartite graphs, such as (α, β)-core [7, 26, 41, 42], bitruss [32, 38, 45], and biclique [9, 22, 25, 48, 51], has been proved useful in many real-world applications ranging from graph neural networks (GNNs), cohesive subgraph analysis, and so on [48]. Naturally, as a fundamental community structure in bipartite networks, biclique counting becomes a fundamental issue in bipartite network analysis and has drawn a lot of attention recently.

(1) *blueCohesive Subgroup Analysis.* Identifying cohesive subgroups or communities within a network is essential to understand its structure and function. Borgatti et al. [5] used bicliques to identify cohesive subgroups and revealed the embedded structure in a bipartite graph.
(2) *Higher-order Clustering Coefficient.* Traditional clustering coefficients measure the density of triangles in a network but it fails to capture higher-order structures. Benson et al. [2] extended the concept of clustering coefficients to higher-order structures and demonstrated that these structures provide a more comprehensive understanding of network topology. The study in [52] shows that networks in the same field usually have similar higher-order clustering coefficients. In a bipartite graph, the higher-order clustering coefficient is defined as the ratio between the counts of bicliques and wedges, where wedges is a special structure close to bicliques. This ratio measures the probability of a wedge becoming a biclique, characterizing the intrinsic properties of a bipartite graph.

(3) *Network Embedding Learning.* Network embedding aims to map nodes into a low-dimensional vector space, preserving structural information. Similarly, the bipartite network embedding (BNE) [49] maps each node to compact embedding vectors that capture hidden topological features surrounding the nodes, to facilitate downstream tasks. Wang et al. [36] pushed forward the progress of Structural Deep Network Embedding (SDNE) by catching the local and global structural properties based on biclique counting, which provides richer structural information.

The approaches for the biclique counting problem include exact counting and approximate estimation. Exact counting methods, such as BCList (BCL for short) and BCList++ (BCL++ for short) proposed by Yang et al.[48], as well as Epivot (EP for short) introduced by Ye et al [51], commonly based on backtracking techniques, which involve recursive searches to complete partial bicliques and performs well on sparse bipartite graphs. However, due to the high connectivity of dense bipartite graphs, the search space exhibits exponential growth, Furthermore, in large and dense graphs, the number of bicliques also grows exponentially. Attempting to enumerate all bicliques requires an impractically large amount of computational time. Therefore, rendering exhaustive enumeration is infeasible for the large and dense graphs. Using an approximate algorithm is an efficient approach to address this problem. Currently, the state-of-the-art approximate algorithm for biclique estimation is Zigzag++ (Zz++ for short), proposed by Ye et al [51]. The key technique in Zigzag++ involves mapping bicliques to paths based on their structure properties and estimating the number of bicliques through path sampling. Although Zigzag++ has an excellent performance on sparse graphs, it necessitates recording all paths, which will be exceedingly large in dense graphs. Thus, Zigzag++ also faces the problem of a combinatorial explosion in the search space, which is the problem we study in this paper.

1.2 Contributions

In this paper, we give an efficient approximate method called **Z-Shadow** which maps target bicliques to smaller ones and reduces the task of counting bicliques in large graphs to smaller subgraphs. Moreover, the accuracy is improved, by decomposing the graph into smaller components and pruning irrelevant sections. Compared to traditional methods that struggle with the exponential growth of the search space in massive bipartite graphs, our approach incorporates the Füredi Theorem to terminate iterations for sufficiently dense subgraphs as soon as possible. This integration not only mitigates the problem of combinatorial explosion but also establishes a solid theoretical basis for ensuring the accuracy of our approximate algorithms. By dynamically adjusting the termination criteria based on graph density, our method achieves significant reductions in computational complexity. Moreover, compared to existing algorithms, such as EPivot and Zigzag++, our approach is more feasible for large scalability and more efficient.

Our main contributions are as follows:

(1) *Integrating Extremal Graph Theory into Biclique Estimation.* Motivated by the k-clique counting randomized algorithm proposed in [15], our algorithm **Z-Shadow(Gloabl)** utilizes results for the Zarankiewicz problem in extremal graph theory. Seminal results by Füredi [13] provided an upper bound for the Zarankiewicz problem, which serves as the theoretical basis to end the space search early. The effect of this theorem will be shown in numeric experiments in Section 6.8.

(2) *Efficient and Memory-Saving Sampling.* To count k-cliques, the shadow construction algorithm presented in [15] has a sufficient theoretical basis to construct the sample sets. However, it requires significant memory to store shadows for a sparse bipartite graph. To overcome this obstacle, Jain and Seshadhri proposed an online algorithm [17] to address this issue. However, it is not the most space-efficient. In order to substantially reduce memory usage, we modified the online algorithm [17] with an unbiased sampling method. Even more, we release the space storing shadows as soon as they are fully utilized. The improved algorithm **Z-Shadow(Online)** can reduce memory usage by up to 1500 times compared to **Z-Shadow(Global)**.

(3) *Efficient for Massive Graphs.* For the two approximation algorithms that perform best on dense graphs, Zz++ and EP/Zz++, **Z-S(AD)** achieves a speedup of over 100 times in the best case, with an average speedup exceeding 37 times. For the most efficient exact algorithm, BCL++, **Z-S(AD)** delivers a maximum speedup of up to 500 times and an average speedup of 130 times. For example, it takes only twenty minutes for $(4, 4)$-biclique estimation on a dataset with 151 million edges, whereas Zz++ [51], BCL++ [48] and other comparison algorithms take more than 24 hours. Moreover, the variations of graph parameters have less impact on the speed of our algorithm.

(4) *Outstanding Accuracy.* Our algorithm achieves accurate results with low variance. The errors of our algorithm are consistently below 0.5% and 1% on dense graphs and sparse graphs, respectively, where BCL++ is used to compute the exact results for $K_{4,4}$.

In summary, our algorithm is efficient, accurate, and memory-efficient, drawing on theoretical results from graph theory.

This paper is organized as follows. We list some related results in the following section and present definitions and notation in graph theory in Section 3. In Section 4, we propose the **Z-Shadow** algorithm with a theoretical analysis. In Section 5, we improve our algorithm by reordering graph vertices and modifying the online method to address memory usage issues. Section 6 gives several numerical experiments and Section 7 summarizes our work.

2 RELATED WORK

Subgraph enumeration is known to be a challenging task in graph analysis. The literature numerical algorithms are proposed to count the number of the subgraphs with a small order, such as triangle [57], butterfly, diamonds, etc [31]. The k-clique problem has received significant attention [6, 16, 50], due to its importance in various network-based applications, such as community detection [27, 54], graph partitioning [14, 55], network embedding [29, 53], and recommendation systems [28, 35]. Exact k-clique counting algorithms, such as those proposed in [8, 16, 24, 34, 40], generally rely on exhaustive search tree traversals for each vertex to identify k-cliques. In contrast, approximation methods for k-clique estimation, particularly sampling techniques [6, 18, 33, 44], aim to reduce computational costs by trading off some precision.

It is worth mentioning that Jain and Seshadhri developed a fast approximation algorithm for k-clique counting using extremal graph theory [15], which is currently one of the fastest available methods for this task. They further extended this approach to

near-clique estimation in [17], by proposing an online sampling algorithm to handle large graphs more efficiently.

While cliques are fully connected subgraphs in general graphs, the analogous structure in bipartite graphs is known as a biclique. Butterflies, the simplest biclique, have been widely studied as a fundamental motif in bipartite graphs. Sanei-Mehri et al. proposed a fast butterfly counting algorithm in 2018 [30], sparking significant interest in accelerated and parallel approaches for butterfly counting across both CPU and GPU architectures [23, 39, 43, 47]. Additionally, this problem has been extended to (α, β)-core queries [7, 26] and fraud detection [37, 56].

Yang et al. [48] introduced the biclique counting problem and developed the BCList algorithm, which relies on depth-first exploration through multi-layer iteration. The core of this algorithm involves maintaining a partial biclique and iteratively expanding it by adding vertices from a candidate set. An improved version, BCList++, was also proposed in [48] to reduce the consumption of finding the candidate set by constructing a 2-hop graph. However, due to the high connectivity of dense bipartite graphs, the search space exhibits exponential growth. To reduce the search space, Ye et al. proposed a pivot-based method in [51], called EPivot. The search space in the algorithm is determined by the partially ordered neighbors of each edge. Notice that the key point is to select appropriate pivots during the process of graph traversal, which becomes significantly more challenging.

Exhaustive enumeration for all biclique is infeasible for the large and dense bipartite graphs. A feasible solution is to estimate the total number. A typical example, given by Ye et al.[51], is an approximate algorithm called Zigzag, which leverages the structural properties of bicliques and maps each biclique to a path with specified lengths. The enhanced version, Zigzag++, introduces a more efficient sampling space construction. It is noteworthy that Zigzag and Zigzag++ can estimate all (s, t)-bicliques in a graph where $s, t < 10$ and represent a remarkable achievement. However, when it comes to dealing with dense graphs, they also face significant computational challenges in collecting all paths of the specified length making the task highly resource-intensive.

3 PRELIMINARIES

This section outlines some concepts and formal definitions. A graph $G = (L, R, E)$ is called *bipartite* if its vertices set can be divided into two disjoint sets L and R such that every edge in the edge set E has one end in L and the other in R. Let $e(G) = |E|$ be the number of edges in a graph G. For a vertex $v \in L \cup R$, let $N_G(v)$ be the *neighborhood* of v in G and $d_G(v) = |N_G(v)|$ be the *degree* of v in G. *The set of 2-hop vertices of* v in G, denoted by $N_G^{2+}(v)$, is the collection of vertices of distance 2 with v. A graph $H = (L', R', E')$ is called a *subgraph* of $G = (L, R, E)$, denoted by $H \subseteq G$, if $L' \subseteq L$, $R' \subseteq R$ and $E' \subseteq E$.

Definition 3.1. For positive integers s, t, a (s, t)*-biclique* $K_{s,t}$ is a bipartite graph $G = (L, R, E)$ with $|L| = s$, $|R| = t$ and E consisting of all edges with one end in L and the other in R.

For a bipartite graph $G = (L, R, E)$ and $S \subseteq L \cup R$, $G[S]$ is the subgraph of G induced by S, whose vertex set is S and the edge set consists of all edges of G with two ends in S. Let $\mathcal{N}(K_{s,t}, S)$ denote the number of $K_{s,t}$ in $G[S]$ with s vertices in L and t vertices in R. We use $\mathcal{N}(K_{s,t}, G)$ to denote $\mathcal{N}(K_{s,t}, L \cup R)$ for convenience. The problem of estimating bicliques in bipartite graphs can be formally defined as follows: Given a bipartite graph $G = (L, R, E)$ and parameters s, t, estimate the number of (s, t)-bicliques with s vertices in L and the t vertices in R.

4 PROPOSED ALGORITHM: GLOBAL SAMPLING FROM A SHADOW

Finding special substructures in discrete structures is a classical topic in extremal graph theory, which was initiated as a separate subarea of combinatorics by Turán in 1941. Given a forbidden subgraph H, the classical Turán-type question is to determine the maximum number of edges in an n-vertex graph that does not contain H as a subgraph, which is denoted by $\mathrm{ex}(n, H)$. The Erdős-Stone Theorem gives an asymptotic formula for $\mathrm{ex}(n, H)$ when the chromatic number of H is at least 3. On the other hand, the Kővári-Sós-Turán Theorem implies that for any bipartite graph H, there is a positive constant δ such that $\mathrm{ex}(n, H) = O(n^{2-\delta})$.

Turán type problems forbidding bipartite graphs are often called *degenerate*. Determining the growth rate of $\mathrm{ex}(n, H)$ for a bipartite graph H is a central and notoriously difficult topic in Extremal Combinatorics, and it remains open for most families. For example, the Even Cycle Problem, proposed by Erdős [4, 12], asks for the exponent of $\mathrm{ex}(n, C_{2k})$ is open for every k not in $\{2, 3, 5\}$ (see e.g. [3, 11, 20, 21, 46]).

To study the degenerate Turán type problems, we usually reduce it to find some specific structures in a bipartite graph, which is called *Zarankiewicz type problems*.

Definition 4.1. For positive integers m, n, s, t, the *Zarankiewicz number* $Z(m, n, s, t)$, is defined as the maximum number of edges in a bipartite graph $G = (L, R, E)$ with $|L| = m$ and $|R| = n$ that does not contain any complete bipartite subgraph $K_{s,t}$ such that s vertices in L and t vertices in R.

Determining $Z(m, n, s, t)$ is known to be notoriously hard in general. In [13], Füredi gave an upper bound of $Z(m, n, s, t)$, on which our algorithm relied.

Theorem 4.2 (Füredi(1996) [13]). *For positive numbers* $m \geq s, n \geq t, s \geq t \geq 2$,

$$Z(m, n, s, t) < (s - t - 1)^{1/t} n m^{1-1/t} + (t - 2)n + (t - 1)m^{2-2/t}.$$

Definition 4.3. Let $G = (L, R, E)$ be a bipartite graph. A shadow $\mathbf{S}$ for $K_{s,t}$ is a multi-set of tuples (S, q) for some $1 \leq q \leq t - 1$ and $S \subseteq (L \cup R)$ such that $|S \cap L| \geq s$, $|S \cap R| \geq q$ and $G[S]$ contains a copy of $K_{s,q}$ with s vertices in L and q vertices in R.

Following Jain and Seshadhri's strategy in [15], the key idea of the proposed algorithm is to find a proper shadow $\mathbf{S}$ for $K_{s,t}$ in a given bipartite graph $G = (L, R, E)$ such that

$$\sum_{(S,q) \in \mathbf{S}} \mathcal{N}(K_{s,q}, S) = \mathcal{N}(K_{s,t}, G).$$

and sample randomly from it. Note that for $S \subseteq L \cup R$, the existence of $K_{s,q}$ is guaranteed by Theorem 4.2. Let $S_L = S \cap L$ and $S_R = S \cap R$ and $|S_L| = m, |S_R| = n$. We define the *threshold function* as

$$\alpha(S, s, q) = (s - q - 1)^{1/q} n m^{1-1/q} + (q - 2)n + (t - 1)m^{2-2/q}.$$

The threshold function, $\alpha(S, s, q)$ can be used to detect whether a subgraph definitely contains $K_{s,q}$. If the number of edges for $G[S]$ exceeds $\alpha(S, s, q)$, it can tell that the subgraph $G[S]$ must contain $K_{s,q}$, and it can enter the sampling stage. In the sampling stage, we pick a set of vertices of size $s + q$ randomly for each element $(S, q) \in \mathbf{S}$ and check whether (s, q)-biclique could be formed.

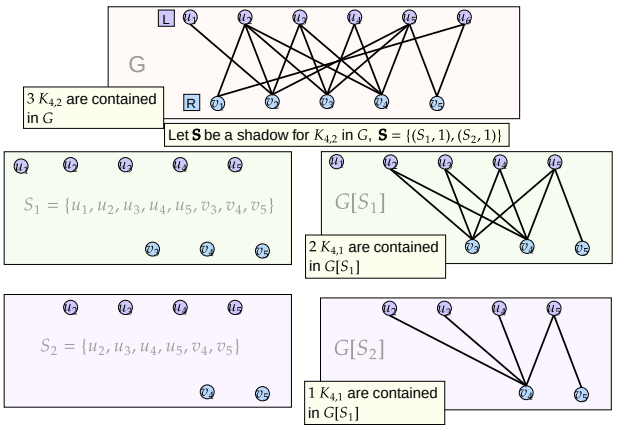


Figure 2: An example of shadow S for $k_{4,2}$

4.1 Vertex-based Shadow Constructor

Let $G = (L, R, E)$ be a bipartite graph with labeled vertices in R as $\{v_1, \cdots, v_{|R|}\}$. For a subset $S \subseteq L \cup R$ and a vertex $v_i \in S_R$, let $N_S(v_i) = N_G(v_i) \cap S$, and $N_S^{2+}(v_i)$ be the set of 2-hop neighbors of v_i in S_R, whose label is greater than v_i. Denote $N_S^+(v_i) = N_S(v_i) \cup N_S^{2+}(v_i)$ and $G_S^{v_i}$ as the subgraph induced by $N_S^+(v_i)$. We call $G_S^{v_i}$ as the subgraph deriving from v_i in S. And the subgraph induced by $N_G^+(v_i)$ is denoted as G_{v_i}. Figure 3 shows a concrete example of these notations.

Algorithm 1 Z-Shadow-Finder (G, s, t)

Input: A bipartite graph $G = (L, R, E)$, integer s, t
Output: A shadow S for $K_{s,t}$ in G

1: **if** $s < t$ **then**
2: $s \Leftrightarrow t, L \Leftrightarrow R$
3: $\mathbf{T} = \{(V(G), t)\}$ and $\mathbf{S} = \emptyset$
4: **while** $\exists (S, q) \in \mathbf{T}$ **do**
5: Delete (S, q) from $\mathbf{T}$
6: **for** each $v_i \in S_R$ with $d_S(v_i) \geq s$ **do**
7: **if** $q > 3$ **then**
8: **if** $e(G_S^{v_i}) \geq \alpha(N_S^+(v_i), s, q-1)$ **then**
9: Add $(N_S^+(v_i), q-1)$ to **S**
10: **else**
11: Add $(N_S^+(v_i), q-1)$ to **T**
12: **if** $q = 3$ **then**
13: Select the two vertices $v_j, v_k \in N_S^{2+}(v_i)$ of maximal degree in $G_S^{v_i}$
14: **if** $|N_S(v_j) \cap N_S(v_k)| > s - 1$ **then**
15: Add $(N_S^+(v_i), 2)$ to **S**
16: **else**
17: Add $(N_S^+(v_i), 2)$ to **T**
18: **if** $q = 2 \wedge$ there is a $K_{s,1}$ in $G_S^{v_i}$ **then**
19: Add $(N_S^+(v_i), 1)$ to **S**

Z-Shadow-Finder is the shadow construction, which aims to get a reasonable sample set S. First, the threshold function requires $s \geq t$, as we can swap L and R, and s, t, respectively (line 1-2) if $s < t$. Then we are given a temporary set T and initialize it to $\mathbf{T} = \{V(G), t\}$ (line 3). And the sample set S is initialized to an empty set (line 3). Each tuple $(S, q) \in \mathbf{T}$ needs to be iterated down

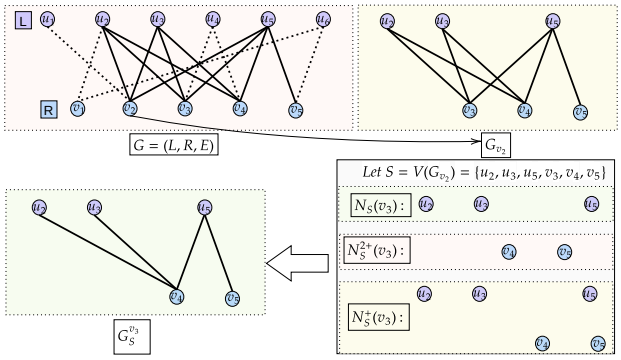


Figure 3: Examples of notations mentioned in section 4.1

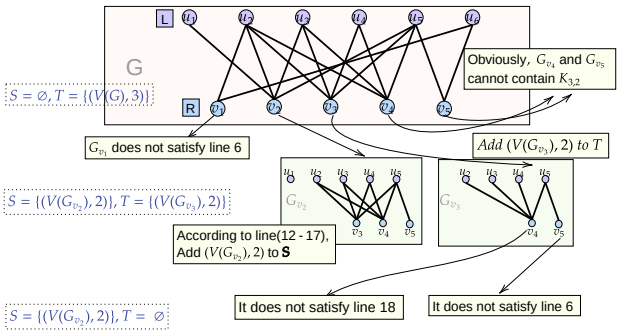


Figure 4: An example of an implementation of Z-Shadow-Finder $(G, 3, 3)$

to get new tuples by decomposing $G[S]$, and the original tuple will be deleted (lines 4-5). For each $v_i \in S_R$, only if $d_S(v_i) > s$, it is possible for $G_S^{v_i}$ to contain a $K_{s,q-1}$ (line 6). Here, we divided the value of q into three cases in the algorithm:

Case 1: q > 3 The threshold function is used to determine whether $G_S^{v_i}$ could enter the sample set S (lines 7-11).

Case 2: q = 3 The threshold function cannot play a good role in this case, because $\alpha(S, s, q)$ is not tight for a small q. Therefore, we directly verify whether the two vertices with the largest degrees and their neighbors in $G_S^{v_i}$ contain a biclique $K_{s,2}$ by (lines 12-17).

Case 3: q = 2 It is equivalent to verifying $G_S^{v_i}$ contains $K_{s,1}$ in the induced subgraph (lines 18-19). And it could be determined by line 6.

Figure 4 shows an example of constructing a shadow for $K_{3,3}$ in G.

For a tuple $(S, q) \in \mathbf{S}$, if S contains a copy of $K_{s,q} = (P, Q, E')$ with $Q = \{v_{i_1}, v_{i_2}, \cdots, v_{i_q}\}$, where $i_1 < i_2 < \cdots < i_q$. Then $G_S^{v_i}$ contains a copy of $K_{s,q-1}$.

Theorem 4.4. *Let* ***S*** *be the shadow of $K_{s,t}$ constructed in* ***Z-Shadow-Finder****. Then, for any $(S, q) \in \boldsymbol{S}$, $G[S]$ contains a copy of $K_{s,q}$ and $\sum_{(S,q)\in \boldsymbol{S}} \mathcal{N}(K_{s,q}, S) = \mathcal{N}(K_{s,t}, G)$.*

Proof. The existence of $K_{s,q}$ in $G[S]$ is guaranteed by Theorem 4.2 if $q > 3$ and is trivial for the other case. Now it suffices to show that, there exists a unique $K_{s,t}$ in G corresponding to any $K_{s,q}$ in S. In fact, for a copy B of $K_{s,t}$ in G, the vertices in the t-part of B have been ordered. In **Z-Shadow-Finder**, we check the vertices in the t-part one by one in this order according to

the threshold function $\alpha(\cdot)$ and q, and output a unique $(S, q) \in \mathbf{S}$. Then $S \cap V(B)$ forms a $K_{s,q}$. The process is reversible and we are done. □

4.2 Sample Procedure

To count the number of $K_{s,t}$ in a bipartite graph G, it suffices to count the number of $K_{s,q}$ in each (S, q), which belongs to the shadow constructed in **Z-Shadow-Finder**(G, s, t). Referring to the strategy for estimating the number of k-cliques in [15], we propose a theoretical probability distribution for the biclique sampling and modify the sampling method in [15] slightly, which are shown in Algorithm 2 and 3.

Algorithm 2 Sample (S)

Output: A set of vertices A

1: For each $(S, q) \in \mathbf{S}$, set $\omega(S) = \binom{|S_L|}{s}\binom{|S_R|}{q}$ and $p(S) = \omega(S)/\sum_{(S',q)\in\mathbf{S}} \omega(S')$
2: Sample (S, q) independently from $\mathbf{S}$ with the probability $p(S)$
3: Uniformly random choose s vertices from S_L and q vertices from S_R to form a vertex subset A.

Algorithm 3 Z-Shadow(Global)

Input: number of samples τ

Output: An estimated value $\hat{K} = \frac{\sum_i X_i}{\tau} \sum_{(S',q)\in\mathbf{S}} \omega(S')$ for K

1: $\mathbf{S} \leftarrow$ **Z-Shadow-Finder**(G, s, t)
2: **for** $i = 1, 2, \ldots, \tau$ **do**
3: $A \leftarrow$ **Sample(S)**
4: **if** $G[A]$ is an (s, q)-biclique **then**
5: $X_i = 1$
6: **else**
7: $X_i = 0$

The algorithm **Sample (S)** firstly samples the tuple $(S, q) \in \mathbf{S}$ with the probability $p(S)$ and then selects certain vertices from the vertices set S to verify whether it forms a biclique. Consequently, **Z-Shadow(Global)** results in a complete biclique estimation algorithm by calling **Z-Shadow-Finder** and **Sample (S)**. The effect of this algorithm is guaranteed by the following result.

Theorem 4.5. *Suppose that ϵ, δ are positive real numbers with $\delta < 1$, and $G = (L, R, E)$ is a bipartite graph with $K = \mathcal{N}(K_{s,t}, G)$. Let $\mathbf{S}$ be the shadow of $K_{s,t}$ constructed in **Z-Shadow-Finder**. If*

$$\tau > \frac{3}{\epsilon^2\gamma}\log\frac{2}{\delta}, \text{ where } \gamma = \min_{(S,q)\in\mathbf{S}} \frac{1}{\binom{|S_L|}{s}\binom{|S_R|}{q}},$$

then Algorithm 3 outputs an estimate $\hat{K}$ for K such that $|\hat{K} - K| \le \epsilon K$ with probability $> 1 - \delta$.

The proof of Theorem 4.5 is given in Appendix A.1.

4.3 Implementations

In our experiments, we mainly use compressed sparse rows (CSR) format, which is the most common for the graphs. It reveals the adjacency relationship between vertices in a graph and contains three arrays:

(1) *Val* index: stores the weight of edges in G;
(2) *Col* index: stores the indices of the neighbors for a vertex;
(3) *Rowptr*: the i-th element records the total number of neighbors of the previous vertices.

In this paper, the default value of the weights for an edge is always equal to 1, so the *Val* array is an all-one array, which can be ignored.

Coordinate format (COO), a sparse matrix storage format, is also used in our experiments. As we know, the COO format is the predecessor of the CSR. In the COO format, the non-zero elements of the matrix are stored as coordinates and consist of three arrays:
(1) *Data* index: stores the weight of an edge in G,
(2) *Col* and *Row* index: Used to store the two endpoints of an edge.

Similarly, the Data index also can be ignored in our experiments. It is worth mentioning that the COO's data have no concern with the order of the vertices. With respect to the vertex labels, We reorder the *Row* index *Col* index, with the neighbors of the same vertex to get the sorted *Col*, which is the *Col* in CSR. *Rowptr* can be obtained by compressing *Row*.

Theorem 4.6. *Let $G = (L, R, E)$ be a bipartite graph and $S \subseteq (L \cap R)$. For $v \in S_R$, the running time of finding G_S^v is*

$$O\left(d_G(v)\log_2|S_L| + \sum_{u\in N_S(v)} d_G(u)\right).$$

Proof. The running time of finding all neighbors of v in S (finding all neighbors of v and take a intersection with S_L) is $O(d_G(v)\log_2|S_L|)$. Then we need to collect all neighbors of u for each $u \in N_S(v)$, whose running time is $O\left(\sum_{u\in N_S(v)} d_G(u)\right)$. □

4.4 Algorithm Analysis

The cost of **Z-Shadow-Finder** mainly includes the cost of inducing subgraphs and verifying the existence of a (p, q)-biclique when $q = 2$. The running time of the remaining operations in the algorithm is $O(1)$. We therefore analyze the runtime by dividing it into two parts as follows.

Since the consuming time for inducing subgraphs is related to

$$|\mathbf{T}| = |\{(S, q) | (S, q) \text{ has existed in } \mathbf{T}\}|,$$

we calculate the upper bound of $|\mathbf{T}|$, before analyzing the cost of **Z-Shadow-Finder**.

Definition 4.7. A subgraph $H = (L', R', E')$ of $G = (L, R, E)$ is called a (α, β)-core candidate, if for any $u \in L'$, and $v \in R'$, $d_H(u) \ge \alpha$ and $d_H(v) \ge \beta$. Moreover, H is a (α, β)-core of G if H is a maximal (α, β)-core candidate.

Let $G = (L, R, E)$ be a bipartite graph and H be its (s, t)-core. Note that to find all copies of $K_{s,t}$, it suffices to find them in H. Let

$$R' = V(H) \cap R, \ \mathbf{T}_i = \{(S, i) | (S, i) \text{ has existed in } \mathbf{T}\}$$

and

$$\Delta_R^{2+} = \max\left\{|N_G^{2+}(v)|, v \in R\right\}.$$

Theorem 4.8. *For $i \ge 1$, $|\mathbf{T}_{t-i}| \le |R'|(\Delta_R^{2+})^{i-1}$.*

Proof. By induction on i, it is trivial if $i = 1$. Assume that the inequality holds for $i-1$. Note that each tuple $(S, t-i+1) \in \mathbf{T}_{t-i+1}$ generates at most Δ_R^{2+} subgraphs. So $|\mathbf{T}_{t-i}| \le |\mathbf{T}_{t-i+1}|\Delta_R^{2+} \le |R'|(\Delta_R^{2+})^{i-1}$. □

Corollary 4.9. $|\mathbf{T}| \le |R'|\sum_{i=1}^{t-1}(\Delta_R^{2+})^{i-1}$.

Theorem 4.10. *The runtime for constructing a shadow is*

$$O(|R'|\Delta_R(\Delta_R^{2+})^{t-1}(\log_2 \Delta_R + \Delta_L)),$$

where $\Delta_R = \max\{d_G(v), v \in R\}$ *and* $\Delta_L = \max\{d_G(v), v \in L\}$.

Proof. For a tuple (S, q) that exists in $\mathbf{T}$, we need to pick one vertex $v \in S_R$ and check the number of edges in G_S^v. Thus the total running time of finding subgraphs in **Z-Shadow-Finder** is

$$W_1 = \sum_{(S,q)\in T} \sum_{v\in S_R} (\text{the running time of inducing } G_S^v).$$

By Theorem 4.6, the running time of inducing G_S^v for a vertex $v \in S_R$ is $O(d_G(v)\log_2 S_L + \sum_{u\in N_S(v)} d_G(u))$. Therefore,

$$\begin{aligned} W_1 &= \sum_{(S,q)\in T} \sum_{v\in S_R} (d_G(v)\log_2 |S_L| + \sum_{u\in N_S(v)} d_G(u)) \\ &\le \sum_{(S,q)\in T} |S_R|(\Delta_R \log_2 \Delta_R + \Delta_L\Delta_R) \\ &\le |T|\Delta_R^{2+}(\Delta_R \log_2 \Delta_R + \Delta_L\Delta_R). \end{aligned} \tag{1}$$

On the other hand, we need to calculate the consuming time for verifying the existence of $(s, 2)$-bicliques, which is

$$W_2 = \sum_{(S,q)\in \mathbf{T}_2} \Delta_R \log_2 \Delta_R = |\mathbf{T}_2|(\Delta_R \log_2 \Delta_R).$$

$$\begin{aligned} O(W_1 + W_2) &= O(|T|\Delta_R^{2+}(\Delta_R \log_2 \Delta_R + \Delta_L\Delta_R)) \\ &= O(|R'|\Delta_R(\Delta_R^{2+})^{t-1}(\log_2 \Delta_R + \Delta_L)) \end{aligned} \tag{2}$$

This completes the proof. □

Theorem 4.11. *The memory complexity of* ***Z-Shadow(Global)*** *is* $O(\Delta_L\Delta_R|R'|(\Delta_R^{2+})^{t-2})$.

Proof. It can be easy to obatain by $\Delta_L\Delta_R|T|$. □

5 ALGORITHM OPTIMIZATIONS

In this section, we optimize our algorithm by reordering the vertices in the graphs to adjust the structure of the graphs. Furthermore, we proposed a modified online method to solve the memory occupation problem.

5.1 Graph Reordering

Actually, the upper bound in Theorem 4.10 is not tight, even much larger than the number of edges in the induced subgraph before reordering. Therefore, for most tuples (S, q), the condition in line 8 of **Z-Shadow-Finder** is hard to be satisfied. For example, for a fixed graph, we choose 10 vertices randomly to form the induced subgraph G_S^v. Table 1 shows the different situations of selected vertices before and after reordering. From observation, $e(G_S^v)$ is much smaller than $\alpha(N_S^+(v), 3, 4)$ before reordering. By Theorem 5.2, we note that $|N_S^{2+}(v)|$ plays a decisive role in $\alpha(N_S^+(v), s, q)$. In order to reduce the value of $\alpha(N_S^+(v), s, q)$, we are going to reduce $|N_S^{2+}(v)|$. To solve this problem, we reorder the vertices in R in increasing degree, such that $d(v_1) \le d(v_2) \cdots \le d(v_{|R|})$, i.e. reorder in ascending order of degree. Take Figure 1 for example, the graph after reordering is shown as Figure 10.

There are two benefits to this:

- Assume that $v_1 \in R$ $G_S^{v_1}$ is likely to be a dense graph and it is easy to achieve the upper bound $\alpha(G_S^v, s, q)$. That will cause more shadow in s.

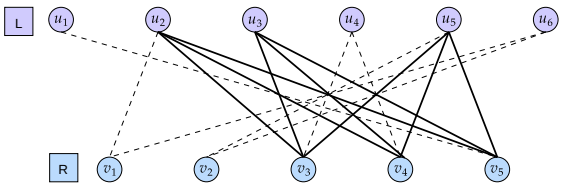


Figure 5: An example for graph reordering

Table 1: The information before and after reordering 10 vertices in a graph, according to the degrees.

Before				After			
id	$\lvert N_G^{2+}(v)\rvert$	$e(G_v)$	$\alpha(G_v, 3, 4)$	id	$\lvert N_G^{2+}(v)\rvert$	$e(G_v)$	$\alpha(G_v, 3, 4)$
7110	2374	61116	156760	8829	918	102870	63677
8012	1250	8004	44002	8005	1741	55672	58285
8435	923	5145	28496	7822	1923	50090	53731
8757	499	973	6255	6897	2690	17658	31318
7558	1949	34491	121724	8758	989	95644	63941
9192	262	480	7142	7572	2174	43948	48079
8490	942	5588	32885	7940	1807	65111	56981
7187	2376	95188	289924	9469	278	97442	38039
7127	2383	73358	170980	8950	797	111680	61094
8774	615	1303	11246	7271	2442	22667	40353

- For the last $t - 1$ vertices in R, the iteration will be interrupted in the pruning process due to the inadequate number of 2-hop neighbors. It effectively avoids the increase of the size of $\mathbf{T}$.

5.2 Online Sampling

In **Z-Shadow(Global)**, most of the time is spent on constructing the shadow, and storing the shadow takes up a lot of storage space. Such defects make our hardware load quite high, which impedes us from dealing with large graphs. To solve this problem, Jain and Seshadhri proposed the online sampling algorithm in [17]. We modify the online sampling algorithm, called **Z-Shadow(online)** , by setting the probability of a vertex $v \in R$ is chosen equal to $p(v) = \frac{\Phi_v}{\Phi}$, where $\Phi_v = \binom{|N_G(v)|}{s}\binom{|N_G^{2+}(v)|}{q}$ and $\Phi = \sum_{v\in R}\Phi_v$. With this probability, the online shadow construction method is with an unbiased sampling. Moreover, we also use the vertex sampling sequences P (line 6) to determine whether a shadow can be reused or abandoned. In this way, the modified online sampling algorithm substantially reduces memory usage and releases the space storing shadows as soon as they are fully utilized. The following two results guarantee the effect of our algorithm.

Theorem 5.1. *$G = (L, R, E)$ is a bipartite graph with $K = \mathcal{N}(K_{s,t}, G)$. The value $\hat{K}$ returned by Algorithm 4 satisfies $\mathbb{E}(\hat{K}) = K$.*

Theorem 5.2. *Suppose that ϵ, δ are positive real numbers with $\delta < 1$, and $G = (L, R, E)$ is a bipartite graph with $K = \mathcal{N}(K_{s,t}, G)$. If $\tau \ge \frac{3\Phi}{\epsilon^2 K}\ln\frac{2}{\delta}$, then Algorithm 4 outputs an estimate $\hat{K}$ for K such $\left|\hat{K} - K\right| < \epsilon K$ with probability $> 1 - \delta$.*

We will leave the proof of these two theorems in Appendix A.2 and A.3, respectively. Meanwhile, we can obtain the memory complexity of online sampling as follows, by the similar discussion as Theorem 4.10.

Algorithm 4 Z-Shadow(Online)

Input: A bipartite graph $G = (L, R, E)$ and integer s, t, τ
Output: An estimate $\hat{K} = \frac{W}{\tau}\Phi$ for K

Order R by degree
$W = 0$
Set the probability of a vertex v be chosen is $p(v) = \frac{\Phi_v}{\Phi}$, where $\Phi_v = \binom{|N_G(v)|}{s}\binom{|N_G^{2+}(v)|}{q}$ and $\Phi = \sum_{v \in R} \Phi_v$
Independently sample τ vertices in R, say $w_1, \cdots, w_\tau$, with probability $p(w_i)$
for $i = 1, 2, \ldots, \tau$ **do**
    $v \leftarrow w_i$
    if $w_i \neq w_{i-1}$ **then**
        $\mathbf{S} \leftarrow$ **Z-Shadow-Finder** $(G_v, s, t-1)$
    Let $\omega_v = \sum_{(S,q) \in \mathbf{S}} \binom{|S_L|}{p}\binom{|S_R|}{q}$
    if Sample$(\mathbf{S})$ is a biclique **then**
        Set $X_i = \omega_v / \Phi_v$
    else
        Set $X_i = 0$
    $W = W + X_i$
    if $w_{i+1} \neq w_i$ **then**
        Delete $\mathbf{S}$

Theorem 5.3. *The memory complexity of **Z-Shadow(Online)** is* $O(\Delta_L \Delta_R (\Delta_R^{2+})^{t-2})$.

6 EXPERIMENTS

In this section, we elaborate on our experiments, including the experiment setup, and evaluation results for running time, accuracy, and memory consumption. Moreover, we also conduct the ablation study for sample size and the effect of vertex reordering. The datasets and demo code are publicized at Github[1].

6.1 Experiments Setup

We will conduct our algorithms in C++ and on a machine equipped with an E5-2697v3(2.30GHz, 3.6GHz turbo) with 16 cores and 36 threads, 4.5MB L2 cache, 45MB L3 cache, and 64GB quad-channel memory. The real datasets are all from KONECT [2], which is bipartite and dense. Moreover, the number of vertices on one side is significantly larger than on the other. We also generate a bunch of artificial graphs with a given number of vertices and edges, whose degrees are in a normal distribution. The statistics of the real and artificial datasets are presented in Table.2, where $|E|$ is the number of edges, $|L|$ and $|R|$ represent the number of vertex in L and R, respectively. And $\overline{d}$, $\overline{d_L}$ and $\overline{d_R}$ represent the average degree of $L \cup R$, L and R, respectively.

Table 2: Basic properties of the dataset

dataset	|E|	|L|	|R|	$\overline{d}$	$\overline{d_L}$	$\overline{d_R}$	origins
movielens-10m	10M	69878	10677	248	143	936	KONECT
flickr	8M	395979	103631	34	21	82	KONECT
movielens-1m	1M	6040	3706	205	165	270	KONECT
movielens-100k	100K	943	1682	76	106	59	KONECT
FilmTrust	35K	1508	2071	20	24	17	KONECT
WikiLens	27k	326	5111	10	83	5	KONECT
moreno-hens	496	32	32	31	31	31	KONECT
S-151M	151M	300000	1000	1007	505	151672	Synthetic
S-2M	2M	10000	1000	364	200	2006	Synthetic
S-917k	917K	3000	500	524	305	1834	Synthetic

[1]https://github.com/Sarabeacon/ZS
[2]http://konect.cc/

Algorithms. In this experiment, we evaluate the following algorithms.

- **Z-S(AD)**: **Z-Shadow(Online)** proposed in Section 5.2, which adopts ascending order of degree.
- **EP++**: An enhanced version of EPivot incorporates additional optimizations such as degree-ordering, which was proposed in [51].
- **Zz**: A dynamic programming-based approximation algorithm is proposed in [51].
- **Zz++**: An improved version of Zigzag proposed in [51].
- **EP/Zz** : A hybrid framework combining EPivot and Zigzag.
- **EP/Zz++** : A hybrid framework combining EPivot with Zigzag++ is proposed in [51].
- **BCL**: The baseline method proposed in [48].
- **BCL++**: The state-of-the-art biclique counting algorithm proposed in [48].
- **PMBE**: Adapted algorithm from maximal biclique enumeration proposed in [1].
- **VSample**: A vertex-based estimation algorithm designed concerning section 5.1 in [30].
- **ESample**: An edge-based estimation algorithm designed concerning section 5.2 in [30].
- **ESpar**: An estimation algorithm based on edge sparsification, which is designed concerning section 6.1 in [30].

Algorithm 4 VSamp

Input: A bipartite graph $G = (L, R, E)$
Output: An estimated value $\hat{K}$ of K

for $i = 1, \cdots, \tau$ **do**
    Choose a vertex w from R uniformly at random
    $A_i \leftarrow$ **BCL++**$(G_w, s, t-1)$
$\hat{K} \leftarrow \frac{|R| \sum_{i=1}^{\tau} A_i}{\tau}$

Algorithm 6 ESamp

Input: A bipartite graph $G = (L, R, E)$
Output: An estimated value $\hat{K}$ of K

for $i = 1, \cdots, \tau$ **do**
    Choose an edge $e = (u, v)$ from E uniformly at random, which $u \in L, v \in R$.
    $N_{>v}(u) = \{w | w \in N(u) \wedge w > v\}$, $G[N^+(e)] = G[N_{>v}(u) \cup N_{>u}(v)]$
    $A_i \leftarrow$ **BCL++**$(G[N^+(e)], s-1, t-1)$
$\hat{K} \leftarrow \frac{|E| \sum_{i=1}^{\tau} A_i}{\tau}$

Algorithm 7 ESpar

Input: A bipartite graph $G = (L, R, E)$, parament $p, 0 \leq q \leq 1$
Output: An estimated value $\hat{K}$ of K

for $i = 1, \cdots, \tau$ **do**
    Construct E_i' by including each edge $e \in E$ independently with probability p
    $A_i \leftarrow$ **BCL++**$((L, R, E_i'), s, t)$
$\hat{K} \leftarrow \frac{\sum_{i=1}^{\tau} A_i}{\tau p^{st}}$

We will leave the analysis related to VSample, ESample, and ESpar in the Appendix A.4.

6.2 Running Time

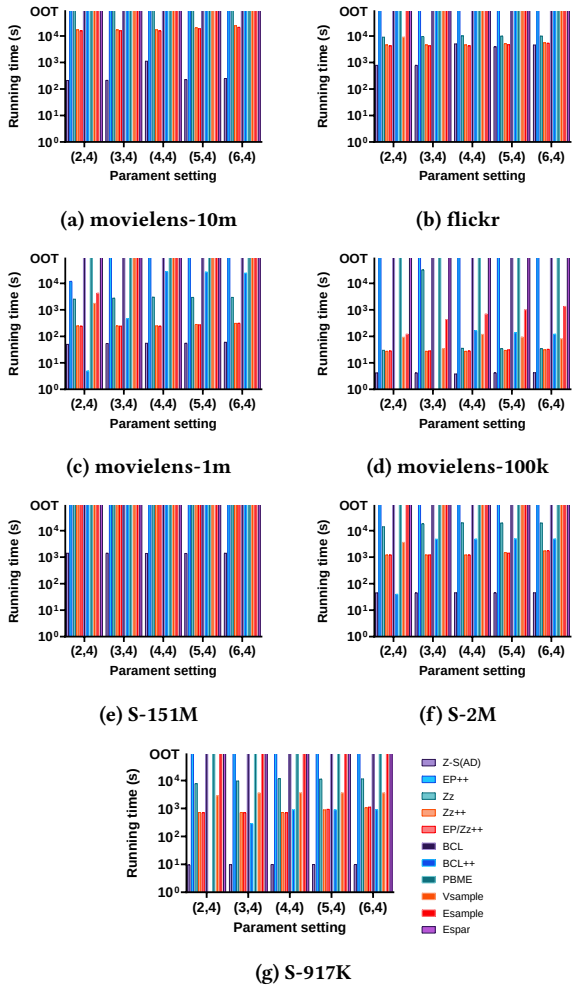


Figure 6: Quantitative evaluation of running time in terms of second ↓

In this subsection, we evaluate the performance of algorithms on the parameter pair (s, t), where t is fixed at 4 and s varies from 2 to 6. In these experiments, the sampling size is set to $10M$. The experimental results are presented in Figure 6. If the runtime exceeds 24 hours, it is recorded as OOT and the corresponding bars are not shown.

In this article, the speedup is defined as a multiple of time reduction. As observed, **Z-Shadow** outperforms other algorithms in most cases, particularly on large, dense graphs. The acceleration results for each comparison algorithm are shown in Table 3. It is shown that compared to two approximation algorithms Zz++

Table 3: The speedup of Z-S(AD) compared to other algorithms.

	Zz++	EP/Zz++	Zz	EP++	BCL++	BCL
MAX	108.9	113.4	1179.5	23622.1	511.2	23622.1
AVERAGE	38.1	37.1	274.9	4953.4	130	4997.9

and EP/Zz++, **Z-S(AD)** achieves a speedup of over 100 times in the best case, with an average speedup exceeding 37 times. For the most efficient exact algorithm, BCL++, **Z-S(AD)** delivers a maximum speedup of up to 500 times and an average speedup of 130 times. Furthermore, the results demonstrate that our algorithm is not sensitive to variations s with fixed t. However, this is not the case for BCL, which is shown in Figure 6a.

6.3 Accuracy of Z-Shadow

To evaluate the accuracy of **Z-Shadow**, we compared it with VSample, ESample, and ESpar. The sampling sizes of all algorithms are set to be $10M$. Recalling that K is the exact value of $\mathcal{N}(K_{s,t}, G)$ given by BCL++ and $\hat{K}$ is the estimated value, the accuracy of $\hat{K}$ is defined as $1 - \frac{|K-\hat{K}|}{K}$. The results, presented in Figure 7, show that **Z-Shadow** maintains a relatively stable accuracy, with an error rate of always within 0.5% on all experimental data sets.

6.4 Memory Reduction

We investigate the memory usage of the **Z-Shadow** algorithm with online sampling and global sampling. Table 4 illustrates the multiple of memory reduction comparing **Z-S(AD)** and **Z-Shadow(Global)**, which is mentioned in Section 4.2 and with ascending degree ordering.

Table 4: Z-S(AD) The multiple of memory reduction compared to Z-S(G)

	M-10M	Flickr	M-1M	M-100K	S-151M	S-2M	S-917K
(2,4)	2493.36	5294.42	774.62	391.45	505.95	25.45	330.60
(3,4)	2528.33	4634.42	773.75	392.83	505.95	30.78	28.18
(4,4)	2672.52	3869.48	832.22	405.62	505.95	49.82	18.91
(5,4)	2813.07	3357.85	951.85	406.91	505.96	65.45	22.50
(6,4)	2947.56	2766.17	988.01	399.12	505.96	73.79	16.40

It is easy to see that the **Z-Shadow** algorithm with online sampling is much more memory-saving than without online sampling. For example, it can be up to 5,000 times for counting the number of $K_{2,4}$ in the dataset movielens-10m.

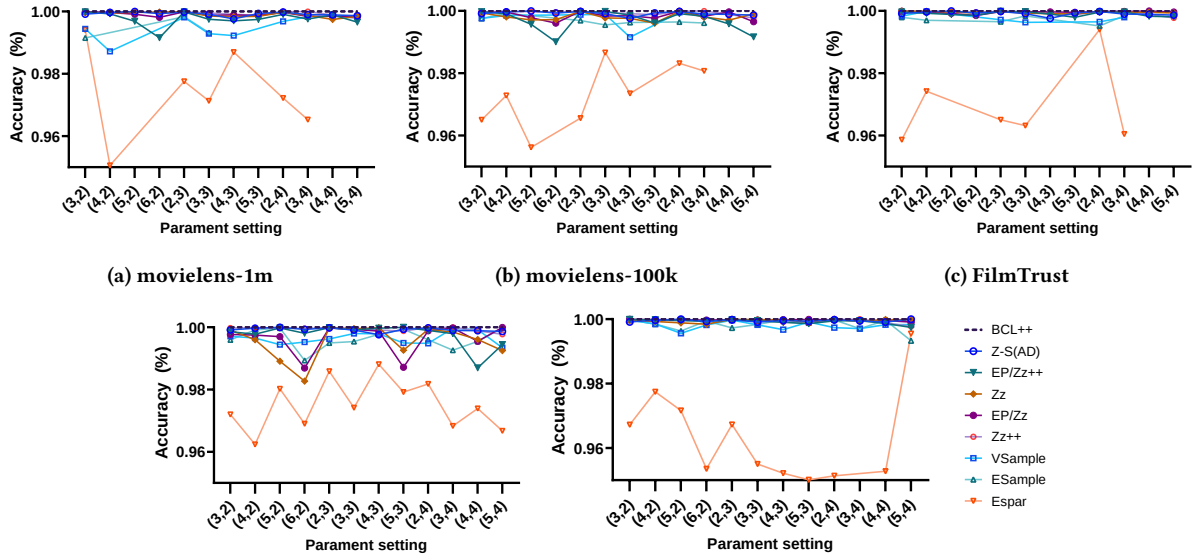
6.5 Impact of Sampling Times

In this subsection, we will explore the effect of sample size on both accuracy and running time. The experiments are conducted on the same five datasets used in Section 5.3. We adjust the sample size $\tau \in \{100K, 1M, 10M, 100M\}$. Figure 8 shows the results on the accuracy. Two datasets, movielens-1m and movielens-100k, are used as examples to investigate the effect of sample size on the run time, whose results are presented in Figure 9.

As shown in Figure 8, when the sample size is down to $100K$, all error rates fall within 5%. Furthermore, as the sample size increases to $10M$, the error rate stabilizes within 1%. When the sample size reaches $100M$, the error rate further reduces to below 0.5%. Overall, accuracy improves with increasing sample size, and it tends to stabilize when the sample size becomes sufficiently large.

As shown in Figure 9, when the sampling size is between $100K$ and $10M$, the running time remains stable with minimal variation. However, a significant increase is observed when the sampling size reaches $100M$.

The growth in time is primarily due to the additional sampling time. To illustrate this, Table 5 shows the time consumption for datasets M-1m and M-100k. The second to last row shows the difference in the time consumption when we do 100M samplings and 100K sampling. The last row shows the difference of 10M



(a) movielens-1m (b) movielens-100k (c) FilmTrust

(d) WikiLens (e) moreno_hens

Figure 7: Quantitative evaluation of accuracy in tems of percent ↑

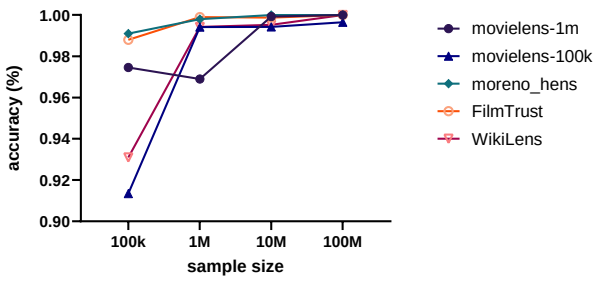


Figure 8: Effect of sample size on accuracy

and 100K samplings. They are both approximately 10 times of the difference in the sampling size. It is evident that most of the shadows are already constructed with $100K$ samplings, and increasing the sampling size further does not significantly affect the time required for shadow construction.

In summary, setting the sampling size to $10M$ is appropriate to balance the accuracy and the time consumption.

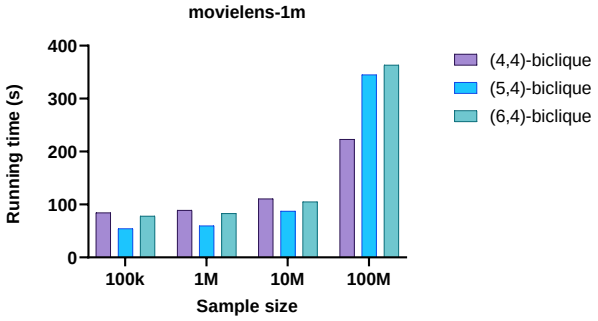
Table 5: Effect of Sample Size on Time

M-1m	$K_{4,4}$	$K_{5,4}$	$K_{6,4}$	M-100k	$K_{4,4}$	$K_{5,4}$	$K_{6,4}$
100K	84.46	54.363	78.016	**100K**	5.841	4.507	4.877
1M	89.158	59.686	83.004	**1M**	6.532	5.321	5.731
10M	110.723	87.493	105.309	**10M**	10.367	14.118	14.33
100M	223.121	345.276	363.634	**100M**	56.42	107.385	104.852
100M-100K	138.661	290.913	285.618	**100M-100K**	50.579	102.878	99.975
10M-100K	26.263	33.13	27.293	**10M-100K**	4.526	9.611	9.453

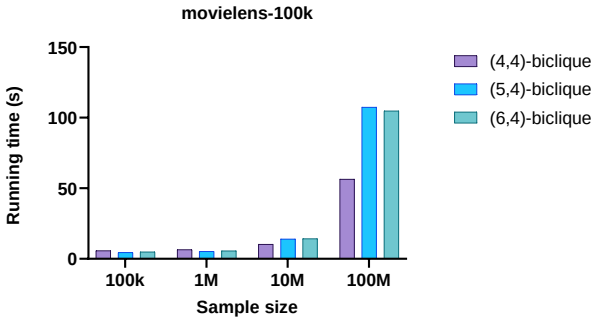
6.6 Impact of reordering

Recalling that $\mathbf{T}$ is the temporary set when we construct the shadow and

$$|\mathbf{T}| = |\{(S,q)|(S,q) \text{ has existed in } \mathbf{T}\}|.$$



(a)



(b)

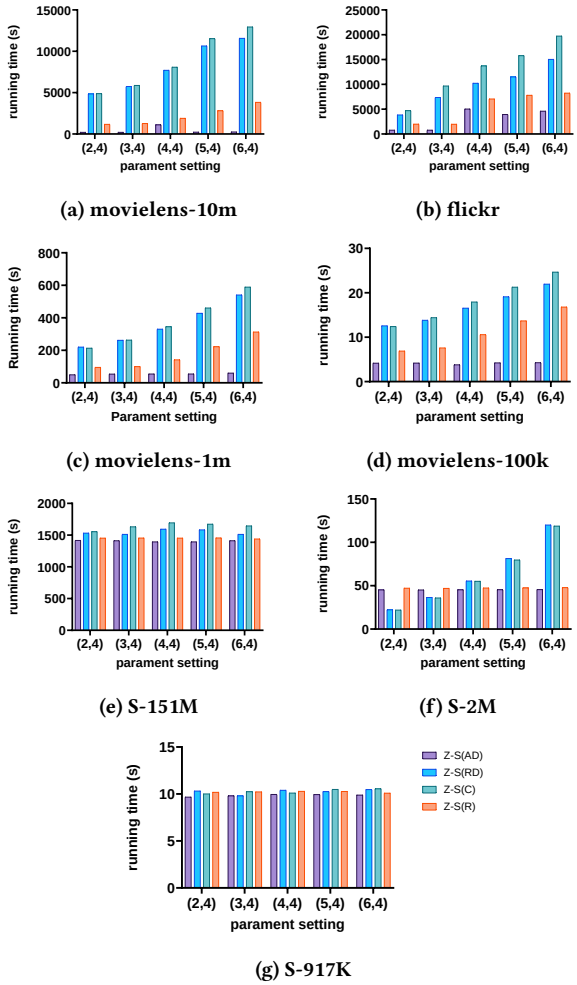
Figure 9: Effect of sample size on time

We define $\widetilde{\mathbf{T}} = \{(S,q)|(S,q) \in \mathbf{T} \text{ and } |S| \geq \alpha(S,s,q)\}$. The *hit ratio* of the thresholds function is defined by

$$\theta = \frac{|\widetilde{\mathbf{T}}|}{|\mathbf{T}|}.$$

In this subsection, we explore the effect of vertex reordering on acceleration, which is illustrated in two aspects. First, Table 6

shows the hit ratio of the thresholds function before and after the ascending degree reordering. It is easy to see that the effect is very significant. The higher the hit ratio, the earlier the graph traversal terminates. On the other hand, Figure 10 shows the effects of different types of vertex reordering applied on **Z-S**, including ascending degree ordering, descending degree ordering, descending core ordering, and random ordering, which are denoted by **Z-S(AD)**, **Z-S(DD)**, **Z-S(C)**, and **Z-S(R)**, respectively. From the observation, **Z-S(AD)** dominates. Although **Z-S(C)** is similar to **Z-S(AD)**, core ordering is an expensive process. It also can be observed that, for some synthetic graphs, such as S-151M, S-2M, and S-917K, there is no significant influence by the vertex reordering. That is because the degree of synthetic graphs are generated by normal distribution and the edge distribution is more evenly distributed. However, the real dataset is not the case. Therefore, vertex ordering is more effective on graphs with high degree heterogeneity.



(a) movielens-10m (b) flickr (c) movielens-1m (d) movielens-100k (e) S-151M (f) S-2M (g) S-917K

Figure 10: The running time of various reordering methods is quantitatively evaluated in seconds↓

6.7 Evaluation for Sparse Graph

As shown in Figure 12, **Z-Shadow** achieves outstanding performance in accuracy. However, BCL++ has an absolute advantage in efficiency as shown in Figure 11. From the perspective of graph theory, here are the reasons: 1) Sampling algorithms fail to play a pruning role on sparse graphs but cause redundant time consumption. 2) There are fewer bicliques in the sparse graphs which causes the probability of extracting biclique to be smaller. This wastes a lot of sampling time.

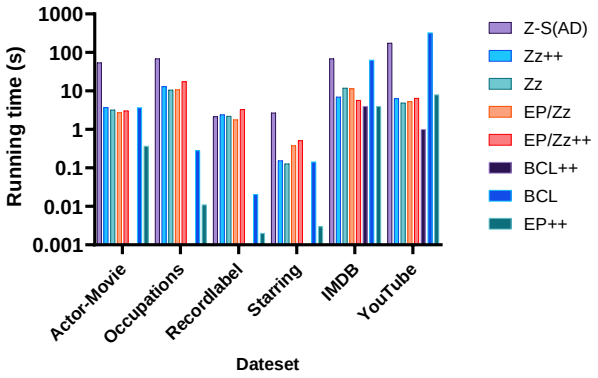


Figure 11: Efficiency evaluation of algorithms on sparse graphs

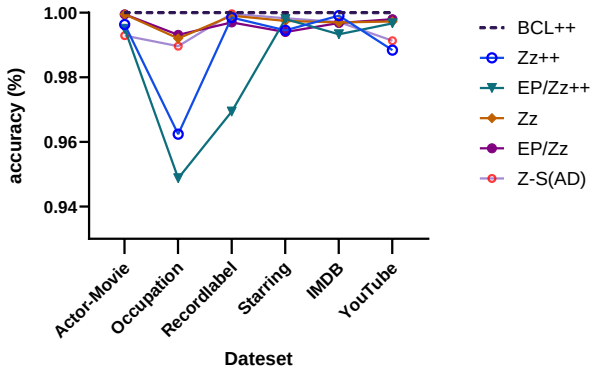


Figure 12: Accuracy evaluation of algorithms on sparse graphs

6.8 Impact of the Threshold Function

Even for butterfly, determining its Zarankiewicz number is a notoriously hard. Füredi's Theorem [13] gives a reasonable upper bound in general. In this subsection, we will evaluate the effect of threshold functions by reducing the exponent of m and n of $\alpha(S, s, q)$. For $0 \le i \le 5$, define

$$\beta(S, s, q, i) = (s-q-1)^{1/q} n^{\frac{20-i}{20}} m^{\frac{q-1}{q+i}} + (q-2) n^{\frac{20-i}{20}} + (q-1) m^{\frac{2q-2}{q+i}}.$$

In the experiment, we selected 6 datasets with different densities and sizes to count $(4, 4)$-bicliques and recorded the performance when applying different threshold functions. Observing Figures 14, it can be seen that the threshold function has a significant impact on the accuracy of the algorithm: the threshold function value is reduced, and its accuracy also decreases. On the other hand, as shown in Figure 13, the reduction of the thresholds has less positive effect on the efficiency. It seems that Füredi's Theorem, the theoretical basis, has played a practical role.

Table 6: The Effect of Vertex Reordering on the Hit Ratio of Bounds

	movielens-1m		movielens-10m		movielens-100k		flickr		S-151M		S-917K		S-2M	
	Z-S(AD)	Z-S(R)	Z-S(AD)	Z-S(R)	Z-S(AD)	Z-S(R)	Z-S(AD)	Z-S(R)	Z-S(AD)	Z-S(R)	Z-S(AD)	Z-S(R)	Z-S(AD)	Z-S(R)
(2,4)	0.779295	0.086654	0.845974	0.020166	0.708452	0.03067	0.061223	0.000637	1	0.532172	1	0.271429	0.556059	0.044055
(3,4)	0.557439	0.070317	0.623576	0.012547	0.436735	0.023486	0.051396	0.000884	1	0.514071	0.990765	0.265896	0.47631	0.057733
(4,4)	0.519205	0.035783	0.560764	0.005585	0.403616	0.006272	0.042911	0.000256	1	0.517996	0.964497	0.251057	0.411262	0.072459
(5,4)	0.498581	0.019913	0.534148	0.003044	0.383442	0.00146	0.041365	0.000027	1	0.492105	0.965866	0.239822	0.327948	0.077176
(6,4)	0.479953	0.01201	0.485802	0.001911	0.365389	0.000287	0.042272	0.000028	1	0.464111	0.933751	0.23808	0.267023	0.080347

Table 7: Basic properties of the sparse dataset

Dataset	$\|L\|$	$\|R\|$	$\|E\|$	$\bar{d}$	$\bar{d}_L$	$\bar{d}_R$	Origins
Actor-Movie	127823	383640	1470404	5.75	11.50	3.83	KONECT
Occupations	127577	101730	250945	2.19	1.97	2.47	KONECT
Recordlabel	168337	18421	233286	2.50	1.38	12.66	KONECT
Starring	76099	81085	281396	3.58	3.69	3.47	KONECT
IMDB	685568	186414	2715604	6.23	3.96	14.56	KONECT
YouTube	94238	30087	293360	4.72	3.11	9.75	KONECT

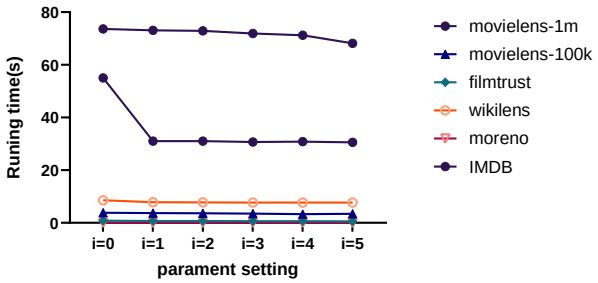


Figure 13: Effect of threshold function reduction on the efficiency of Z-S(AD)

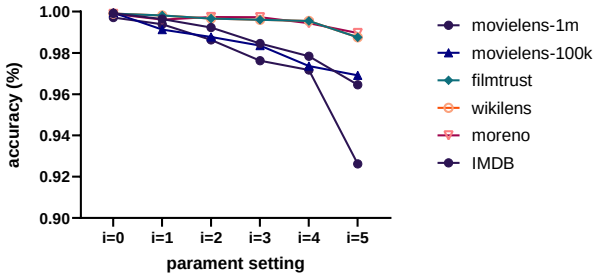


Figure 14: Effect of threshold function reduction on the accuracy of Z-S(AD)

7 CONCLUSION AND FUTURE WORK

This paper presents a provable method for estimating bicliques in massive bipartite graphs using the classical Füredi's Theorem and a novel online sampling method. Our algorithm offers substantial improvements in both computational efficiency and memory usage, achieving up to 100x and 500x speedup over the state-of-the-art approximate and exact methods in experiments, respectively. Its combination of speed, accuracy, and memory efficiency makes it a useful tool in network analysis, particularly in domains where large bipartite structures are common, such as bioinformatics, social networks, and recommendation systems.

The experimental results, conducted on both real-world and synthetic datasets, further validate the robustness and practicality of our approach. The significant reduction in memory consumption, combined with the ability to handle dense datasets, highlights the scalability and effectiveness of **Z-Shadow** in real-world graph analysis tasks. This work provides a practical tool for the large-scale analysis of bipartite networks, with potential applications across fields where such structures are prevalent.

Looking forward, there are several avenues for future research. Firstly, only simple bipartite graphs without loops and multiple edges are considered in this paper. We believe **Z-Shadow** also works on weighted graphs with a minor revision. Another possibility is to extend the **Z-Shadow** algorithm to other types of motif, such as near-biclique, which would broaden its applicability to a wider range of network analysis problems. Additionally, **Z-Shadow** is still an iteration-based algorithm. For small biclique estimation, few number of iterations is sufficient. However, it becomes an obstacle when estimating large bicliques. A reasonable solution is to integrate parallel and distributed computing techniques to further enhance the scalability of **Z-Shadow**. Finally, investigating potential optimizations in the sampling process could further reduce computational overhead, making the algorithm even more efficient.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China (Grant No. 2023YFA1010202), the Central Guidance on Local Science and Technology Development Fund of Fujian Province (Grant No. 2023L3003), the National Natural Science Foundation of China (No.11901094). We are particularly grateful to the authors of [48] for providing the code for BCL and BCL++ and the EDBT reviewers for their insightful comments and constructive suggestions.

A MISSING PROOFS

In this section, we present all the missing proofs from the main sections.

A.1 Proof of Theorem 4.5

Recall that $\omega(S) = \binom{|S_L|}{s}\binom{|S_R|}{q}$ and $\gamma = \min_{(S,q)\in \mathbf{S}} \frac{1}{\binom{|S_L|}{s}\binom{|S_R|}{q}}$. For $(S,q) \in \mathbf{S}$, the probability (S,q) be chosen is $p(S) = \frac{\omega(S)}{\sum_{(S,q)\in\mathbf{S}}\omega(S)}$. Note that for $1 \le r \le \tau$,

$$\begin{aligned}
&\mathbb{P}[X_r = 1] \qquad (3)\\
&= \sum_{(S,q)\in\mathbf{S}} (\mathbb{P}[(S,q) \text{ is chosen}] \times \mathbb{P}[G[A] \text{ is an } (s,q)\text{-biclique}])\\
&= \sum_{(S,q)\in\mathbf{S}} \frac{\omega(S)}{\sum_{(S,q)\in\mathbf{S}}\omega(S)} \times \frac{\mathcal{N}(K_{s,q}, G[S])}{\omega(S)}\\
&= \sum_{(S,q)\in\mathbf{S}} \frac{\mathcal{N}(K_{s,q}, G[S])}{\sum_{(S,q)\in\mathbf{S}}\omega(S)} = \theta := \frac{\mathcal{N}(K_{s,t}, G)}{\sum_{(S,q)\in\mathbf{S}}\omega(S)}
\end{aligned}$$

where the last equation holds by Theorem 4.4. Again, by Theorem 4.4, we know $\mathcal{N}(K_{s,q}, G[S]) \geq \gamma\omega(S)$, which implies that

$$\sum_{(S,q)\in\mathbf{S}} \mathcal{N}(K_{s,q}, G[S]) \geq \gamma \sum_{(S,q)\in\mathbf{S}} \omega(S). \tag{4}$$

Combining (3) and (4) yields that $\mathbb{P}[X_r = 1] \geq \gamma$. Let $X = \sum_{1\leq i\leq\tau} X_i$. Then by the linearity of the expectation, we have

$$\mathbb{E}[X] = \sum_{1\leq r\leq\tau} \mathbb{E}[X_r] \geq \gamma\tau.$$

Note that all X_r are independent. Applying multiplicative Chernoff bound [10] for X yields that

$$\max\{\mathbb{P}[X > (1+\epsilon)\mathbb{E}[X]], \mathbb{P}[X < (1-\epsilon)\mathbb{E}[X]]\} \leq \exp\left(-\frac{\epsilon^2\gamma\tau}{3}\right).$$

Recall that $\tau > \frac{3}{\epsilon^2\gamma}\log\frac{2}{\delta}$. Thus,

$$\max\left\{\mathbb{P}\left[\frac{\sum_{1\leq r\leq\tau} X_r}{\tau} < \theta(1-\epsilon)\right], \mathbb{P}\left[\frac{\sum_{1\leq r\leq\tau} X_r}{\tau} > \theta(1+\epsilon)\right]\right\} \leq \frac{\delta}{2}, \tag{5}$$

and then $\mathbb{P}\left[\theta(1-\epsilon) \leq \frac{\sum_{1\leq r\leq\tau} X_r}{\tau} \leq \theta(1+\epsilon)\right] \geq 1-\frac{\delta}{2}$. Recall that $\hat{K} = \frac{\sum_{1\leq r\leq\tau} X_r}{\tau}\sum_{(S,q)\in\mathbf{S}}\binom{|S_L|}{s}\binom{|S_R|}{q}$. We conclude that

$$\mathbb{P}\left[\theta(1-\epsilon)\sum_{(S,q)\in\mathbf{S}}\omega(S) \leq \hat{K} \leq \theta(1+\epsilon)\sum_{(S,q)\in\mathbf{S}}\omega(S)\right] \geq 1-\delta.$$

which is equivalent $\mathbb{P}\left[(1-\epsilon)K \leq \hat{K} \leq K(1+\epsilon)\right] \geq 1-\delta$. This completes the proof.

A.2 Proof of Theorem 5.1

In this subsection, we prove Theorem 5.1. Let q be a integer and $1 \leq q < t$. Recall that $\omega_v = \sum_{(S,q)\in\mathbf{S}}\binom{|S_L|}{s}\binom{|S_R|}{q}$, where $\mathbf{S}$ is the shadow constructed in $N_G^+(v)$ (line 6). In each sampling, $\mathbb{P}[X_i \neq 0] = \frac{\mathcal{N}(K_{s,q-1},G_v)}{\omega_v}$.

Given a vertex v, let $\mathbf{S}$ be the shadow constructed in G_v and B be a particular $(s,t-1)$-biclique in G_v. By Theorem 4.6, there must exist a pair $(S',q') \in \mathbf{S}$ corresponding to B. Therefore,

$$\begin{aligned}
&\mathbb{P}[B \text{ is chosen}] \\
&= \mathbb{P}[v \text{ is chosen}]\cdot\mathbb{P}[(S',q') \text{ is chosen}]\cdot\mathbb{P}[S'\cap V(B) \text{ is chosen}] \\
&= \frac{\Phi_v}{\Phi}\cdot\frac{\binom{S'_L}{s}\binom{S'_R}{q'}}{\omega_v}\cdot\frac{1}{\binom{S'_L}{s}\binom{S'_R}{q'}} = \frac{\Phi_v}{\Phi\omega_v},
\end{aligned}$$

which implies that

$$\begin{aligned}
\mathbb{E}[X_i] &= \sum_{v\in R}\mathbb{P}[K_{s,t-1}\in G_v \text{ is sampled}]\cdot X_i \\
&= \sum_{v\in R}\mathcal{N}(K_{s,t-1}, G-v)\cdot\frac{\Phi_v}{\Phi\omega_v}\cdot\frac{\omega_v}{\Phi_v} \\
&= \frac{\mathcal{N}(K_{s,t},G)}{\Phi}.
\end{aligned}$$

Since $W = \sum_{i=1}^{\tau} X_i$, we get

$$\mathbb{E}[W] = \frac{\tau K}{\Phi}.$$

Recall that $\hat{K} = \frac{W\Phi}{\tau}$, which implies that $\mathbb{E}[\hat{K}] = K$.

A.3 Proof of Theorem 5.2

Now we prove Theorem 5.2. Note that $\hat{K} = \frac{W}{\tau}\Phi = \frac{\sum_{1\leq i\leq\tau} X_i}{\tau}\Phi$, $\mathbb{E}\left[\sum_{i=1}^{\tau} X_i\right] = \frac{\tau K}{\Phi}$ in A.2. Let $X = \sum_{1\leq i\leq\tau} X_i$. Then

$$\begin{aligned}
\mathbb{P}\left[|\hat{K}-K| \leq \epsilon K\right] &= 1 - \mathbb{P}\left[\hat{K} > (1+\epsilon)K\right] - \mathbb{P}\left[\hat{K} < (1-\epsilon)K\right] \\
&= 1 - \mathbb{P}\left[X > (1+\epsilon)\frac{K\tau}{\Phi}\right] - \mathbb{P}\left[X > (1-\epsilon)\frac{K\tau}{\Phi}\right] \\
&= 1 - \mathbb{P}\left[X > (1+\epsilon)[X]\right] - \mathbb{P}\left[X > (1-\epsilon)\mathbb{E}[X]\right]
\end{aligned}$$

Applying multiplicative Chernoff bound [10] for X yields that

$$\max\{\mathbb{P}[X > (1+\epsilon)\mathbb{E}[X]], \mathbb{P}[X < (1-\epsilon)\mathbb{E}[X]]\} \leq \exp\left(-\frac{\epsilon^2\tau K}{3\Phi}\right).$$

This together with $\tau \geq \frac{3\Phi}{\epsilon^2 K}\ln\frac{2}{\delta}$ yields that

$$\mathbb{P}\left[|\hat{K}-K| \leq \epsilon K\right] \geq 1-\delta.$$

A.4 Analysis Related to Other Sampling Algorithms

Lemma A.1. *Let $G = (L,R,E)$ is a bipartite graph with $K = \mathcal{N}(K_{s,t},G)$ and $\hat{K}$ be the output of **VSamp**, then $E\left[\hat{K}\right] = K$.*

Proof. Let $R = \{v_1,\cdots,v_{|R|}\}$. Suppose a biclique $H = (L',R',E')$, and $R' = \{v_{i1},\cdots,v_{it}\}$. We denote v_i as start vertices of a biclique $H = (L',R',E')$ if $v_i = v_{i1}$. Let X_i be the number of bicliques where v_i is a start vertex. Then $\sum_{i=1}^{|R|} X_i = K$.

For $1 \leq i \leq \tau$, $\mathbb{E}[A_i] = \frac{\sum_{i=1}^{|R|} X_i}{|R|} = \frac{K}{|R|}$. Therefore,

$$\mathbb{E}[\hat{K}] = \mathbb{E}\left[\frac{|R|\sum_{i=1}^{\tau} A_i}{\tau}\right] = \frac{|R|}{\tau}\sum_{i=1}^{\tau}\mathbb{E}[A_i] = K$$

□

Lemma A.2. *Let $G = (L,R,E)$ is a bipartite graph with $K = \mathcal{N}(K_{s,t},G)$ and $\hat{K}$ be the output of **ESamp**, then $E\left[\hat{K}\right] = K$.*

By analogous analysis with Lemma A.1, we also can the following result.

Lemma A.3. *Let $G = (L,R,E)$ is a bipartite graph with $K = \mathcal{N}(K_{s,t},G)$ and $\hat{K}$ be the output of **ESpar**, then $E\left[\hat{K}\right] = K$.*

Proof. For each A_i, $i = 1,\ldots,\tau$, let X_{ij} be a random variable indicating the j-th $K_{s,t}$, $j = 1,\cdots,K$, such that $X_{ij} = 1$ if and only if the j-th $K_{s,t}$ exists in the fixed sampled graph (L,R,E'_i) and 0 otherwise. We have $\mathbb{E}[A_i] = \sum_{j=1}^{K}\mathbb{P}[X_{ij}=1]$, Then

$$\mathbb{E}[\hat{K}] = \frac{1}{\tau p^{st}}\sum_{i=1}^{\tau}\sum_{j=1}^{K}\mathbb{P}[X_{ij}=1] = \frac{1}{\tau p^{st}}\sum_{i=1}^{\tau}\sum_{j=1}^{K}p^{st} = K.$$

□

REFERENCES

[1] Aman Abidi, Rui Zhou, Lu Chen, and Chengfei Liu. 2020. Pivot-based Maximal Biclique Enumeration. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, Christian Bessiere (Ed.). International Joint Conferences on Artificial Intelligence Organization, 3558–3564. https://doi.org/10.24963/ijcai.2020/492 Main track.

[2] Austin R. Benson, David F. Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166. https://doi.org/10.1126/science.aad9029

[3] Clark T. Benson. 1966. Minimal Regular Graphs of Girths Eight and Twelve. *Canadian Journal of Mathematics* 18 (1966), 1091–1094. https://doi.org/10.4153/CJM-1966-109-8

[4] John A. Bondy and Miklós Simonovits. 1974. Cycles of even length in graphs. *Journal of Combinatorial Theory, Series B* 16, 2 (1974), 97–105. https://doi.org/10.1016/0095-8956(74)90052-5

[5] Stephen P. Borgatti and Martin G. Everett. 1997. Network analysis of 2-mode data. *Social Networks* 19, 3 (1997), 243–269. https://doi.org/10.1016/S0378-8733(96)00301-2

[6] Lijun Chang, Rashmika Gamage, and Jeffrey Xu Yu. 2024. Efficient k-Clique Count Estimation with Accuracy Guarantee. *Proc. VLDB Endow.* 17, 11 (July 2024), 3707–3719. https://doi.org/10.14778/3681954.3682032

[7] Chen Chen, Qiuyu Zhu, Yanping Wu, Renjie Sun, Xiaoyang Wang, and Xijuan Liu. 2022. Efficient critical relationships identification in bipartite networks. *World Wide Web* 25, 2 (March 2022), 741–761. https://doi.org/10.1007/s11280-021-00914-2

[8] Maximilien Danisch, Oana Balalau, and Mauro Sozio. 2018. Listing k-cliques in Sparse Real-World Graphs* *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 589–598. https://doi.org/10.1145/3178876.3186125

[9] Apurba Das and Srikanta Tirthapura. 2018. Incremental Maintenance of Maximal Bicliques in a Dynamic Bipartite Graph. *IEEE Transactions on Multi-Scale Computing Systems* 4, 3 (2018), 231–242. https://doi.org/10.1109/TMSCS.2018.2802920

[10] Devdatt P. Dubhashi and Alessandro Panconesi. 2009. *Concentration of Measure for the Analysis of Randomized Algorithms.* Cambridge University Press.

[11] Paul Erdös. 1963. On a Problem in Graph Theory. *The Mathematical Gazette* 47, 361 (1963), 220–223. http://www.jstor.org/stable/3613396

[12] Paul Erdös. 1964. Extremal problems in graph theory. *Publ. House Cszechoslovak Acad. Sci., Prague* (1964), 29–36.

[13] Zoltán Füredi. 1996. An Upper Bound on Zarankiewicz' Problem. *Combinatorics, Probability and Computing* 5, 1 (1996), 29–33. https://doi.org/10.1017/S0963548300001814

[14] Felipe Glaria, Cecilia Hernández, Susana Ladra, Gonzalo Navarro, and Lilian Salinas. 2021. Compact structure for sparse undirected graphs based on a clique graph partition. *Information Sciences* 544 (2021), 485–499. https://doi.org/10.1016/j.ins.2020.09.010

[15] Shweta Jain and Comander Seshadri. 2017. A Fast and Provable Method for Estimating Clique Counts Using Turán's Theorem *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 441–449. https://doi.org/10.1145/3038912.3052636

[16] Shweta Jain and Comander Seshadri. 2020. The Power of Pivoting for Exact Clique Counting. In *Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM '20)*. Association for Computing Machinery, New York, NY, USA, 268–276. https://doi.org/10.1145/3336191.3371839

[17] Shweta Jain and Comander Seshadri. 2020. Provably and Efficiently Approximating Near-cliques using the Turán Shadow: PEANUTS *(WWW '20)*. Association for Computing Machinery, New York, NY, USA, 1966–1976. https://doi.org/10.1145/3366423.3380264

[18] Madhav Jha, Comander Seshadri, 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 (WWW '15)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 495–505. https://doi.org/10.1145/2736277.2741101

[19] Matthieu Latapy, Clémence Magnien, and Nathalie Del Vecchio. 2008. Basic notions for the analysis of large two-mode networks. *Social Networks* 30, 1 (2008), 31–48. https://doi.org/10.1016/j.socnet.2007.04.006

[20] Felix Lazebnik and Vasiliy A. Ustimenko. 1993. New Examples of Graphs without Small Cycles and of Large Size. *European Journal of Combinatorics* 14, 5 (1993), 445–460. https://doi.org/10.1006/eujc.1993.1048

[21] Felix Lazebnik, Vasiliy A. Ustimenko, and Andrew J. Woldar. 1999. Polarities and 2k-cycle-free graphs. *Discrete Mathematics* 197-198 (1999), 503–513. https://doi.org/10.1016/S0012-365X(99)90107-3 16th British Combinatorial Conference.

[22] Jinyan Li, Guimei Liu, Haiquan Li, and Limsoon Wong. 2007. Maximal Biclique Subgraphs and Closed Pattern Pairs of the Adjacency Matrix: A One-to-One Correspondence and Mining Algorithms. *IEEE Transactions on Knowledge and Data Engineering* 19, 12 (2007), 1625–1637. https://doi.org/10.1109/TKDE.2007.190660

[23] Rundong Li, Pinghui Wang, Peng Jia, Xiangliang Zhang, Junzhou Zhao, Jing Tao, Ye Yuan, and Xiaohong Guan. 2022. Approximately Counting Butterflies in Large Bipartite Graph Streams. *IEEE Transactions on Knowledge and Data Engineering* 34, 12 (2022), 5621–5635. https://doi.org/10.1109/TKDE.2021.3062987

[24] Rong-Hua Li, Sen Gao, Lu Qin, Guoren Wang, Weihua Yang, and Jeffrey Xu Yu. 2020. Ordering heuristics for k-clique listing. *Proc. VLDB Endow.* 13, 12 (July 2020), 2536–2548. https://doi.org/10.14778/3407790.3407843

[25] Bingkai Lin. 2018. The Parameterized Complexity of the k-Biclique Problem. *J. ACM* 65, 5, Article 34 (Aug. 2018), 23 pages. https://doi.org/10.1145/3212622

[26] Boge Liu, Long Yuan, Xuemin Lin, Lu Qin, Wenjie Zhang, and Jingren Zhou. 2019. Efficient (,)-core Computation: an Index-based Approach. In *The World Wide Web Conference (WWW '19)*. Association for Computing Machinery, New York, NY, USA, 1130–1141. https://doi.org/10.1145/3308558.3313522

[27] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks V.S. Lakshmanan, Wenjie Zhang, and Xuemin Lin. 2020. Efficient Algorithms for Densest Subgraph Discovery on Large Directed Graphs. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 1051–1066. https://doi.org/10.1145/3318464.3389697

[28] Samuel Manoharan and Sathesh Ammaiappan. 2020. Patient Diet Recommendation System Using K Clique and Deep learning Classifiers. (01 2020), 121–130.

[29] Ryan Rossi, Nesreen Ahmed, and Eunyee Koh. 2018. Higher-order Network Representation Learning. *WWW '18: Companion Proceedings of the The Web Conference 2018*, 3–4. https://doi.org/10.1145/3184558.3186900

[30] Seyed-Vahid Sanei-Mehri, Ahmet Erdem Sariyuce, and Srikanta Tirthapura. 2018. Butterfly Counting in Bipartite Networks *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 2150–2159. https://doi.org/10.1145/3219819.3220097

[31] Yudi Santoso, Venkatesh Srinivasan, and Alex Thomo. 2020. Efficient Enumeration of Four Node Graphlets at Trillion-Scale. In *International Conference on Extending Database Technology*. https://api.semanticscholar.org/CorpusID:214613382

[32] Ahmet Erdem Sarıyüce and Ali Pinar. 2018. Peeling Bipartite Networks for Dense Subgraph Discovery. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*. Association for Computing Machinery, New York, NY, USA, 504–512. https://doi.org/10.1145/3159652.3159678

[33] Comander. Seshadri, Ali Pinar, and Tamara G. Kolda. 2014. Wedge sampling for computing clustering coefficients and triangle counts on large graphs. *Stat. Anal. Data Min.* 7, 4 (Aug. 2014), 294–307. https://doi.org/10.1002/sam.11224

[34] Jessica Shi, Laxman Dhulipala, and Julian Shun. 2021. Parallel clique counting and peeling algorithms. In *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA21)*. SIAM, 135–146.

[35] Phonexay Vilakone, Doo-Soon Park, Khamphaphone Xinchang, and Fei Hao. 2018. An Efficient movie recommendation algorithm based on improved k-clique. *Hum.-Centric Comput. Inf. Sci.* 8, 1, Article 161 (Dec. 2018), 15 pages. https://doi.org/10.1186/s13673-018-0161-6

[36] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1225–1234. https://doi.org/10.1145/2939672.2939753

[37] Haibo Wang, Chuan Zhou, Jia Wu, Weizhen Dang, Xingquan Zhu, and Jilong Wang. 2018. Deep Structure Learning for Fraud Detection. In *2018 IEEE International Conference on Data Mining (ICDM)*. 567–576. https://doi.org/10.1109/ICDM.2018.00072

[38] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. 2020. Efficient Bitruss Decomposition for Large-scale Bipartite Graphs. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 661–672. https://doi.org/10.1109/ICDE48307.2020.00063

[39] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. 2022. Accelerated butterfly counting with vertex priority on bipartite graphs. *The VLDB Journal* 32, 2 (May 2022), 257–281. https://doi.org/10.1007/s00778-022-00746-0

[40] Kaixin Wang, Kaiqiang Yu, and Cheng Long. 2024. Efficient k-Clique Listing: An Edge-Oriented Branching Strategy. *Proc. ACM Manag. Data* 2, 1, Article 7 (March 2024), 26 pages. https://doi.org/10.1145/3639262

[41] Kai Wang, Wenjie Zhang, Xuemin Lin, Ying Zhang, and Shunyang Li. 2022. Discovering Hierarchy of Bipartite Graphs with Cohesive Subgraphs. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 2291–2305. https://doi.org/10.1109/ICDE53745.2022.00217

[42] Kai Wang, Wenjie Zhang, Xuemin Lin, Ying Zhang, Lu Qin, and Yuting Zhang. 2021. Efficient and Effective Community Search on Large-scale Bipartite Graphs. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 85–96. https://doi.org/10.1109/ICDE51399.2021.00015

[43] Mengyuan Wang, Hongbo Jiang, Peng Peng, Youhuan Li, and Wenbin Huang. 2024. Toward Accurate Butterfly Counting with Edge Privacy Preserving in Bipartite Networks. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*. 2289–2298. https://doi.org/10.1109/INFOCOM52122.2024.10621436

[44] Pinghui Wang, Junzhou Zhao, Xiangliang Zhang, Zhenguo Li, Jiefeng Cheng, John C.S. Lui, Don Towsley, Jing Tao, and Xiaohong Guan. 2018. MOSS-5: A Fast Method of Approximating Counts of 5-Node Graphlets in Large Graphs. *IEEE Transactions on Knowledge and Data Engineering* 30, 1 (2018), 73–86. https://doi.org/10.1109/TKDE.2017.2756836

[45] Yue Wang, Ruiqi Xu, Xun Jian, Alexander Zhou, and Lei Chen. 2022. Towards distributed bitruss decomposition on bipartite graphs. *Proc. VLDB Endow.* 15, 9 (May 2022), 1889–1901. https://doi.org/10.14778/3538598.3538610

[46] Rephael Wenger. 1991. Extremal graphs with no C4's, C6's, or C10's. *Journal of Combinatorial Theory, Series B* 52, 1 (1991), 113–116. https://doi.org/10.1016/0095-8956(91)90097-4

[47] Qingyu Xu, Feng Zhang, Zhiming Yao, Lv Lu, Xiaoyong Du, Dong Deng, and Bingsheng He. 2022. Efficient load-balanced butterfly counting on GPU. *Proc. VLDB Endow.* 15, 11 (July 2022), 2450–2462. https://doi.org/10.14778/3551793.3551806

[48] Jianye Yang, Yun Peng, and Wenjie Zhang. 2021. (p,q)-biclique counting and enumeration for large sparse bipartite graphs. *Proc. VLDB Endow.* 15, 2 (Oct. 2021), 141–153. https://doi.org/10.14778/3489496.3489497

[49] Renchi Yang, Jieming Shi, Keke Huang, and Xiaokui Xiao. 2022. Scalable and Effective Bipartite Network Embedding. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 1977–1991. https://doi.org/10.1145/3514221.3517838

[50] Xiaowei Ye, Rong-Hua Li, Qiangqiang Dai, Hongzhi Chen, and Guoren Wang. 2024. Efficient k-Clique Counting on Large Graphs: The Power of Color-Based Sampling Approaches. *IEEE Transactions on Knowledge and Data Engineering* 36, 4 (2024), 1518–1536. https://doi.org/10.1109/TKDE.2023.3314643

[51] Xiaowei Ye, Rong-Hua Li, Qiangqiang Dai, Hongchao Qin, and Guoren Wang. 2023. Efficient Biclique Counting in Large Bipartite Graphs. *Proc. ACM Manag. Data* 1, 1, Article 78 (May 2023), 26 pages. https://doi.org/10.1145/3588932

[52] Hao Yin, Austin R. Benson, and Jure Leskovec. 2018. Higher-order clustering in networks. *Phys. Rev. E* 97 (May 2018), 052306. Issue 5. https://doi.org/10.1103/PhysRevE.97.052306

[53] Yanlei Yu, Zhiwu Lu, Jiajun Liu, Guoping Zhao, and Ji-rong Wen. 2019. RUM: Network Representation Learning Using Motifs. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 1382–1393. https://doi.org/10.1109/ICDE.2019.00125

[54] Long Yuan, Lu Qin, Wenjie Zhang, Lijun Chang, and Jianye Yang. 2018. Index-Based Densest Clique Percolation Community Search in Networks. *IEEE Transactions on Knowledge and Data Engineering* 30, 5 (2018), 922–935. https://doi.org/10.1109/TKDE.2017.2783933

[55] Long Yuan, Lu Qin, Wenjie Zhang, Lijun Chang, and Jianye Yang. 2018. Index-Based Densest Clique Percolation Community Search in Networks. *IEEE Transactions on Knowledge and Data Engineering* 30, 5 (2018), 922–935. https://doi.org/10.1109/TKDE.2017.2783933

[56] Ge Zhang, Zhao Li, Jiaming Huang, Jia Wu, Chuan Zhou, Jian Yang, and Jianliang Gao. 2022. eFraudCom: An E-commerce Fraud Detection System via Competitive Graph Neural Networks. *ACM Trans. Inf. Syst.* 40, 3, Article 47 (March 2022), 29 pages. https://doi.org/10.1145/3474379

[57] Lingling Zhang, Zhiwei Zhang, Guoren Wang, Ye Yuan, and Kangfei Zhao. 2023. Efficiently Counting Triangles for Hypergraph Streams by Reservoir-Based Sampling. *IEEE Transactions on Knowledge and Data Engineering* 35, 11 (2023), 11328–11341. https://doi.org/10.1109/TKDE.2023.3236335